

Multicopter Design and Control Practice Experiments

RflySim Advanced Courses Lesson 03: External Control Interface

Dr. Xunhua Dai, Associate Professor, School of Computer Science and Engineering, Central South University, China;

Email: <u>dai.xh@csu.edu.cn</u>;

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





- 1. Configuration of software & hardware
- 2. MAVLink communication analysis
- 3. PX4 official controller communication
- Path of demo source code of this lesson: " RflySimAPIs\SimulinkC ontrolAPI"
- 4. Code generation controller communication
- 5. Summary





Note: The examples in this section need to use the official firmware of PX4, any version is fine, the latest firmware is selected here

1.1 Pixhawk hardware configuration (1) — online firmware download

1) Open the QGC ground station software;

2) As shown in the figure on the right, click the **gear icon** in the toolbar to enter the settings page, and then click the "**Firmware**" label to enter the firmware burning page;

3) Connect the Pixhawk autopilot to the computer with a USB cable. The software will automatically recognize the Pixhawk hardware, as shown in the lower right figure, the firmware configuration window will pop up on the right side of the interface, check "**PX4** ***", and then click "**OK**", QGC will start to download automatically (Internet connection is required; if no internet, please refer to the next page to use the local firmware file) and

install the latest PX4 firmware into Pixhawk;



QGroundCor	itrol	-		×
🕲 <mark>🌼</mark> 💐	e 🖉 🗟	Waiting For Vehicl	.e Connec	tion
	Firmware Setup	Firmware Setup	Cancel	0k
Summary		Detected Pixhawk board. from the following flig	You can se ht stacks:	elect
Firmware	QGroundControl can upgrade the firmware on Flow Smart Cameras.	Flight Stack		_
	All QGroundcontrol connections to vehicles upgrade.	●PX4 Pro Stable Relea	ıse v1.11.1	
	Please unplug your Pixhawk and/or Radi Plug in your device via USB to start f	●ArduPilot		
	Plug in your device via USB to start f Found device: PX4 FMU V2 Connected to bootloader: Version: 5 Board ID: 255 Flash size: 2080768	Advanced settin	gs ——	

16/36 년) 크튼



1.2 Pixhawk hardware configuration (2) — offline firmware download (when the firmware cannot be downloaded online due to the network)

1) Open the QGC ground station software;

2) Click the "**Firmware**" tab, and connect the Pixhawk autopilot with the USB data cable. The ground station will automatically detect the autopilot

3) Check the "advanced settings" checkbox;_

4) Click triangle icon besides the "Standard Version (stable)" tab – "Custom firmware file", then click "OK";

5) In the file selection page that pops up, if you use Pixhawk1 flight control, select "RflySimAPIs \ FlightControlExpCourse \ code \ e0 \4.PX4Firmwares \ px4fmuv3_default1.10.1Stable.px4", if you use Pixhawk 4, select fmu-v5 firmware





1. Configuration of software C:VPX4PSP

- 1.3 Pixhawk hardware configuration (3) offline firmware generation
- Re-run the "OnekeyScript.p" script in MATLAB;
- Enter "px4_fmu-v3_default" on the second line (here for Pixhawk 1, please select the compilation command according to your flight control hardware)
- As shown in the figure on the right, set the 9th and 10th items to "yes" and "no" respectively, and keep the other options as default. Click the "OK" button to compile the official PX4 firmware without blocking the output of the PX4 itself.
- Run the "PX4Upload" command in MATLAB to pop up the firmware burning page. At this time, insert the Pixhawk flight controller to burn the official firmware.

Note: Try this method only when the first two methods are invalid; the following figure is for the advanced version of RflySim, if it is the basic version of RflySim, please use the default firmware and compiler version.

4	Toolbox	one-key	instal	lation	script
*	1001008	One-key	ilistai	auon	script

(1) Software package installation directory

(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_fmu-v3_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) PX4 firmware compiling toolchain (1: Win10WSL[suitable for all versions],2: Msys2[suitable for <= PX4-1.8], 3: Cygwin[for >=PX4-1.8])

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

no

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

no

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

no

ves

4 no

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

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

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



飞行控



1. Configuration of software & hardware

1.4 Pixhawk HIL simulation mode

- After the firmware is uploaded, the autopilot will automatically restart and re-connect to the QGC; at this time, as shown on the right, enter the "Airframe" tab, select the airframe as "HIL Quadcopter X", and then click on the " Apply and Restart" button, the autopilot will restart automatically at this time;
- After restarting, QGC will automatically look for the serial port and connect to Pixhawk. At this time, check each configuration page to ensure that Pixhawk enters hardware-in-theloop (HIL) simulation mode.
- After finishing the radio control (RC) transmitter calibration and mode setting, unplug the RC receiver on Pixhawk, this course does not need to connect the RC receiver









1. Configuration of

1.6 PX4 HITL simulation test

- If you use RflySim Advanced Edition, please insert Pixhawk, and then directly run the HITLRun shortcut on the desktop, enter the serial port number in the popup window, and press Enter to start the hardware-in-the-loop(HIL) simulation system
- If you use RflySim basic version, please insert Pixhawk, open the CopterSim software, select the flight control serial port in the "**Select Pixhawk Com.**" dropdown box, and click "**Start Simulation**", and then manually open QGC and 3DDisplay
- In QGC, click the paper plane icon -Takeoff - Slide to confirm, you can see that the drone takes off automatically in the view, indicating that the HIL configuration is correct.

Note: no matter SIL or HIL simulation, you should wait until CopterSim show message "** EKF initialization finished" on the UI, then you can use QGC/Simulink/Python to control the drone.





1. Configuration of software & hardw

1.7 PX4 SITL simulation (Pixhawk hardware is not required)

- This function is only available in the advanced version of RflySim (corresponding to the setting in Section 1.5)
- Double-click the "SITLRun" shortcut on the desktop and enter the number "1" to start one vehicle SIL simulation system
- Same as the previous page, control the drone to takeoff in QGC. If it can takeoff automatically, it means that the platform is configured correctly.
- **Principle**: PX4 SITL is a real-time operating system that simulates Pixhawk in the Ubuntu environment of Win10WSL, thereby running a complete PX4 controller, and connecting with CopterSim through the network to realize the interaction of sensors/control commands, forming a control simulation closed-loop system, and inserting Pixhawk HITL by hardware has the same effect
- Note: Under PX4 SITL simulation, QGC can also be used to configure participation and obtain log files (stored in the installation directory: Firmware \build \px4_sitl_default \ instance_1)



>STERun Please input UAV swarm number:1 Start QGroundControl Kill all CopterSims Starting PX4 Build [1/1] Generating ../../logs killing running instances starting instance 1 in /mnt/c/PX4PSPFul1/Firmware/ PX4 instances start finished Press any key to exit



SimulinkControlAPI > VS2017Installer

vs_community2 017.exe

1.8 Install Visual Studio 2017 (other versions can also be used, MATLAB can recognize it)

- The Visual Studio compiler is needed in many places in subsequent courses, such as MATLAB
- The use of S-Function Builder module, Simulink automatically generates C/C++ model code, etc.
- It is recommended to install Visual Studio 2017. The online installation steps (internet required) are as follows:
- Double-click "RflySimAPIs \ SimulinkControlAPI \ VS2017Installer \ vs_community2017.exe"
- This course content only needs to check the "Desktop development with C++" on the right.
- Note: If you want to use UE4 C++ development in the future, you can also check the latest Window 10 SDK in the "Installation details" on the right; then click the "Individual components" tab and check .NET 4.7.2 (or the latest version)
- and the corresponding pack package.
 Click install again.



воаа кенable Flight Control Group



1. Configuration of software & hardware

1.9 Configure C++ Compiler for MATLAB

- Enter the command "mex setup" in the MATLAB command line window
- Generally speaking, the 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 2013/2015/2017

_	
Co	ommand Window
	>> mex -setup
	MEX configured to use ['Microsoft Visual C++ 2017 (C)'] for C language compilation.
	Warning: The MATLAB 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:
	<pre>Microsoft Visual C++ 2013 (C) mex -setup:D:\MATLAB\R2017b\bin\win64\mexopts\msvc2</pre>
1	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:
	<u>mex -setup C++</u>
	mex -setup FORTRAN



 $f_{x} >>$



- 1. Configuration of software & hardware
- 2. MAVLink communication analysis
- 3. PX4 official controller communication
- Path of demo source code of this section: "RflySimAPIs\SimulinkCo ntrolAPI\MavlinkDemo"
- 4. Code generation controller communication

5. Summary





2.1 MAVLink (Micro Air Vehicle Link)

- It is a communication protocol for small unmanned vehicles, first released in 2009. This protocol is widely used in the communication between Ground Control Station (GCS) and Unmanned vehicles, as well as in the internal communication between the onboard computer and the Pixhawk. The protocol is defined in the form of a message library rules for parameter transmission. The MAVLink protocol supports a variety of vehicles such as unmanned fixed-wing aircraft, unmanned rotorcraft, and unmanned vehicles.
- Official use file website:
 https://mavlink.io/en/messages/common.html
- MAVLink source code:
 https://github.com/mavlink/mavlink
- QGroundControl ground station source code based on MAVLink: <u>https://github.com/mavlink/qgroundcontrol</u>





2.2 The essence of MAVLink

- It is the encapsulation and analysis protocol of byte stream
- The packet format of MAVLink 1 shown as follow:

MAVLink v1 Frame (8 - 263 bytes)



• The packet format of MAVLink 2 shown as follow:

MAVLink v2 Frame (11 - 279)

STX	LEN	INC FLAGS	CMP FLAGS	SEQ	SYS ID	COMP ID	MSG ID (3 bytes)	PAYLOAD (0 - 255 bytes)	CHECKSUM (2 bytes)	SIGNATURE (13 bytes)	
	北航可靠飞行控制研究组 BUAA Beliable Elight Control Group										



2. MAVLink cor



PAYLOAD

STX LEN SEQ SYS COMP MSG

CKA CKB

- Definition of bytes in the MAVLink 1 package.
- **STX** \rightarrow Packet start sign
- LEN \rightarrow Payload Length
- **SEQ** \rightarrow Packet sequence
- SYS \rightarrow System ID
- COMP → Component ID
- **MSG** \rightarrow Message ID
- PAYLOAD → Data
- **CKA** \rightarrow Checksum A
- **CKB** \rightarrow Checksum B

Byte	Content	Value	Explanation
Index			
0	Packet start	v1.0: 0xFE	Indicates the start of a new packet.
	sign	(v0.9: 0x55)	
1	Payload length	0 - 255	Indicates length of the following payload.
2	Packet	0 - 255	Each component counts up his send sequence. Allows to detect
	sequence		packet loss
3	System ID	1 - 255	ID of the SENDING system. Allows to differentiate different MAVs
			on the same network.
4	Component ID	0 - 255	ID of the SENDING component. Allows to differentiate different
			components of the same system, e.g. the IMU and the autopilot.
5	Message ID	0 - 255	ID of the message - the id defines what the payload "means" and
			how it should be correctly decoded.
6 to	Data	(0 - 255) bytes	Data of the message, depends on the message id.
(n+6)			
(n+7) to	Checksum (low	ITU X.25/SAE AS	6-4 hash, excluding packet start sign, so bytes 1(n+6) Note:
(n+8)	byte, high byte)	The checksum a	Iso includes MAVLINK_CRC_EXTRA (Number computed from
		message fields.	Protects the packet from decoding a different version of the same
		packet but with d	lifferent variables).





2.3 Analysis principle

- Read: All byte streams are stored in the buffer, and the byte data in the buffer is read sequentially. When the STX flag bit is encountered (the flag bit of MAVLink v1 is **0xFE**, the flag bit of v2 is **0xFD**), it starts to recognize a message until the end of the message. If the message verification is correct, send the message to the handler
- Send: follow the previous page to convert the message into a byte stream







2.4 receive analyze source code analysis

Given a byte stream buffer of a certain length, the length simply called length, with the following script analysis, the **onMavLinkMessage** function will execute every time a MAVLink packet is parsed.

```
for(int i = 0; i < length; ++i){
```

msgReceived = mavlink_parse_char(MAVLINK_COMM_1, (uint8_t)buffer[i], &message, &status);

```
if(msgReceived){
```

```
emit onMavLinkMessage(message);
```

```
}
```

Among them:

void onMavLinkMessage(mavlink_message_t message);

It is the processing function after a MAVLink message package is obtained. Users need to identify the purpose of the current package (heartbeat package, GPS location, posture, etc.) according to its ID, and extract the important data.





2.4 receive analyze source code analysis

The analysis function is implemented as follows, jump to the corresponding _decode function according to message.msgid, and decode the data void onMavLinkMessage(mavlink_message_t message){ switch (message.msgid){ case MAVLINK_MSG_ID_GLOBAL_POSITION_INT: { mavlink global position int t gp; mavlink_msg_global_position_int_decode(&message, &gp); outHilData.time_boot_ms = m_LastReceiveMavMsg; outHilData.GpsPos[0]=gp.lat; outHilData.GpsPos[1]=gp.lon; outHilData.GpsPos[2]=gp.alt; outHilData.relative_alt = gp.relative_alt; outHilData.GpsVel[0]=gp.vx; outHilData.GpsVel[1]=gp.vy; outHilData.GpsVel[2]=gp.vz; outHilData.hdg = gp.hdg; break:



2.5 Send source code analysis — send a MAVLink_hil_actuator_controls message void sendHILCtrlMessage(uint8_t modes, uint64_t flags, float ctrl[])

```
mavlink_hil_actuator_controls_t hilctrl;
hilctrl.mode = modes:
hilctrl.flags = flags;
for(int i=0;i<16;i++){
  hilctrl.controls[i]=ctrl[i];
mavlink_message_t mess;
mavlink_msg_hil_actuator_controls_encode(SystemID, TargetCompID, & mess, & hilctrl);
char buffer[500];
memset(buffer,0,500);
unsigned int length = mavlink_msg_to_send_buffer((uint8_t*)buffer, & mess);
udp.writeDatagram(buffer,length);//send the buffer out through USP or interface is ok
```





2.6 MAVLink ID list of message package

• <u>https://mavlink.io/en/messages/common.html</u>

HEARTBEAT (#0) Heartbeat package, ID = 0



20

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

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



2.7 QGC ground station view MAVLink messages

On the MAVLink Inspector page of QGC, you can browse all the MAVLink packages sent by Pixhawk, check the frequency and specific values of each package

QGroundControl							
🕼 🍫 🍪	Ł	- 🛃 🔏 📥 💷] N/	A 🔷 未解锁 -	Manual 🗸		
分析	查看	实时 MAVLink 消息。				1	
	1	AUTOPILOT_VERSION	0.0Hz	信息: 组件:	HEARTBEAT (0) 1.0Hz 1	J	
🥥 GeoTag Images	1	COMMAND_ACK	$0.0 \mathrm{Hz}$	计数:	25		
≫ MAVLink Console	1	EXTENDED_SYS_STATE	2.0Hz	Name		Value 2	
₩ MAVLink Inspector	1	HEARTBEAT	1.0Hz	autopilot base mode		12 113	
	1	MISSION_COUNT	0.0Hz	custom_mode		65536 3	
	1	PARAM_VALUE	0.0Hz	mavlink_version		3	
	1	PING	1.0Hz				
	1	PROTOCOL_VERSION	0.0Hz				
	1	SYSTEM_TIME	1.0 Hz				
	1	SYS_STATUS	1.0Hz				
	1	TIMESYNC	10.0Hz				





2.8 Source code of MAVLink 2

Open folder

'RflySimAPIs\SimulinkControlAPI\MavlinkDemo\mavlink\v2.0\common'. You can see C++ source code of MAVLink, including all definition of all message package.

			> mavlink > v2.0 > common	~ O	♀ 搜索"common"
IAPI > MavlinkDemo > mavlink > v2.0 >	× ت	J		修改日期	类型
名称 ^	修改日期		C common.h	2019/6/29 21:55	C Header 源文件
📕 ardupilotmega	2020/8/9 16:31		C mavlink.h	2019/6/29 21:55	C Header 源文件
ASLUAV	2020/8/9 16:31		mavlink_msg_actuator_control_targe	2019/6/29 21:55	C Header 源文件
📕 autoquad	2020/8/9 16:31		C mavlink_msg_adsb_vehicle.h	2019/6/29 21:55	C Header 源文件
Common	2020/8/9 16:31		mavlink_msg_altitude.h	2019/6/29 21:55	C Header 源文件
📕 matrixpilot	2020/8/9 16:31		mavlink_msg_att_pos_mocap.h	2019/6/29 21:55	C Header 源文件
message_definitions	2020/8/9 16:31		mavlink_msg_attitude.h	2019/6/29 21:55	C Header 源文件
📕 minimal	2020/8/9 16:31		C mavlink_msg_attitude_quaternion.h	2019/6/29 21:55	C Header 源文件
📕 slugs	2020/8/9 16:31		mavlink_msg_attitude_quaternion_co	2019/6/29 21:55	C Header 源文件
📕 standard	2020/8/9 16:31		mavlink_msg_attitude_target.h	2019/6/29 21:55	C Header 源文件
📕 test	2020/8/9 16:31	ίĒ	mavlink_msg_auth_key.h	2019/6/29 21:55	C Header 源文件
📕 uAvionix	2020/8/9 16:31	R	mavlink_msg_autopilot_version.h	2019/6/29 21:55	C Header 源文件



2.9 Simulink Encapsulation and Analysis Implementation of MAVLink Protocol Open demo

 $`` RflySimAPIs \ SimulinkControlAPI \ MavlinkDemo \ MavlinkCodeDecode.slx"$





- After clicking Run, we can see that we encapsulate the data into byte stream data (uint8 byte stream) and len (byte stream length) in Simulink, and then pass a parsing function to parse the byte stream into sending data.
- This example is implemented by **S-Function Builder**, it will automatically call the MAVLink header file when it is running, and compile it into the **.c/.mexw64/.tlc** files shown on the right
- This demo can teach you how to call external C/C++ header files in Simulink to implement your own algorithms.

S-Eunction Builder: MaylinkCodeDecode/S-Eunction Builder1

	• S Function Builden IV	aviinkeedebeeede, 5 Taneaon b	anderi		
	Parameters				MavlinkSerialInputOutput.m
🗢 🧵 er er lårk	S-function name: n	navlink_msg_receiver		Build	MavSfunTest Arm.slx
🗄 🚊 mavlink	S-function parameters-				MaySfunTest_controlBC sly
🛀 MavCmdDefines.m	Name	Data type	Value		
MaylinkCodeDecode sly					MavSfunTest SerialCom.six
					🚺 rtwmakecfg.m
🔠 MavlinkSerialInputOutput.m				X	SFR maylink msg receiver SFR ma
MaySfunTest Arm.slx	Port/Parameter St	art Outputs Derivativ	ves Update Termina	ate Build Info	GED_mandlah_man_an_dan_CED_mat
May Char Test sentre IDC shy	data	Initialization	Data Properties	Libraries	The second secon
WavStuniest_controlkc.six	en Ente	r any library/object or source file	s used by the S-function. Then, s	pecify any necessary inc	
🔊 MavSfunTest SerialCom.slx	Output Ports	nter the external function declara ate methods	tions. These functions can be cal	illed in the Outputs, Deri	
	→ timeStamp ↓ mode	rary/Object/Source files (one per	line) Include files and exten	rnal function declarat	
			#define inlineinline	e	24
			#include . \mavlink\v2.	. U\common\maviink.h	

mavlink slpri

MavCmdDefines.m

mavlink msg receiver.c

mavlink msg receiver.tlc

mavlink msg sender.c

mavlink msg sender.tlc

MavlinkCodeDecode.slx

MavlinkCodeDecode.slxc

mavlink msg receiver.mexw64

mavlink msg receiver wrapper.c

mavlink msg sender.mexw64

mavlink msg sender wrapper.c

🎦 MavlinkCodeDecode * - Simulink

<u>File Edit View Display Diagram Simulation Analysis Code Tools Help</u>



2.10 Simulink S-Function program method

 Open the .slx file and drag in an S-Function Builder module from Simulink-User-Defined Functions, doubleclick it to get the picture on the right

Simulink Library Browser

💠 🧼 serial 🗸 🗸 🗸 🚽 🔞 🗸 🛏 🔇

Simulink/User-Defined Functions



 Below picture shows how to use this module to generate the MAVLink message mentioned above: MAVLINK_MSG_ID_HIL_ACTUATOR_CONTROLS





S-Function Build	ler: MavlinkCodeI	Decode/S-Func	tion Builder			-		×		
rameters function name:								Build		
-function parame	ters									
lame		Data type			Value					
								x		
ort/Parameter Input Ports ↓ u0 Output Ports	Start Our Initializ	tputs De ation	rivatives [Update Data Proper	e Termina rties	te	Build Librarie	l Info es		
i output Ports └-∲ y0 ∲ Parameters	The S-Function input ports, out output ports ca 2-D signals. This have the block	Builder block c put ports, and n propagate Si s block also sup generate a TLC	reates a wraj a variable nu mulink built- ports discre file to be us	pper C-ME imber of sc in data typ ite and con ed with Sin	X S-function from alar, vector, or r es, fixed-point of tinuous states of hulink Coder for	n your natrix latatyp f type code q	supplie parame pes, com real. You generat	d C coc ters. Thi iplex, fr u can or ion.		
	S-function settings									
	Number of disc Discrete states	rete states: IC:	0	Sam Sam	ple mode: ple time value:	Inher	ited	~		
	Number of con	tinuous states:	0	Nun	nber of PWorks:	0				
						Cance	el	Help		

😽 S-Function Build	ler: MavlinkCodeD	ecode/S-Function Bu	uilder	_	- 🗆 ×					
Parameters										
S-function name:	mavlink_msg_	sender			Build	tion c	inalysis			
S-function parame	ters						1101 y 515			
Name		Data type		Value						
						• No	ame the mo	odule and	set the inpu	t and
					X	OL	itput paran	neter name	es, dimensio	ns, and
						de	nta tupos o	n the Data		
Port/Parameter —	l Start Out	touto Dorivativ	uas Undata	Torminato	Puild Info		ind types of		Type page	
🖃 🔙 Input Ports	Initializ	ation	Data Proper	ties	Libraries					
timeStamp	Description									
- ♥ mode	Use the Add and	Delete buttons to a	dd/remove ports	and parameters to	the S-function. Use					
♦ controls	to configure the the data type an	data type, dimension d complexity of each	ns, complexity an 1 parameter.	d frameness of eac	h S-function port a	Start	Outputs	Derivatives	Update	Terminate
	Port and Parame	eter properties	•				Initialization		Data Properties	
Parameters	Input port	s Output ports Para	ameters Data typ	e attributes		Descrip	tion———			
	Port name	Dimensi Rows	Columns	Complex Bus	Bus Name	lise the	Add and Delete	buttons to add/	remove ports and	Inarameters

✓ off

✓ off

✓ off

✓ off

Terminate

Complex... Bus

~ off

✓ off

real

real

 \sim

 \sim

 \sim

 \sim

Build Info

Bus Name

gh

 \sim

 \sim

Libraries

real

real

real

real

Undate

Data Properties

Use the Add and Delete buttons to add/remove ports and parameters to the S-function. Use

to configure the data type, dimensions, complexity and frameness of each S-function port a

Columns

timeStamp 1-D

mode

controls

flags

Start

<u>i</u>+

×

1

T

data

len

Description-

1-D

1-D

1-D

Outputs

Initialization

Port and Parameter properties-

±

Ŧ

Port/Parameter-

🖃 🚺 Input Ports

🔷 mode

🔷 flags

🔷 data

Parameters

🔶 len

controls

😼 Output Ports

🗣 timeStamp

× 1

× 1

~ 1

the data type and complexity of each parameter.

Port name Dimensi... Rows

1-D

1-D

~ 16

Derivatives

Input ports Output ports Parameters Data type attributes

~ 300

 ~ 1

Use the Add and Delete buttons to add/remove ports and parameters to configure the data type, dimensions, complexity and frameness of each the data type and complexity of each parameter.

Port and Parameter properties-

Input ports	Output ports	Parameter	rs	Data ty	pe attr	ibutes
Port	Data type			Nord I	Sig	Fraction
In_1: timeStamp	uint32	\sim	1:		\checkmark	3
In_2: mode	uint8	\sim	8		\sim	3
In_3: flags	uint32	\sim	8		\sim	3
In_4: controls	single	\sim	8		\sim	3
Out_1: data	uint8	\sim	1:		\sim	3
Out_2: len	uint16	\sim	8		\sim	3



- Import the MAVLink header file
- Enter the "Libraries" tab, and add the following code in the "includes" box
- #define inline __inline
- #include ".\mavlink\v2.0\common\mavlink.h"

Start	Outputs	Derivatives	Update	Terminate	Build Info
	Initialization		Data Propertie	s	Libraries
Enter any or enter t Update m	library/object he external fun ethods.	or source files use ction declarations	d by the S-func s. These function	tion. Then, specify ns can be called in	any necessary in the Outputs, Der
Library/C	bject/Source	files (one per line)		es and external fur	nction declarat ·
			#define in #include "	lineinline .\mavlink\v2.0\comm	on\mavlink.h"
					1



- In the Outputs tab, add the C/C++ code that obtains the input data and packs it into a MAVLink message, and puts it in the output ports data and len. Where data is the uint8 matrix, and len is the effective length of the data.
- Note: Simulink S-function signals have no concept of scalar, and all input/output signals are vectors. Therefore, although an output "len" is an one-dimensional scalar, the assignment statement of "len=***" is wrong, so use "len[0]=***" instead.

```
Initialization Data Properties Libraries Start Outputs Perivatives Update Terminate Build Info
Code description
Enter your C-code or call your algorithm. If available, discrete and continuous states should be referenced
as xD[0]...xD[n], xC[0]...xC[n] respectively. Input ports, output ports and parameters should be referenced
using symbols specified in Data Properties. These references appear directly in the generated S-function.
uint8_T out_msg[MAVLINK_MAX_PACKET_LEN]={0};
mavlink_message_t out_msg_m1;
uint8_T compid = 0;
mavlink_msg_hil_actuator_controls_pack(sysid, compid, &out_msg_m1, timeStamp[0], controls, mode[0], flags[0]);
uint16_t MavlinkMessageSizes = mavlink_msg_to_send_buffer(out_msg, &out_msg_m1);
len[0] = MavlinkMessageSizes;
for(int i=0;i<MavlinkMessageSizes;i++){ data[i]=out_msg[i]; }</pre>
```



• Check the option to generate **TLC** and **MEX**-file, and then click the compile button, you can get the file that can be called by Simulink as shown on the right.

🐱 S-Function Build	ler: MavlinkCodeDecode/S-Function E	Builder	- □ >	×			
Parameters		1					
S-function name:	mavlink_msg_sender	J	3 Build				
S-function parame	ters				🕀 📒 mavlink		
Name	Data type Value		e		C maylink r	nsa receiver.c	
					M maylink r	nsg_receiver.mexw64	
D / D				×	B maylink r	nsa receiver mexw64 pdf	6
Port/Parameter	Initialization	Data Properties	Libraries		maylink r	nsg_receiver tic	- I
timeStamp	Start Outputs Derivati	ives Update	Terminate Build Inid)	a maylink_r	nsg_receiver.wrapper.c	
∳ mode	Compliation diagnostics					nsg_receiver_wrapper.c	
∳ flags	### 'mavlink msg sender.c' created successfu	illy					
Output Ports	### <u>mavlink msg_sender wrapper.c'</u> created s ### <u>'mavlink_msg_sender.tlc'</u> created successf	fully 4			rtwmaked	rtg.m	
data	### S-function 'mavlink_msg_sender.mexw64' (created successfully			H SFB_mav	link_msg_receiverSFB.m	nat
∳ len							
Parameters	Build options						
	Show compile steps Ge	enerate wrapper TLC	Enable access to SimStru	u			
	Create a debuggable ME	ave code only	L Enable support for cove	r			
2		we code only	Enable support for Desig	9		20	
			Close Help			29	

Help



- Let's build a decoding module for MAVLink messages
- Name it "mavlink_msg_receiver"
- Input and output ports, completely opposite to the previous module

ľ		21	ر ا	· · · · ·				
I	Por	t and Parame	ter properti	es——				
	-	Input ports	Output po	rts Paramet	ers Data ty	pe attributes		
	\times	Port name	Dimensi	Rows	Columns	Complex	Bus	
		data	1-D 🗠	300		real ~	off	\sim
	:=::	len	1-D 🗠	1		real \sim	off	\sim
	Ŧ							

r Port	and Parame	ter propertie	es —				
	Input ports	Output po	rts Paramete	rs Data typ	e attributes		
	Port name	Dimensi	Rows	Columns	Complex	Bus	
	timeStamp	1-D 🗠	1		real 🛛 🗠	off	2
: : :::	mode	1-D 🗠	1		real ~	off	~
Ŧ	flags	1-D 🗸	1		real ~	off	~
	controls	1-D 🗠	16		real ~	off	~

🐱 S-Function Build	der: M	lavlink	CodeDe	code/	S-Func	tion Build	ler	1			_		×
Parameters													
S-function name:	m	navlink	c_msg_re	ceiver	1								Build
S-function parame	ters-												
Name			D)ata ty	pe				Valu	e			
													x
Port/Parameter	Sta	art	Outpu	its	De	rivatives		Update	-	Terminate		Buil	d Info
🖃 🚂 Input Ports		In	nitializatio	on			C	Data Prope	rties		L	ibrari	es
🗝 🔶 data	Des	Description											
🛶 🔶 len	Use	the Ad	dd and D	eletel	button	s to add/	′rer	nove ports	and	oarameters t	o the :	S-fun	ction. Us
🗎 🚺 Output Ports	to co	onfigu	re the da	ata typ	e, dim	ensions,	cor	nplexity ar	nd frar	meness of ea	ch S-fi	uncti	on port a
🔷 timeStamp	the d	the data type and complexity of each parameter.											
• mode	Port	Port and Parameter properties											
flags		Inpu	t ports (Outou	t norts	Parame	oter	s Data tvr	oe atti	ributes			
			e porto -	outpu	c porto	, rarante			.				1
	\times	Port		Data	type			Word I	Sig	Fraction le.	Slo	pe	Bias
	+	In_1: (data	uint8			\sim	8	\sim	3	2^-3		0
		In_2: 1	len	uint1	6		\sim	8	\sim	3	2^-3		0
		0ut_1:	timeStamp	uint3	2		~	12	\sim	3	2~-3		0
		0ut_2:	mode	uint8	2		~	8		3	2 -3		0
		Out_3:	flags	uint3	2		~	0		3	2 -3		0
		Jut_4:	CONTROLS	single	:			0		9	2 -3		V

北航可靠飞行控制研究组



2. MAVLink commu

- As shown in the figure on the right, set the decoded byte stream in the Outputs tab and parse out the code of the MAVLink message.
- In the same way, on the library file page, import the MAVLink library file
- After setting the compilation options, click the "**build**" button to check whether the tlc and **mex** files can be generated correctly.

```
Initialization Data Properties Libraries Start Outputs Derivatives Update Terminate Build Info
Code description-
Enter your C-code or call your algorithm. If available, discrete and continuous states should be referenced
as xD[0]...xD[n], xC[0]...xC[n] respectively. Input ports, output ports and parameters should be referenced
using symbols specified in Data Properties. These references appear directly in the generated S-function.
mavlink message t message;
mavlink_status_t status;
bool msgReceived = false;
for(int i=0:i<1en[0]:i++) {
    msgReceived = mavlink parse char(MAVLINK COMM 1, (uint8 t)data[i], &message, &status);
    if (msgReceived) {
         switch(message.msgid)
             case MAVLINK_MSG_ID_HIL_ACTUATOR_CONTROLS: {
                  mavlink_hil_actuator_controls_t hil_actuator_control;
                  mavlink_msg_hil_actuator_controls_decode(&message, &hil_actuator_control);
                  timeStamp[0] = hi1_actuator_control.time_usec;
                  mode[0] = hi1_actuator_control.mode;
                 flags[0] = hi1_actuator_control.flags;
                  for(int i=0:i<16:i++) {</pre>
                      controls[i]=hi1_actuator_control.controls[i];
                  break:
              default:{
                  break:
         msgReceived = false;
```





2.11 Send arm command to the flight controller via Simulink/MAVLink (*RflySim* Advanced Version only)

- Plug in Pixhawk, open CopterSim, set HITL simulation, set UDP_Mode to Mavlink_FULL (RflySim Advanced version only), and click the "Start Simulation" button
- Open the demo "MavlinkDemo\MavSfunTest_Arm.slx" and run it, you can see "Command ARM/DISARM ACCECPTED" in the message box of CopterSim, indicating that the experiment was successful





2.12 Simulate sending RC data via MAVLink (RflySim advanced version only) Same as the previous step, open CopterSim hardware-in-the-loop, and run the "MavlinkDemo\MavSfunTest_control RC.slx" demo to control the arming of Pixhawk and send RC data to control drone take-off and landing, flight, etc.





2.13 Simulink sends and receives MAVLink messages through the serial port

Connect Pixhawk to computer, use CopterSim to start HIL simulation (you do not need to set UDP_Mode so the basic version of RflySim is also applicable), use a digital transmission (radio telemetry) module to connect Pixhawk to the computer, remember the serial number of the radio telemetry module (if you don't have a radio telemetry, you can plug in Pixhawk directly without opening CopterSim, and enter the serial port number of Pixhawk here). Note, the Baud rate of a radio is usually 57600.

 \times

 Open "MavlinkDemo\MavSfunTest_SerialCom.slx", double-click "Mavlink Serial Input&Output", and enter the serial port number in it Block Parameters: Mavlink Serial Input&Output





• In this example, you can obtain Pixhawk data through the serial port and send control commands. This example can be directly used for real-time (through data transmission) control of the Pixhawk multicopter real vehicle.



35



- 1. Configuration of software & hardware
- 2. MAVLink communication analysis
- 3. PX4 official controller communication
- Path of demo source code of this section: "RflySimAPIs\SimulinkCo ntroIAPI"

4. Code generation controller communication

5. Summary





3.1 Simulink simulates sending the raw data of the RC to control the drone First connect Pixhawk with CopterSim and start HIL simulation and open the 3D software at the same time (RflySim Advanced version can directly run the Desktop **HITLRun** to quickly open the HIL simulation, or run **SITLRun** to open the SIL simulation), and then open "**RflySimAPIs\SimulinkControlAPI\RadioControlAPI.slx**" through MATLAB and run.





3. PX4 official controller communic

BUAA Reliable Flight Contro

3.1 Simulink simulates sending the raw data of the RC to control the drone

- First, drag the **Vez** slider to the right (simulating pushing up the throttle to pass the midpoint), you can control the speed of the drone in the Z direction and make the drone take off vertically;
- Then, drag the **VeX** (simulate forward and backward pitch stick) and VeY (simulate left and right roll stick) sliders to achieve forward and backward movement,
- Similarly, drag the **Yaw** slider (simulating the • left and right yaw sticks) to control the yaw speed and the drone deflection.







3. PX4 official controller communication





3. PX4 official controller comr

3.1 Simulink simulates sending the raw data of the RC to control the drone

Double-click the "RCControlTrans" module, you can see the internal information shown below



CopterID

RCControlTran

1 VbX

2 VbY CopterID

CmdUDP

CMDPack



int

char

ConsultoP ConsultoP

3.1 Simulink simulates sending the raw data of the RC to control the drone

 Double-click to enter the "CMDPack" module, you can see the internal information shown in the figure below

If the the PX4 is not armed, CopterSim will send the arm command repeatly.

• Two-layer encapsulation (to ensure data security)

struct inHILCMDData{
 uint32_t time_boot_ms;
 uint32_t copterID;
 uint32_t modes;
 uint32_t flags;
 float ctrls[16];
};
struct netDataShortShort {
 TargetType ts;
};

len:

InetDataShortShort;

payload[112];



Set here to be consistent with the calibration value of the RC system, otherwise there will be a problem of response deviation. First, send the data amplitude to the **inHILCMDData** structure, and then store the structure data in the payload data segment of the **netDataShortShort** structure, and finally send the data.

If the TargetTpye is set to 1, the CopterSim will use Ctrs[1:8] to generate mavlink rc_channels_override to simulator RC signals, (mode,flags,and Ctrls[9:16] are reserved)

Senerally, the RC should be calibrated in QGC; the RC 5 or 6th channels of PX4 should be the mode switch in QGC, corresponding to Stablized, Altitude, and Position Mod



3. PX4 official controller communication

3.1 Simulink simulates sending the raw data of the RC to control the drone

 After CopterSim receives the UDP message from Simulink, it will generate the MAVLink message RC_CHANNELS_OVERRIDE (RC channel coverage), and forward it to the Pixhawk module that implements the RC signal

℀ RC_CHANNELS_OVERRIDE (#70)

The RAW values of the RC channels sent to the MAV to override info received from the RC radio. A value of UINT16_MAX means no change to that channel. A value of 0 means control of that channel should be released back to the RC radio. The standard PPM modulation is as follows: 1000 microseconds: 0%, 2000 microseconds: 100%. Individual receivers/transmitters might violate this specification.

Field Name	Туре	Units	Description
target_system	uint8_t		System ID
target_component	uint8_t		Component ID
chan1_raw	uint16_t	us	RC channel 1 value. A value of UINT16_MAX means to ignore this field.
chan2_raw	uint16_t	us	RC channel 2 value. A value of UINT16_MAX means to ignore this field.
chan3_raw	uint16_t	us	RC channel 3 value. A value of UINT16_MAX means to ignore this field.
chan4_raw	uint16_t	us	RC channel 4 value. A value of UINT16_MAX means to ignore this field.
chan5_raw	uint16_t	us	RC channel 5 value. A value of UINT16_MAX means to ignore this field.
chan6_raw	uint16_t	us	RC channel 6 value. A value of UINT16_MAX means to ignore this field.
chan7_raw	uint16_t	us	RC channel 7 value. A value of UINT16_MAX means to ignore this field.
chan8_raw	uint16_t	us	RC channel 8 value. A value of UINT16_MAX means to ignore this field.
chan9_raw **	uint16_t	us	RC channel 9 value. A value of 0 or UINT16_MAX means to ignore this field.
chan10_raw **	uint16_t	us	RC channel 10 value. A value of 0 or UINT16_MAX means to ignore this field.
chan11_raw **	uint16_t	us	RC channel 11 value. A value of 0 or UINT16_MAX means to ignore this field.





- 3.1 Simulink simulates sending the raw data of the RC
- Go back to the outermost layer and click to enter the "UDP_SIL_State_Receiver" module



Data check, if data is not OK then keep the last data.





3. PX4 official controller communication

3.1 Simulink simulates sending the raw data of the RC to control the drone

- Principle: Receive the data of the outHILStateData structure sent by CopterSim via UDP, and extract the value of interest from it
- CopterSim data resource: forward MAVLink message from Pixhawk, including
 LOCAL_POSITION_NED,
 ATTITUDE, HOME_POSITIONE,
 STIMATOR_STATUS, and etc.

struct outHILStateData{ // mavlink data forward from Pixhawk 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 and up is positive 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 up is positive 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 (good precision) int32 t resrveInit; //Int, reserve for the future use float AngEular[3]; //Estimated Euler angle, unit: rad/s float localPos[3]; //Estimated locoal position, NED, unit: m float localVel[3]; //Estimated locoal velocity, NED, unit: m/s 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





3.2 Simulink simulates sending a normalized RC signal to control the drone

- The previous example sent the raw data of the RC system, so the PWM output value needs to be consistent with the RC calibration value, otherwise control deviation may occur
- Open "RflySimAPIs\SimulinkControlAPI\ManulControlAPI.slx", and get a demo with the same function as the previous example. The experiment process is the same, but this demo doesn't need to focus on the RC calibration value





3. PX4 official controller communication

3.2 Simulink simulates sending a normalized RC signal to control the drone This message is a MAVLink message that implements **MANUAL_CONTROL**. In actual flight, the signal can also be sent through digital transmission to reproduce the control command.

MANUAL_CONTROL (#69)

This message provides an API for manually controlling the vehicle using standard joystick axes nomenclature, along with a joystick-like input device. Unused axes can be disabled an buttons are also transmit as boolean values of their

Field Name	Туре	Description
target	uint8_t	The system to be controlled.
х	int16_t	X-axis, normalized to the range [-1000,1000]. A value of INT16_MAX indicates that this axis is invalid. Generally corresponds to forward(1000)- backward(-1000) movement on a joystick and the pitch of a vehicle.
у	int16_t	Y-axis, normalized to the range [-1000,1000]. A value of INT16_MAX indicates that this axis is invalid. Generally corresponds to left(-1000)-right(1000) movement on a joystick and the roll of a vehicle.
Z	int16_t	Z-axis, normalized to the range [-1000,1000]. A value of INT16_MAX indicates that this axis is invalid. Generally corresponds to a separate slider movement with maximum being 1000 and minimum being -1000 on a joystick and the thrust of a vehicle. Positive values are positive thrust, negative values are negative thrust.
r	int16_t	R-axis, normalized to the range [-1000,1000]. A value of INT16_MAX indicates that this axis is invalid. Generally corresponds to a twisting of the joystick, with counter-clockwise being 1000 and clockwise being -1000, and the yaw of a vehicle.
buttons	uint16_t	A bitfield corresponding to the joystick buttons' current state, 1 for pressed, 0 for released. The lowest bit corresponds to Button 1.



3.3 Simulink simulates sending a controlled drone using Offboard mode

- Offboard mode is a control mode of drones. Usually the onboard computer or ground computer (host computer) is used to control the speed, position, attitude of the drone in real time. The drone can be treat as a whole object, focusing on the top-level vision and swarm algorithm development.
- The RC signal control cannot quantitatively control the speed of the drone, so it is not convenient to use the Offboard control mode, but the RC signal control mode is the closest way to human operation, and it has better effects in some high-maneuver performance control.
- The follow-up experiments of this course are all to see the drone as a whole subobject (receive and implement speed/position/acceleration/route and other commands), so the subsequent series of experiments will mainly use the Offboard mode to control the drone. Since the Offboard mode is a function provided by the official PX4 controller, you need to make sure that Pixhawk is running the official firmware (mentioned in the previous settings).





3. PX4 official controller communication

3.3 Simulink simulates sending a controlled drone using Offboard mode

- Enable CopterSim's HIL (or SIL) simulation system
- Open "RflySimAPIs \SimulinkControlAPI \OffboardAPI.slx" to run, you can see that the drone will automatically take off to a height of 10m at first, then switch the "speed/position control" switch, drag the slider, you can enter speed 5m/s on direction X (or drag Slider VeX to the desired value), observe whether the speed is consistent with the given speed in QGC





3. PX4 official controller comm









3. PX4 official controller communication

3.3 Simulink simulates sending a controlled drone using Offboard mode (recommended)

- **Principle**: This example will make PX4 enter the Offboard mode, then send a MAVLink message of **SET_POSITION_TARGET_LOCAL_NED** to control the speed, position, and angle of the drone. This command does not require the drone to perform RC calibration or modal settings, just specify the specified speed or position directly.
- Three module examples are shown in the **OffboardAPI.slx** file. **VelEarthFrameOffboardCtrl** is the speed control module in the earth coordinate system
- VelBodyFrameOffboardCtrl is the speed control in the body coordinate system
- **PosTargetEarthFrameOffboardCtrl** is the position control module in the earth coordinate system (given the relative take-off point x, y, z coordinates, the drone will automatically fly to this point and hover).
- The implementation methods of the three modules are exactly the same, except that several parameters of the Offboard message (position/speed control mode & body/earth coordinate system) are different. The Offboard control command is based on the disarmed position as the **Home_Position** coordinate as the origin relative coordinate (**Local_Position**), so the position command sent refers to the relative coordinate value of flying to the relative armed position.





3.3 Simulink simulates sending a controlled drone using Offboard mode (recommended)

• The Offboard mode interface of Simulink is shown in the figure below, you can combine the commands that need to be controlled by yourself





3. PX4 official controller communication

3.3 Simulink simulates sending a controlled drone using Offboard mode https://mavlink.io/en/messages/common.html#SET_POSITION_TARGET_LOCAL_NED achieve method of MAVLink message shows as below picture

SET_POSITION_TARGET_LOCAL_NED(#84)

[Message] Sets a desired vehicle position in a local north-east-down coordinate frame. Used by an external controller to command the vehicle (manual controller or other system).

Field Name	Туре	Units	Values	Description
time_boot_ms	uint32_t	ms		Timestamp (time since system boot).
target_system	uint8_t			System ID
target_component	uint8_t			Component ID
coordinate_frame	uint8_t		MAV_FRAME	Valid options are: MAV_FRAME_LOCAL_NED = 1, MAV_FRAME_LOCAL_OFFSET_NED = 7, MAV_FRAME_BODY_NED 8, MAV_FRAME_BODY_OFFSET_NED = 9
type_mask	uint16_t		POSITION_TARGET_TYPEMASK	Bitmap to indicate which dimensions should be ignored by the vehicle.
			BUAA Reliable Flig	ht Control Group



3. PX4 official controller communication

Х	float	m	X Position in NED frame
у	float	m	Y Position in NED frame
Z	float	m	Z Position in NED frame (note, altitude is negative in NED)
VX	float	m/s	X velocity in NED frame
vy	float	m/s	Y velocity in NED frame
VZ	float	m/s	Z velocity in NED frame
afx	float	m/s/s	X acceleration or force (if bit 10 of type_mask is set) in NED frame in meter / s^2 or N
afy	float	m/s/s	Y acceleration or force (if bit 10 of type_mask is set) in NED frame in meter / s^2 or N
afz	float	m/s/s	Z acceleration or force (if bit 10 of type_mask is set) in NED frame in meter / s^2 or N
yaw	float	rad	yaw setpoint
yaw_rate	float	rad/s	yaw rate setpoint



- 1. Configuration of software & hardware
- 2. MAVLink communication analysis
- 3. PX4 official controller communication
- Path of demo source code of this section: "RflySimAPIs\SimulinkCo ntrolAPI\Rfly_API_CTRL"
- 4. Code generation controller communication

5. Summary





4. Code generation controller

4.1 Automatic code generation environment configuration

- The Pixhawk control algorithm generated for Simulink can also be controlled through the QGC/Simulink API. To run the example in this section, you need to use Simulink code to generate the controller, so you need to re-run the installation script, as shown in the right figure (RflySim Advanced Edition) to block the PX4's own output. Note: Please use px4fmuv3_default, PX4-1.7.3 firmware, and Msys2 compiler for RflySim basic version
- Note: The compilation command needs to be modified according to your own hardware
- Note: The code automatically generated by Simulink currently does not support the **px4_sitl_default** compilation command, so it does not support PX4 SITL simulation. The examples in this section need to use Pixhawk hardware for HIL simulation.

Note: The content of this section is mainly aimed at the external communication problem of the controller developed with Simulink in the course "RflySim_Lesson_02_Flight_Control_Experiments.pdf" (i.e., the series of experiments in the book "Multcopter Design and Control Practice").

承 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_fmu-v3_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])

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

no

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

no

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

no

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

no

(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)

究纠^{yes}

Grou





4.2 Use the RC signal generated by Simulink to control the controller designed by Simulink

You can use the RC signal output by Simulink to control our own designed attitude controller, such as "RflySimAPIs \ FlightControlExpCourse \ code \ e0 \ 3.DesignExps \ Exp4_AttitudeSystemCodeGen_old.slx"

First, open "**Exp4_AttitudeSystemCodeGen_old.slx**" to compile the firmware and burn Pixhawk, then use CopterSim to connect to Pixhawk and start the HIL simulation (the advanced version can directly run the desktop **HITLRun** and quickly open the HIL).

• Open and run the "RadioControlAPI.slx" file, you can control the drone to take off and move forward and backward. Data can also be observed in QGC

🖸 Q	GroundControl		-						—		\times
	😵 💘 🖉	₽		× 15 0.0	.	51% →	Armed -	Position -			
	Analyze		M	AVLink Ir	nspector						-7
	Log Download		HOME_	POSITION		0.2Hz	信息 : 组件:	RC_CHANNELS	(65) 20.0Hz		
\odot	GeoTag Images		LOCAL	_POSITION	_NED	50.0Hz	· 计数:	524			
>	MAVLink Console		PING			0.2Hz	Name	Value	Туре	Plot 1	Plot 2
M	MAVLink Inspector		POSI1	ION_TARGE	T_LOCAL_NED	41.2Hz	time_boot_ms chancount	52988 8	uint32_t uint8_t		
			RC_CH	IANNELS		20.0Hz	chan1_raw chan2_raw	1499	uint16_t uint16_t		
			SERVO	0_OUTPUT_R	AW	50.0Hz	chan3_raw chan4_raw	1929 1499	uint16_t uint16_t		
		j	SYSTE	EM_TIME		0.0Hz	chan5_raw chan6_raw	1929 1929	uint16_t uint16_t		
			eve c	TATIC		1 011-	chan7_raw	1499	uint16_t		



4.2 Use the RC signal generated by Simulink to control the controller designed by Simulink

- Principle: To further explain the principle, the RadioControlAPI.slx file sends the MAVLink message of RC_CHANNELS_OVERRIDE to Pixhawk, and the RC module used in Exp4_AttitudeSystemCodeGen_old.slx will receive the MAVLink message, so it can respond.
- Similarly, if you need your own generated code to respond to the input in ManulControlAPI.slx and OffboardAPI.slx, you need to receive the corresponding uORB messages in the Simulink controller respectively. Among them, ManulControlAPI.slx corresponds to the uORB message of "MANUAL_CONTROL_SETPOINT", and OffboardAPI.slx corresponds to the uORB message of "POSITION_SETPOINT".
- Note: ManulControlAPI.slx corresponds to the demo of "Exp4_AttitudeSystemCodeGen.slx" code generation, and can respond to the "MANUAL_CONTROL_SETPOINT" message.





4.3 Send and receive customized messages using Simulink

 The RflySim platform also provides an external program to communicate with its Simulink controller. It sends uORB messages of rfly_ctrl.msg (see Firmware\msg folder). It is defined as follows:

1 2	uint64 timestamp uint64 flags	<pre># time since system start (microseconds) # control flag</pre>
3	uint8 modes	# mode flag
4	float32[16] controls	# 16D control signals

 The uORB message can be sent by an external program to send MAVLink messages to achieve the "HIL_ACTUATOR_CONTROLS" message, and its 16-dimensional control amount corresponds to the following definition.

HIL_ACTUATOR_CONTROLS (#93)

Sent from autopilot to simulation. Hardware in the loop control outputs (replacement for HIL_CONTROLS)

Field Name	Туре	Units	Values	Description
time_usec	uint64_t	us		Timestamp (UNIX Epoch time or time since system boot). The receiving end can infer timestamp format (since 1.1.1970 or since system boot) by checking for the magnitude the number.
controls	float[16]			Control outputs -1 1. Channel assignment depends on the simulated hardware.
mode	uint8_t		MAV_MODE_FLAG	System mode. Includes arming state.
flags	uint64_t			Flags as bitfield, reserved for future use.

58



4.3 Send and receive customized messages using Simulink

• Open RflySimAPIs\SimulinkControlAPI\Rfly_API_CTRL\Rfly_ Mavlink_Receiver.slx with MATLAB, compile the generated code, and burn it to Pixhawk. As shown in the figure below, it received the uORB message of rfly_ctrl, replacing the original





4.3 Send and receive customized messages using Simulink

- Enable the hardware-in-the-loop simulation of CopterSim and Pixhawk, run the
 "Rfly_API_CTRL\Rfly_Mavlink_API_Sender.slx" file, which can send control signals to CopterSim,
 and CopterSim will forward the "HIL_ACTUATOR_CONTROLS" MAVLink message to Pixhawk,
 and then Pixhawk will publish it to 'rfly_ctrl' in the pool, used by the px4_simulink_app
 generated by the code in the above figure. As shown in the figure below, this slx demo
 simulates the RC data of CH1~Ch5 and sends it to the rfly_ctrl message.
- The experimental effect of this demo is consistent with the operation process of the analog RC signal control PX4 official controller in Section 3.1
- The experimental phenomenon is the same as the effect of sending the RC signal to the Simulink code generation controller in **Section 4.2**, because the two examples both send the RC signal, but the communication path is different.
- Note: In the case of real flight, just use MAVLink to send HIL_ACTUATOR_CONTROLS message to the flight controller, this module can send data to the control





Thanks

