Multicopter Design and Control Practice Experiments

RflySim Advanced Courses
Lesson 07: UAV Swarm Control

Dr. Xunhua Dai, Associate Professor,

School of Computer Science and Engineering,
Central South University, China;

Email: dai.xh@csu.edu.cn ;

https://faculty.csu.edu.cn/daixunhua

北航可靠飞行控制研究组
BUAA Reliable Flight Control Group

Content

1. Setup Instructions

2. Interface Introduction

3. Basic examples

4. Advanced examples

5. summary

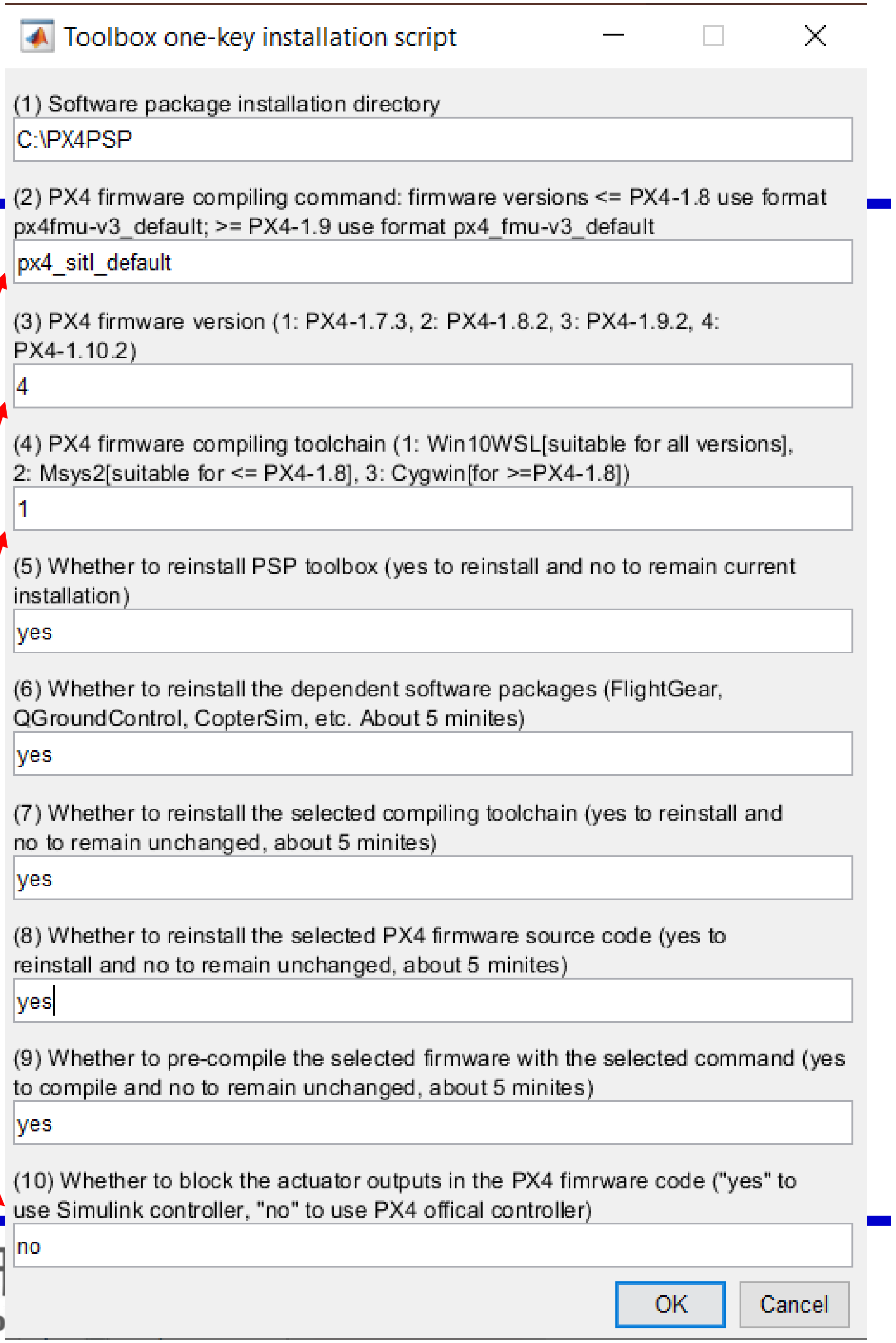
Path of demo source code of this Lesson: RflySimAPIs\SimulinkSwarmAPI

1. Setup Instructions

1.1 RflySim platform configuration

- When this platform is used for vision or swarm control algorithm development, it is recommended to re-run the "**OnekeyScript.p**" script in the installation package and use the configuration as shown in the right figure:
- Use the PX4_SITL software-in-the-loop firmware to compile, where the compiling command is "**px4_sitl_default**"
- Use the latest PX4 firmware **PX4-1.10.2**, select "**4**" for the firmware version
- Use the **Win10WSL** compiler, so select "**1**" for the compiler
- Whether to shield PX4 output items, select “**no**"
- Click the "**OK**" button to start the installation.

If you want to use Pixhawk for Hardware-In-the-Loop (HIL) simulation, you also need to correctly configure the Pixhawk autopilot through QGC according to the method in **Section 2.5** of "**RflySim_Lesson_01_Introduction.pdf**"



Toolbox one-key installation script

(1) Software package installation directory

C:\PX4PSP

(2) PX4 firmware compiling command: firmware versions <= PX4-1.8 use format px4fmu-v3_default; >= PX4-1.9 use format px4_fmu-v3_default

px4_sitl_default

(3) PX4 firmware version (1: PX4-1.7.3, 2: PX4-1.8.2, 3: PX4-1.9.2, 4: PX4-1.10.2)

4

(4) PX4 firmware compiling toolchain (1: Win10WSL[suitable for all versions], 2: Msys2[suitable for <= PX4-1.8], 3: Cygwin[for >=PX4-1.8])

1

(5) Whether to reinstall PSP toolbox (yes to reinstall and no to remain current installation)

yes

(6) Whether to reinstall the dependent software packages (FlightGear, QGroundControl, CopterSim, etc. About 5 minites)

yes

(7) Whether to reinstall the selected compiling toolchain (yes to reinstall and no to remain unchanged, about 5 minites)

yes

(8) Whether to reinstall the selected PX4 firmware source code (yes to reinstall and no to remain unchanged, about 5 minites)

yes

(9) Whether to pre-compile the selected firmware with the selected command (yes to compile and no to remain unchanged, about 5 minites)

yes

(10) Whether to block the actuator outputs in the PX4 fimrware code ("yes" to use Simulink controller, "no" to use PX4 offical controller)

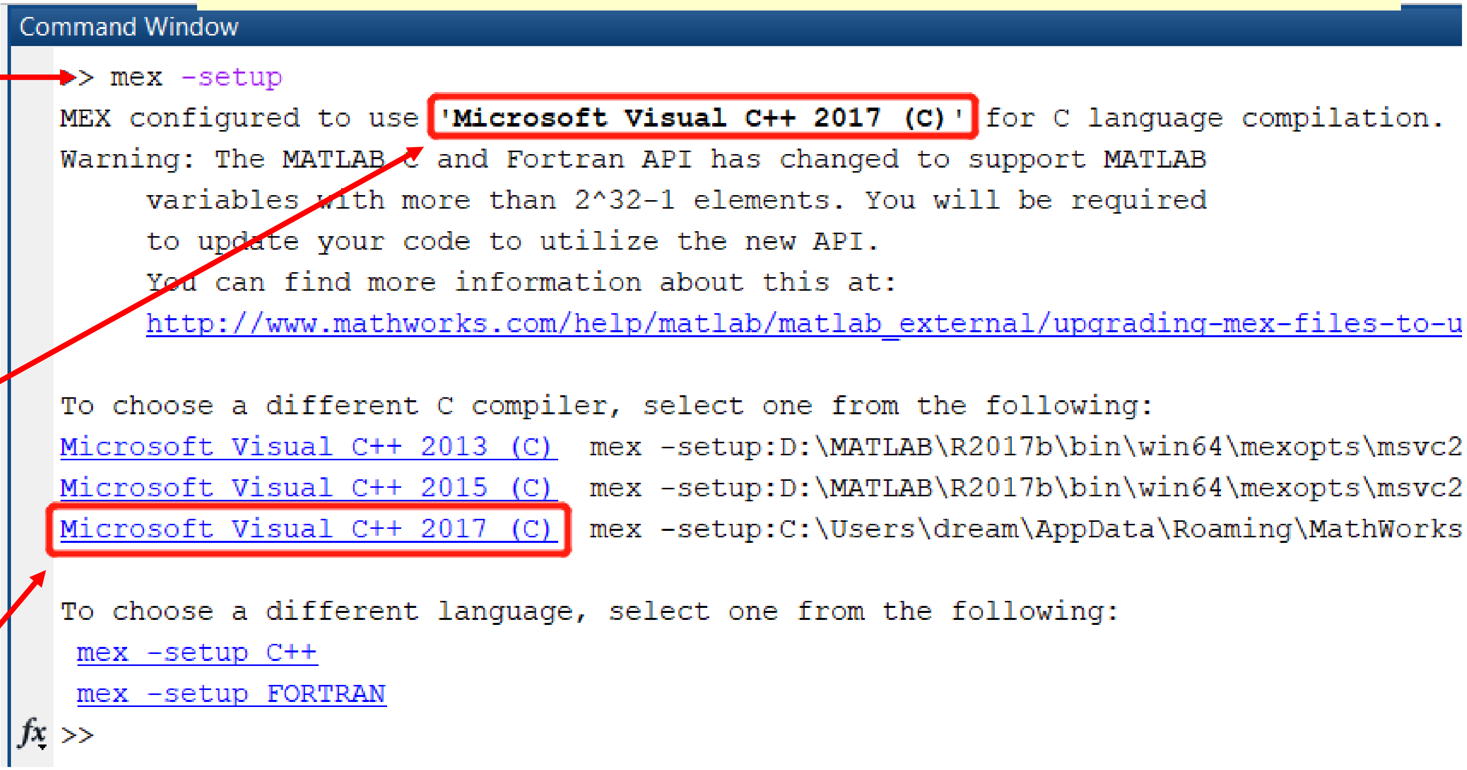
no

1. Configuration of software & hardware

This platform needs to use the C++ compiler of Visual Studio 2015, 2017 or above (MinGW is not supported), so it is necessary to install Visual Studio in advance and configure MATLAB for identification. Please follow the steps in **Section 1.8** of **Lesson 1** of the Advanced Course to install the VS compiler

1.2 Configure C++ Compiler for MATLAB

- Enter the command "**mex -setup**" in the MATLAB "**Command Window**"
- Generally speaking, the Visual Studio (VS) 2017 compiler will be automatically recognized and installed. As shown in the right figure, "MEX configured to use 'Microsoft Visual C++ 2017' for", indicating that the installation is correct
- This page can also switch to other compilers such as Visual Studio 2013/2015

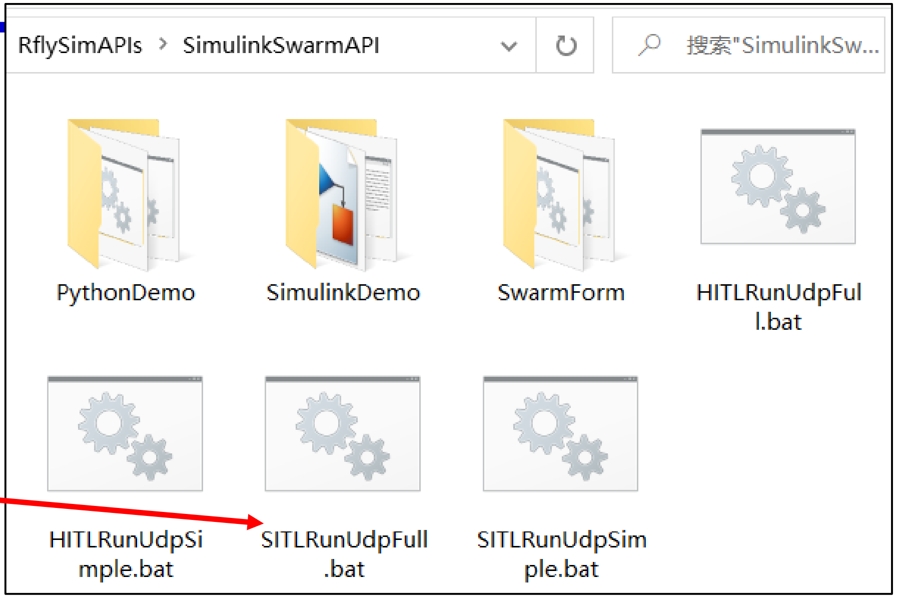


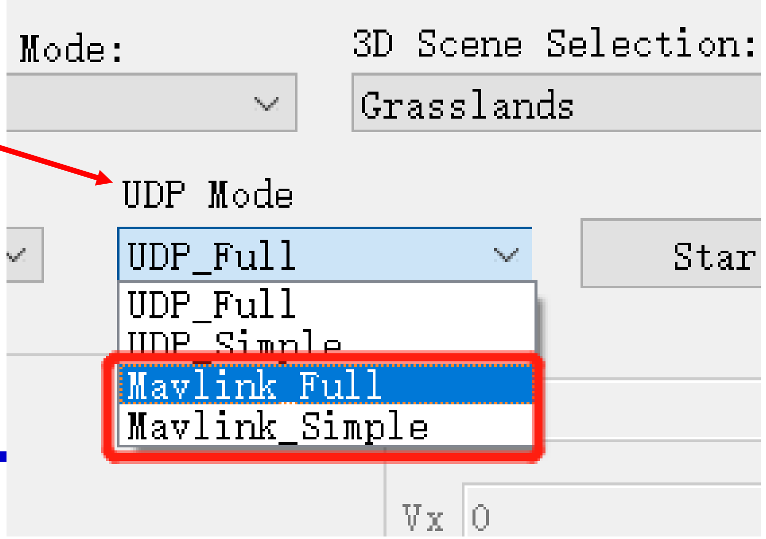
```
>> mex -setup
MEX configured to use 'Microsoft Visual C++ 2017 (C)' for C language compilation.
Warning: The MATLAB C and Fortran API has changed to support MATLAB
     variables with more than 2^32-1 elements. You will be required
     to update your code to utilize the new API.
     You can find more information about this at:
     http://www.mathworks.com/help/matlab/matlab_external/upgrading-mex-files-to-u

To choose a different C compiler, select one from the following:
Microsoft Visual C++ 2013 (C)  mex -setup:D:\MATLAB\R2017b\bin\win64\mexopts\msvc2
Microsoft Visual C++ 2015 (C)  mex -setup:D:\MATLAB\R2017b\bin\win64\mexopts\msvc2
Microsoft Visual C++ 2017 (C)  mex -setup:C:\Users\dream\AppData\Roaming\MathWorks

To choose a different language, select one from the following:
 mex -setup C++
 mex -setup FORTRAN
fx >>
```

1. Setup Instructions

1.3 Introduction to the swarm demo folder

- The swarm control demos of this course are stored under the path "**RflySimAPIs\SimulinkSwarmAPI**". The swarm control algorithms inside are mainly implemented by Simulink, and a Python interface is also provided.
- As shown in the figure on the right, the folder contains **bat** startup scripts with two suffixes "**Full**" and "**Simple**", which correspond to the "**UDP_Mode**" option in CopterSim in the figure below, aiming at the communication requirements of swarm simulations of different sizes.
- "**Full**" mode (CopterSim default) uses a complete communication protocol with detailed data, but too many vehicles will cause network saturation and large delays; the "**Simple**" mode is recommended for large-scale swarm simulation (the number of vehicle on per computer ≥8). Using the simplified protocol, the amount of transmitted data is small with low delay.





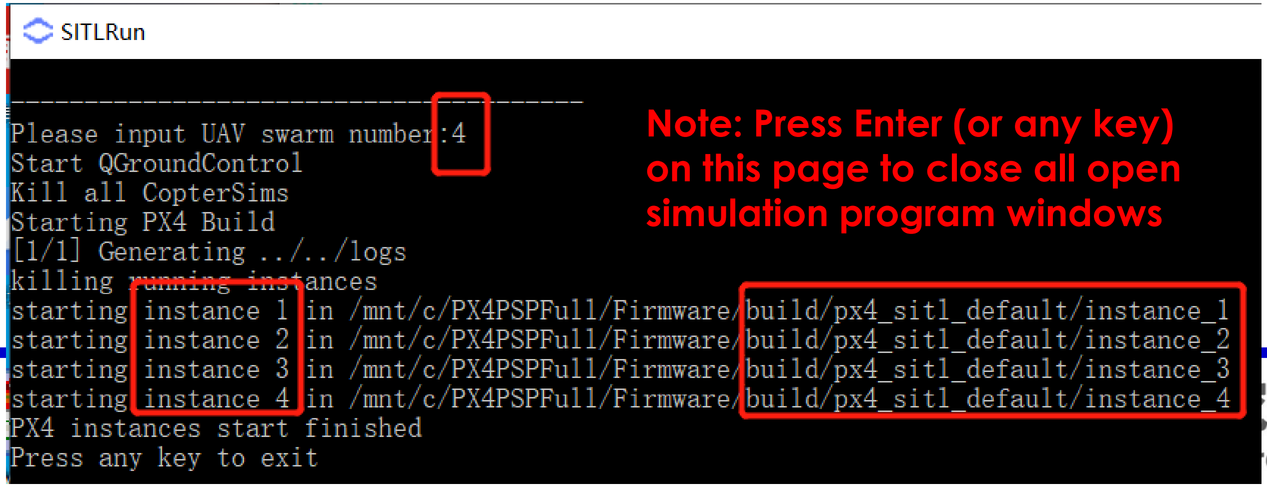
1. Setup Instructions

SITLRun.bat

SITLRun

1.4 Swarm software in the loop simulation test

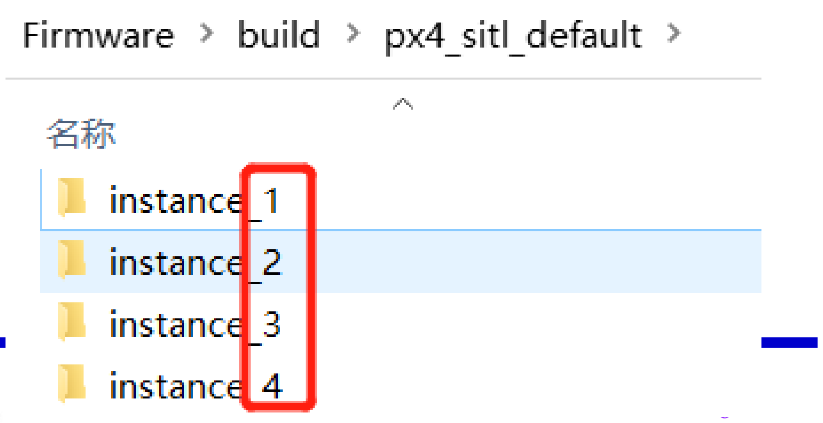
- Double-click to run "**RflySimAPIs\SITLRun.bat**" (or the desktop shortcut with the same name), enter "**4**" in the pop-up command prompt, and click keyboard "**Enter**" to quickly start the swarm simulation system of 4 vehicles
- The automatically opened software includes 4 CopterSim (one for each vehicle), 1 RflySim3D (all vehicles will be displayed at the same time), 1 QGC ground station (display and control all vehicles at the same time), and 1 command prompt window (call **Win10WSL** compiler enables four PX4_SITL controllers, and the log and parameters of each controller can be accessed folder "**Firmware\build\px4_sitl_default\instance_***")



SITLRun

```
-----------------------------------
Please input UAV swarm number:4
Start QGroundControl
Kill all CopterSims
Starting PX4 Build
[1/1] Generating ../../logs
killing running instances
starting instance 1 in /mnt/c/PX4PSPFull/Firmware/build/px4_sitl_default/instance_1
starting instance 2 in /mnt/c/PX4PSPFull/Firmware/build/px4_sitl_default/instance_2
starting instance 3 in /mnt/c/PX4PSPFull/Firmware/build/px4_sitl_default/instance_3
starting instance 4 in /mnt/c/PX4PSPFull/Firmware/build/px4_sitl_default/instance_4
PX4 instances start finished
Press any key to exit
```

Note: Press Enter (or any key) on this page to close all open simulation program windows



Firmware › build › px4_sitl_default ›

名称

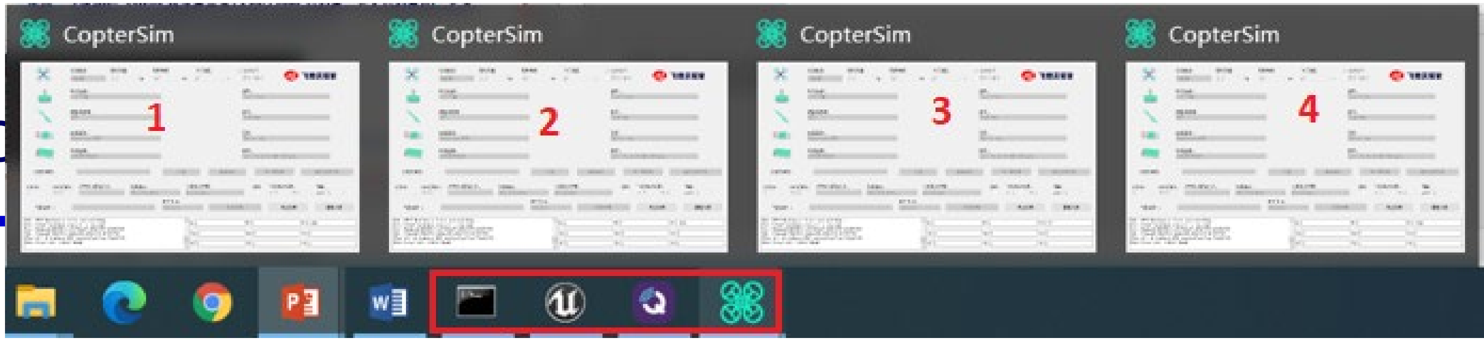
instance_1

instance_2

instance_3

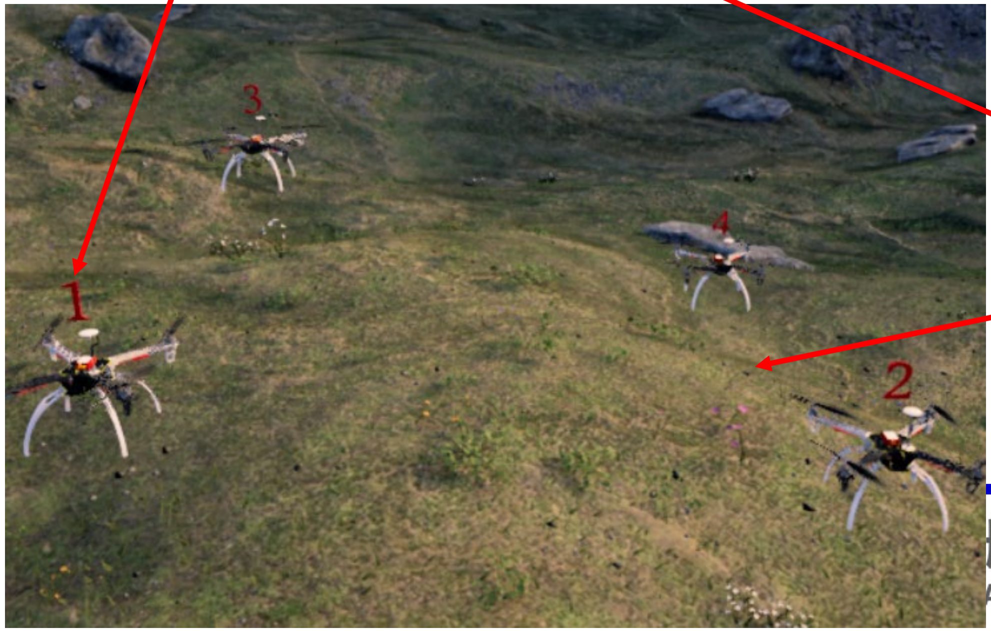
instance_4

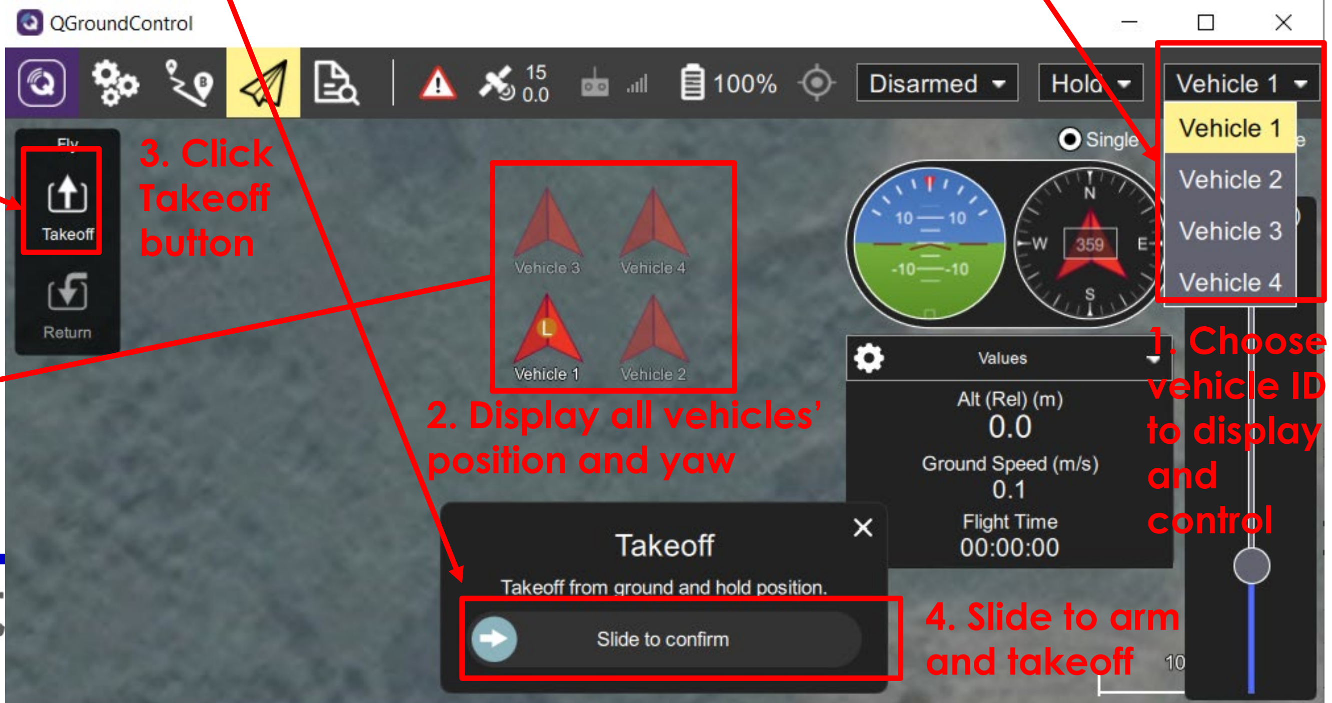
1. Setup Instructio



1.5 Swarm SIL simulation results

- As shown in the upper right picture, all open programs can be seen on the Windows taskbar
- As shown in the figure below Right, switch each drone (Vehicle 1~4) in turn, and click the “**Takeoff**" button in turn, and then click “**Slide to confirm**" to control the 4 vehicle to take off in turn
- After taking off, press the key "**S**" in RflySim3D to display the ID of each vehicle, and the effect is shown on the left





Takeoff

Takeoff from ground and hold position.

Slide to confirm

1. Choose vehicle ID to display and control

2. Display all vehicles' position and yaw

3. Click Takeoff button

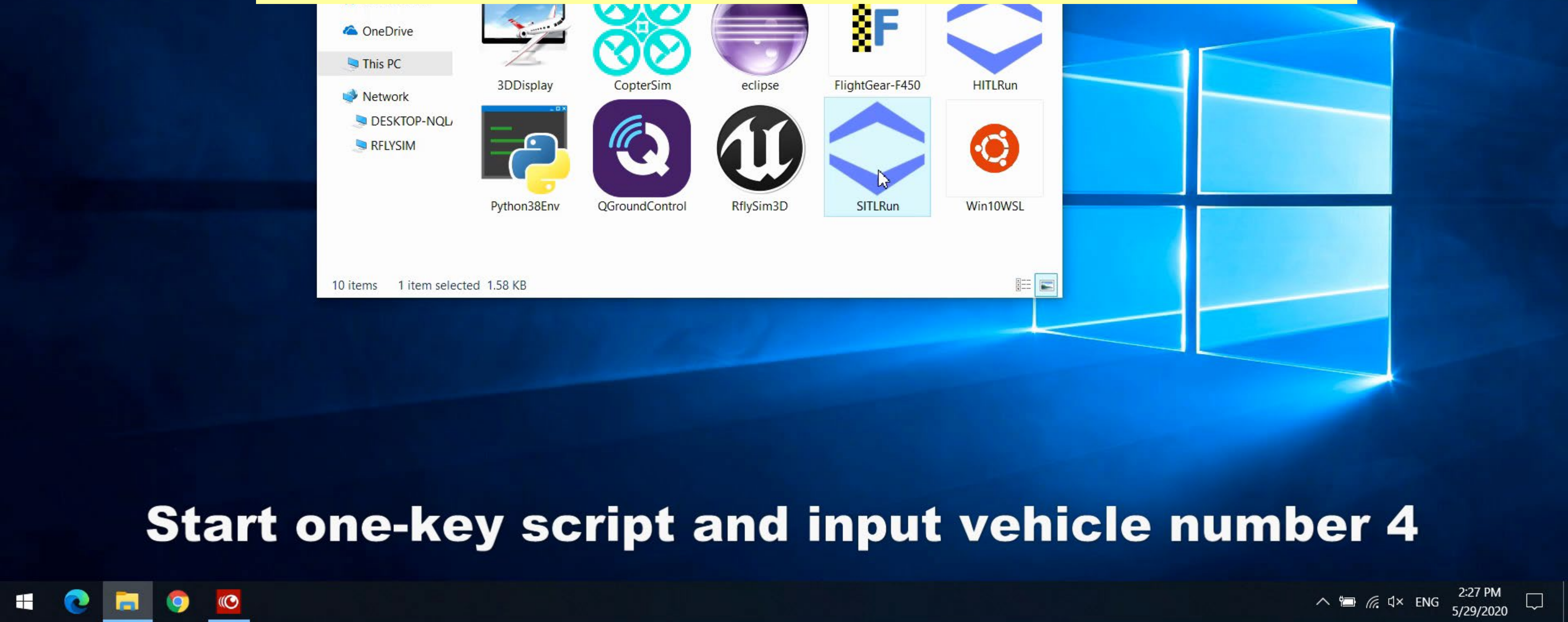
4. Slide to arm and takeoff

RflySim: How to quickly perform software-in-the-loop (SIL) simulation for UAV swarm

Watch this video by clicking the following links:

YouTube: https: //youtu.be/88dGpErxPJ8

Youku: https: //v.youku.com/v_show/id_XNDcwNjA4MzA0NA==.html



10 items   1 item selected 1.58 KB

2:27 PM
5/29/2020

Start one-key script and input vehicle number 4

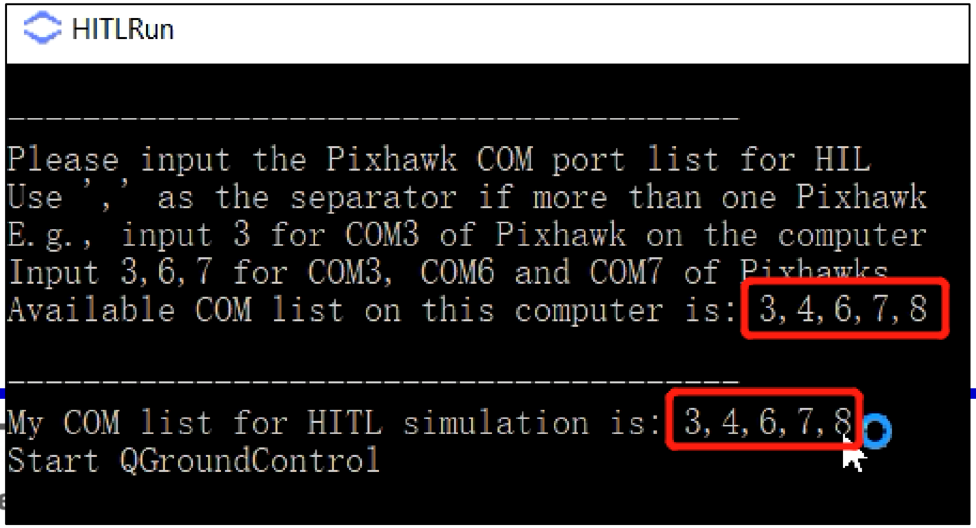
1. Setup Instructions

HITLRun.bat

HITLRun

1.6 Swarm hardware-in-the-loop simulation

- As shown in the figure on the left below, insert all Pixhawk flight controllers (please set the Pixhawk to HITL mode in advance according to the steps in **Section 2.5~2.7** of **Lesson 1**) into the computer
- Double-click to run "**RflySimAPIs\HITLRun.bat**" (or a shortcut with the same name on the desktop), enter the Pixhawk serial port number (separated by commas) in the pop-up window shown on the right, and click the key “**Enter**“ to quickly open multiple vehicles swarm simulation system
- The following two figures correspond to the situation of connecting five Pixhawks for hardware-in-the-loop (HIL) simulation. The running results are the same as those in **Section 1.5**



HITLRun

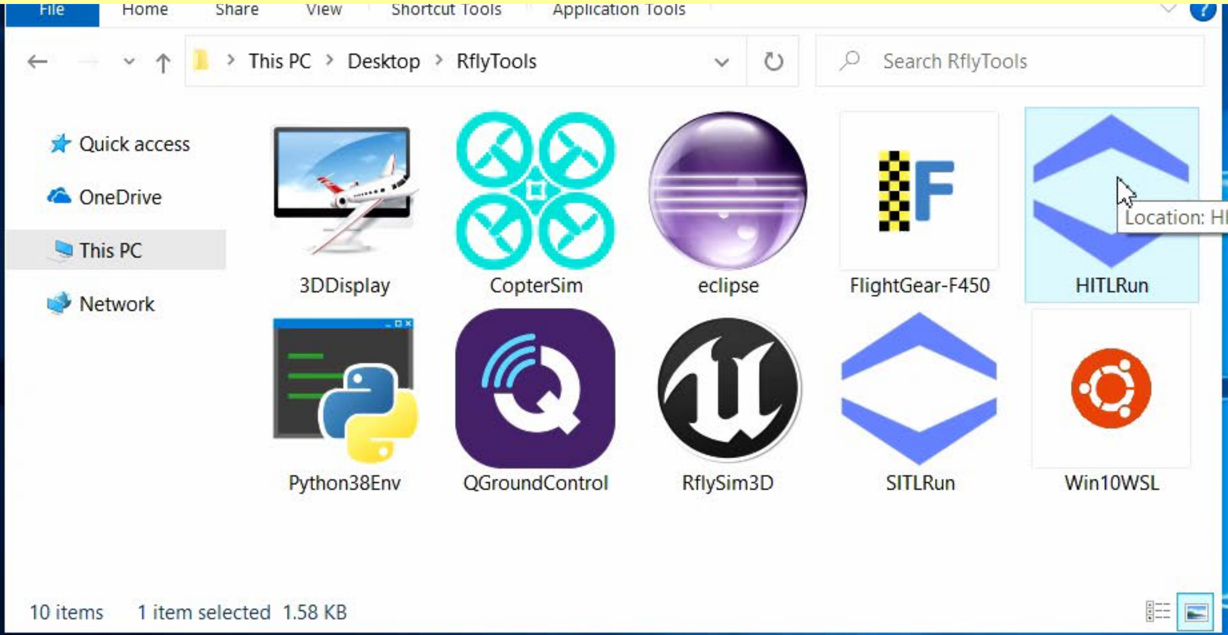
```
----------------------------------------
Please input the Pixhawk COM port list for HIL
Use ',' as the separator if more than one Pixhawk
E.g., input 3 for COM3 of Pixhawk on the computer
Input 3,6,7 for COM3, COM6 and COM7 of Pixhawks
Available COM list on this computer is: 3,4,6,7,8

----------------------------------------
My COM list for HITL simulation is: 3,4,6,7,8
Start QGroundControl
```

RflySim: How to use hardware-in-the-loop simulation for UAV swarm

Watch this video by clicking the following links:

YouTube: https: //youtu.be/oZ_-yhEgebA

Youku: https: //v.youku.com/v_show/id_XNDcwNjA3OTQ0MA==.html



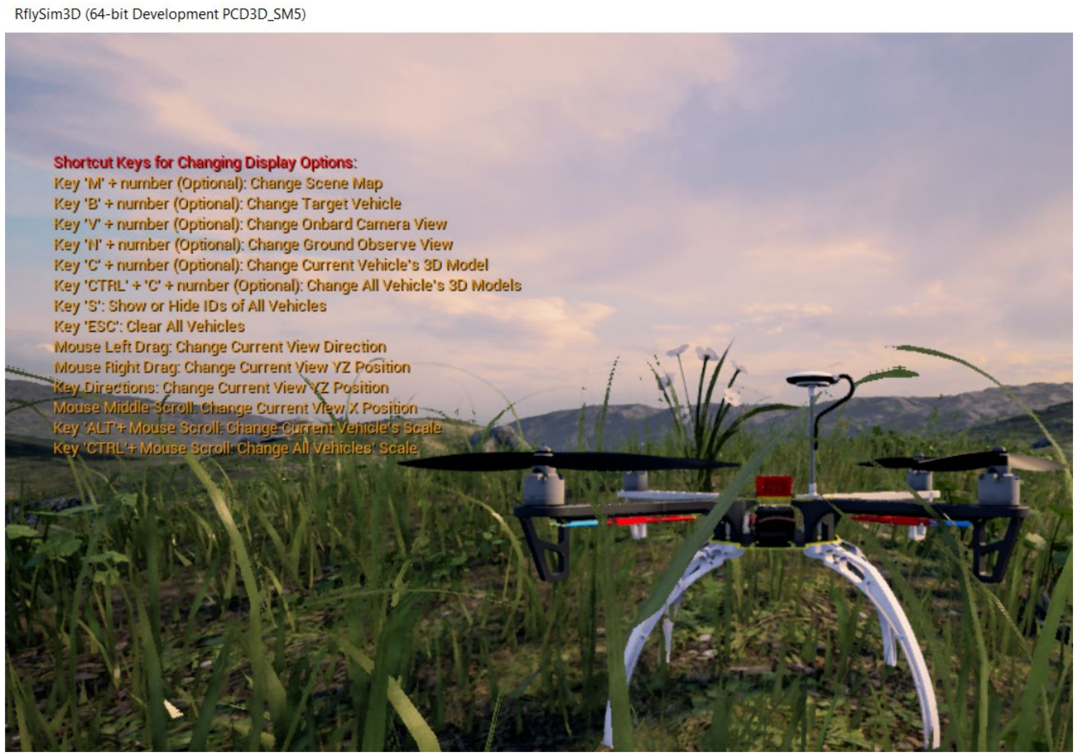
Connect all Pixhawks to computer with USB

1. Setup Instructions

Note: *You can open multiple CopterSims and one RflySim3D at the same time, and then try the following shortcuts*

1.7 RflySim3D’s basic keyboard shortcuts

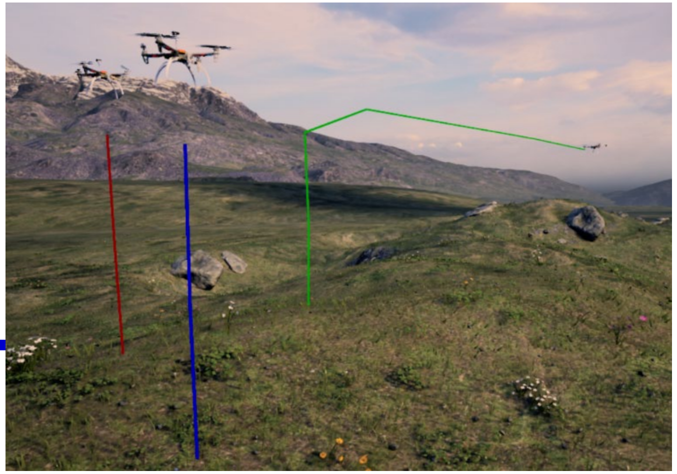
- **F1**: pop-up help menu notice;
- **ESC**: clear all vehicles
- **S**: show/hide vehicle ID;
- **M**: switch among maps (All CopterSim should be closed to observe effect)；
- **M+number***: switch to the ***th** map;
- **B**: switch among different vehicles；
- **B+number***: switch to the ***th** vehicle;
- **C**: switch current vehicle 3D model；
- **C+number***: switch to the ***th** 3D model;
- **CTRL + C**: switch all vehicle’s 3D Model
- **V**: switch among onboard cameras of the vehicle，**0**: chasing view, **1**: front camera, **2**: right camera, …；
- **V+number***: switch to the ***th** viewpoint


RflySim3D (64-bit Development PCD3D_SM5)

Shortcut Keys for Changing Display Options:
Key 'M' + number (Optional): Change Scene Map
Key 'B' + number (Optional): Change Target Vehicle
Key 'V' + number (Optional): Change Onbard Camera View
Key 'N' + number (Optional): Change Ground Observe View
Key 'C' + number (Optional): Change Current Vehicle's 3D Model
Key 'CTRL' + 'C' + number (Optional): Change All Vehicle's 3D Models
Key 'S': Show or Hide IDs of All Vehicles
Key 'ESC': Clear All Vehicles
Mouse Left Drag: Change Current View Direction
Mouse Right Drag: Change Current View YZ Position
Key Directions: Change Current View YZ Position
Mouse Middle Scroll: Change Current View X Position
Key 'ALT'+ Mouse Scroll: Change Current Vehicle's Scale
Key 'CTRL'+ Mouse Scroll: Change All Vehicles' Scale

1. Setup Instructions

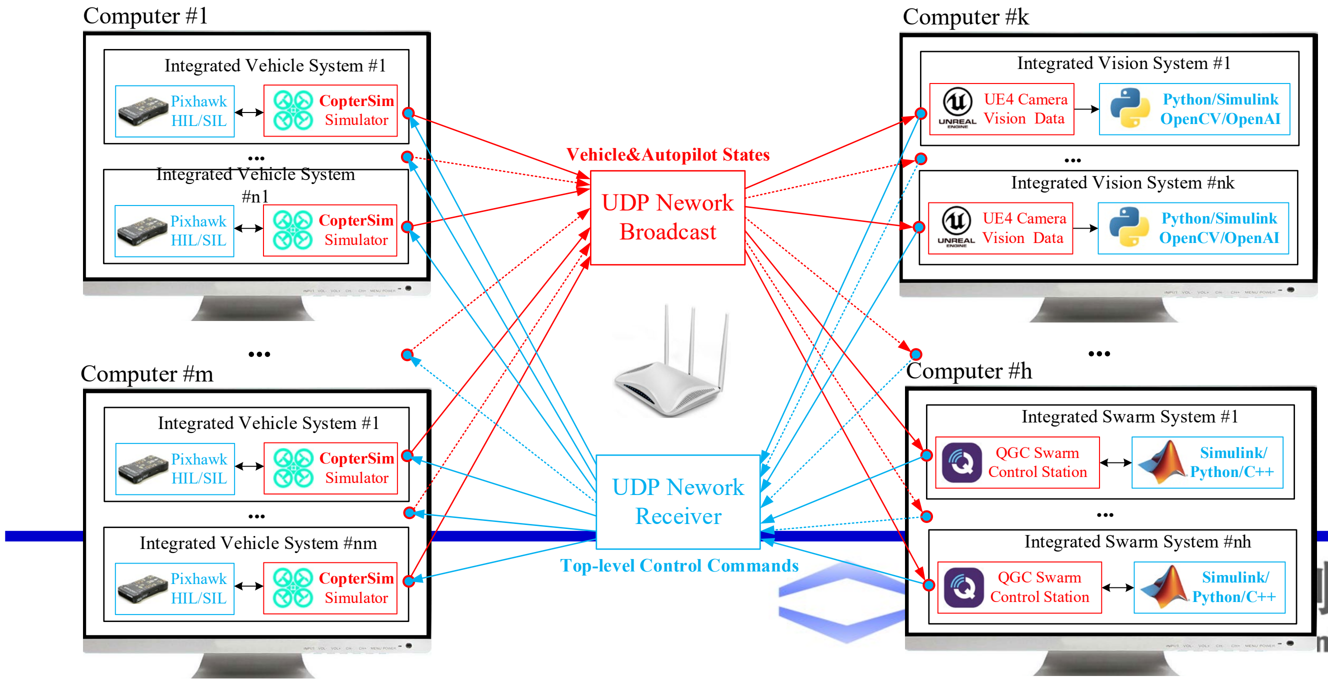
- **N**: switch to vehicle god’s viewpoint， **0:** follow vehicle perspective(not changing perspective angle with posture changing), **1:** fixed ground perspective and keep looking at current vehicle, **2:** fixed ground northern perspective, **3:** fixed ground southern perspective…
- **N+number***: switch to *th god’s perspective,
- **Mouse left key press to move**: switch perspective angle; Mouse right key press to move: switch vertical yz location of the perspective
- **Mouse middle wheel**: switch horizontal x location of the perspective
- **CRTL + Mouse middle wheel**: zoom in/out all the vehicles’ size (easy to observe multi vehicles’ move);
- **ALT + Mouse middle wheel**: zoom in/out current vehicle.
- **T:** show or hide vehicles’ trajectories
- **T+ number*:** change the width of trajectories.



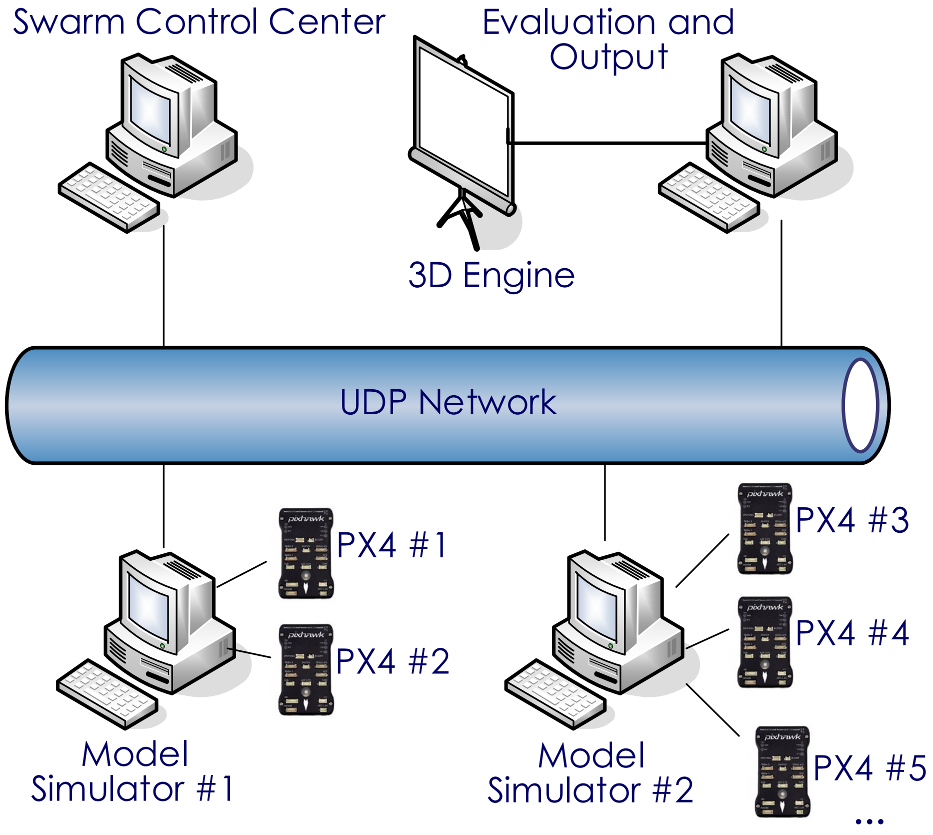
1. Setup Instructions

1.8 RflySim swarm system distributed architecture

- Each APP is an independent process, which can be opened on the LAN computer
- Insert multiple Pixhawk flight controllers to perform multi-computer distributed swarm simulation
- Internal networking simulates actual hardware data transmission networking



Computer #1
Integrated Vehicle System #1
Pixhawk HIL/SIL
CopterSim Simulator
...
Integrated Vehicle System #n1
Pixhawk HIL/SIL
CopterSim Simulator
...
Computer #m
Integrated Vehicle System #1
Pixhawk HIL/SIL
CopterSim Simulator
...
Integrated Vehicle System #nm
Pixhawk HIL/SIL
CopterSim Simulator
Vehicle&Autopilot States
UDP Nework Broadcast
UDP Nework Receiver
Top-level Control Commands
Computer #k
Integrated Vision System #1
UE4 Camera
Vision Data
Python/Simulink OpenCV/OpenAI
...
Integrated Vision System #nk
UE4 Camera
Vision Data
Python/Simulink OpenCV/OpenAI
...
Computer #h
Integrated Swarm System #1
QGC Swarm Control Station
Simulink/ Python/C++
...
Integrated Swarm System #nh
QGC Swarm Control Station
Simulink/ Python/C++



1. Setup Instructions

1.8 Swarm simulation difficulties and RflySim platform solutions

- Question 1: It is too burden to start N vehicles for simulation?
  - **Solution**: One-click script to start all programs and complete parameter configuration
- Question 2: Multiple vehicles are displayed in the same scene at the same time?
  - **Solution**: Use UDP broadcast mode. RflySim3D automatically generate new vehicles when receiving vehicle data with different vehicle ID. Theoretically, it can expand to any number of vehicles, and it also supports the creation of auxiliary objects such as obstacles and pedestrians;
- Question 3: Too many vehicles and network congestion?
  - **Solution**: Provide a variety of data protocols, and the most simplified data mode can be selected for large-scale swarm to ensure smooth network
- Question 4: If a multi-vehicle swarm uses hardware in the loop, the cost is high and the operation is complicated?
  - **Solution**: Support PX4 SITL software-in-the-loop simulation, create and run multiple complete PX4 controllers directly on the computer, and can access the logs, flight parameters, etc. of each controller
- Question 5: How about the performance of the simulation platform?
  - **Solution**: mainly QGC and RflySim3D occupy computing resources, while models and controllers occupy less resources. Preliminary test: one high performance personal computer (PC) can smoothly run more than 15 vehicles in software-in-the-loop simulation, the number will increase if QGC and RflySim3D are not running, and the number can be doubled again if use Pixhawk hardware-in-the-loop simulation.

1. Setup Instructions

1.8 Swarm simulation difficulties and RflySim platform solutions

- Question 6: Simulink slows down with the increase in the control of vehicle, so it cannot be controlled in real time?
  - **Solution**: Support Simulink automatic code generation exe executable file, computer performance occupancy is close to 0, so the real-time simulation problem is solved.
- Question 7: With limited performance of a single computer, it can only simulate a specific number of vehicles?
  - **Solution**: Support online simulation of swarm in LAN to ensure the smooth progress of large-scale swarm, support LAN broadcast and designated IP methods to reduce communication delay
- Question 8: How can swarm simulation be as close to actual flight as possible?
  - **Solution 1**: The Simulink swarm interface receives the internal estimated state sent by Pixhawk through MAVLink instead of the simulated ideal value, and the control command sent is also in the MAVLink control format, so the swarm controller can be directly used for real flight.
  - **Solution 2**: The Simulink controller supports the generation of C/C++ code to be embedded in the original swarm system, and also supports the direct use of Simulink to receive MAVLink messages sent from the LAN Pixhawk for swarm control during actual flight. That is: after the simulation, it can be used directly in real flight without modification.

Content

1. Setup Instructions

2. Interface Introduction

3. Basic examples

4. Advanced examples

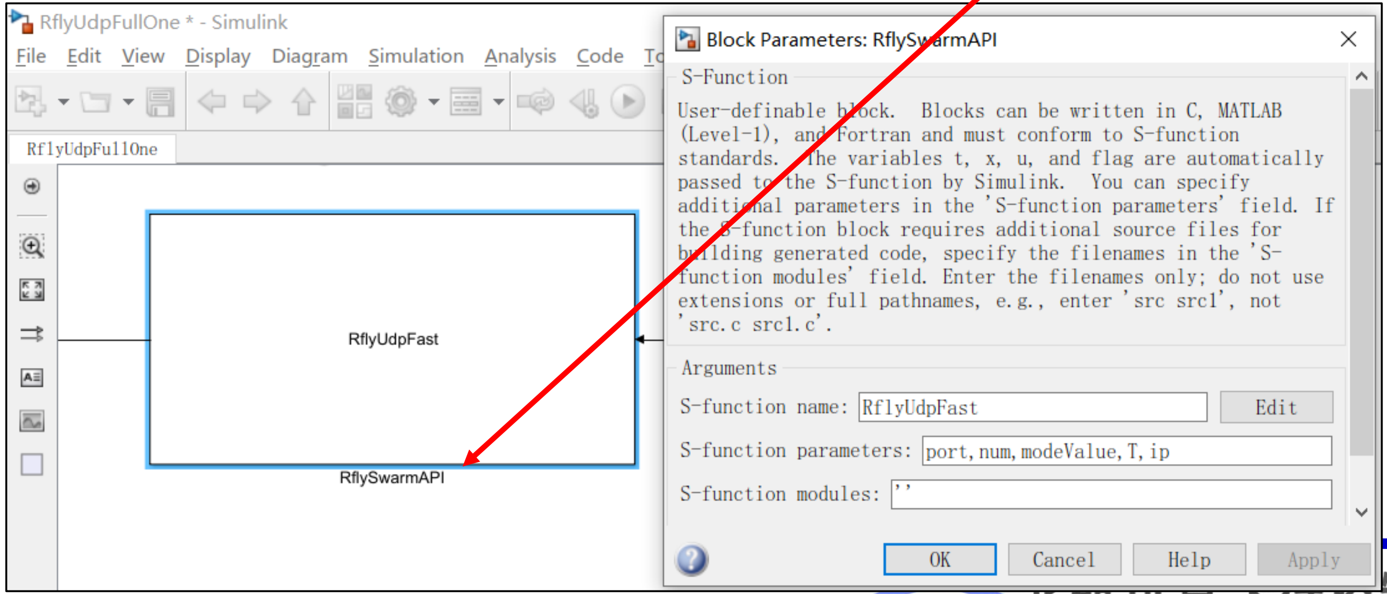
5. summary

Path of demo of source code of this Section:
RflySimAPIs\SimulinkSwarmAPI\SimulinkDemo

2. Interface Introduction

2.1 Simulink swarm communication interface

- The platform provides two types of communication interfaces, Simulink and Python, and their bottom layer is implemented through the **MAVLink** protocol, so the simulated algorithm can be quickly deployed on the experimental platform. Here we introduce the interface of Simulink first:
- Open the "**RflySimAPIs\SimulinkSwarmAPI\SimulinkDemo\RflyUdpFullOne.slx**" file, as shown in the lower left picture, you can see that the "**RflySwarmAPI**" module is the swarm communication module.
- This module has Simulink S functions implemented by C++ mixed compilation, and the source file is "**RflyUdpFast.cpp**".


RflyUdpFullOne * - Simulink
File Edit View Display Diagram Simulation Analysis Code To
RflyUdpFullOne
RflyUdpFast
RflySwarmAPI
Block Parameters: RflySwarmAPI
OK
Cancel
Help
Apply

S-Function

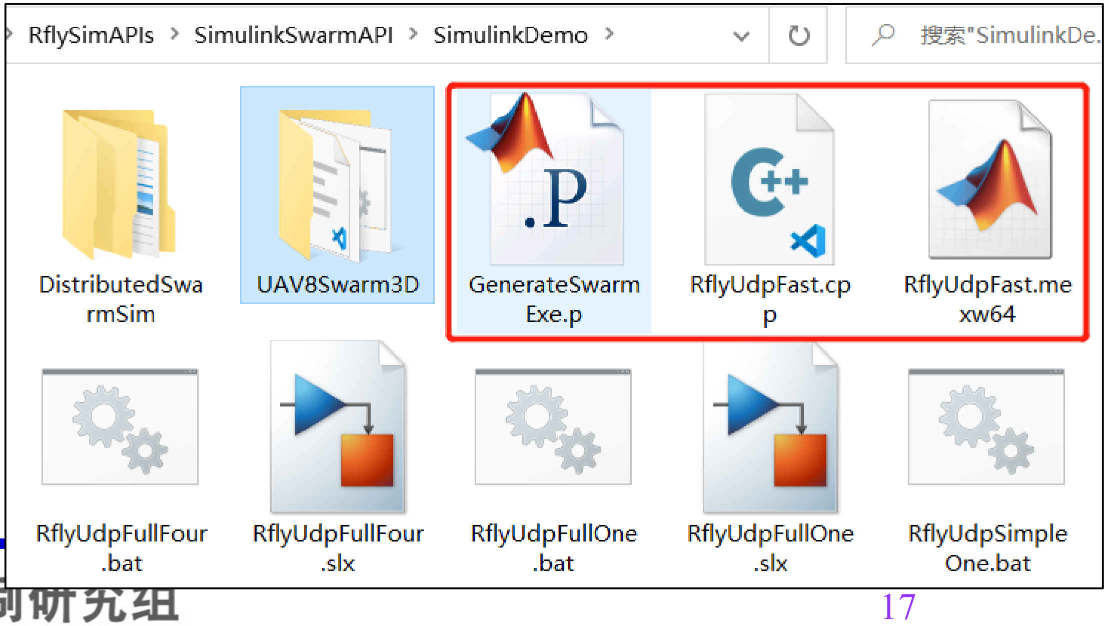
User-definable block. Blocks can be written in C, MATLAB (Level-1), and Fortran and must conform to S-function standards. The variables t, x, u, and flag are automatically passed to the S-function by Simulink. You can specify additional parameters in the 'S-function parameters' field. If the S-function block requires additional source files for building generated code, specify the filenames in the 'S-function modules' field. Enter the filenames only; do not use extensions or full pathnames, e.g., enter 'src src1', not 'src.c src1.c'.

Arguments

S-function name: RflyUdpFast

S-function parameters: port, num, modeValue, T, ip

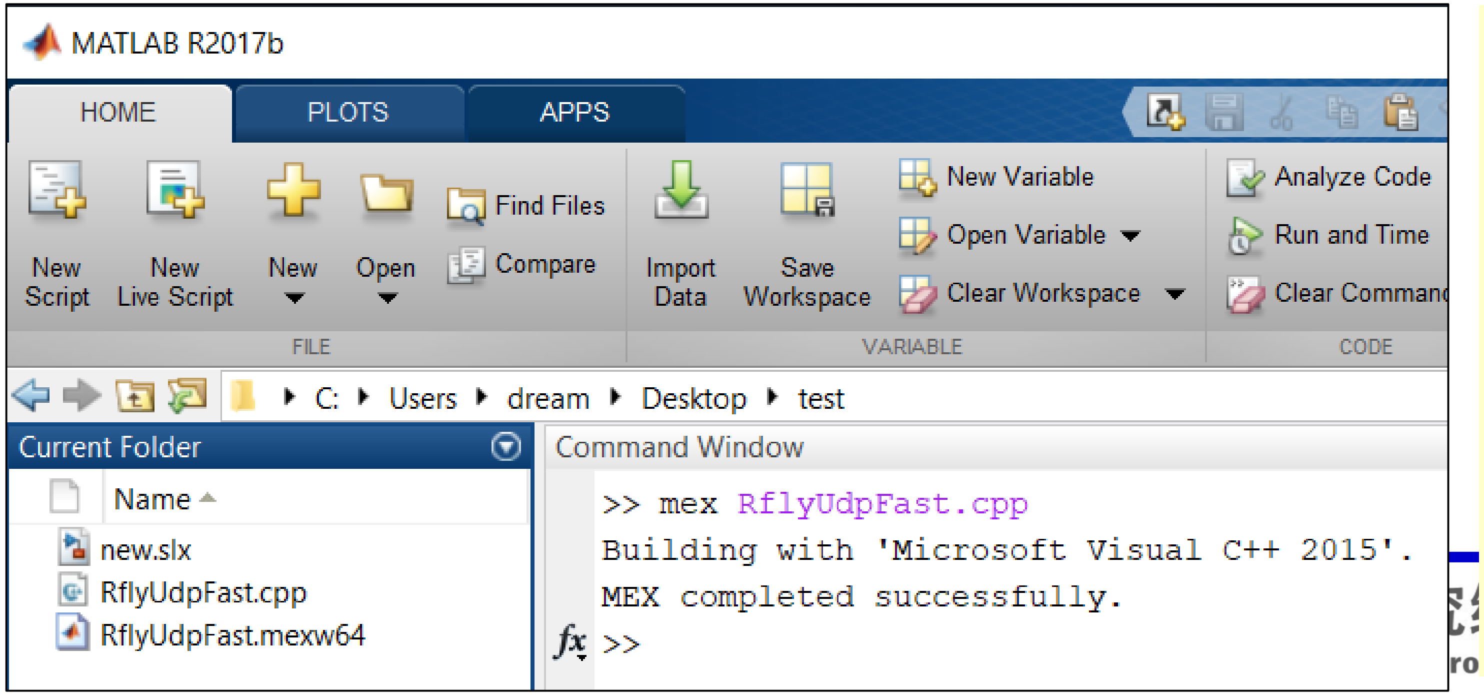
S-function modules: ''



2. Interface Introduction

2.2 Simulink swarm communication interface usage

- The "**RflySwarmAPI**" module needs to be placed in the same folder as "**RflyUdpFast.mexw64**" to be called. Therefore, when creating a new **.slx** project, while copying the "**RflySwarmAPI**" module, you also need to copy the "**RflyUdpFast.mexw64**" file to the directory where the new **.slx** file is located.
- "**RflyUdpFast.mexw64**" is the source file "**RflyUdpFast.cpp**" compiled in MATLAB. If you modify the **.cpp** file, you also need to use the command "**mex RflyUdpFast.cpp**" to recompile



```
>> mex RflyUdpFast.cpp
Building with 'Microsoft Visual C++ 2015'.
MEX completed successfully.
>>
```

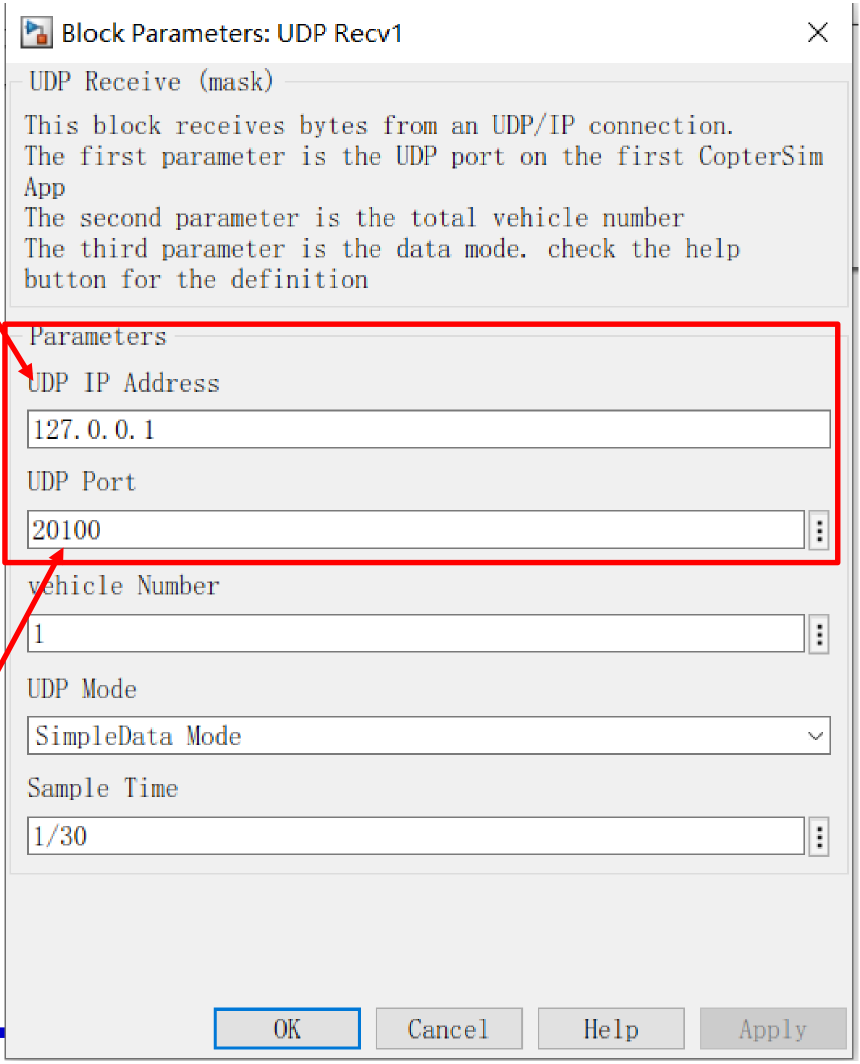
- As shown on the left, MATLAB input "**mex RflyUdpFast.cpp**" to get the "**RflyUdpFast.mexw64**" file (you can modify the **cpp** file to develop ROS, serial port, TCP and other communications by yourself)
- Copy "**RflyUdpFast.mexw64**" to a folder, create a **new.slx** (or other name), copy "**RflySwarmAPI**" in "**RflyUdpFullOne.slx**" into it, and develop the swarm algorithm inside.

2. Interface Introduction

Double-click the "RflySwarmAPI" module, and this dialog will pop up

2.3 swarm communication interface configuration analysis

- The first item "**UDP IP Address**" is the IP address of the target computer. Enter "**127.0.0.1**" to accept and control the Pixhawk autopilot status forwarded by CopterSim on this computer; "**255.255.255.255**" can receive and control the CopterSim running in all LAN computer (other computers needs to check the “**Link**" button on CopterSim); the designated IP such as "**192.168.1.12**" will only send control commands to the host with the IP address.
- Generally speaking, the "**255.255.255.255**" broadcast has been able to meet the demand in a small swarm. When the number of vehicles continues to increase, the designated IP needs to be enabled to reduce network load and improve communication speed and reliability.
- The second item "**UDP Port**" is the initial port number of the first vehicle, and the default initial port is 20100. Each CopterSim sending and receiving messages needs to occupy a port, so if this module needs to simulate a vehicle with a vehicle ID of 10~15, then this item needs to be filled in **20100+10*2=20120**, and the latter item is the number of "**Vehicle number**" vehicle Need to fill in 5.



Block Parameters: UDP Recv1

UDP Receive (mask)

This block receives bytes from an UDP/IP connection.
The first parameter is the UDP port on the first CopterSim App
The second parameter is the total vehicle number
The third parameter is the data mode. check the help button for the definition

Parameters

UDP IP Address: 127.0.0.1

UDP Port: 20100

Vehicle Number: 1

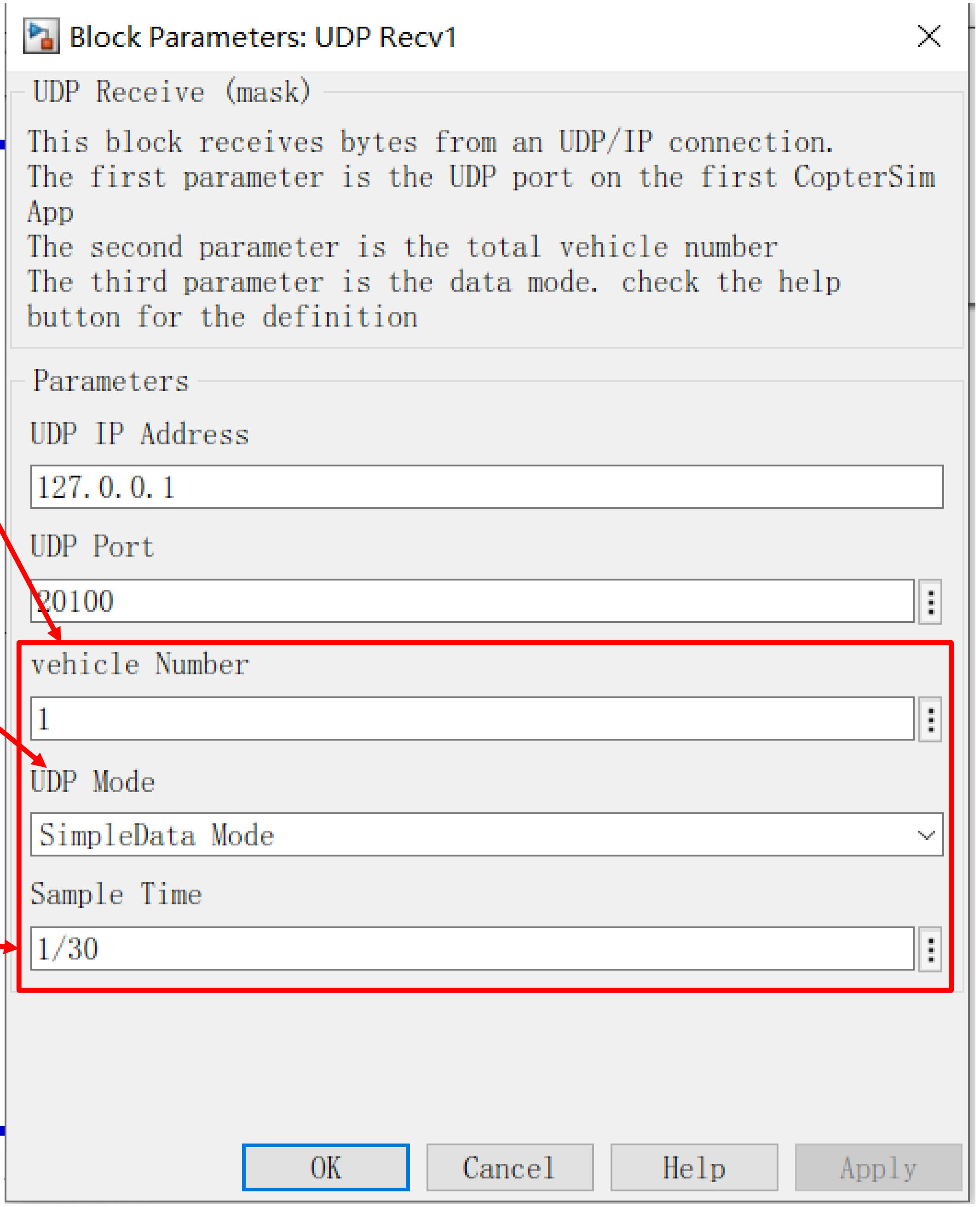
UDP Mode: SimpleData Mode

Sample Time: 1/30

2. Interface Introduction

2.3 Swarm communication interface configuration analysis

- The third item "**Vehicle number**" indicates the number of CopterSim that need to be connected. The number of input and output ports of the module is controlled by this option. If 10 is input, the module will automatically generate 10 pairs of input and output ports.
- The fourth item "**UDP mode**" is the data mode protocol of the input and output interfaces, mainly including "**FullData Mode**" : complete mode (the most complete data, but the amount of data transmitted is larger); "**SimpleData Mode**" : simplified data mode (more vehicle>8, avoiding excessive data network Blocking) and "**UltraSimple Mode**" : ultra-compact mode (the number of vehicle per computer>20), the delay is smaller.
- The fifth item is the "**Sample Time**" sampling time, which should correspond to the Simulink simulation time.



Block Parameters: UDP Recv1

UDP Receive (mask)

This block receives bytes from an UDP/IP connection.
The first parameter is the UDP port on the first CopterSim App
The second parameter is the total vehicle number
The third parameter is the data mode. check the help button for the definition

Parameters

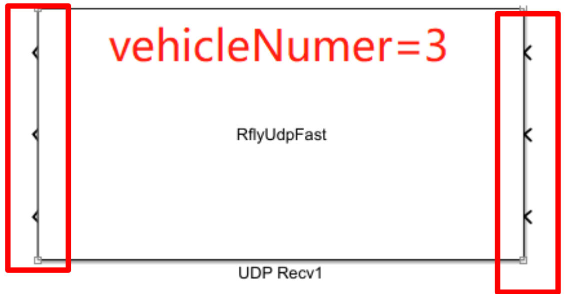
UDP IP Address: 127.0.0.1

UDP Port: 20100

vehicle Number: 1

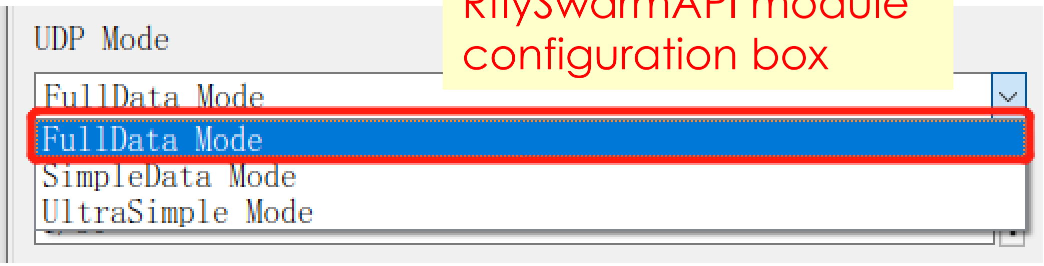
UDP Mode: SimpleData Mode

Sample Time: 1/30



vehicleNumer=3

2. Interface Introduction



UDP Mode

FullData Mode

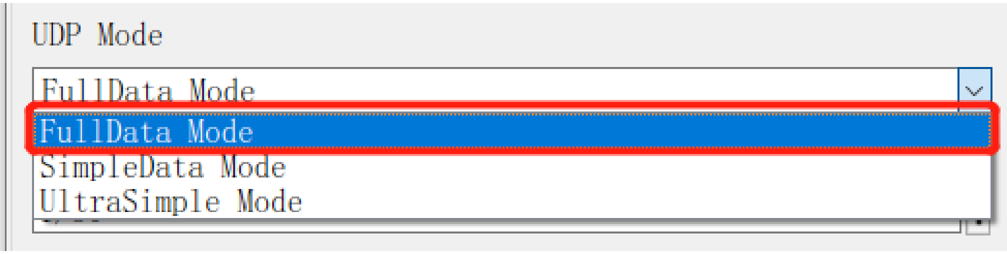
- FullData Mode
- SimpleData Mode
- UltraSimple Mode

RflySwarmAPI module configuration box

2.4 **FullData Mode** of Communication Interface-Inp

- The **input** of the module is a **15-dimensional** double type vector, and the specific definition (to achieve the Offboard message of MAVLink) is as follows
- Dimension **1** : **time_boot_ms**;% current timestamp (fill in 0, not currently used)
- Dimension **2: copterID**; %vehicle ID (just fill in 1, it is not currently used)
- Dimension **3: type_mask**;% input control mode (same as Offboard definition)
- Dimension **4** : **coordinate_frame**;% coordinate system mode (same as Offboard MAVLink message definition)
- Dimension **5 to 15: ctrls[11]**;% respectively correspond to the 3-dimensional desired position “**pos**”, 3-dimensional desired velocity “**vel**”, 3-dimensional desired acceleration “**acc**”, 1-dimensional desired yaw angle “**yaw**”, 1-dimensional desired yaw Angular speed “**yawRate**”. (Same as Offboard definition)
- This module has the same function as "**RflySimAPIs\SimulinkControlAPI\OffboardAPI.slx**" (see **Section 3.3** of **Lesson 3** of the advanced course), and achieves all the functions of Offboard control, and can achieve position, velocity, acceleration tracking, etc.
- The detailed definition of Offboard message can be seen
- https: //mavlink.io/en/messages/common.html#SET_POSITION_TARGET_LOCAL_NED

2. Interface Introduction



UDP Mode

FullData Mode

2.5 **FullData Mode** of communication interface --- output

- The **output** of the module is a **28-dimensional** double vector (all forwarded from the Pixhawk internal filter values), which are defined as follows

① Dimension **1** to **3**: **gpsHome[3]**; The latitude and longitude coordinates of the Home position (will not change after power-on or armed), the latitude and longitude must be divided by 1e7 to get the latitude and longitude in degrees, and the height must be divided by 1e3 to get the unit of m (upwards is positive).

② Dimension **4** to **6**: **AngEular[3]**; Euler angle (rad) of the posture estimated by Pixhawk

③ Dimension **7** to **9**: **localPos[3]**; %Pixhawk estimated relative northeast position vector with gpsHome as the origin, unit m, z-axis is positive when downward

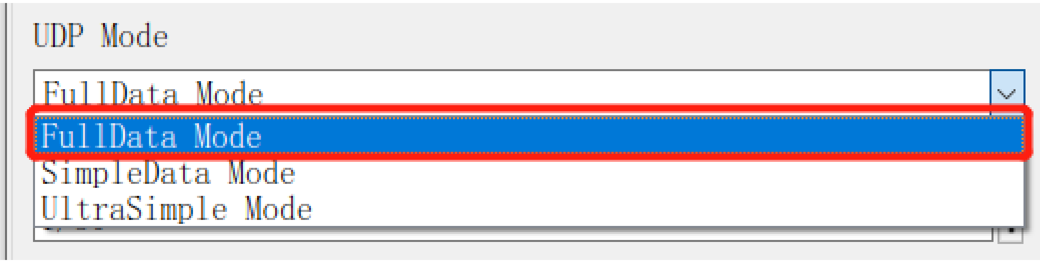
④ Dimension **10** to **12**: **localVel[3]**;% the velocity vector of the northeast, unit m/s

⑤ Dimension **13** to **15**: **GpsPos[3]**;% Real-time GPS position, the unit is the same as **gpsHome**, but it will change in real time

⑥ Dimension **16** to **18**: **GpsVel[3]**; %GPS speed, need to be divided by 100 to get the speed in m/s

⑦ Dimension **19**: **time_boot_ms**;% power-on time

2. Interface Introduction



2.5 **FullData Mode** of communication interface --- output

⑧ Dimension **20: copterID**; % vehicle ID

⑨ Dimension **21**: **relative_alt**; % GPS relative altitude, need to be divided by 1000 to get the altitude in unit: m, upward is positive

⑩ Dimension **22: hdg**; % GPS heading angle, need to be divided by 1000 to get an angle in the range of 0~360 degrees

⑪ Dimension **23: satellites_visible**; % number of visible satellites

⑫ Dimension **24: fix_type**; % positioning accuracy

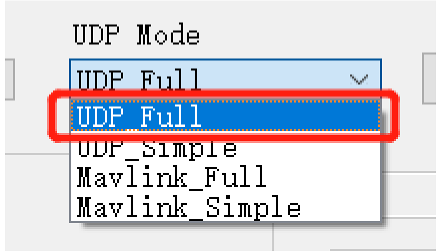
⑬ Dimension **25: resrveInit**; % reserved bits of int type

⑭ Dimension **26: pos_horiz_accuracy**; % horizontal positioning accuracy, unit m

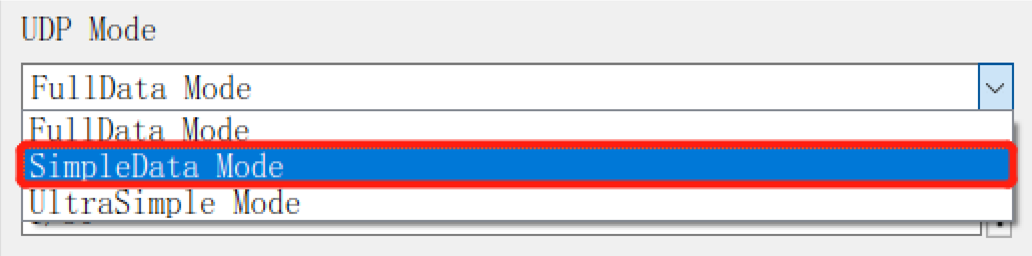
⑮ Dimension **27**: **pos_vert_accuracy**; % vertical positioning accuracy, unit m

⑯ Dimension **28: resrveFloat**; % float type reserved bit, not enabled

- To use this mode, select "**UDP_Full**" in the "**UDP Mode**" option on CopterSim



2. Interface Introduction



UDP Mode

FullData Mode

FullData Mode

SimpleData Mode

UltraSimple Mode

2.6 **SimpleData Mode** of Communication Interface --- Input

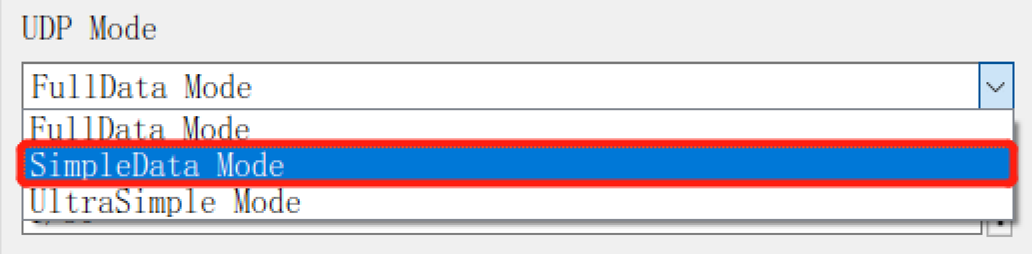
- The **input** is a **5-dimensional** double type vector, the specific definition (to achieve the Offboard message of MAVLink) is as follows

① Dimension **1**: **ctrlMode**;% The first digit is the flag bit, **0**: represents the earth speed control mode "**Earth Vel**"; 1: the body speed control mode "**Body Vel**"; 2: the earth position control mode "**Earth Pos**"; 3: the body position control mode "**Body Pos**"

② Dimension **2** to **5**: If **ctrlMode =0**, these four dimensions correspond to the **x, y, z** position + ***yaw*** angle in the **earth frame** (the position when armed); if **ctrlMode =1**, these four dimensions correspond to The **x, y, z** position + ***yaw*** angle in the **body frame** (based on the body attitude and position when armed); if **ctrlMode = 2**, then these four dimensions correspond to the **earth frame** (take the armed position as the origin) ***Vx, vy, vz*** velocity+ ***yawRate*** angular rate; if **ctrlMode = 3**, these four dimensions correspond to the ***vx, vy, vz*** velocity + ***yawRate*** angular rate in the body coordinate system (to arm the body attitude and position) signal.

- Generally, **ctrlMode=0** is more, and the global speed is directly given for trajectory control. If you switch between position and speed, you need to modify the value of ctrlMode in real time and adjust the definition of the 2 to 5 dimension signals.

2. Interface Introduction



2.7 **SimpleData Mode** of Communication Interface --- Output

- The **output** is a **12-dimensional** double vector, and the sequence is defined as follows

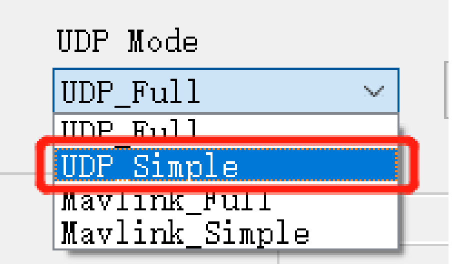
① Dimension **1** to **3**: **gpsHome[3]**; % Home point (will not change after power-on) latitude and longitude coordinates, longitude and latitude need to be divided by 1e7 to get the longitude and latitude in degrees, and the height needs to be divided by 1e3 to get m as the unit High (upwards is positive)

② Dimension **4** to **6**: **AngEular[3]**;% Pixhawk's estimated Euler angle, in radians

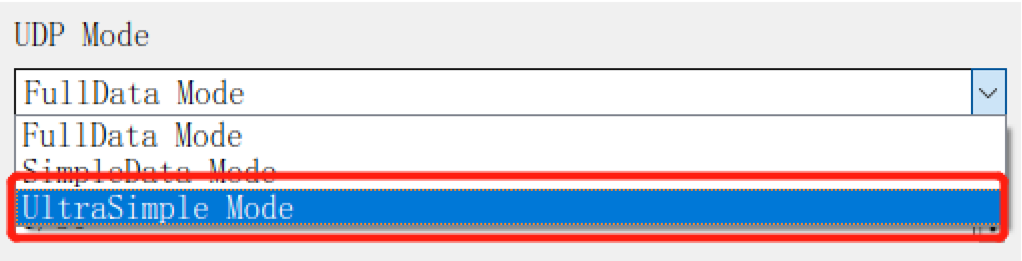
③ Dimension **7** to **9**: **localPos[3]**;% Pixhawk estimated relative northeast position vector with gpsHome as the origin, unit m, z-axis is positive downward

④ Dimension **10** to **12**: **localVel[3]**;% The velocity vector of the northeast region, in m/s

- **Note**: **gpsHome** is the latitude and longitude coordinates. If you want to convert to the map coordinates in CopterSim, you need to subtract the initial point latitude and longitude (**116.2593683**° , **40.1540302**° , **0**m), and then convert it to the unit m
- **Note**: **SimpleData Mode** is not only simplified input and output, but also simplified UDP transmission structure, so there is no other information in **FullData Mode**. Besides, the "**UDP Mode**" option on CopterSim should select "**UDP_Simple**"



2. Interface Introduction



2.8 **UltraSimple Mode** of communication interface

- The **input** is a **5-dimensional** double type vector, the specific definition is exactly the same as the “**SimpleData Mode**” in **Section 2.5**
- The **output** is a **12-dimensional** double vector, and the sequence is defined as follows

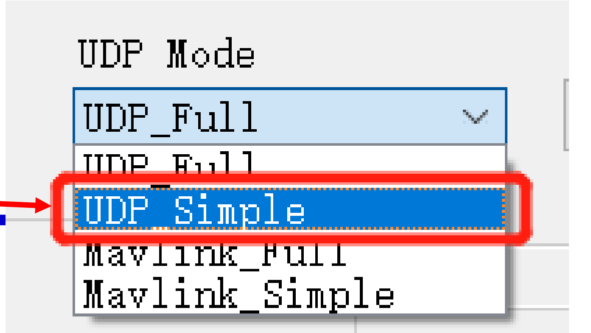
① Dimension **1** to **3**: **GlobalPos [3]**;% CopterSim global position, unit m

② 4th~6th dimension: **AngEular[3]**;% Pixhawk's estimated Euler angle, in unit rad

③ Dimension **7** to **9**: **localPos[3]**;% Pixhawk estimated relative northeast position vector with **gpsHome** as the origin, unit m, z-axis is positive when downward

④ Dimension **10** to **12**: **localVel[3]**;% The velocity vector of the northeast region, in m/s

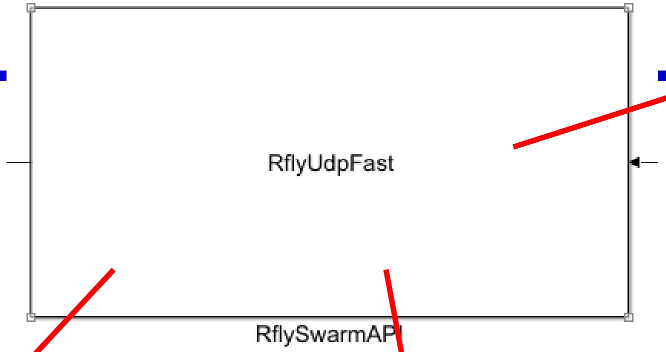
- **Note**: The only difference between “**UltraSimple Mode**” and “**SimpleData Mode**” is that the first to third dimensions of the output are changed from the original **gpsHome** to **GlobalPos**, where **GlobalPos** is the global **x, y, z** position in the CopperSim calculated after **gpsHome** has undergone the coordinate conversion of latitude and longitude. (Northeast coordinate system) coordinates
- **Note**: The intranet transmission data in **SimpleData** and **UltraSimple** modes are the same, and the "**UDP Mode**" option on CopterSim should be selected as "**UDP_Simple**"

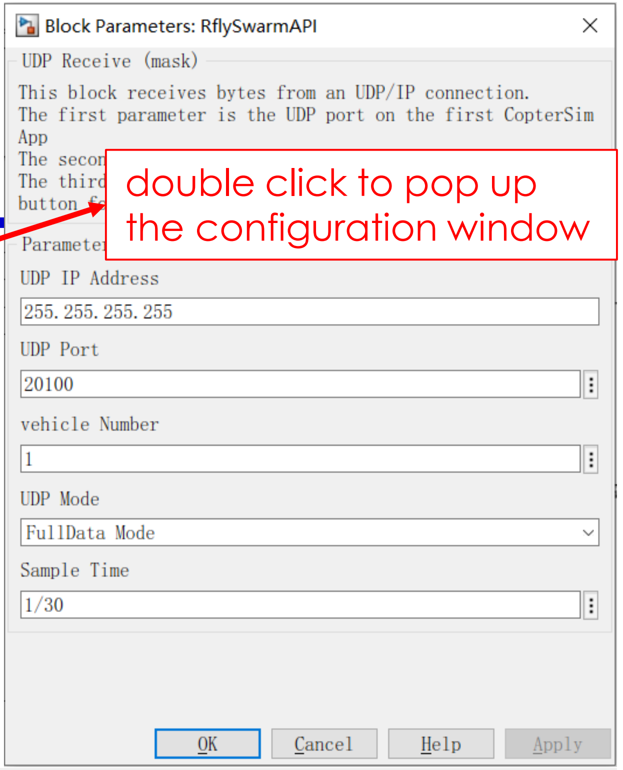


2. Interface Introducti

2.9 Analysis of RflySwarmAPI Module Implementation Principle

- The **S-function** called by RflySwarmAPI module is set by the module parameter page in the lower left figure
- The configuration menu in the right picture of this module is set by the Mask editor in the bottom right picture.





Block Parameters: RflySwarmAPI

UDP Receive (mask)

This block receives bytes from an UDP/IP connection.
The first parameter is the UDP port on the first CopterSim App
The secon
The third
button f

Paramete

UDP IP Address: 255.255.255.255

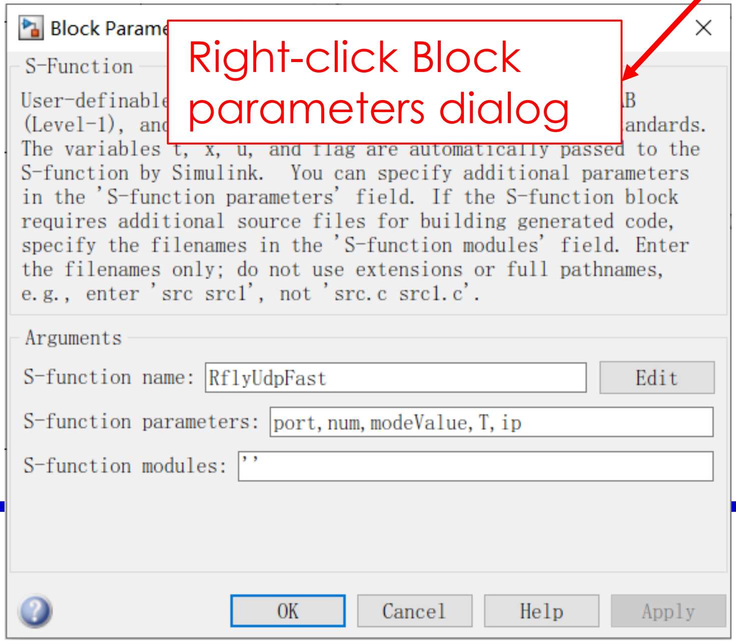
UDP Port: 20100

vehicle Number: 1

UDP Mode: FullData Mode

Sample Time: 1/30

double click to pop up the configuration window



Block Parame

S-Function

User-definable ... (Level-1), and ... B ... andards. The variables t, x, u, and flag are automatically passed to the S-function by Simulink. You can specify additional parameters in the 'S-function parameters' field. If the S-function block requires additional source files for building generated code, specify the filenames in the 'S-function modules' field. Enter the filenames only; do not use extensions or full pathnames, e.g., enter 'src src1', not 'src.c src1.c'.

Arguments

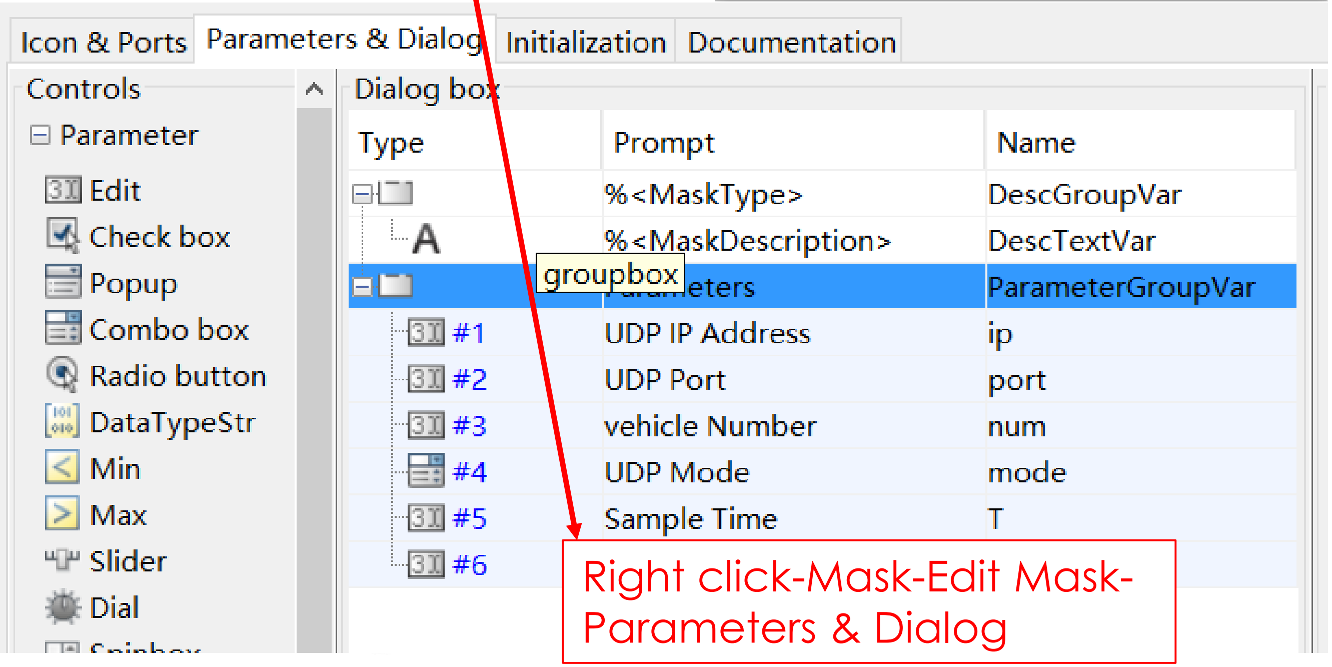
S-function name: RflyUdpFast

S-function parameters: port,num,modeValue,T,ip

S-function modules: ''

Right-click Block parameters dialog

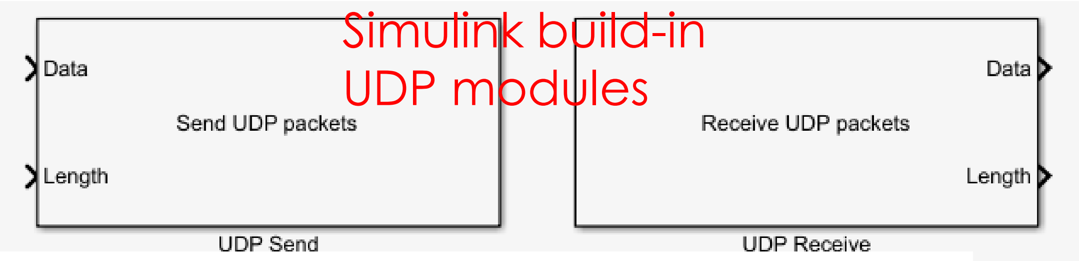
Mask Editor : RflySwarmAPI



Type	Prompt	Name
	%<MaskType>	DescGroupVar
A	%<MaskDescription>	DescTextVar
	Parameters	ParameterGroupVar
#1	UDP IP Address	ip
#2	UDP Port	port
#3	vehicle Number	num
#4	UDP Mode	mode
#5	Sample Time	T
#6		

Right click-Mask-Edit Mask-Parameters & Dialog

2. Interface Introduction



Simulink build-in UDP modules

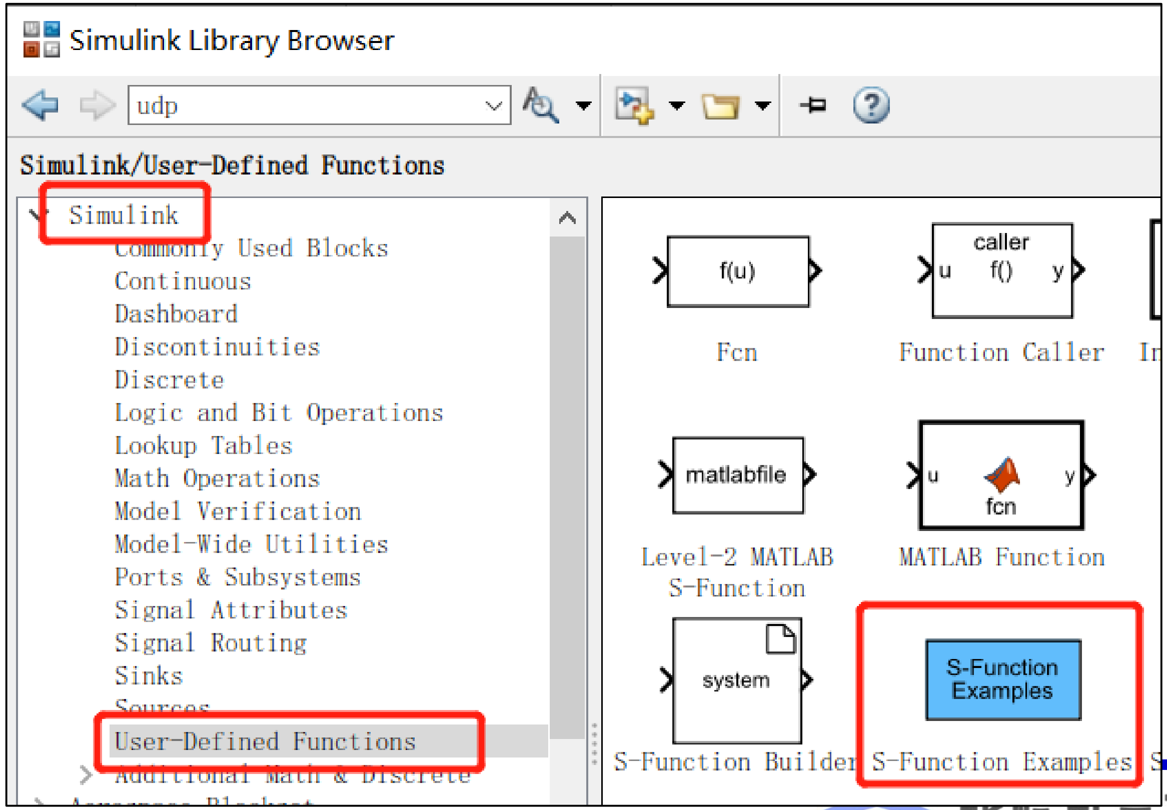
2.10 Advantages of **C++ S-function** communication module

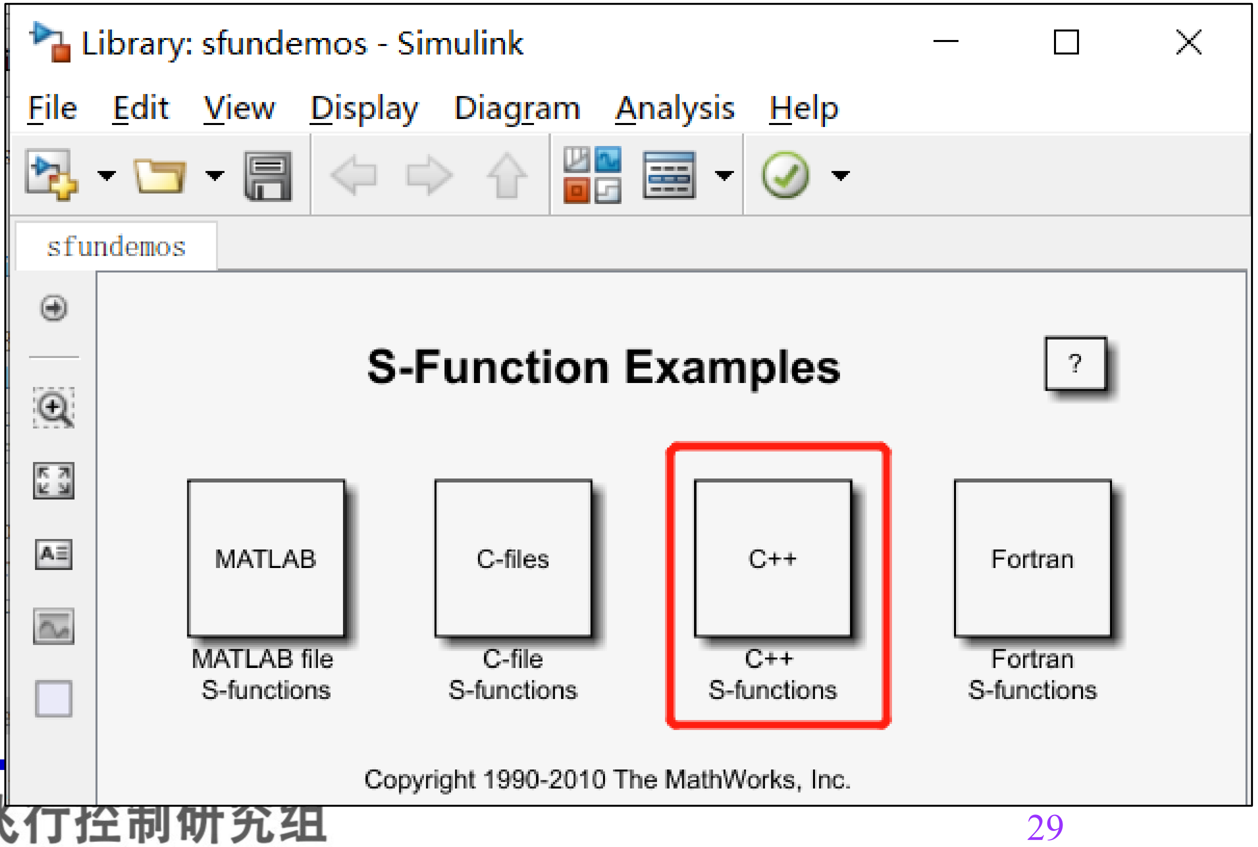
- The main content of **RflySwarmAPI S-function** includes two aspects: **UDP network communication** programming + **Simulink S-function** programming
- Advantages of the module **RflySwarmAPI** related to the **Simulink's build-in** UDP module
- **High efficiency**: **C/C++** language is directly implemented in **RflySwarmAPI**, which is more efficient than MATLAB m language
- **Small operation**: Simulink's build-in UDP module input and output are all high-dimensional (several hundred dimensions) unit8 vectors. When the number of vehicle increases, the overall project dimension expands sharply; while the s-function method directly outputs double-type position, speed and other data. The dimensions is low (several dimensions) with small amount of calculation
- **Low latency**: In order to prevent data loss, Simulink's built-in UDP module will save all the received data in the buffer and call them in turn, so that when the frequency of external program transmission is greater than the operating frequency of Simulink. The old data will be called to cause large delay. S-function method can avoid this problem
- **More reliable**: Simulink's built-in UDP module reads one data per simulation step. If the sending frequency of the external program is less than the operating frequency of Simulink, Simulink does not read the data and will output 0, resulting in an operation error; while writing the S-function yourself. This problem can be avoided.
- **Strong scalability**: S-function can be easily extended to other communication protocols such as serial port, TCP, shared memory, MAVLink, etc.

2. Interface Introduction

2.11 S-function learning example

- Click the **Simulink library browser** - **Simulink** - **User Defined Functions** – **S-Function Examples** – **C++ S-functions**, from which you can view the writing methods of S-functions, and easily achieve Simulink control other systems

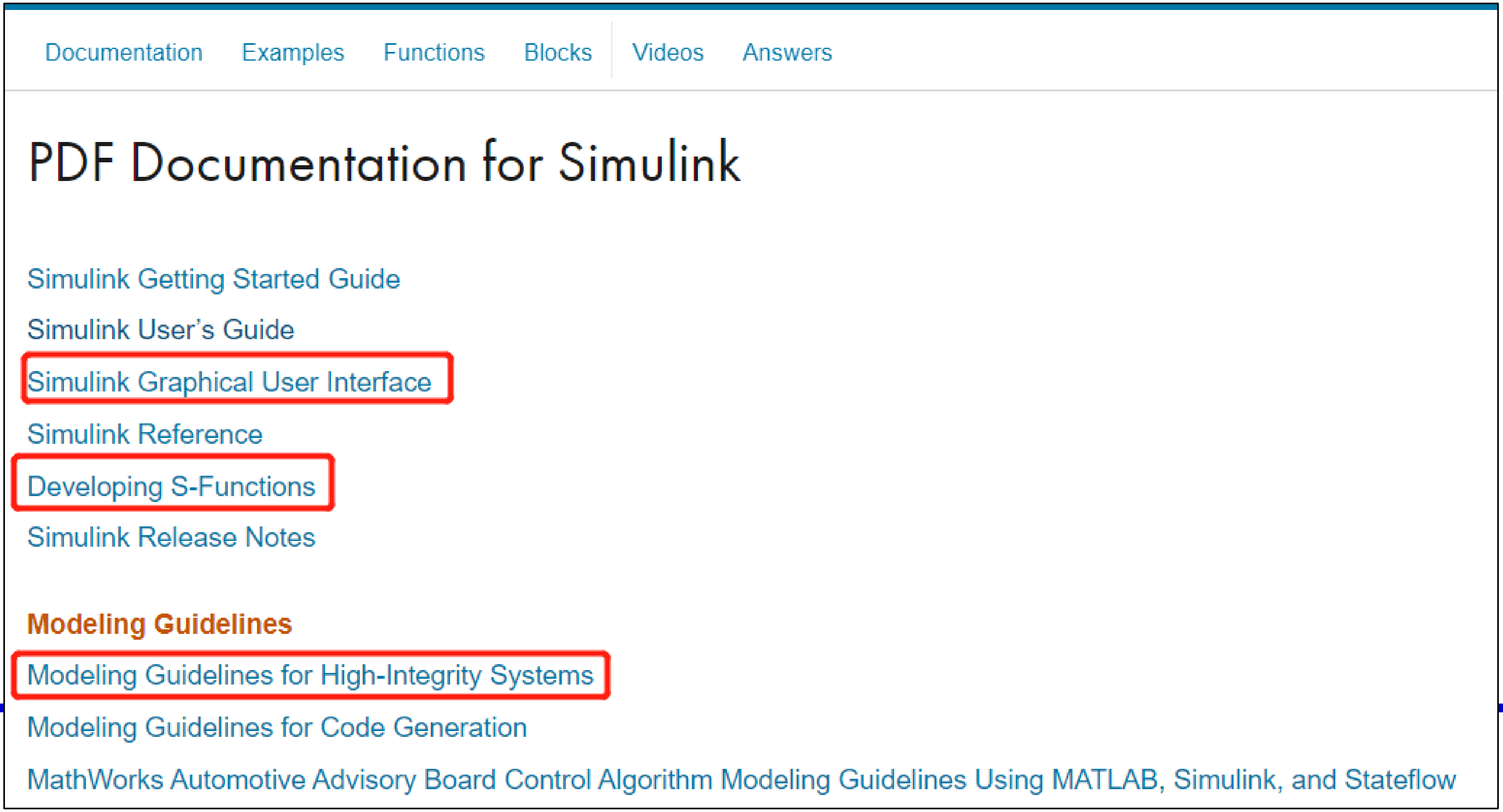




2. Interface Introduction

2.12 Official PDF document of S function

https: //www.mathworks.com/help/pdf_doc/simulink/index.html



Documentation Examples Functions Blocks Videos Answers

PDF Documentation for Simulink

Simulink Getting Started Guide

Simulink User's Guide

Simulink Graphical User Interface

Simulink Reference

Developing S-Functions

Simulink Release Notes

Modeling Guidelines

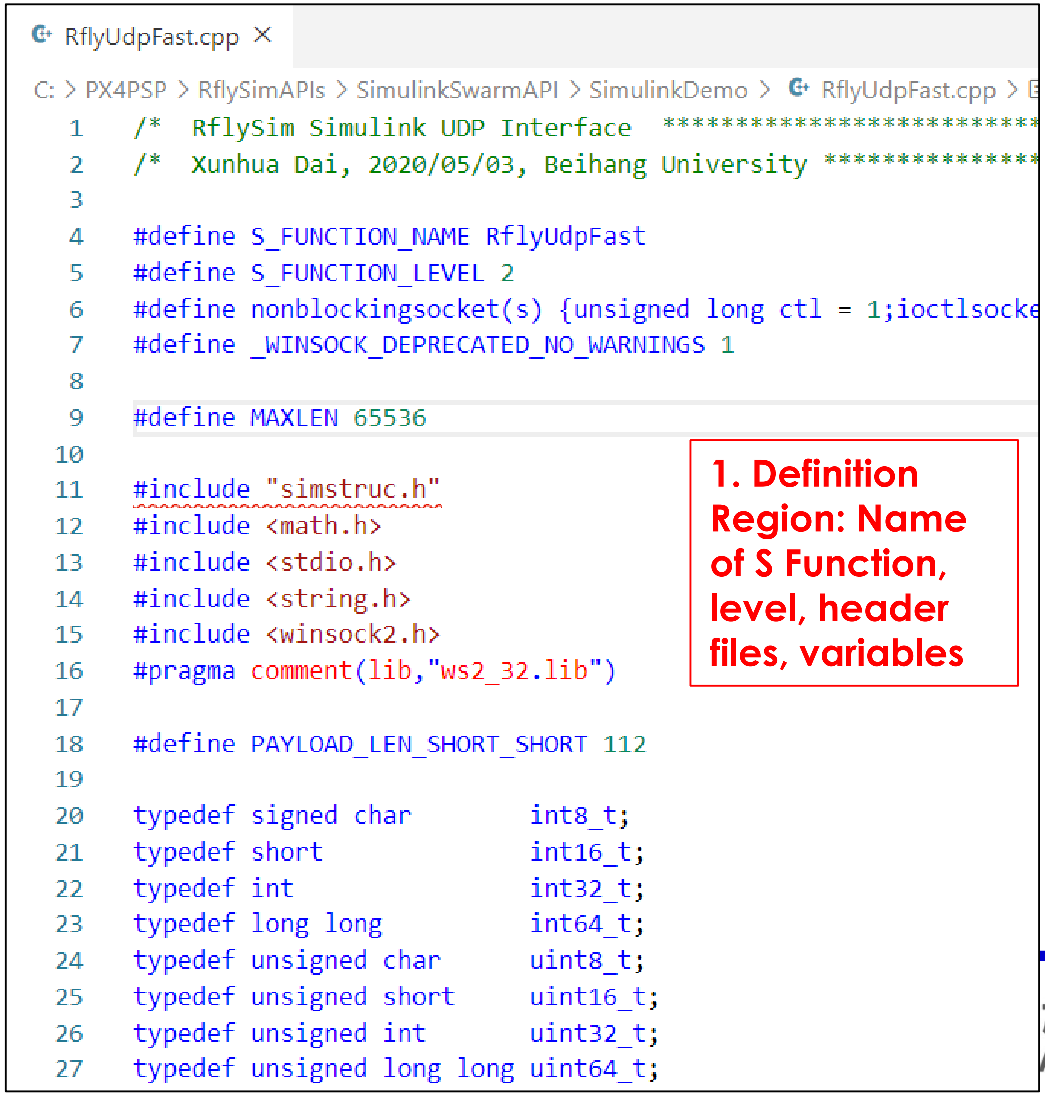
Modeling Guidelines for High-Integrity Systems

Modeling Guidelines for Code Generation

MathWorks Automotive Advisory Board Control Algorithm Modeling Guidelines Using MATLAB, Simulink, and Stateflow

2. Interface Introduction

2.13 RflyUdpFast.cpp code analysis

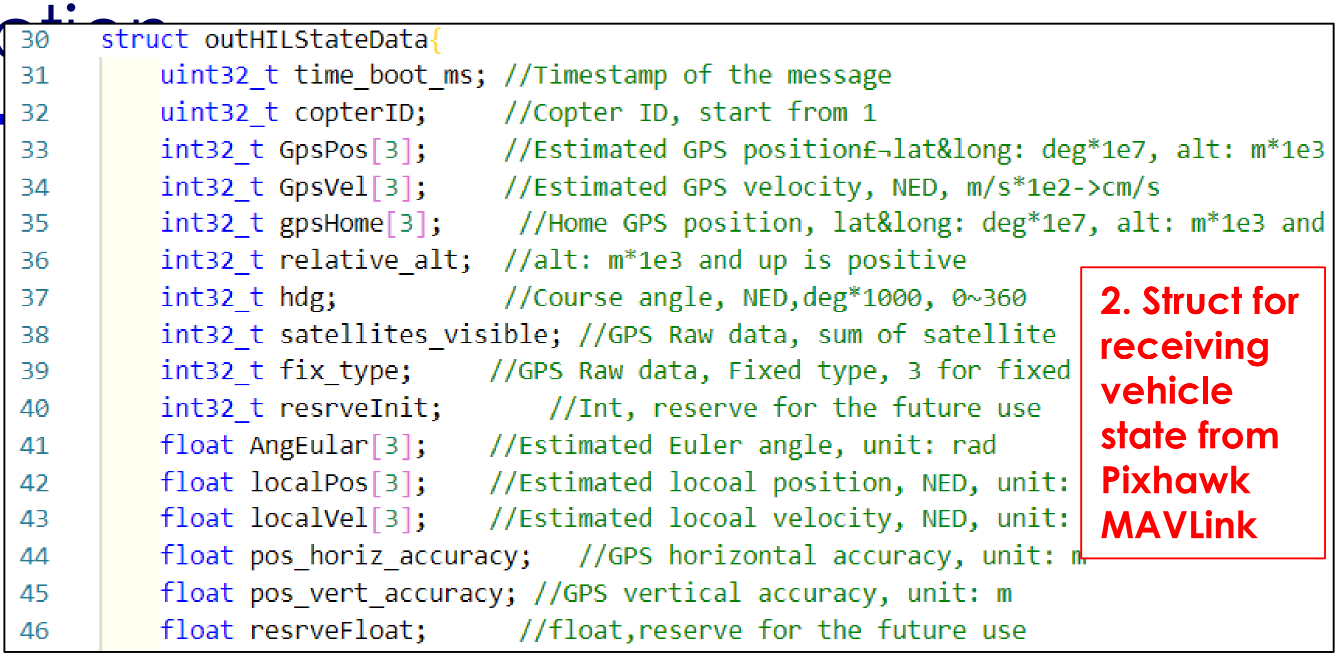


RflyUdpFast.cpp ×

C: > PX4PSP > RflySimAPIs > SimulinkSwarmAPI > SimulinkDemo > RflyUdpFast.cpp >

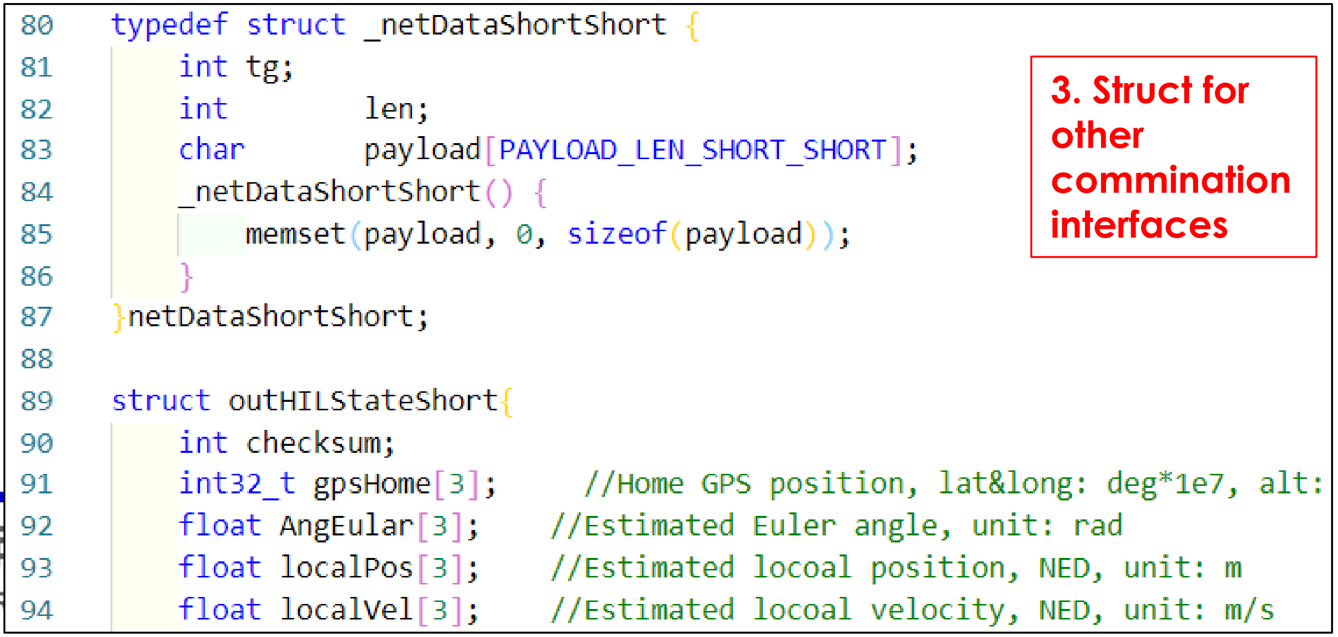
```
/*  RflySim Simulink UDP Interface  **************************
/*  Xunhua Dai, 2020/05/03, Beihang University ***************

#define S_FUNCTION_NAME RflyUdpFast
#define S_FUNCTION_LEVEL 2
#define nonblockingsocket(s) {unsigned long ctl = 1;ioctlsocke
#define _WINSOCK_DEPRECATED_NO_WARNINGS 1

#define MAXLEN 65536

#include "simstruc.h"
#include <math.h>
#include <stdio.h>
#include <string.h>
#include <winsock2.h>
#pragma comment(lib,"ws2_32.lib")

#define PAYLOAD_LEN_SHORT_SHORT 112

typedef signed char         int8_t;
typedef short               int16_t;
typedef int                 int32_t;
typedef long long           int64_t;
typedef unsigned char       uint8_t;
typedef unsigned short      uint16_t;
typedef unsigned int        uint32_t;
typedef unsigned long long uint64_t;
```

1. Definition Region: Name of S Function, level, header files, variables



```
struct outHILStateData{
    uint32_t time_boot_ms; //Timestamp of the message
    uint32_t copterID;     //Copter ID, start from 1
    int32_t GpsPos[3];     //Estimated GPS position£¬lat&long: deg*1e7, alt: m*1e3
    int32_t GpsVel[3];     //Estimated GPS velocity, NED, m/s*1e2->cm/s
    int32_t gpsHome[3];     //Home GPS position, lat&long: deg*1e7, alt: m*1e3 and
    int32_t relative_alt;  //alt: m*1e3 and up is positive
    int32_t hdg;           //Course angle, NED,deg*1000, 0~360
    int32_t satellites_visible; //GPS Raw data, sum of satellite
    int32_t fix_type;     //GPS Raw data, Fixed type, 3 for fixed
    int32_t resrveInit;       //Int, reserve for the future use
    float AngEular[3];    //Estimated Euler angle, unit: rad
    float localPos[3];    //Estimated locoal position, NED, unit:
    float localVel[3];    //Estimated locoal velocity, NED, unit:
    float pos_horiz_accuracy;   //GPS horizontal accuracy, unit: m
    float pos_vert_accuracy; //GPS vertical accuracy, unit: m
    float resrveFloat;      //float,reserve for the future use
```

2. Struct for receiving vehicle state from Pixhawk MAVLink

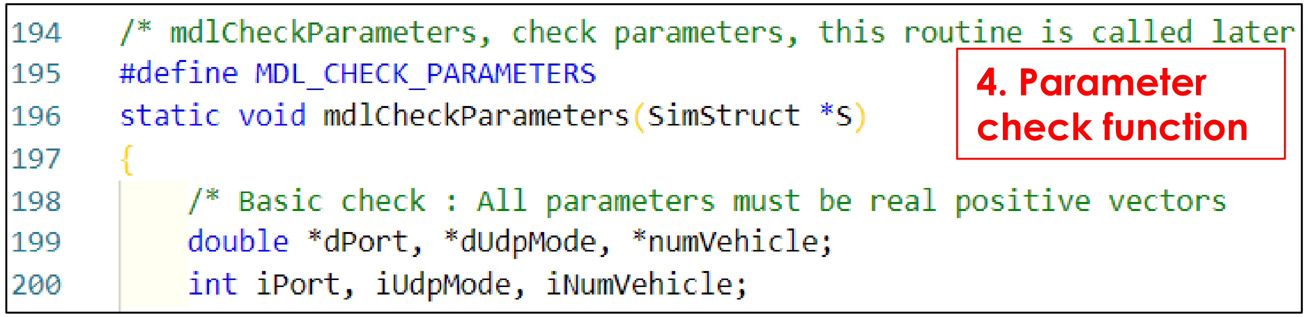


```
typedef struct _netDataShortShort {
    int tg;
    int         len;
    char        payload[PAYLOAD_LEN_SHORT_SHORT];
    _netDataShortShort() {
        memset(payload, 0, sizeof(payload));
    }
}netDataShortShort;

struct outHILStateShort{
    int checksum;
    int32_t gpsHome[3];     //Home GPS position, lat&long: deg*1e7, alt:
    float AngEular[3];    //Estimated Euler angle, unit: rad
    float localPos[3];    //Estimated locoal position, NED, unit: m
    float localVel[3];    //Estimated locoal velocity, NED, unit: m/s
```

3. Struct for other commination interfaces

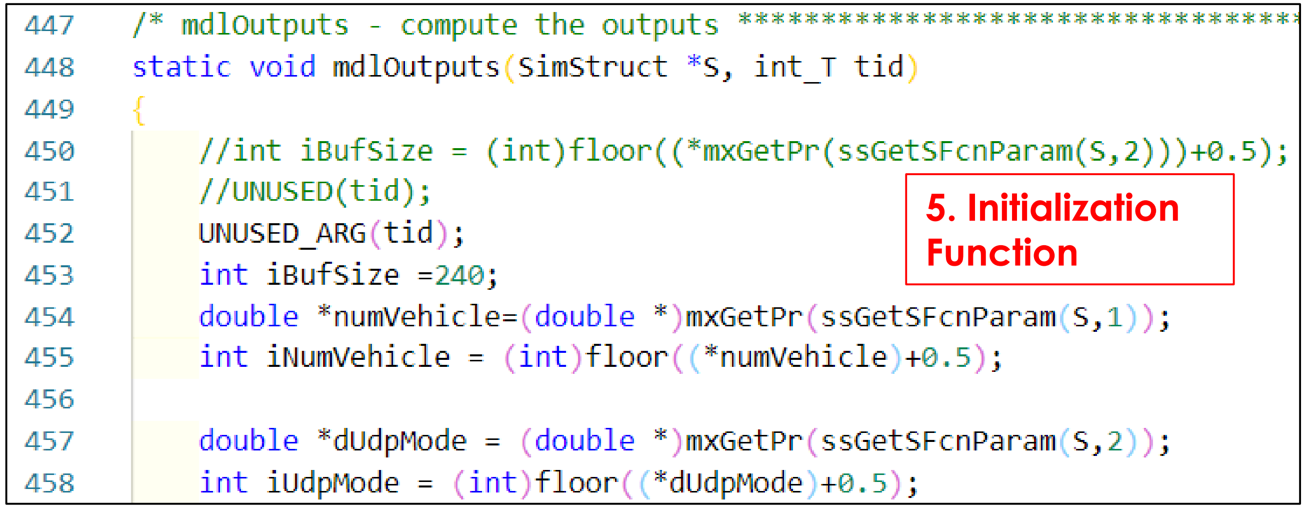
2. Interface Introduction

2.13 RflyUdpFast.cpp code analysis



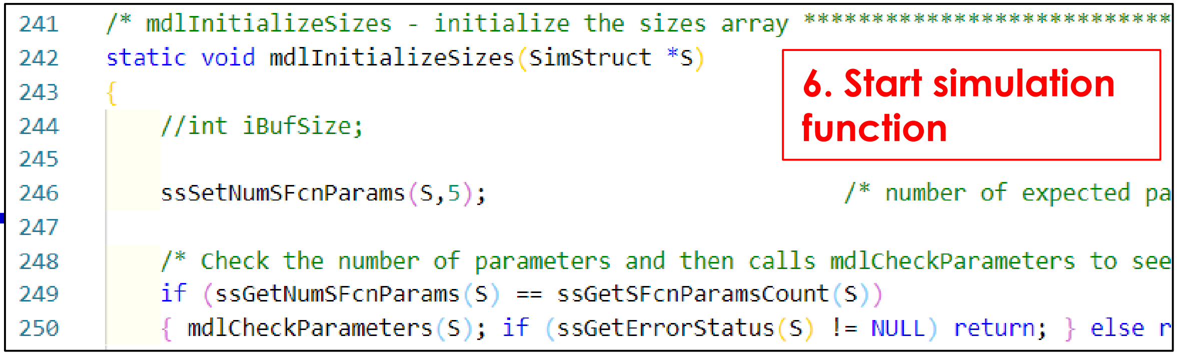
4. Parameter check function

```
/* mdlCheckParameters, check parameters, this routine is called later
#define MDL_CHECK_PARAMETERS
static void mdlCheckParameters(SimStruct *S)
{
    /* Basic check : All parameters must be real positive vectors
    double *dPort, *dUdpMode, *numVehicle;
    int iPort, iUdpMode, iNumVehicle;
```



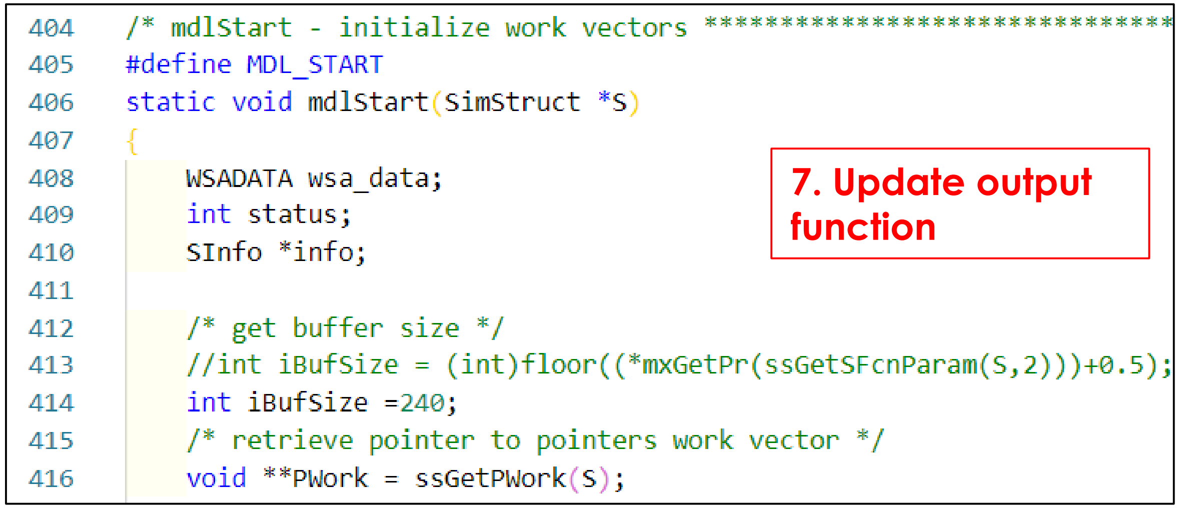
5. Initialization Function

```
/* mdlOutputs - compute the outputs ********************************
static void mdlOutputs(SimStruct *S, int_T tid)
{
    //int iBufSize = (int)floor((*mxGetPr(ssGetSFcnParam(S,2)))+0.5);
    //UNUSED(tid);
    UNUSED_ARG(tid);
    int iBufSize =240;
    double *numVehicle=(double *)mxGetPr(ssGetSFcnParam(S,1));
    int iNumVehicle = (int)floor((*numVehicle)+0.5);

    double *dUdpMode = (double *)mxGetPr(ssGetSFcnParam(S,2));
    int iUdpMode = (int)floor((*dUdpMode)+0.5);
```



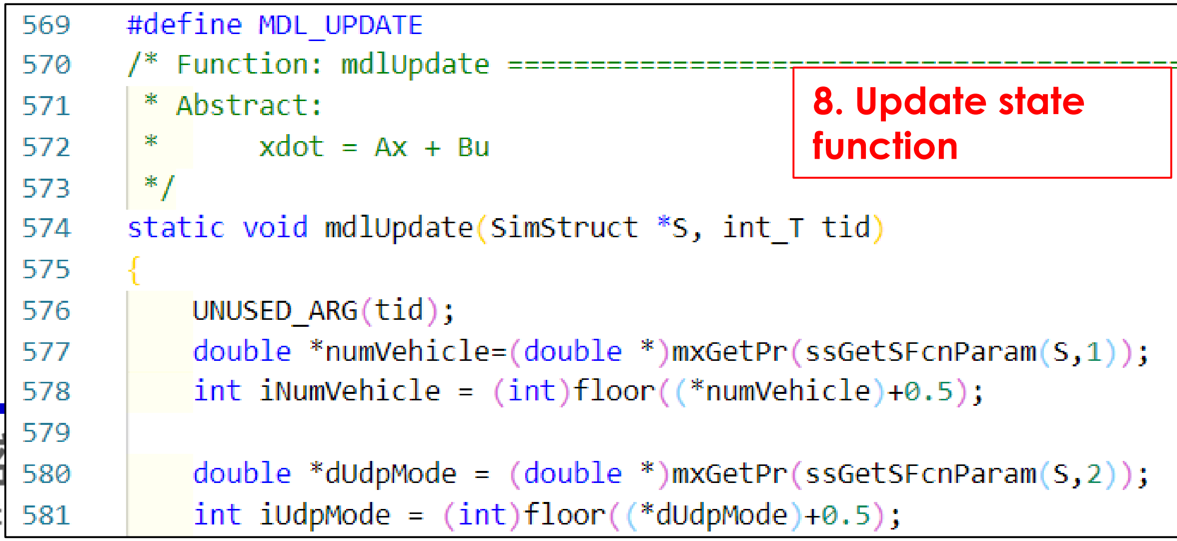
6. Start simulation function

```
/* mdlInitializeSizes - initialize the sizes array ***************************
static void mdlInitializeSizes(SimStruct *S)
{
    //int iBufSize;

    ssSetNumSFcnParams(S,5);                          /* number of expected pa

    /* Check the number of parameters and then calls mdlCheckParameters to see
    if (ssGetNumSFcnParams(S) == ssGetSFcnParamsCount(S))
    { mdlCheckParameters(S); if (ssGetErrorStatus(S) != NULL) return; } else r
```



7. Update output function

```
/* mdlStart - initialize work vectors *****************************
#define MDL_START
static void mdlStart(SimStruct *S)
{
    WSADATA wsa_data;
    int status;
    SInfo *info;

    /* get buffer size */
    //int iBufSize = (int)floor((*mxGetPr(ssGetSFcnParam(S,2)))+0.5);
    int iBufSize =240;
    /* retrieve pointer to pointers work vector */
    void **PWork = ssGetPWork(S);
```



8. Update state function

```
#define MDL_UPDATE
/* Function: mdlUpdate ===========================================
 * Abstract:
 *      xdot = Ax + Bu
 */
static void mdlUpdate(SimStruct *S, int_T tid)
{
    UNUSED_ARG(tid);
    double *numVehicle=(double *)mxGetPr(ssGetSFcnParam(S,1));
    int iNumVehicle = (int)floor((*numVehicle)+0.5);

    double *dUdpMode = (double *)mxGetPr(ssGetSFcnParam(S,2));
    int iUdpMode = (int)floor((*dUdpMode)+0.5);
```

Content

1. Setup Instructions

2. Interface Introduction

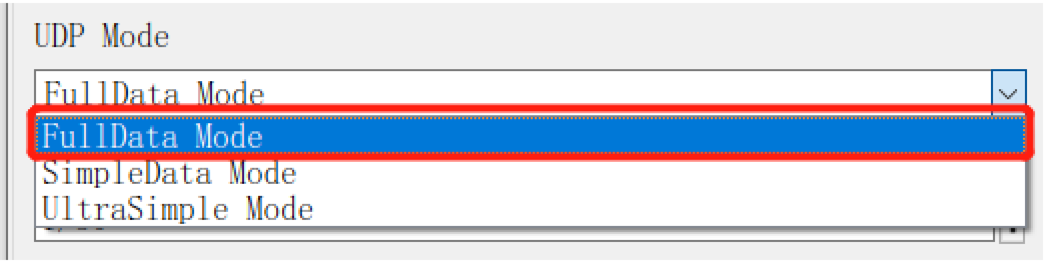
3. Basic examples

4. Advanced examples

5. summary

Path of demo source code of this Section:
RflySimAPIs\SimulinkSwarmAPI\SimulinkDemo

3. Basic examples



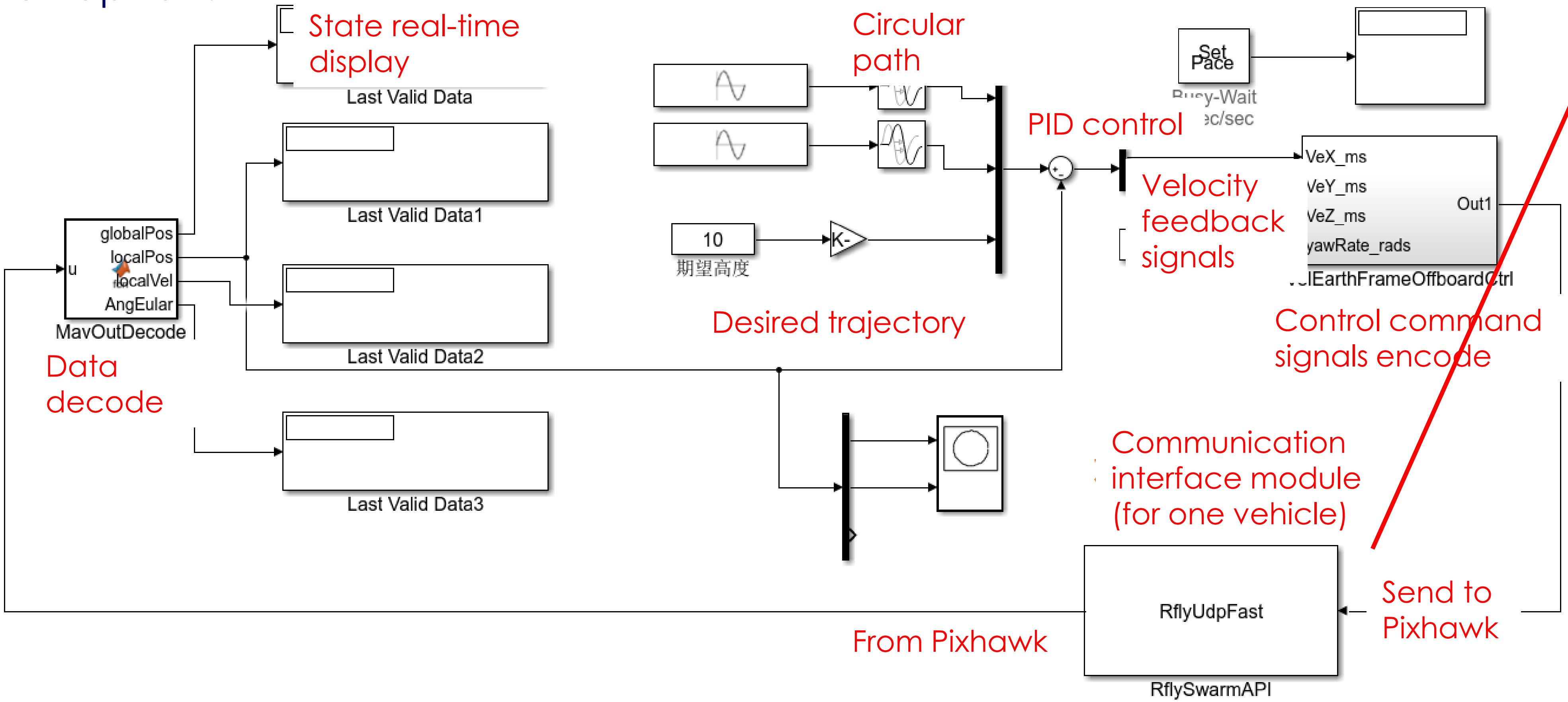
UDP Mode

FullData Mode

FullData Mode
SimpleData Mode
UltraSimple Mode

3.1 Example of **FullData Mode** of communication interface

Open "**RflySimAPIs\SimulinkSwarmAPI\SimulinkDemo\RflyUdpFullOne.slx**" to see the example when the **RflySwarmAPI** communication interface module uses the "**FullData Mode**" option.



State real-time display

Circular path

PID control

Velocity feedback signals

Desired trajectory

Control command signals encode

Data decode

Communication interface module (for one vehicle)

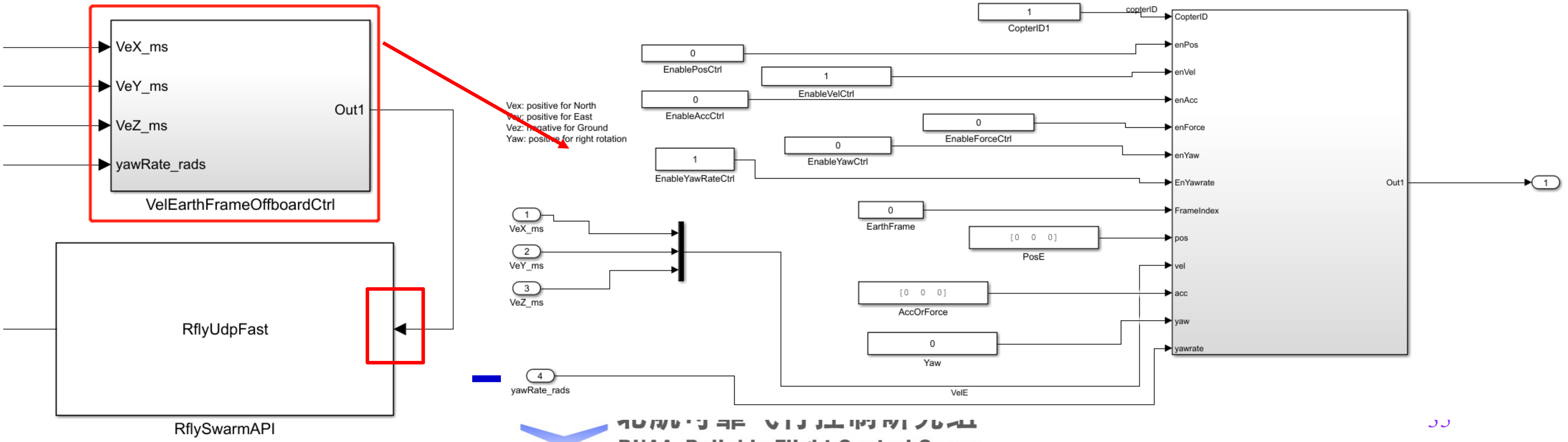
Send to Pixhawk

From Pixhawk

3. Basic examples

3.2 Example of **FullData Mode** of Communication Interface --- Input Mapping Analysis

- The input mapping module converts the 4-dimensional speed (speed and yaw rate control in the earth coordinate system) into a 15-dimensional signal (defined in **Section 2.4**) to the **FullData** Mode mode of the "**RflySwarmAPI**" module.
- “**FullData Mode**” can achieve all Offboard control functions (need to modify some of the configurations shown in the figure below right)

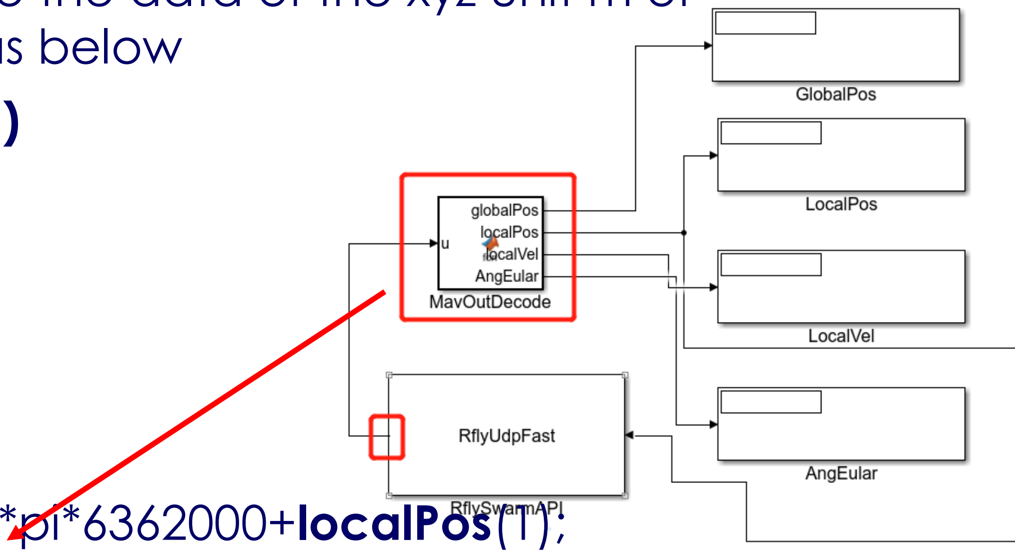


3. Basic examples

3.3 Communication interface **FullData Mode** example --- output extraction analysis

- Extract the **28-dimensional** output of the **RflySwarmAPI** module (see definition in **Section 2.5**) to extract the values of interest: **GpsHome**, **LocalPos**, **AngEular**, etc.
- Convert the latitude and longitude data of **GpsHome** to the data of the xyz unit m of CopterSim's global coordinate **GlobalPos**. code show as below

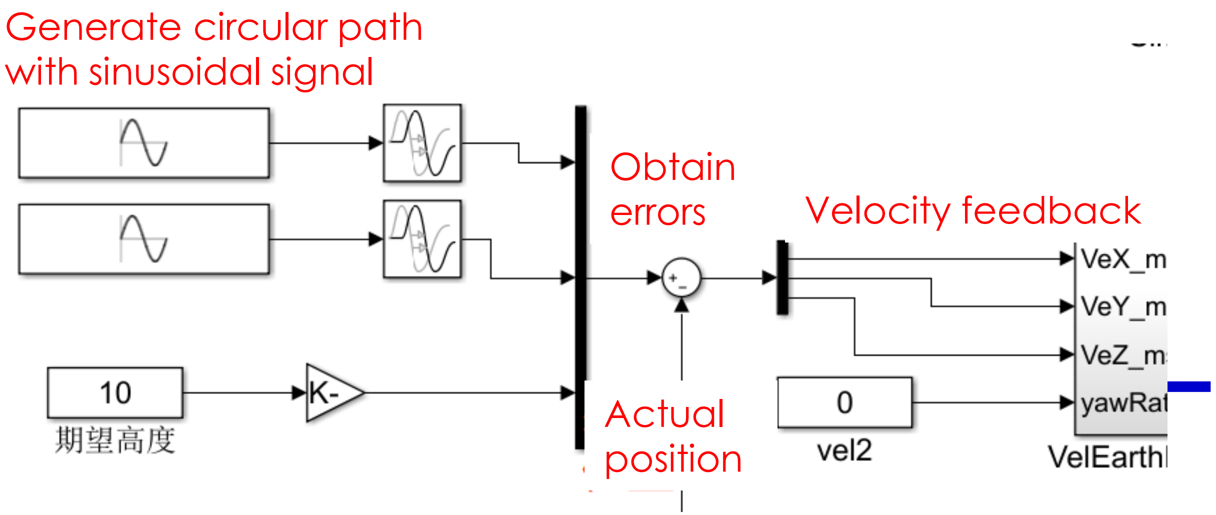
```
function [globalPos,localPos,localVel,AngEular] = fcn(u)
GpsHomePos = u(1: 3);
AngEular = u(4: 6);
localPos = u(7: 9);
localVel = u(10: 12);
globalPos = [0,0,0];
if ~(abs(GpsHomePos(1))<1&&abs(GpsHomePos(2))<1)
    globalPos(1)=(GpsHomePos(1)*1e-7-40.1540302)/180*pi*6362000+localPos(1);
    globalPos(2)=(GpsHomePos(2)*1e-7-116.2593683)/180*pi*4.8823e6+localPos(2);
    globalPos(3)=-GpsHomePos(3)*1e-3+localPos(3);
end
```



3. Basic examples

3.4 Example of **FullData Mode** of Communication Interface --- Control Principle

- **Trajectory generation**: As shown in the figure below, an example trajectory is generated by combining sinusoidal and cosinoidal signals, and the height is always 10m, so it is a fixed-height flying circle trajectory (the trajectory here can also be replaced with any other trajectory, such as a Figure of Eight).
- The expected trajectory minus the real-time position (**LocalPos**, relative to the position of the arm/takeoff point), and the position error is obtained. Then, multiplied the error by **1** to form a proportional (**P**) controller, which feeds the position error back to Pixhawk (as long as there is a position error, the speed is not 0, so the position error tends to zero).



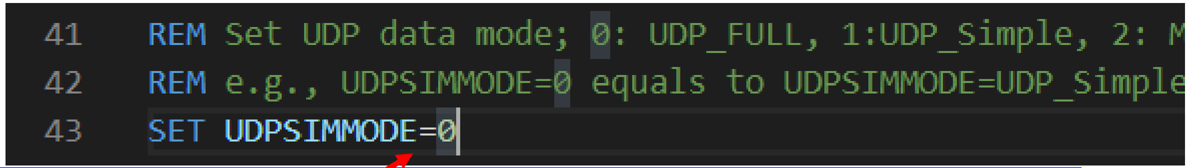
Generate circular path with sinusoidal signal

Obtain errors

Velocity feedback

Actual position

3. Basic examples



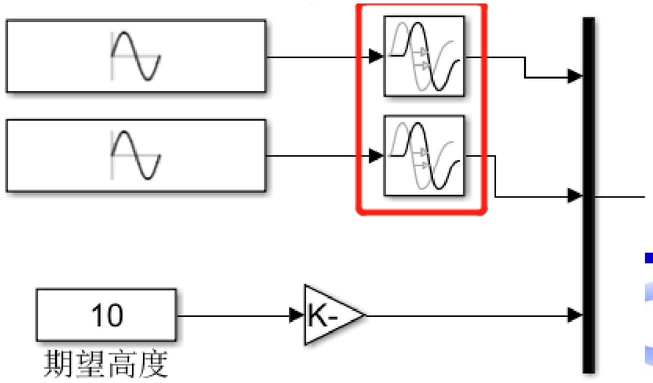
```
REM Set UDP data mode; 0: UDP_FULL, 1:UDP_Simple, 2: M
REM e.g., UDPSIMMODE=0 equals to UDPSIMMODE=UDP_Simple
SET UDPSIMMODE=0
```

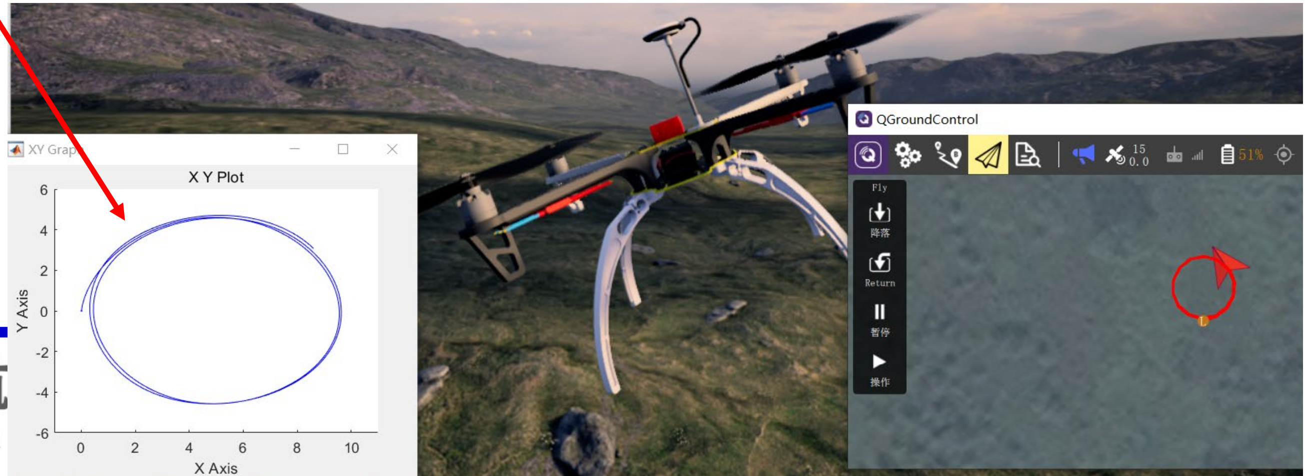
Batch script

3.5 Example of **FullData Mode** of Communication Interface --- Experimental Phenomenon

- Double-click to run “**RflySimAPIs\SimulinkSwarmAPI\SimulinkDemo\RflyUdpFullOne.bat**” (or double-click the script “**SITLRunUdpFull.bat**” in the upper directory, enter the number **1** and press key “**Enter**”) to open the one SITL simulation system (or run the **HITLRunUdpFull.bat** in the upper directory to run a HIL simulation system).
- **Note**: The "**UDPSIMMODE**" option in this script needs to be "**UDP_Full**", which corresponds to the **FullData Mode** in "**RflySwarmAPI**".
- Open the “**RflyUdpFullOne.slx**” demo with MATLAB, click the “**Run**” button in Simulink, you can see that the single vehicle takes off to a certain height, and after a period of time (realized by the delay module before the sin/cos inputs), it starts to fly in a circle.

Delay 15 seconds

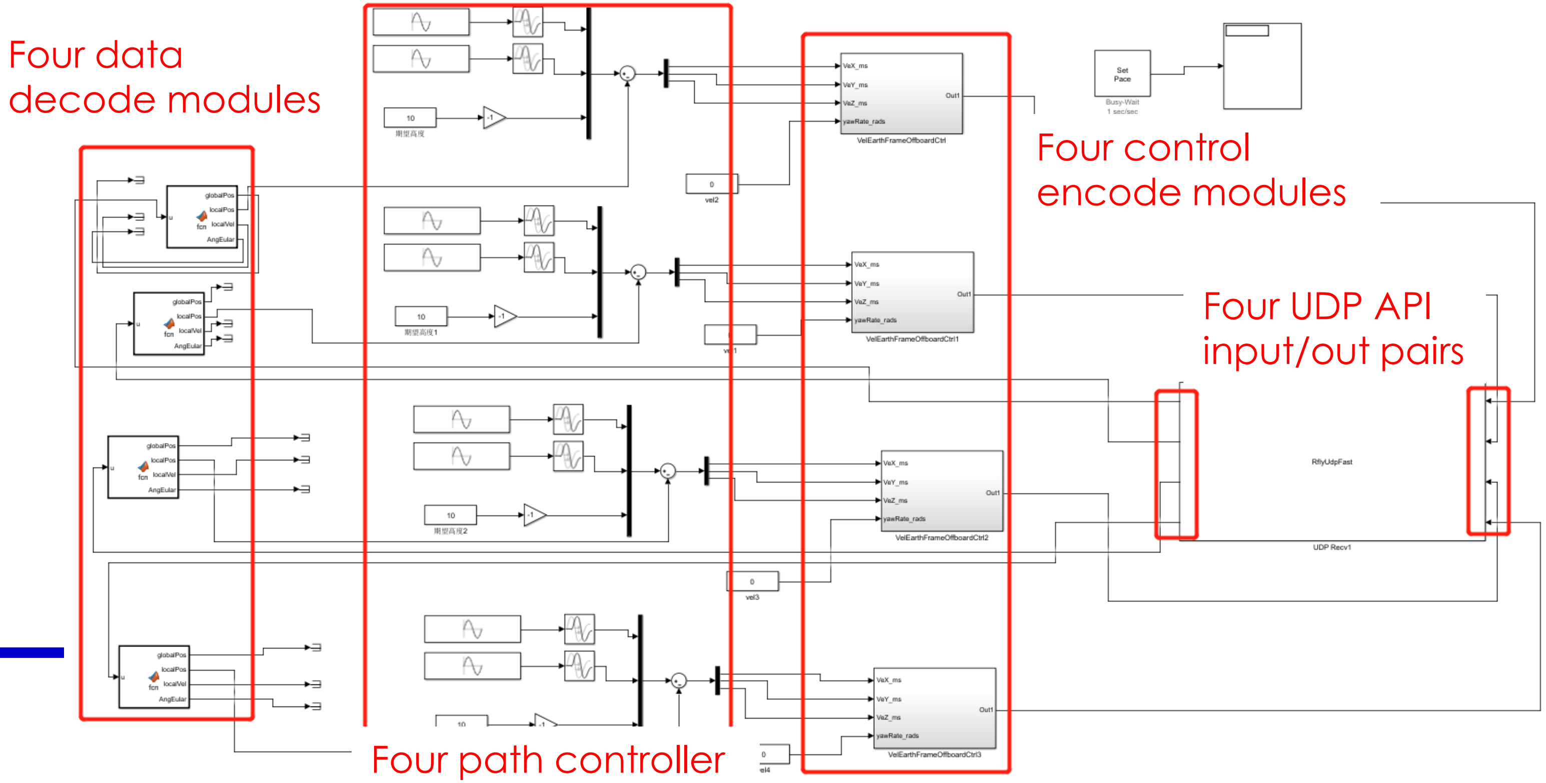




3. Basic examples

3.6 Example of **FullData Mode** of communication interface --- four vehicles

- Open "**RflySimAPIs\SimulinkSwarmAPI\SimulinkDemo\RflyUdpFullFour.slx**"
- Set the vehicle number in the **RflySwarmAPI** module to **4** to get four pairs of input and output, then copy and paste the data encode/decode modules for each input/output port, and then copy four copies of the control algorithm.



Four data decode modules

Four control encode modules

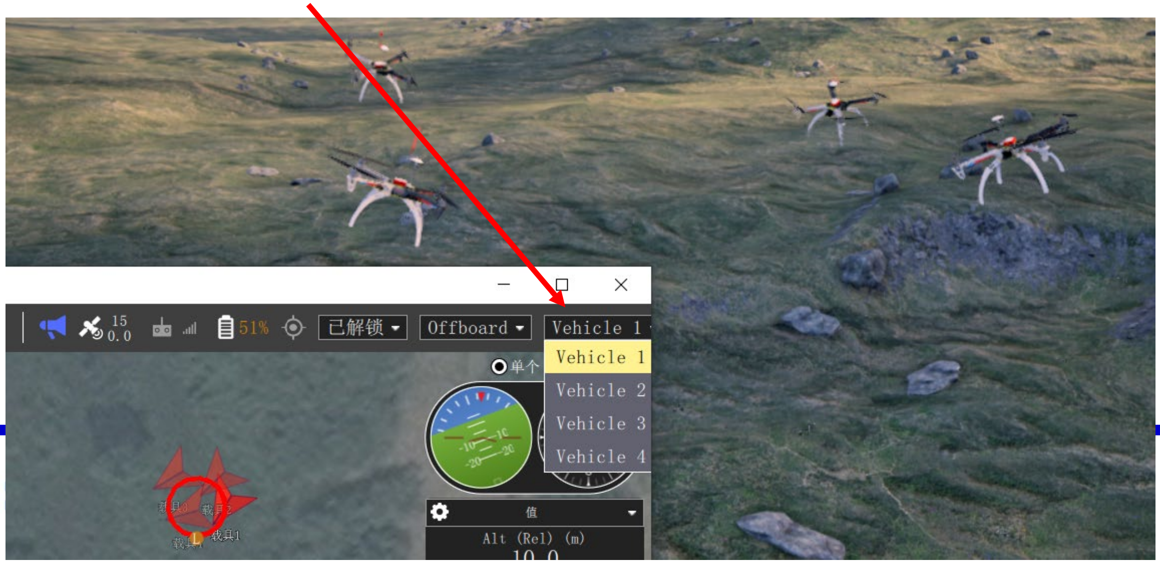
Four UDP API input/out pairs

Four path controller

3. Basic examples

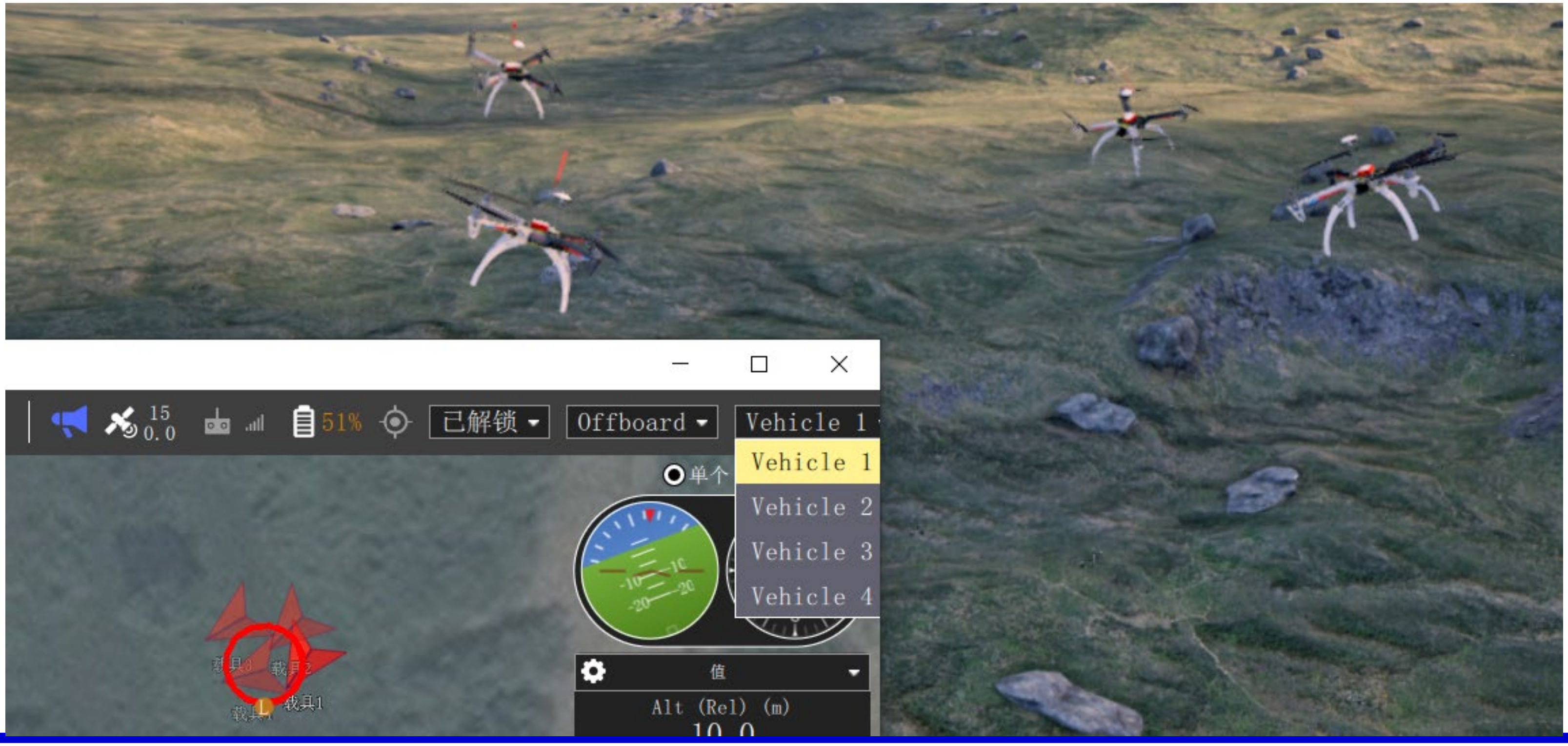
3.7 Example of **FullData Mode** of communication interface --- four vehicles experimental results

- Double-click to run “**RflySimAPIs\SimulinkSwarmAPI\SimulinkDemo\RflyUdpFullFour.bat**” (or double-click the script “**SITLRunUdpFull.bat**” in the upper directory, enter the number **4** and press “**Enter**”) to open the multi-vehicle’s SITL swarm simulation system (or run **HITLRunUdpFull.bat** to open HIL swarm simulation).
- Open the "**RflyUdpFullFour.slx**" demo with MATLAB, click “**Run**” button in Simulink, you can see four drones takeoff to a certain height and start to draw circles (the center of each vehicle is different, switch “**Vehicle ***” on QGC to view the trajectory of each vehicle) .



3. Basic examples

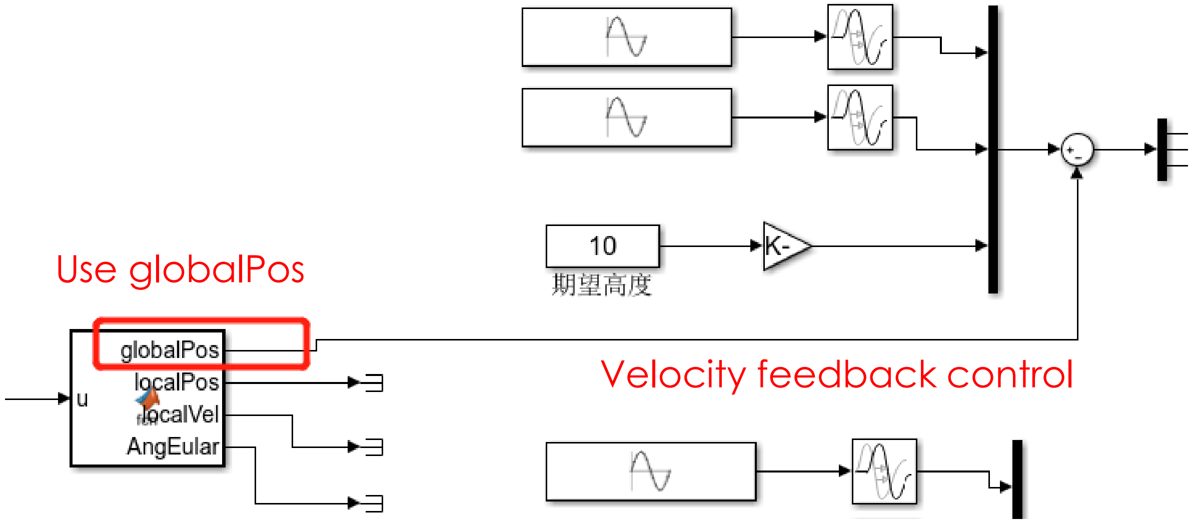
- **Note**: Please modify the **.slx** file to make the trajectory center of each vehicle consistent. (Hint: Change the feedback position signal from **LocalPos** to **GlobalPos**)
- **Note**: **LocalPos** takes the respective armed position as the origin, and **GlobalPos** takes the center of the CopterSim/RflySim3D map as the origin



3. Basic examples

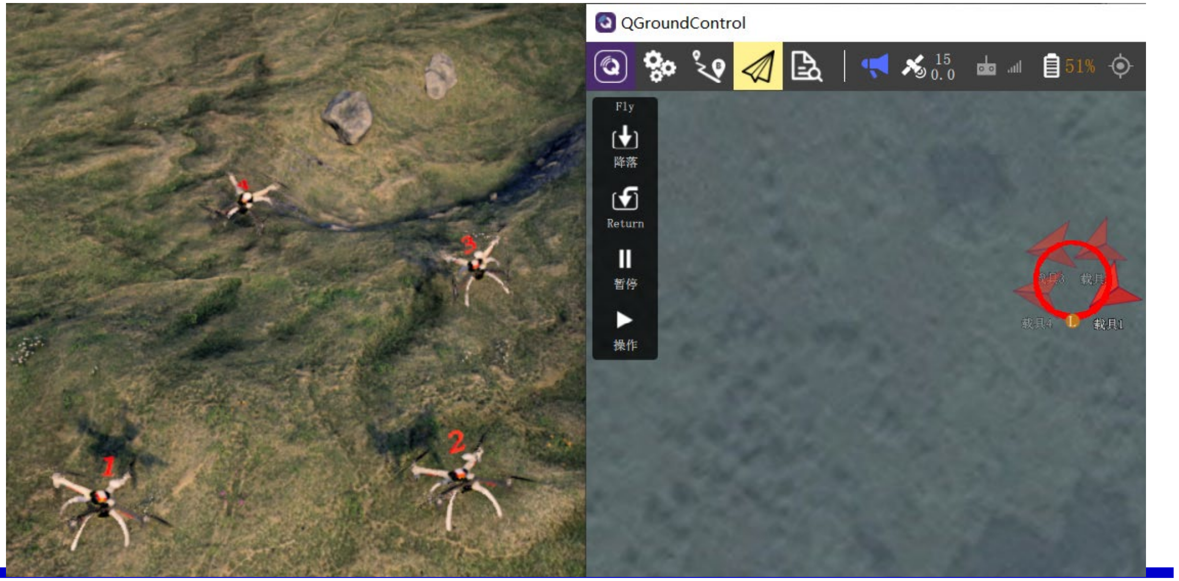
3.7 Example of **FullData Mode** of communication interface --- four vehicles experimental results

- Open the "**RflyUdpFullFour.slx**" demo with MATLAB, modify this file according to the figure below left, and change the feedback position from **LocalPos** to **GlobalPos** to ensure that all vehicles use the CopterSim map coordinate system
- Click "**RflyUdpFullFour.bat**" and then run "**RflyUdpFullFour.slx**", you can see that the vehicles take off and hover to the same point in the air, and then start to fly around the same center.



Use globalPos

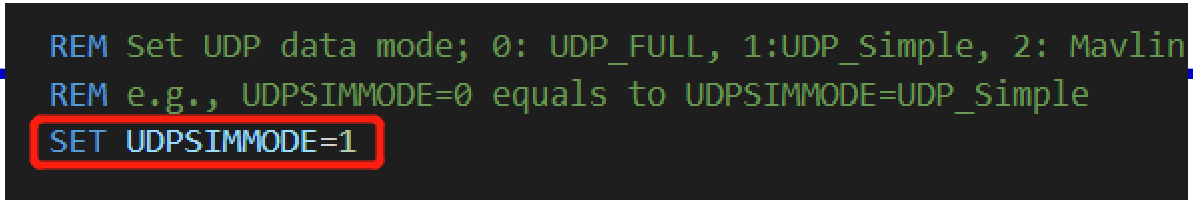
Velocity feedback control



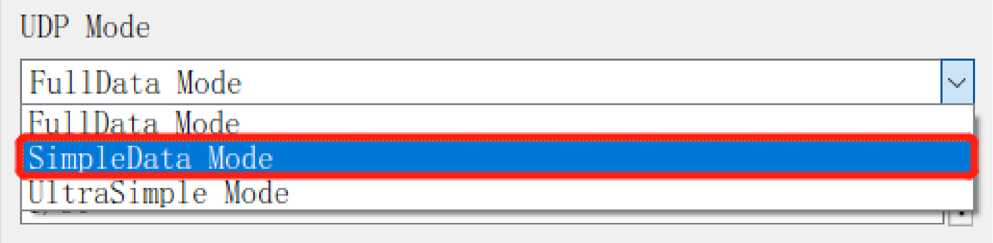
3. Basic examples

3.8 SimpleData Mode example of communication interface --- experimental effect

- Double-click "**RflySimAPIs\SimulinkSwarmAPI\SimulinkDemo\RflyUdpSimpleOne.bat**" (or double-click the script "**SITLRunUdpSimple.bat**" in the upper directory, enter the number **1** and press key "**Enter**") to open the one-vehicle SIL simulation system (or insert Pixhawk and run "**HITLRunUdpSimple.bat**" turns on one-vehicle HIL simulation).
- Open the "**RflyUdpSimpleOne.slx**" demo with MATLAB, click the "**Run**" button in Simulink, you can see that a single drone takeoff vehicle takes off to a certain height, and then starts to draw a circle after a while.
- The result of this experiment is the same as in **Section 3.5**, and the same control function is achieved through two interfaces.
- In this example, the "**UDPSIMMODE**" in the bat script is set to **1**, which corresponds to the "**UDP Mode**" in CopterSim is set to "**UDP_Simple**", and also corresponds to the **SimpleData Mode** in **RflySwarmAPI**

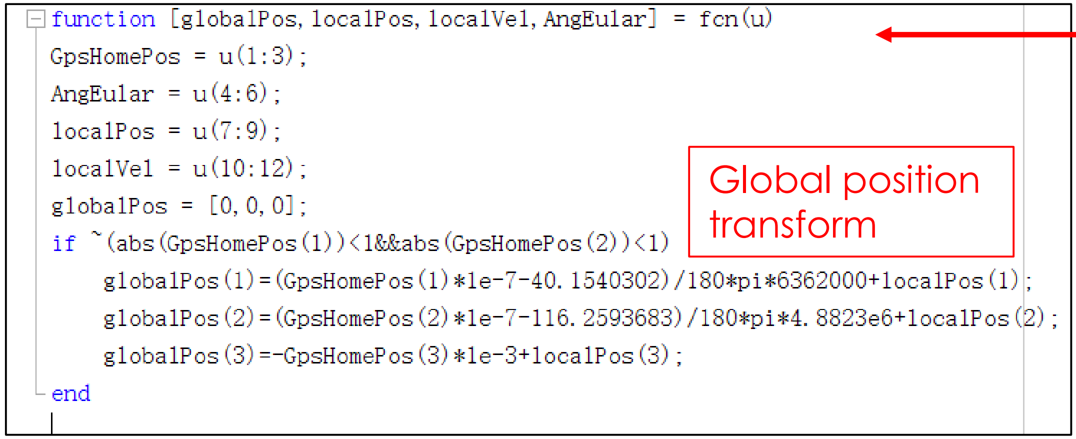


```
REM Set UDP data mode; 0: UDP_FULL, 1:UDP_Simple, 2: Mavlin
REM e.g., UDPSIMMODE=0 equals to UDPSIMMODE=UDP_Simple
SET UDPSIMMODE=1
```



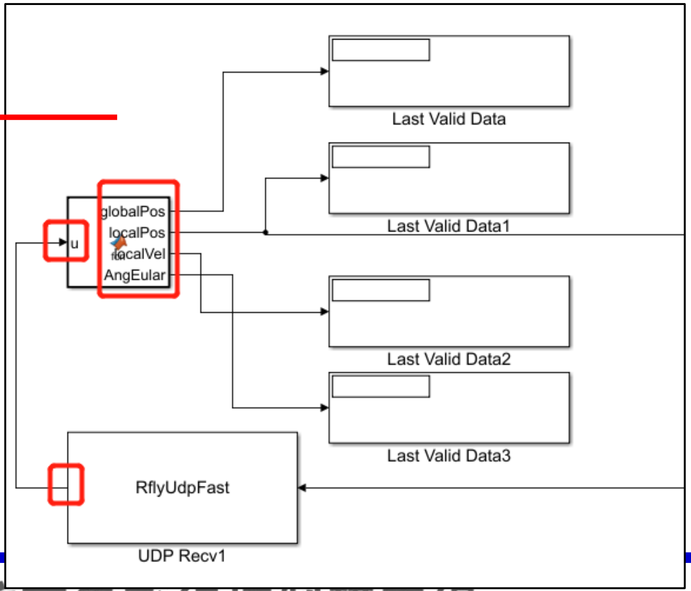
3.9 Communication interface SimpleData Mode example --- input and output mapping analysis

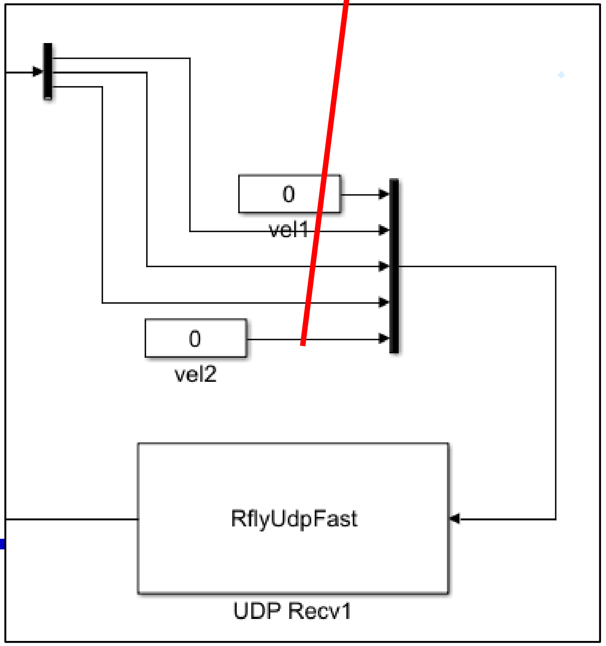
- As shown in the figure on the right, the input part of "**RflySwarmAPI**" does not need to be mapped, just give the 5-dimensional control signal according to the data of **SimpleData** (see the definition in **Section 2.5**).
- As shown in the figure on the left below, the "**RflySwarmAPI**" output module needs to be mapped to obtain the value of CopterSim global coordinates **globalPos** (reserved for multi-vehicle swarms)



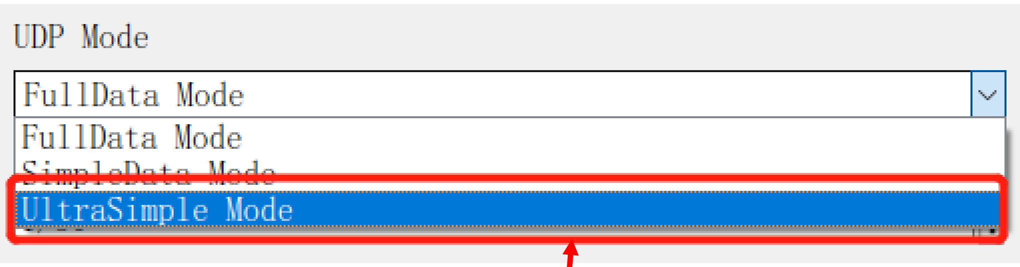
```
function [globalPos, localPos, localVel, AngEular] = fcn(u)
GpsHomePos = u(1:3);
AngEular = u(4:6);
localPos = u(7:9);
localVel = u(10:12);
globalPos = [0,0,0];
if ~(abs(GpsHomePos(1))<1&&abs(GpsHomePos(2))<1)
    globalPos(1)=(GpsHomePos(1)*1e-7-40.1540302)/180*pi*6362000+localPos(1);
    globalPos(2)=(GpsHomePos(2)*1e-7-116.2593683)/180*pi*4.8823e6+localPos(2);
    globalPos(3)=-GpsHomePos(3)*1e-3+localPos(3);
end
```

Global position transform





3. Basic examples



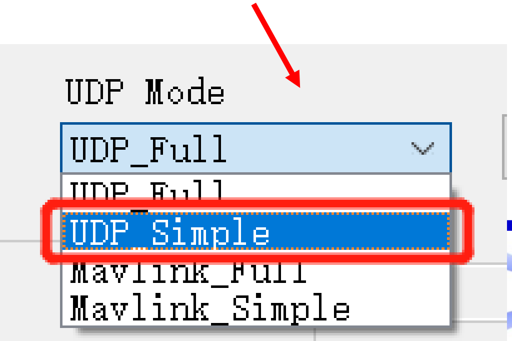
UDP Mode

FullData Mode

FullData Mode
SimpleData Mode
UltraSimple Mode

3.10 Communication interface UltraSimple Mode example --- input and output mapping analysis

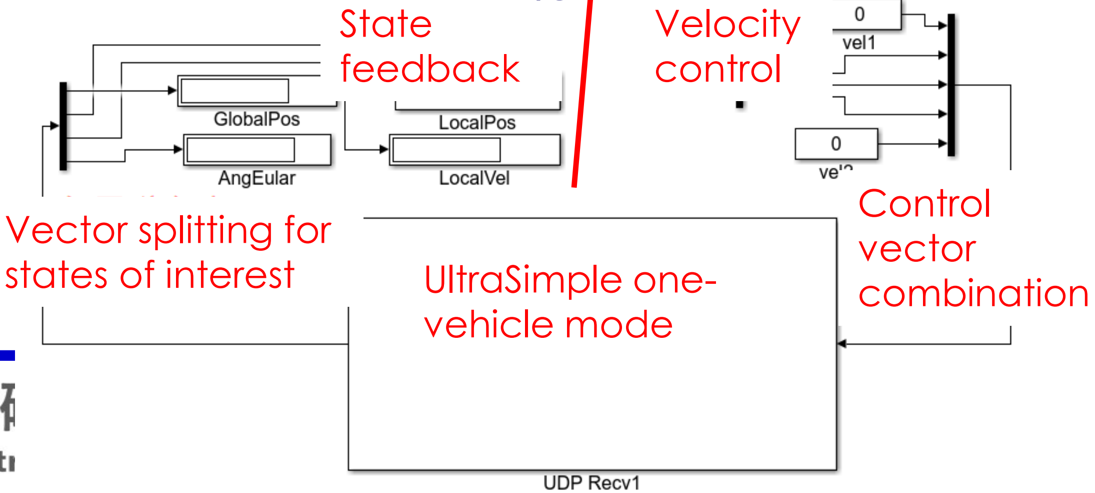
- Open "**RflySimAPIs\SimulinkSwarmAPI\SimulinkDemo\RflyUdpUltraSimpleOne. slx**" to view an example of the **RflySwarmAPI** module using **UltraSimple Mode**. This example has less data transmission and less delay, and is suitable for the needs of more vehicle swarm speed/position/trajectory control.
- The input of the **UltraSimple Mode** of the **RflySwarmAPI** module is a 5-dimensional vector, and its definition is exactly the same as in the previous section; the output is a 12-dimensional vector, and its definition can be referred to **Section 2.8**.
- As shown in the figure on the right, this interface does not require additional processing, and can directly use vector decomposition to obtain the states of interest (global position, local position, attitude, etc.) for control.
- This example requires CopterSim to use **UDP_Simple.**



UDP Mode

UDP_Full

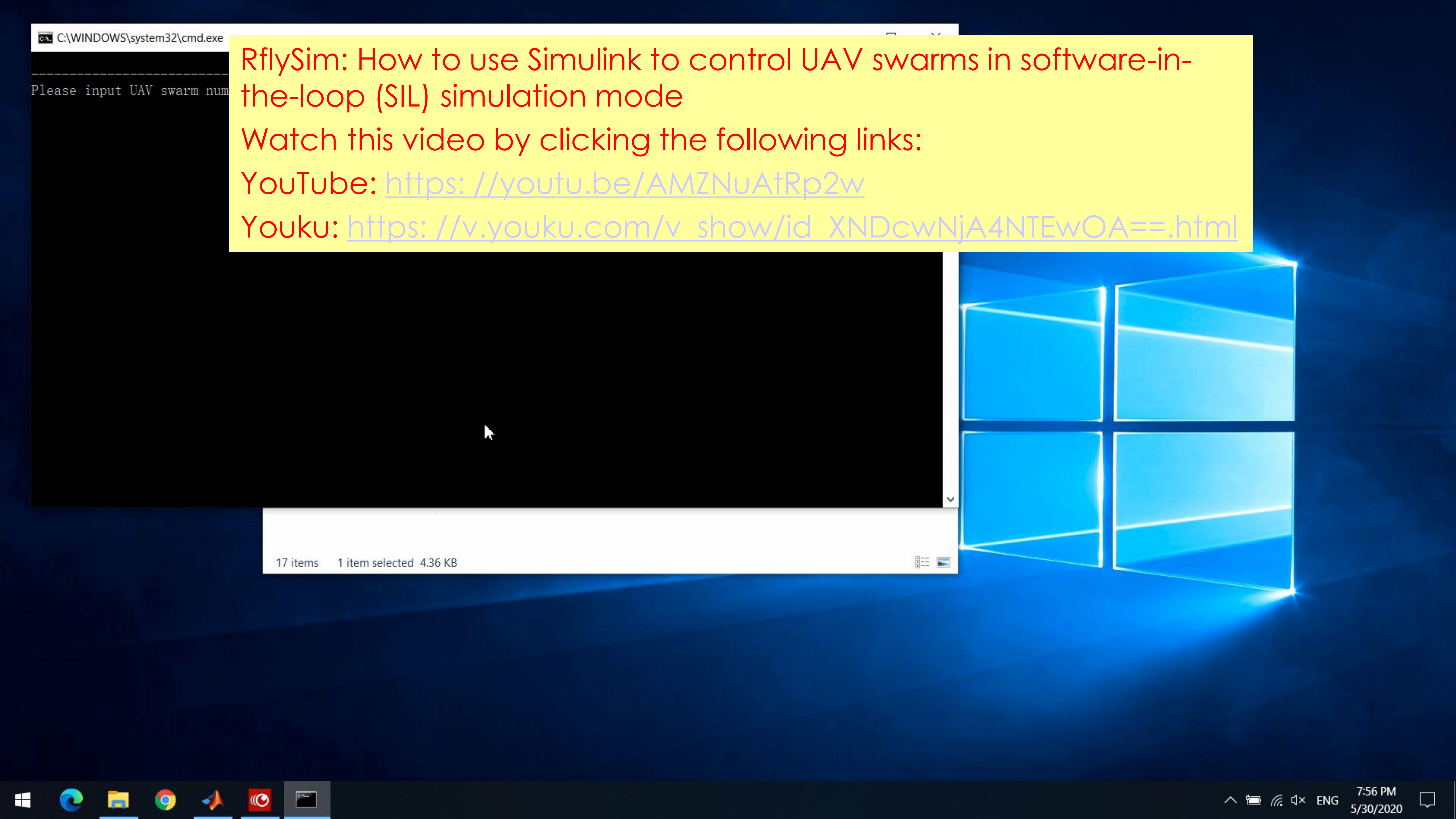
UDP_Full
UDP_Simple
Mavlink_Full
Mavlink_Simple


State feedback
Velocity control
0
vel1
GlobalPos
LocalPos
0
vel2
AngEular
LocalVel
Vector splitting for states of interest
Control vector combination
UltraSimple one-vehicle mode
UDP Recv1

3. Basic examples

3.11 Communication interface UltraSimple Mode example --- experimental results

- Enter "**RflySimAPIs\SimulinkSwarmAPI\SimulinkDemo**" folder
- Double-click "**RflyUdpUltraSimpleOne.bat**" (or double-click the "**SITLRunUdpSimple.bat**" script in the upper directory, enter the number **1** and press key “**Enter**”) and then run "**RflyUdpUltraSimpleOne.slx**" to see one vehicle taking off and drawing a circle.
- Double-click "**RflyUdpUltraSimpleFour.bat**" (or double-click the "**RflyUdpUltraSimpleFour.bat**" script in the upper directory, enter the number 4 and press Enter) and then run "**RflyUdpUltraSimpleFour.slx**" to see four vehicles taking off and drawing circles.
- Double-click "**RflyUdpUltraSimpleEight.bat**" (or double-click the "**SITLRunUdpSimple.bat**" script in the upper directory, enter the number **8** and press key “**Enter**”) and then run "**RflyUdpUltraSimpleEight.slx**" to see eight vehicles take off and draw a circle. (The flight effect is related to the computer configuration and requires a higher **CPU** and **GPU**)
- The above example can also insert a specific number of Pixhawk, then run "**HITLRunUdpSimple.bat**" and enter the serial port number string to start the multi-vehicle hardware-in-the-loop simulation, and run the corresponding **slx** file to observe the effect.



RflySim: How to use Simulink to control UAV swarms in software-in-the-loop (SIL) simulation mode

Watch this video by clicking the following links:

YouTube: https: //youtu.be/AMZNuAtRp2w

Youku: https: //v.youku.com/v_show/id_XNDcwNjA4NTEwOA==.html

C:\WINDOWS\system32\cmd.exe

```
--------------------------
Please input UAV swarm num
```

17 items   1 item selected 4.36 KB

Content

1. Setup Instructions

2. Interface Introduction

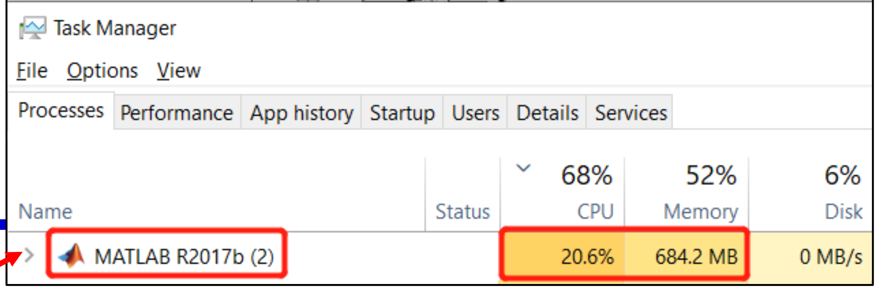
3. Basic examples

4. Advanced examples

5. summary

Path of demo source code of this Section:
RflySimAPIs\SimulinkSwarmAPI

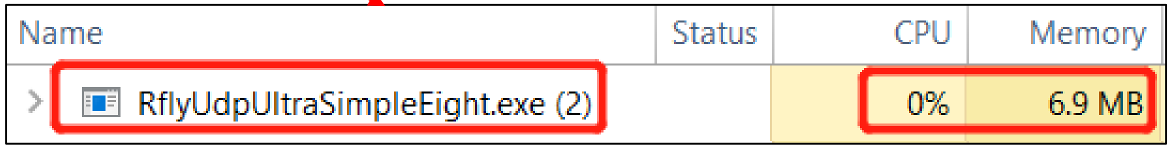
4. Advanced examples


Task Manager
File Options View
Processes Performance App history Startup Users Details Services

Name	Status	68% CPU	52% Memory	6% Disk
MATLAB R2017b (2)		20.6%	684.2 MB	0 MB/s

4.1 Compile Simulink swarm algorithm into exe to speed up operation

- MATLAB itself occupies a lot of CPU and memory resources (see the figure on the right). When running complex Simulink control programs,
- On the one hand, the calculation amount is too large, which causes the algorithm to run slowly and cannot meet the real-time requirements (1s in Simulink is greater than the real clock 1s), so it is impossible to control the swarm vehicle of the real-time simulation system (or real hardware involved system) in real time
- Secondly, if Simulink occupies a large amount of computing resources during simulation, it will lead to less allocation of computing resources for RflySim3D and CopterSim, resulting in poor real-time performance of vehicle simulation, severe shaking and even crashes.
- After compiling the Simulink controller to generate an exe, the algorithm can run without MATLAB, and it is a binary executable file itself, which has very high operating efficiency. Even large-scale control algorithms can guarantee real-time control.
- It is a resource manager image that runs eight-vehicle swarm control algorithms in exe mode. It can be seen that its memory and CPU usage are almost 0, which has a huge performance improvement compared to Simulink running directly.
- **Note**: VS compiler is required to generate exe.

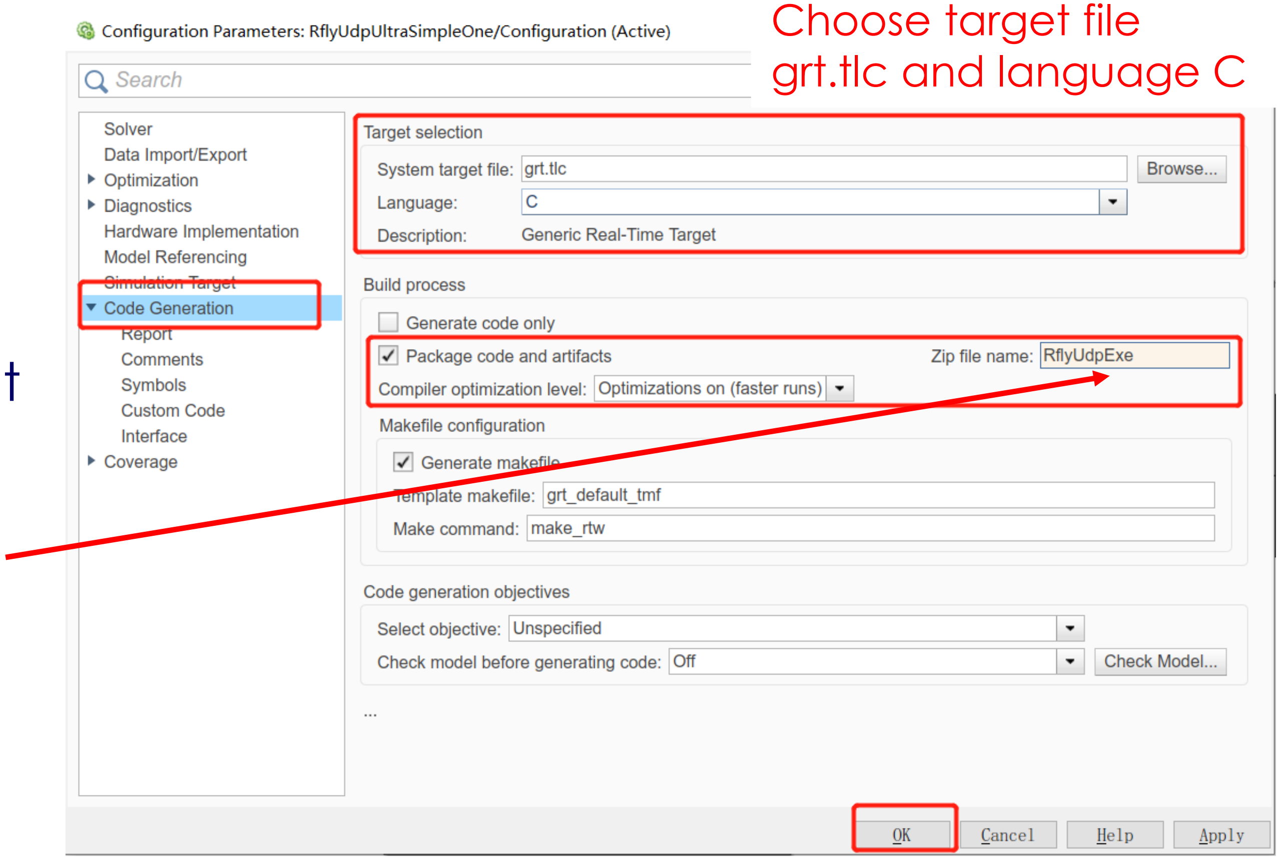


Name	Status	CPU	Memory
RflyUdpUltraSimpleEight.exe (2)		0%	6.9 MB

4. Advanced examples

4.2 Generate exe program Simulink settings

- Open the "**RflyUdpUltraSimpleFour.slx**" file (or create a new **.slx** file, but ensure that **GenerateSwarmExe.p**, **RflyUdpFast.cpp** and **RflyUdpFast.mexw64** are copied to the same directory)
- Click the setting button of Simulink and set as shown in the figure below. Use the default for everything else, the key is to check "**Package Code and …**", enter the name: "**RflyUdpExe**"; then, set the compilation mode to: "**faster runs**" (this item is not required, you can make it run after setting faster).



Configuration Parameters: RflyUdpUltraSimpleOne/Configuration (Active)

Target selection

System target file: grt.tlc

Language: C

Description: Generic Real-Time Target

Build process

□ Generate code only

☑ Package code and artifacts

Zip file name: RflyUdpExe

Compiler optimization level: Optimizations on (faster runs)

Makefile configuration

☑ Generate makefile

Template makefile: grt_default_tmf

Make command: make_rtw

Code generation objectives

Select objective: Unspecified

Check model before generating code: Off

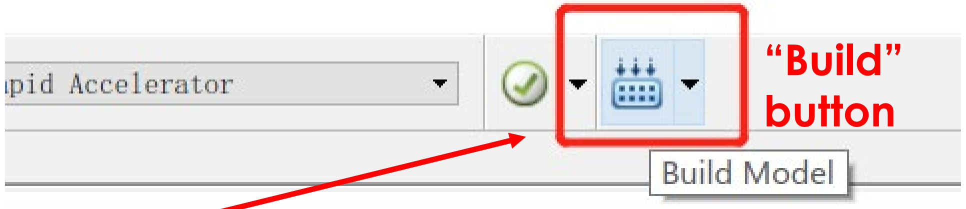
...

Choose target file grt.tlc and language C

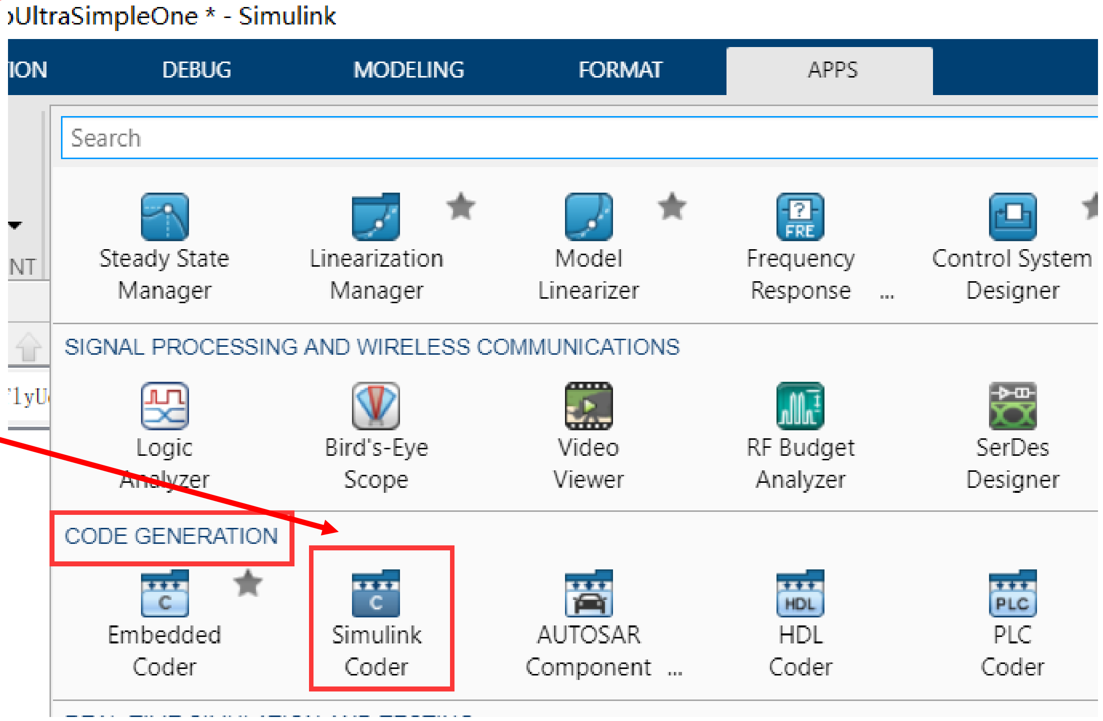
4. Advanced examples

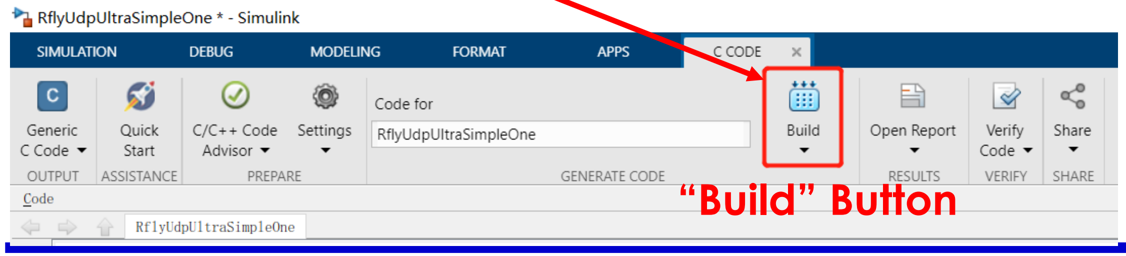
4.2 Generate exe program Simulink settings

- Click the “**Build**” button of Simulink
- For **MATLAB 2019a** and earlier versions, the toolbar style is shown in the upper right picture, just click its “**Build**” button.
- For 2019b and later versions, as shown on the right, click **APPS-CODE GENERATION-Simulink Coder** to pop up the code generation toolbar, in which click the "**Build**" button as shown below to compile and generate the code



“Build” button



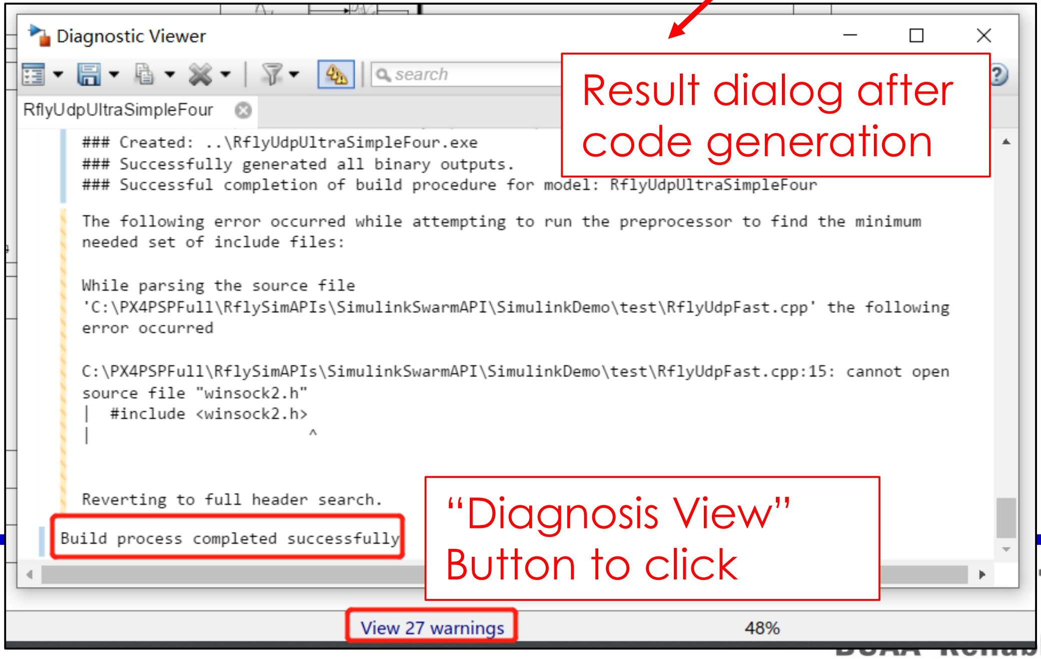


“Build” Button

4. Advanced examples

4.3 Generate exe program compilation results

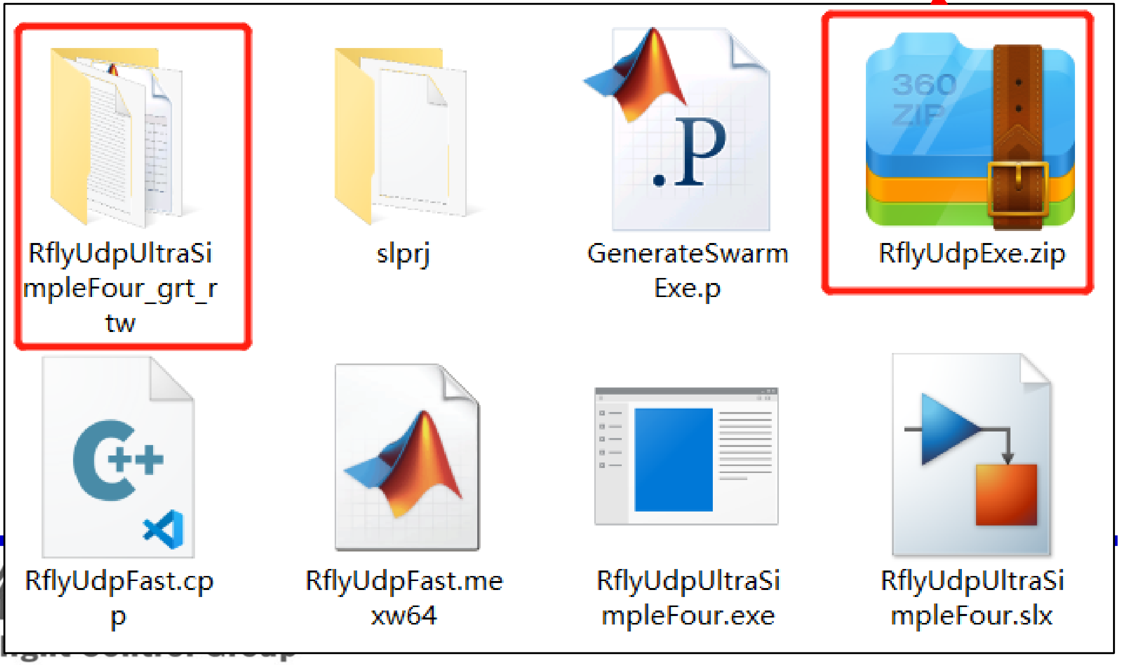
- After the compilation is completed, you will get the prompt of successful completion of compilation in the "**Diagnostic Viewer**" (Diagnostic Viewer) as shown in the figure below (please ignore the yellow warning)
- As shown in the picture on the right, we can get a "**RflyUdpExe.zip**" compressed package file, which contains all the generated codes, which will be used for our subsequent generation of exe files.



```
### Created: ..\RflyUdpUltraSimpleFour.exe
### Successfully generated all binary outputs.
### Successful completion of build procedure for model: RflyUdpUltraSimpleFour

The following error occurred while attempting to run the preprocessor to find the minimum
needed set of include files:

While parsing the source file
'C:\PX4PSPFull\RflySimAPIs\SimulinkSwarmAPI\SimulinkDemo\test\RflyUdpFast.cpp' the following
error occurred

C:\PX4PSPFull\RflySimAPIs\SimulinkSwarmAPI\SimulinkDemo\test\RflyUdpFast.cpp:15: cannot open
source file "winsock2.h"
|  #include <winsock2.h>
|                      ^


Reverting to full header search.

Build process completed successfully
```

Result dialog after code generation

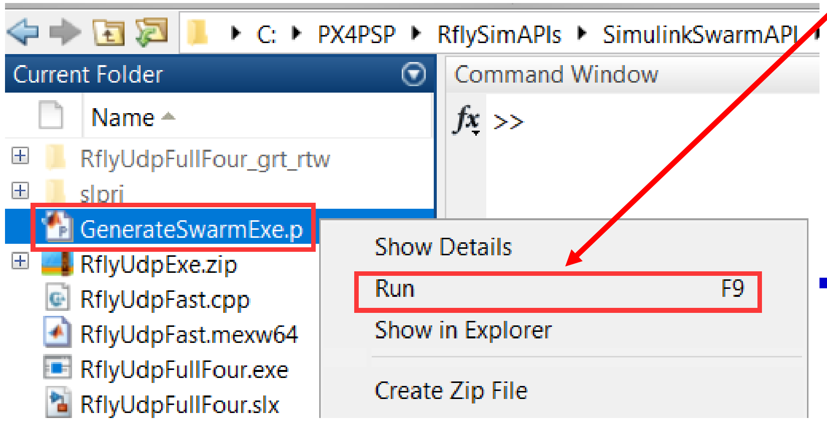
“Diagnosis View” Button to click

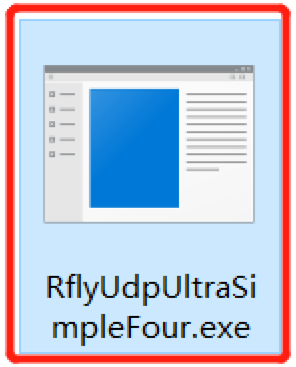

RflyUdpUltraSimpleFour_grt_rtw
slprj
GenerateSwarmExe.p
RflyUdpExe.zip
RflyUdpFast.cpp
RflyUdpFast.mexw64
RflyUdpUltraSimpleFour.exe
RflyUdpUltraSimpleFour.slx

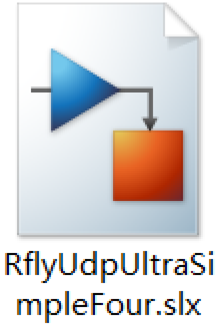
4. Advanced examples

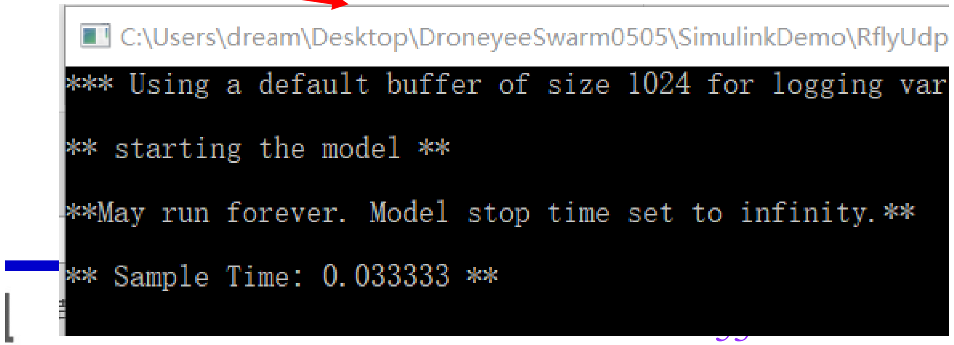
4.3 Generate exe program commands and effects

- **Note**: A "**RflyUdpUltraSimpleFour.exe**" file was generated in the previous step of compiling the command. Double-click it to run, but it does not meet the real-time control requirements (always run at the fastest speed, not synchronized with the real clock), so we need to run our script To generate real-time control exe files.
- As shown in the figure below, right-click the **GenerateSwarmExe.p** file and click "**Run**". Or directly enter "**GenerateSwarmExe**" in the MATLAB command line, and you can get the .exe file "**RflyUdpUltraSimpleFour.exe**" corresponding to the Simulink file name as shown in the figure below.
- Double-click "**RflyUdpUltraSimpleFour.bat**" to start the four-vehicle SITL system, and click the exe file generated in the previous step to control the vehicle to take off and draw a circle like Simulink. The exe running window is as shown in the lower right.







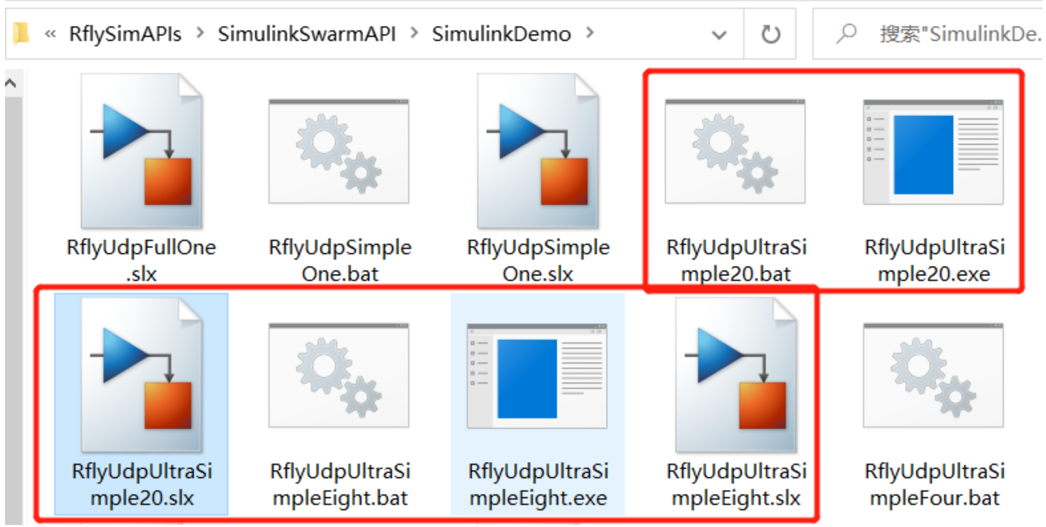

C:\Users\dream\Desktop\DroneyeeSwarm0505\SimulinkDemo\RflyUdp

```
*** Using a default buffer of size 1024 for logging var

** starting the model **

**May run forever. Model stop time set to infinity.**

** Sample Time: 0.033333 **
```

4. Advanced examples

4.3 Other examples of generating exe programs

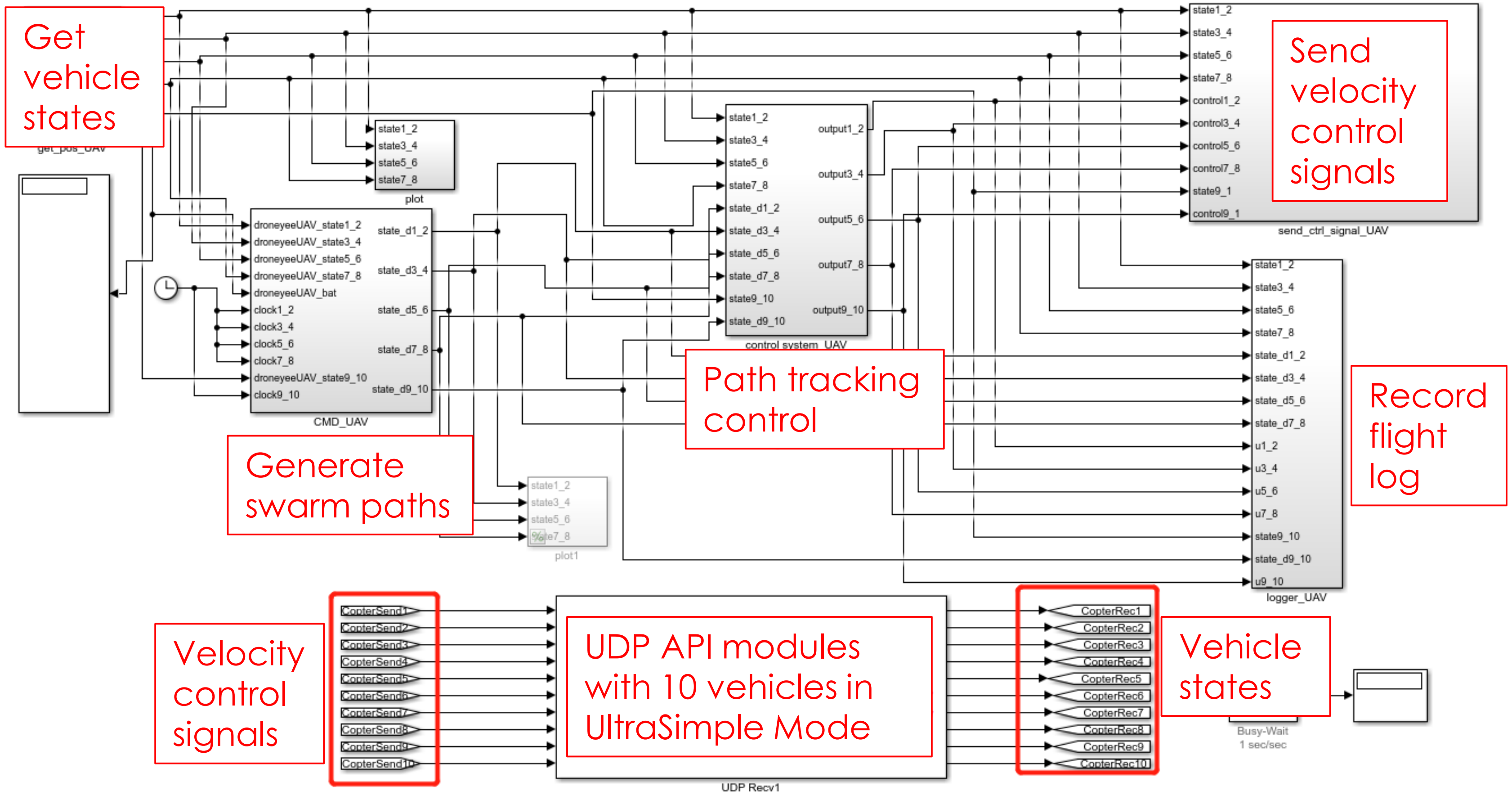
- Please use the exe code generation method to run the 8 vehicle flight example (computer configuration support is required). For the example file, see "**RflyUdpUltraSimpleEight.exe**".
- Please use the exe code generation method to run the 20 vehicle example (only the full version, and computer configuration support is required). For the example file, see "RflyUdpUltraSimple20.exe".
- All the demos under the folder "**RflySimAPIs\SimulinkSwarmAPI\SimulinkDemo**" support the generation of exe files, and then software/hardware-in-the-loop simulation. Please try to run these examples yourself.
- **Note**: The RflySimVision version and the RflySimSwarm version limit the maximum number of vehicle to 8, so it is impossible to run the demo of 20 vehicles


« RflySimAPIs › SimulinkSwarmAPI › SimulinkDemo ›
搜索"SimulinkDe...
RflyUdpFullOne.slx
RflyUdpSimpleOne.bat
RflyUdpSimpleOne.slx
RflyUdpUltraSimple20.bat
RflyUdpUltraSimple20.exe
RflyUdpUltraSimple20.slx
RflyUdpUltraSimpleEight.bat
RflyUdpUltraSimpleEight.exe
RflyUdpUltraSimpleEight.slx
RflyUdpUltraSimpleFour.bat

4. Advanced examples

4.4 Example of eight vehicles flying in figure 8 formation

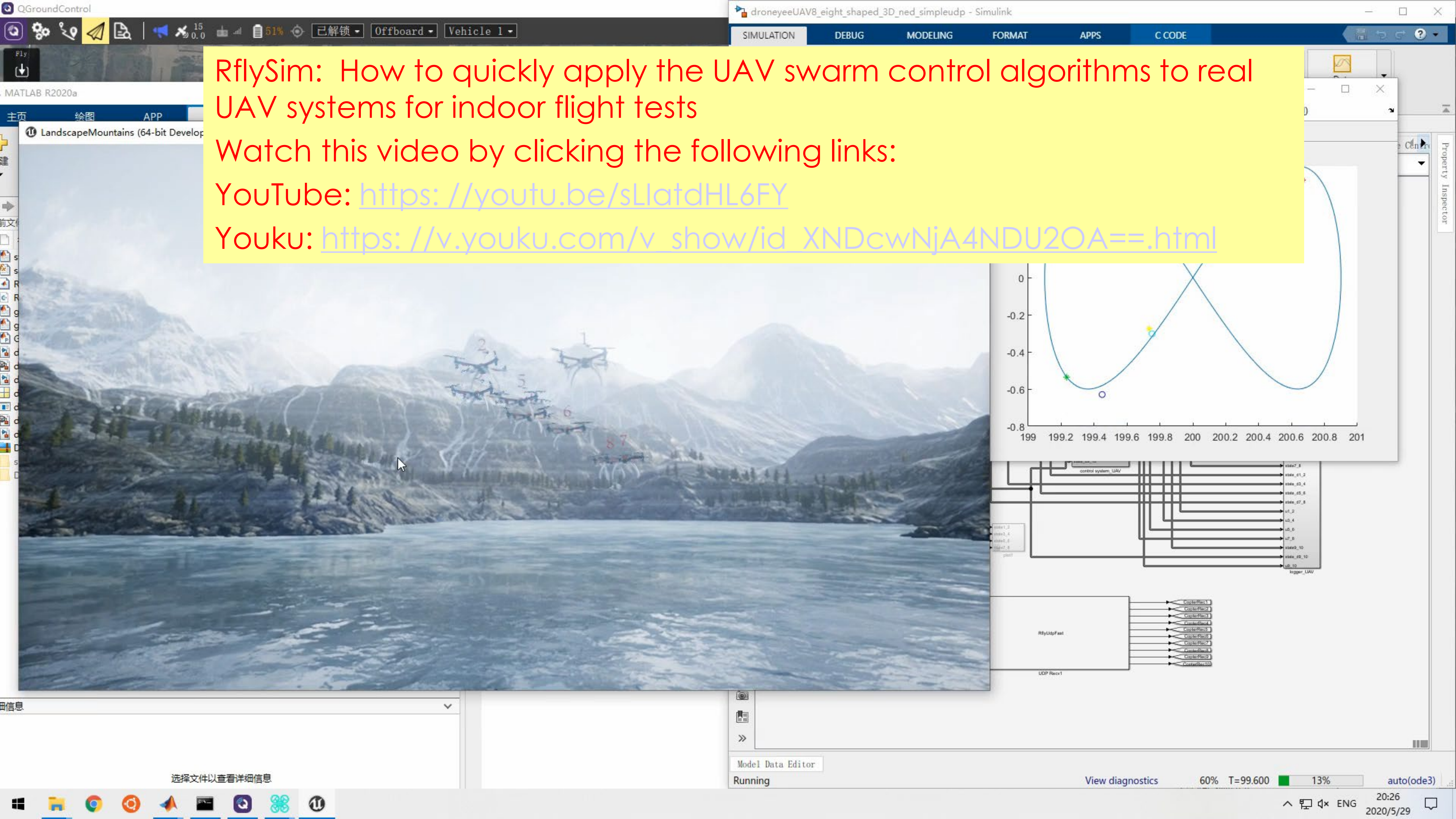
- Enter the "**RflySimAPIs\SimulinkSwarmAPI\SimulinkDemo\UAV8Swarm3D**" folder
- Open the "**UAV8Swarm3D.slx**" file with MATLAB, and you can see the Simulink flying figure 8 formation control algorithm.
- Note: This algorithm can be used for formation control of 1-10 vehicles.
- Read the internal implementation by yourself



4. Advanced examples

4.5 Example of eight vehicles flying in 8-character formation --- experimental results

- Running "**UAV8Swarm3D.bat**" in this directory will automatically open the SITL simulation mode of eight vehicles. Or insert 8 flight controllers and run "**RflySimAPIs\SimulinkSwarmAPI\HITLRunUdpSimple.bat**" to start the hardware-in-the-loop of the eight vehicles.
- **Note**: This example can control a formation of 10 vehicles at most. You can also modify the **VehicleNum** amplitude statement in "**UAV8Swarm3D.bat**" to "**SET VehicleNum=10**".
- Open MATLAB and run the "**UAV8Swarm3D.slx**" file, you can see that the vehicles take off one by one, and the 8 vehicles are flying in formation.
- **Note**: A computer with insufficient configuration may freeze and blow up when running 8 vehicles SITL and MATLAB at the same time. You can also generate "**UAV8Swarm3D.exe**" according to the previous steps to run the 8-character formation without opening MATLAB.



RflySim: How to quickly apply the UAV swarm control algorithms to real UAV systems for indoor flight tests

Watch this video by clicking the following links:

YouTube: https: //youtu.be/sLlatdHL6FY

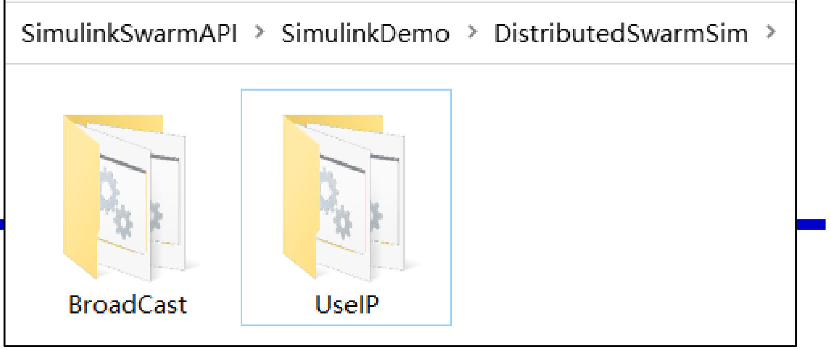
Youku: https: //v.youku.com/v_show/id_XNDcwNjA4NDU2OA==.html

4. Advanced examples

As shown in the figure on the right, the code of this example can also be directly used for formation experiments of more vehicles.

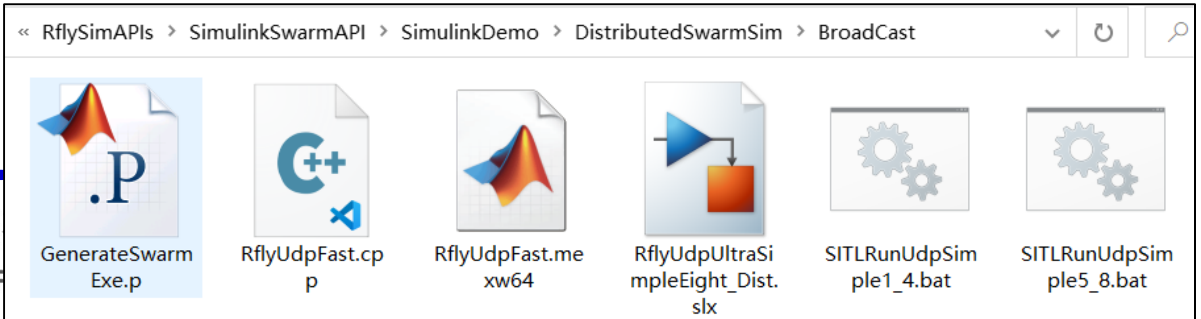
Note: The indoor test needs to use the Optitrack Vison positioning system to obtain the position, and the outdoor GPS positioning can be used to form the UAV swarm.

4. Advanced examples



4.6 Distributed LAN simulation (RflySimFull version only)

- For example files, see "**RflySimAPIs\SimulinkSwarmAPI\SimulinkDemo\DistributedSwarmSim**", which contains two folders "**BroadCast**": use broadcast communication to simulate LAN networking; "**UseIP**": specify the IP addresses of two computers Come to network simulation.
- The advantage of the "**BroadCast**" example is that it is simple to configure and does not need to check the addresses of two computers in the local area network. It can be directly run when copied to other computers, but the communication efficiency is low, unstable, and delayed.
- The "**UseIP**" example is the opposite of the broadcast method. The configuration is more complicated (you need to specify the IP of two computers), but the performance is higher, the communication speed is fast, and the delay is small. This mode is recommended for more vehicle simulations.
- There are two bat files in both code folders, which need to be run on two computers separately


« RflySimAPIs › SimulinkSwarmAPI › SimulinkDemo › DistributedSwarmSim › BroadCast
GenerateSwarmExe.p
RflyUdpFast.cpp
RflyUdpFast.mexw64
RflyUdpUltraSimpleEight_Dist.slx
SITLRunUdpSimple1_4.bat
SITLRunUdpSimple5_8.bat

4. Advanced examples

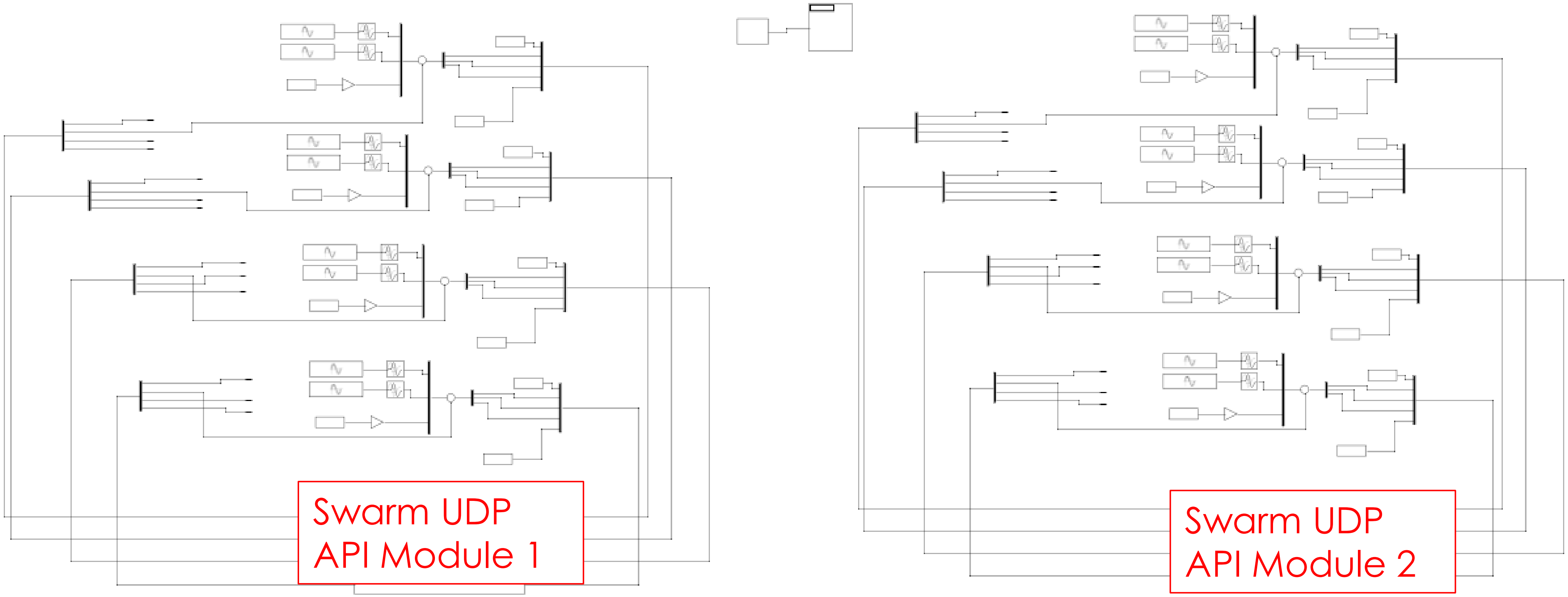
4.6 Precautions for writing distributed LAN simulation bat script

- "**SITLRunUdpSimple1_4.bat**" is used on a computer to start the software-in-the-loop simulation of vehicle 1~4. Compared with other bat scripts, its modifications are as follows
- After the **SET /a START_INDEX=1** statement, add **SET /a TOTOAL_COPTER=8** on a new line; for the "**BroadCast**" demo, modify its **SET IS_BROADCAST=0** to: "**SET IS_BROADCAST=1**"; for the "**UseIP**" demo , Use "**SET IS_BROADCAST= 192.168.1.5**", where "**192.168.1.5**" is the IP address of the computer used for Simulink swarm control.
- "**SITLRunUdpSimple5_8.bat**" is used on the second computer to start the software-in-the-loop simulation of vehicle 5-8. Compared with other bat scripts, its modifications are as follows
- Change its **SET /a START_INDEX=1** to: **SET /a START_INDEX=5**; after the statement, add **SET /a TOTOAL_COPTER=8** to the first line; for the "**BroadCast**" demo, change its **SET IS_BROADCAST=0** to : "**SET IS_BROADCAST=1**"; For the "**UseIP**" demo, use "**SET IS_BROADCAST= 192.168.1.5**", where "**192.168.1.5**" is the IP address of the computer used for Simulink swarm control.

4. Advanced examples

4.6 Simulink example of distributed LAN simulation

Open "**DistributedSwarmSim\BroadCast\RflyUdpUltraSimpleEight_Dist.slx**", the internal implementation of the device is as follows, which contains two UDP communication modules, which communicate with two computers respectively



Swarm UDP API Module 1

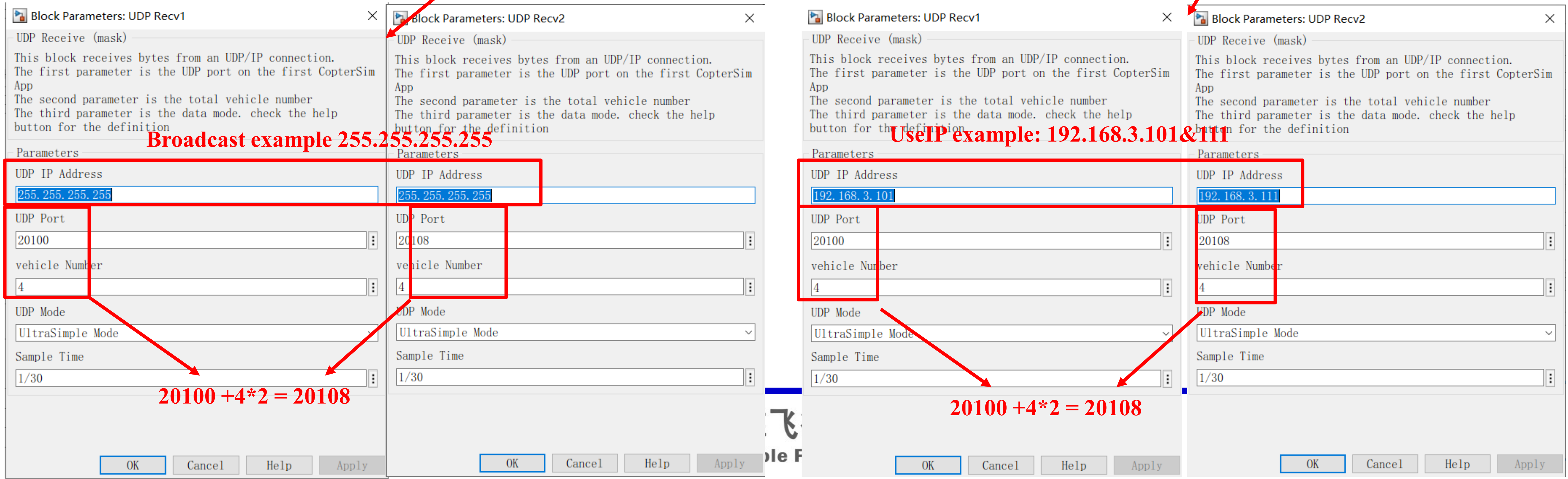
Swarm UDP API Module 2

4. Advanced examples

4.6 Simulink example of distributed LAN simulation

For the **BroadCast** example, enter "**255.255.255.255**" for the IP address, and for the **UseIP** example, enter the IP address of the computer where the controlled vehicle is located (for example, **192.168.3.101**)

The module on the left corresponds to vehicle No. 1~4, the starting port number is **20100**; the vehicle on the right starts from 5, and the starting port is **20108**



Broadcast example 255.255.255.255

UseIP example: 192.168.3.101&111

20100 +4*2 = 20108

20100 +4*2 = 20108

Block Parameters: UDP Recv1

UDP Receive (mask)

This block receives bytes from an UDP/IP connection.
The first parameter is the UDP port on the first CopterSim App
The second parameter is the total vehicle number
The third parameter is the data mode. check the help button for the definition

Parameters

UDP IP Address: 255.255.255.255

UDP Port: 20100

vehicle Number: 4

UDP Mode: UltraSimple Mode

Sample Time: 1/30

Block Parameters: UDP Recv2

UDP Receive (mask)

This block receives bytes from an UDP/IP connection.
The first parameter is the UDP port on the first CopterSim App
The second parameter is the total vehicle number
The third parameter is the data mode. check the help button for the definition

Parameters

UDP IP Address: 255.255.255.255

UDP Port: 20108

vehicle Number: 4

UDP Mode: UltraSimple Mode

Sample Time: 1/30

Block Parameters: UDP Recv1

UDP Receive (mask)

This block receives bytes from an UDP/IP connection.
The first parameter is the UDP port on the first CopterSim App
The second parameter is the total vehicle number
The third parameter is the data mode. check the help button for the definition

Parameters

UDP IP Address: 192.168.3.101

UDP Port: 20100

vehicle Number: 4

UDP Mode: UltraSimple Mode

Sample Time: 1/30

Block Parameters: UDP Recv2

UDP Receive (mask)

This block receives bytes from an UDP/IP connection.
The first parameter is the UDP port on the first CopterSim App
The second parameter is the total vehicle number
The third parameter is the data mode. check the help button for the definition

Parameters

UDP IP Address: 192.168.3.111

UDP Port: 20108

vehicle Number: 4

UDP Mode: UltraSimple Mode

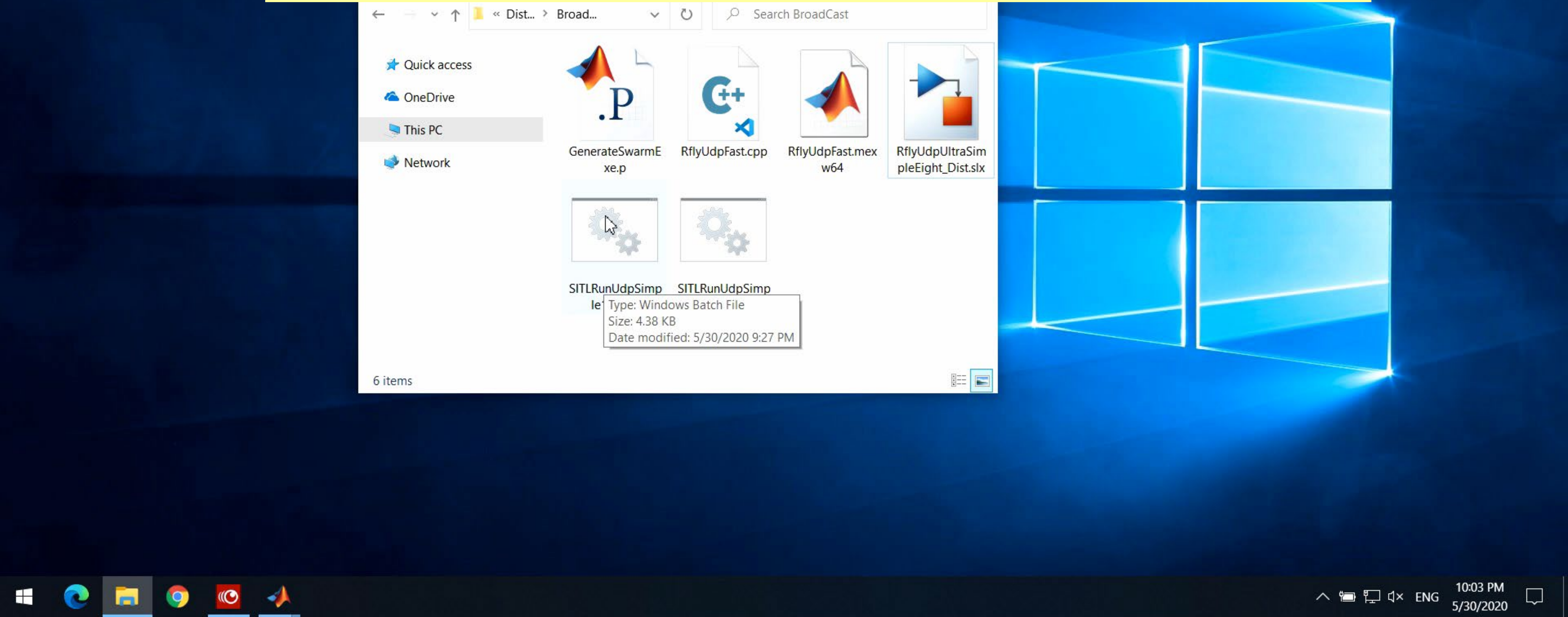
Sample Time: 1/30

RflySim: How to quickly perform distributed software-in-the-loop simulation (SIL) for UAV swarm with multiple computer

Watch this video by clicking the following links:

YouTube: https: //youtu.be/fmzYADSQyj0

Youku: https: //v.youku.com/v_show/id_XNDcwNjA4NDE2OA==.html

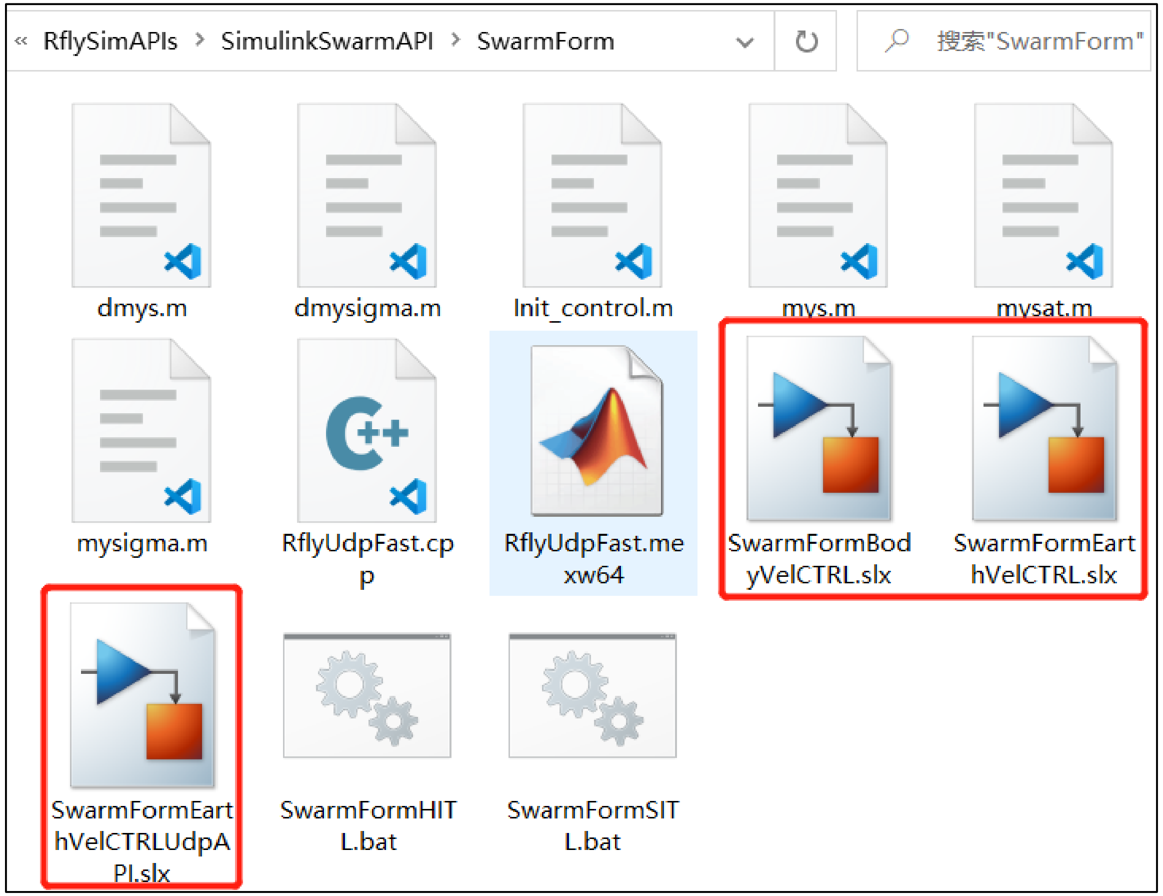


Type: Windows Batch File
Size: 4.38 KB
Date modified: 5/30/2020 9:27 PM

4. Advanced examples

4.7 Implementation of the formation --- free swarm algorithm with automatic collision avoidance

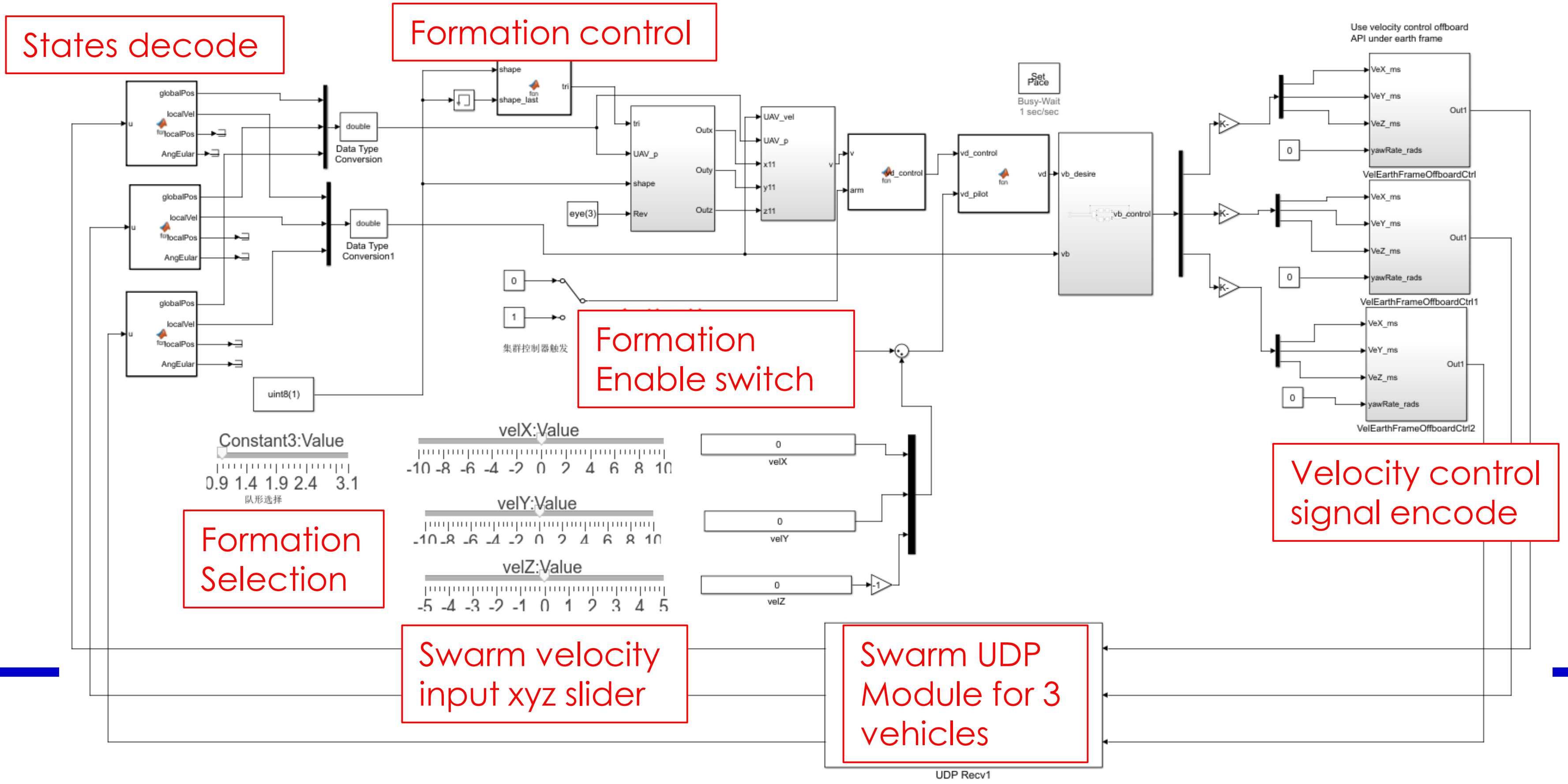
- See the folder "**RflySimAPIs\SimulinkSwarmAPI\SwarmForm**" for the experimental example, which contains three programs:
- "**SwarmFormBodyVelCTRL.slx**" uses Simulnk's UDP module to control vehicle formation by sending airframe speed signals
- "**SwarmFormEarthVelCTRL.slx**" uses Simulnk's UDP module to control vehicle formation by sending earth speed (NED coordinate system) signals
- "**SwarmFormEarthVelCTRLUdpAPI.slx**" uses the multi-vehicle UDP module (**UDP_Full** mode) provided by us to send earth speed (NED coordinate system) control signals.



4. Advanced examples

4.7 Implementation of the formation --- free swarm algorithm with automatic collision avoidance

Let's take "**SwarmFormEarthVelCTRLUdpAPI.slx**" as an example to introduce the operation process



States decode

Formation control

Formation Enable switch

Formation Selection

Swarm velocity input xyz slider

Swarm UDP Module for 3 vehicles

Velocity control signal encode

4. Advanced examples

4.7 achieve of the formation --- free swarm algorithm with automatic collision avoidance-mathematical principle

- Use artificial potential field method to avoid obstacles

$$\sigma(x,d_1,d_2)=\begin{cases}1, & x\le d_1\\ Ax^3+Bx^2+Cx+D, & d_1\le x\le d_2\\ 0, & d_2\le x\end{cases}\qquad s(x,r_s)=\begin{cases}x & 0\le x\le x_1\\ (1-r_s)+\sqrt{{r_s}^2-(x-x_2)^2} & x_1\le x\le x_2\\ 1 & x_2\le x\end{cases}$$

$$V_{m,il}=\frac{k_3\sigma_{m,il}(\|\tilde{\xi}_{m,il}\|,2r_m,r_a)}{(1+\varepsilon)\|\tilde{\xi}_{m,il}\|-2r_m s(\frac{\|\tilde{\xi}_{m,il}\|}{2r_m},r_s)}\qquad b_{il}=-\frac{\partial V_{m,il}}{\partial\|\tilde{\xi}_{m,il}\|}\frac{1}{\|\tilde{\xi}_{m,il}\|}$$

- The speed command of the artificial potential field is expressed as

$$\mathbf{V}_\mathbf{i}^\mathbf{col}=-\sum_{l\ne i}b_{il}\tilde{\xi}_{m,il}$$

- The relative position tracking of the response remote control is

$$\mathbf{V}_\mathbf{i}^\mathbf{pos}=k_p\left(\mathbf{p}_\mathbf{i}-\mathbf{p}_\mathbf{i}^\mathbf{d}\right)+\frac{1}{l}\mathbf{v}_\mathbf{i}+\sum_{j=1}^{N}k_{ij}\left[\left(\mathbf{p}_\mathbf{i}-\mathbf{p}_\mathbf{i}^\mathbf{d}\right)-\left(\mathbf{p}_\mathbf{j}-\mathbf{p}_\mathbf{j}^\mathbf{d}\right)\right]$$

- Therefore, the total speed command can be expressed as

$$\mathbf{V}_\mathbf{i}=\mathbf{V}_\mathbf{i}^\mathbf{pos}+\mathbf{V}_\mathbf{i}^\mathbf{col}=k_p\left(\mathbf{p}_\mathbf{i}-\mathbf{p}_\mathbf{i}^\mathbf{d}\right)+\frac{1}{l}\mathbf{v}_\mathbf{i}+\sum_{j=1}^{N}k_{ij}\left[\left(\mathbf{p}_\mathbf{i}-\mathbf{p}_\mathbf{i}^\mathbf{d}\right)-\left(\mathbf{p}_\mathbf{j}-\mathbf{p}_\mathbf{j}^\mathbf{d}\right)\right]-\sum_{l\ne i}b_{il}\tilde{\xi}_{m,il}$$

4. Advanced examples

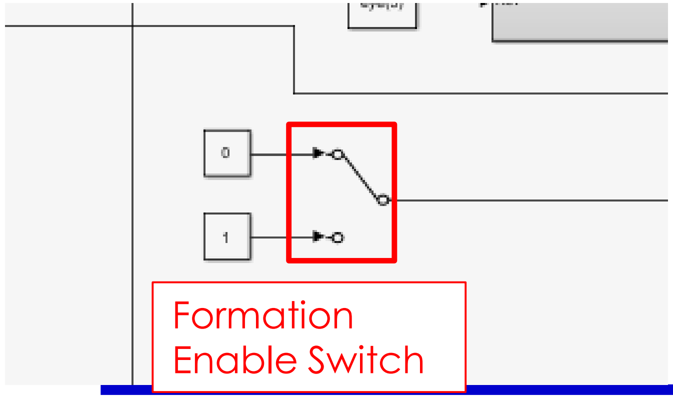
4.8 achieve of the formation --- free swarm algorithm with automatic collision avoidance-experimental results

- Double-click to run the "**SwarmFormSITL.bat**" script to create three vehicle SITL simulations (or insert three Pixhawks, double-click to run "**SwarmFormHITL.bat**" and enter the serial numbers of the three flight controllers to create three HITL simulation systems).
- Wait until CopterSim prompts that EKF initialization is complete (Pixhawk's LED lights flashing green when the hardware is in the loop, and the software is displayed on the ground station when the loop is in the loiter/Hold mode)
- Run the "**SwarmFormEarthVelCTRLUdpAPI.slx**" program at this time, you can observe the vehicle taking off slowly on RflySim3D (press the keyboard "**S**" key to view each vehicle ID)

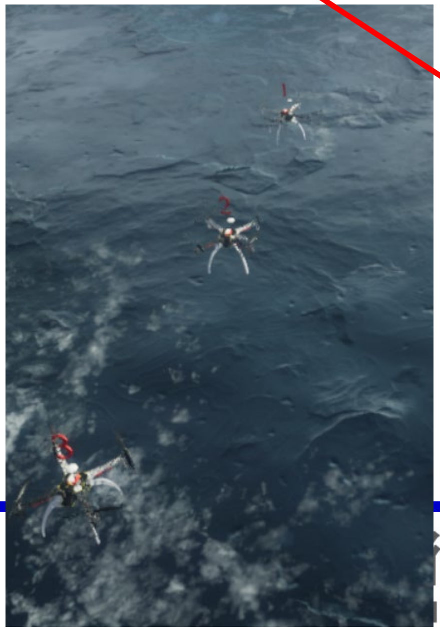
4. Advanced examples

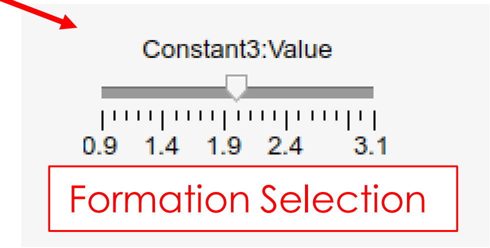
4.8 Achieve of the formation-free swarm algorithm with automatic collision avoidance-experimental results

- Turn the "**Formation Enable Switch**" to "**1**", you can see that three vehicles form a vertical "**1**" formation
- Move the "**formation selection**" slider to the middle, you can see the formation of three vehicles change from "1" to a horizontal "**---**" formation. Note that the principle here is that given the final formation "**---**", the vehicle will automatically fly over through algorithms such as artificial potential fields and avoid obstacles, without specifying the trajectory of each vehicle.

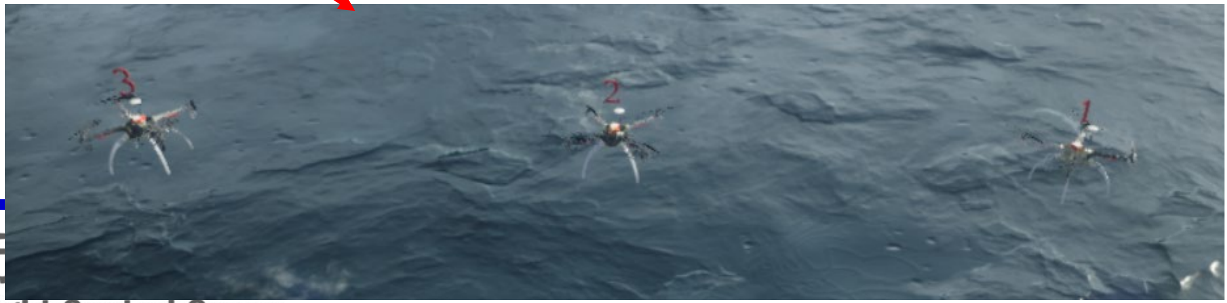


Formation Enable Switch



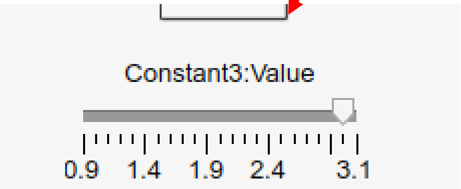


Formation Selection

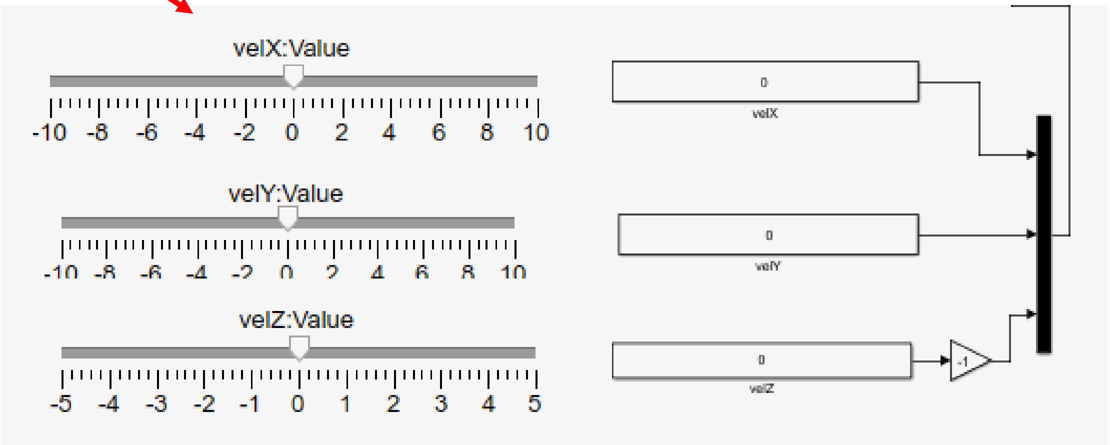


4.8 Realization of the formation-free swarm algorithm with automatic collision avoidance --- experimental results

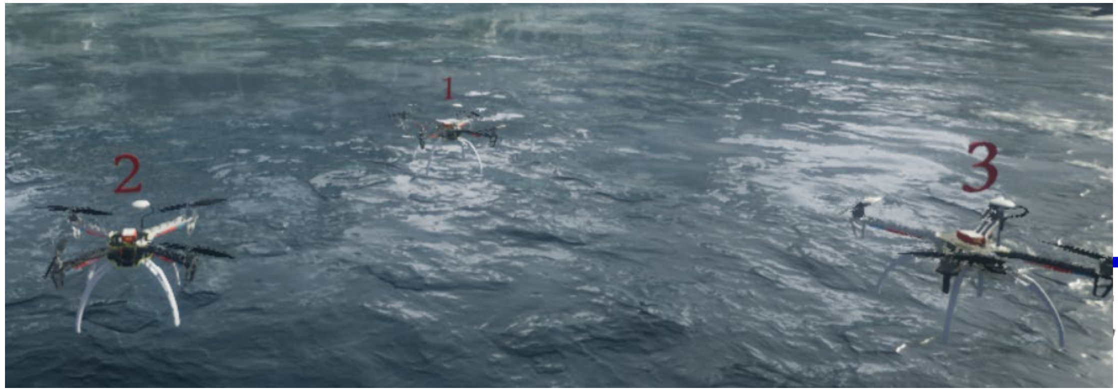
- Move the "**Formation Selection**" slider to the end, you can see the vehicle programming triangle formation.
- Drag the "**Swarm velocity input xyz slider**" to control the overall movement of the triangle formation.



Formation Selection



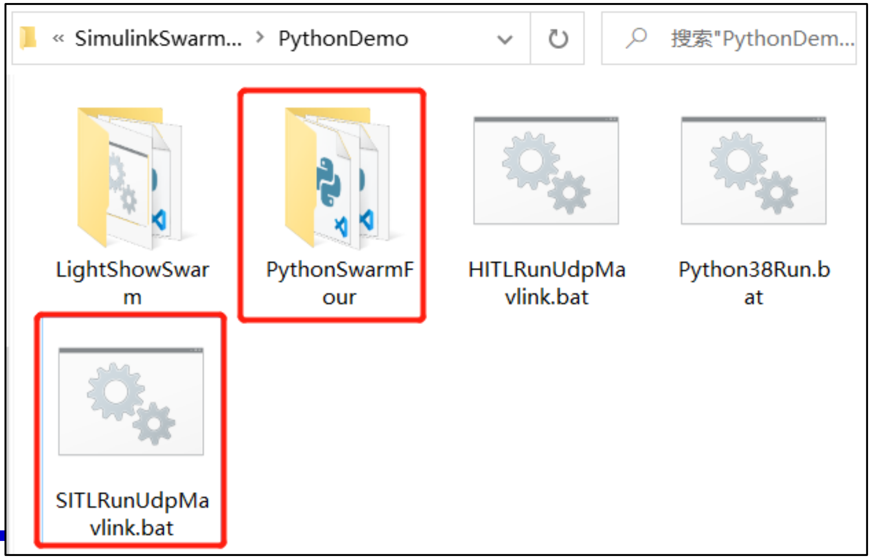
Swarm velocity input xyz slider



4. Advanced examples

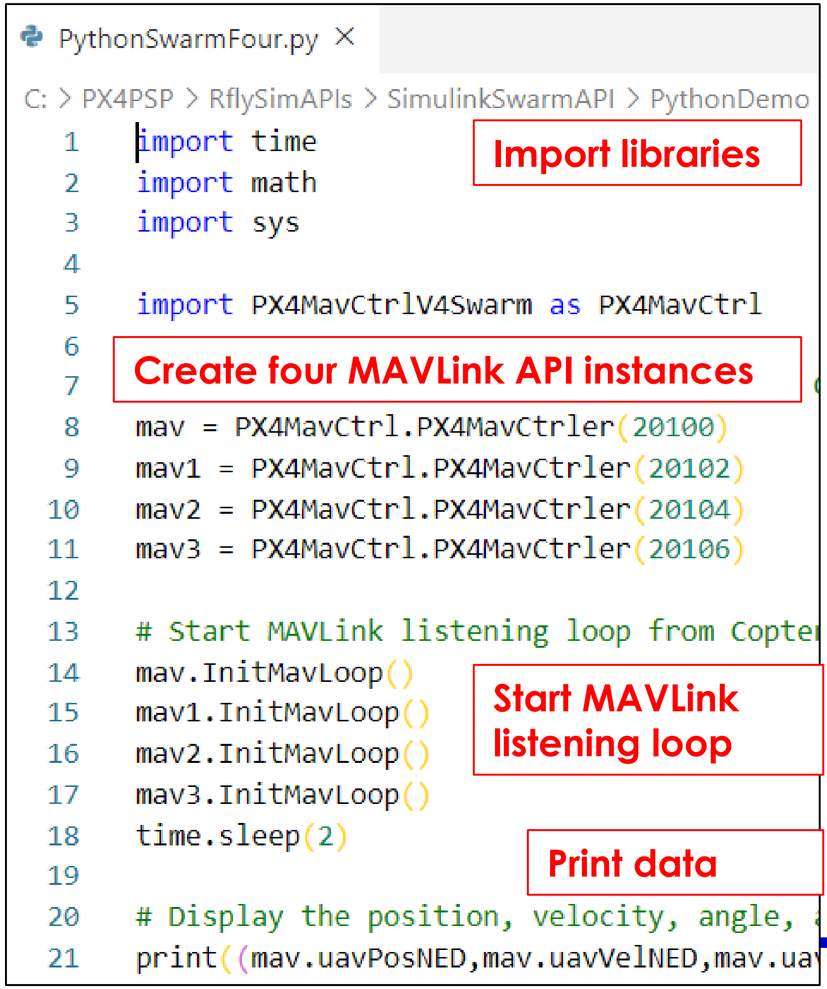
4.9 Python control swarm drone demo

- For the sample folder, see "**RflySimAPIs\SimulinkSwarmAPI\PythonDemo**". Note that VS Code needs to be installed to read and run the Python code. For the specific installation process, please refer to **Section 1.2** in **lesson 6** of the Advanced Course.
- The Python communication interface file for swarm control is "**PythonSwarmFour\PX4MavCtrlV4Swarm.py**". The basic usage method is highly consistent with the "**PX4MavCtrlV4.py**" interface in **Section 2.5** of the Advanced Course, Chapter 6, but the functions such as controlling RflySim3D for viewing angle adjustment are deleted. .
- Python control needs to directly obtain and send the vehicle's MAVLink messages, so you need to use "**HITLRunUdpMavlink.bat**" and "**SITLRunUdpMavlink.bat**“ in this folder to enable hardware or software-in-the-loop simulation. Compared with the example of Simulink swarm control, these two scripts mainly modify "**SET UDPSIMMODE=2**", that is, CopterSim enables the MAVLink forwarding mode.



4. Advanced examples

4.9 Python control swarm drone example-code analysis



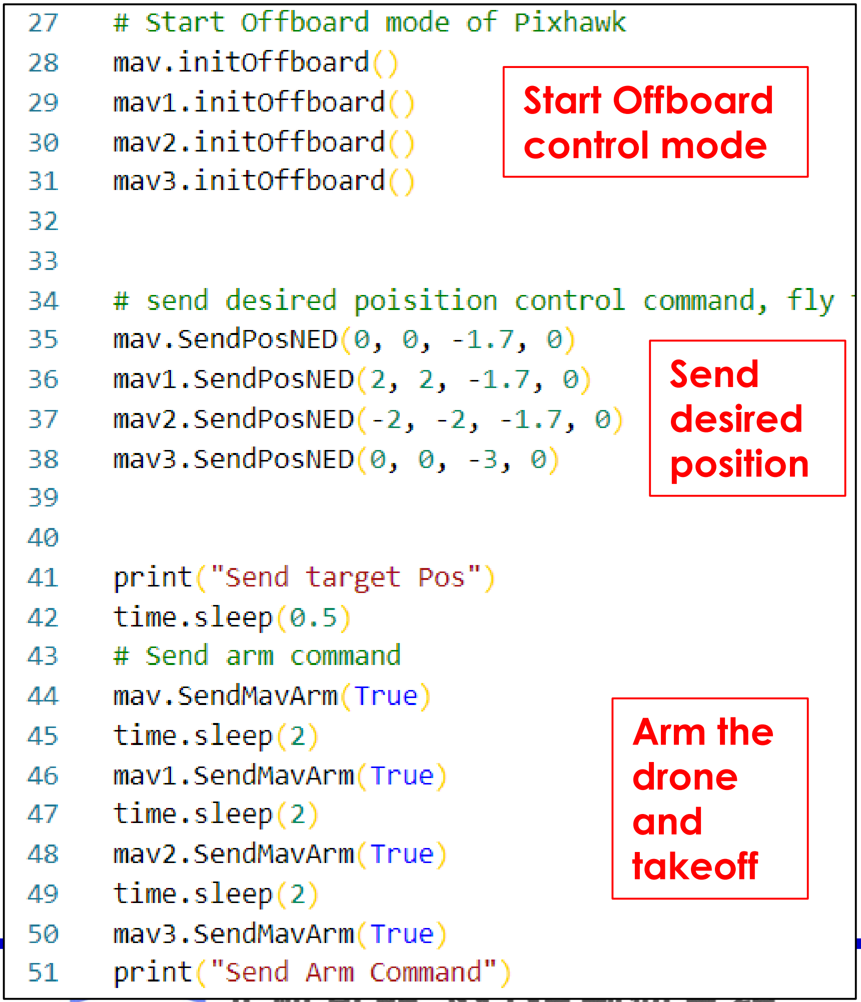
```python
import time
import math
import sys

import PX4MavCtrlV4Swarm as PX4MavCtrl


mav = PX4MavCtrl.PX4MavCtrler(20100)
mav1 = PX4MavCtrl.PX4MavCtrler(20102)
mav2 = PX4MavCtrl.PX4MavCtrler(20104)
mav3 = PX4MavCtrl.PX4MavCtrler(20106)

# Start MAVLink listening loop from Copter
mav.InitMavLoop()
mav1.InitMavLoop()
mav2.InitMavLoop()
mav3.InitMavLoop()
time.sleep(2)

# Display the position, velocity, angle,
print((mav.uavPosNED,mav.uavVelNED,mav.ua
```

Import libraries

Create four MAVLink API instances

Start MAVLink listening loop

Print data

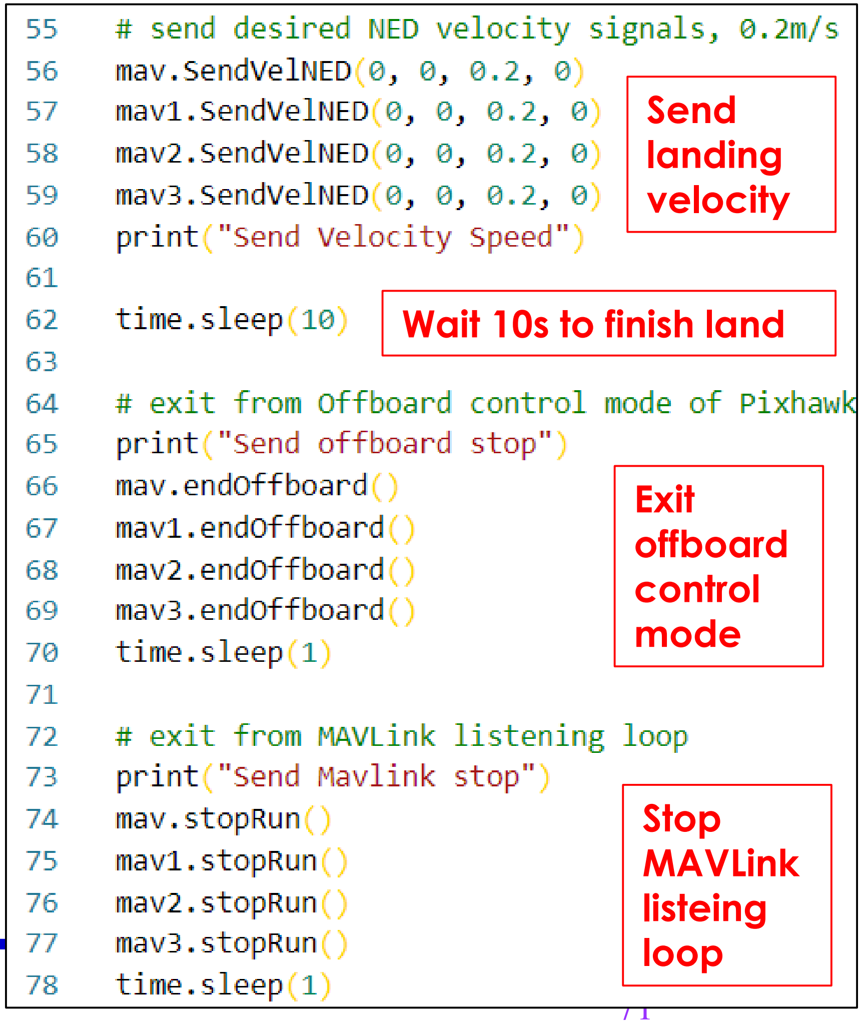


```python
# Start Offboard mode of Pixhawk
mav.initOffboard()
mav1.initOffboard()
mav2.initOffboard()
mav3.initOffboard()


# send desired poisition control command, fly
mav.SendPosNED(0, 0, -1.7, 0)
mav1.SendPosNED(2, 2, -1.7, 0)
mav2.SendPosNED(-2, -2, -1.7, 0)
mav3.SendPosNED(0, 0, -3, 0)


print("Send target Pos")
time.sleep(0.5)
# Send arm command
mav.SendMavArm(True)
time.sleep(2)
mav1.SendMavArm(True)
time.sleep(2)
mav2.SendMavArm(True)
time.sleep(2)
mav3.SendMavArm(True)
print("Send Arm Command")
```

Start Offboard control mode

Send desired position

Arm the drone and takeoff



```python
# send desired NED velocity signals, 0.2m/s
mav.SendVelNED(0, 0, 0.2, 0)
mav1.SendVelNED(0, 0, 0.2, 0)
mav2.SendVelNED(0, 0, 0.2, 0)
mav3.SendVelNED(0, 0, 0.2, 0)
print("Send Velocity Speed")

time.sleep(10)

# exit from Offboard control mode of Pixhawk
print("Send offboard stop")
mav.endOffboard()
mav1.endOffboard()
mav2.endOffboard()
mav3.endOffboard()
time.sleep(1)

# exit from MAVLink listening loop
print("Send Mavlink stop")
mav.stopRun()
mav1.stopRun()
mav2.stopRun()
mav3.stopRun()
time.sleep(1)
```

Send landing velocity

Wait 10s to finish land

Exit offboard control mode

Stop MAVLink listeing loop

4. Advanced examples

4.9 Python control swarm drone example-running effect

- The "**PythonSwarmFour**" folder shows the interface demos for controlling four vehicles with Python, among which the "**PX4MavCtrlV4Swarm.py**" file is the key communication interface module
- Double-click the "**PythonSwarmFour.bat**" script to start the software-in-the-loop (MAVLink data protocol is enabled) of the four vehicles, and run "**PythonSwarmFour.py**" to use the interface to obtain the data of the four vehicles and control their take-off and landing.

Thanks