User Guide for Transparent Data Transmission between Mobile and Onboard Devices

V1.0.0

2015.06

目录

Introduction

This guide gives an overview of the Transparent Data Transmission function between onboard and mobile devices. Topics covered include development purpose, method of use, and sample codes.

Note that the upstream bandwidth (mobile to onboard) is around 1KB/s, while the downstream bandwidth (onboard to mobile) is around 8KB/s.

1 Development Purpose

DJI provides two types of APIs for developers to create their own applications: Mobile API and Onboard API. Mobile API allows developers to monitor and control the UAV from a mobile device, running iOS or Android, that is connected to the remote controller; Onboard API allows developers to monitor and control the UAV from any system directly connected to the UAV through the available serial (UART) port.

Mobile API can be used without any other devices, and allows developers to monitor the flight status easily. However, this configuration has a relatively low computing power, and the wireless link between the mobile device and the UAV restricts the bandwidth for real-time or complex control.

Onboard API is implemented through the computing device which is mounted on the UAV. Communication with the UAV is done directly through their serial ports, providing sufficient computing power and stability for developers to run complex and demanding applications. But due to the computing device being mounted on the UAV, developers will not be able to monitor the flight status. If the program crashes, developers will have to manually control the UAV with the remote controller.

Transparent Data Transmission was developed to combine the benefits of the two APIs, by establishing a connection between the mobile device and the onboard device for the first time. With this technology, developers can send data from the mobile device to the onboard device, enabling control over the program running on the onboard device. The onboard device can also send data to the mobile device for flight status monitoring and other functions.

In short, Transparent Data Transmission serves as a linkage between Mobile API and Onboard API, granting developers greater flexibility in creating their application (Fig.1).

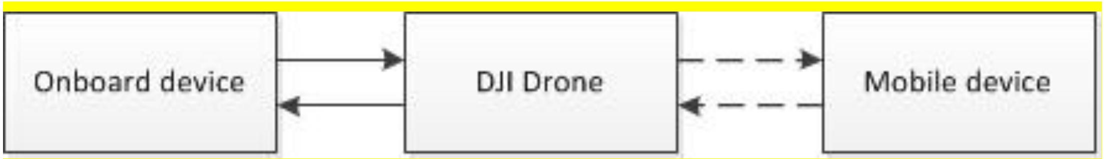


Fig.1 Transparent Data Transmission stream frame

2 Transparent Data Transmission (Onboard to Mobile)

2.1 Onboard **Device** to UAV

The communication protocol between the onboard device and the UAV is introduced in *onboard_SDK_introduction*. This section explains it in more detail.

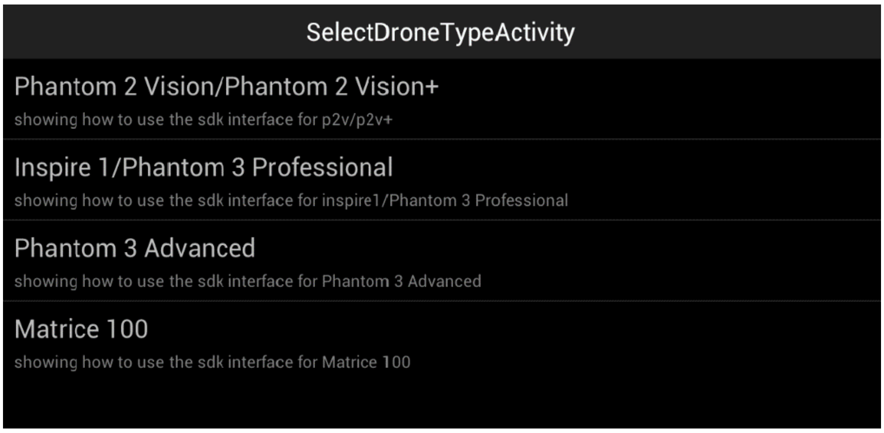
The communication protocol is described below:
Command set: 0X00
Command ID: 0XFE
Sample code:

	Offset	*Size*	*Description*
Request data	**0**	1~100	Data needs to be sent to Mobile device
Return data	**0**	2	Return code 0 Success

```
char cmd_buf[10];
cmd_buf[0] = 0x00;
cmd_buf[1] = 0xFE;
memcpy(&cmd_buf[2], “Hello!”, 7);
Linklayer_Send(
                                 SESSION_MODE3,
                                 cmd_buf,
                                 9,
                                 0，
                                 200，
                                 3，
                                 0
);
```

2.2 UAV to Mobile Device

This section uses the Android system as an example. Users can enter the DJI-SDK-DEMO application (Fig.2), monitor the UAV status through relevant functions (Fig.3), and then use the Transparent Data Transmission function on the *Controller State* page (Fig.4). Note that so far only the DJI Matrice 100 supports this function.



SelectDroneTypeActivity

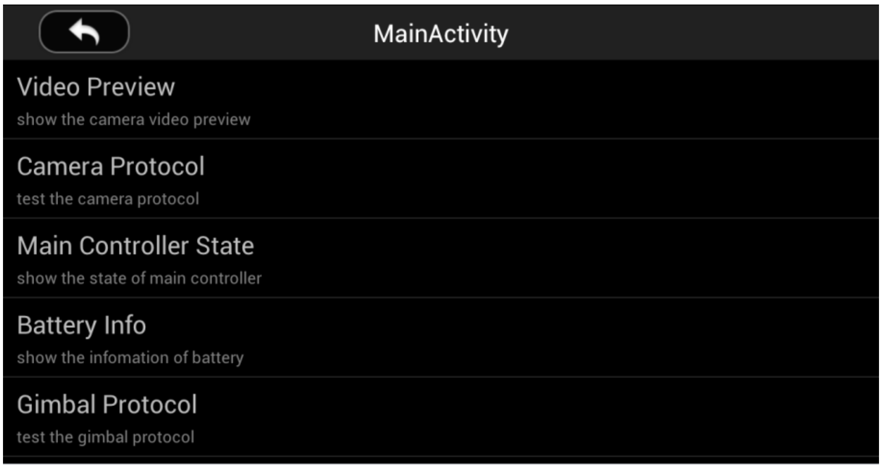
Phantom 2 Vision/Phantom 2 Vision+
showing how to use the sdk interface for p2v/p2v+

Inspire 1/Phantom 3 Professional
showing how to use the sdk interface for inspire1/Phantom 3 Professional

Phantom 3 Advanced
showing how to use the sdk interface for Phantom 3 Advanced

Matrice 100
showing how to use the sdk interface for Matrice 100

Fig.2 DJI-SDK-DEMO main interface



MainActivity

Video Preview
show the camera video preview

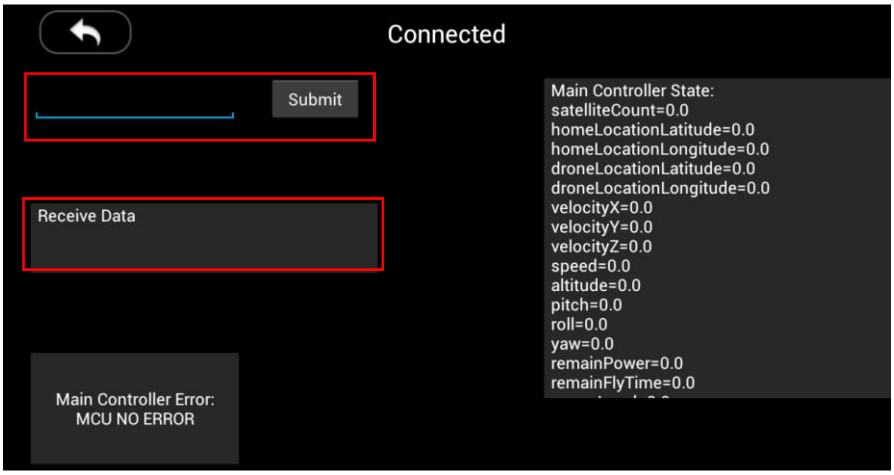
Camera Protocol
test the camera protocol

Main Controller State
show the state of main controller

Battery Info
show the infomation of battery

Gimbal Protocol
test the gimbal protocol

Fig.3 DJI-SDK-DEMO relative functions



Connected

```
Main Controller State:
satelliteCount=0.0
homeLocationLatitude=0.0
homeLocationLongitude=0.0
droneLocationLatitude=0.0
droneLocationLongitude=0.0
velocityX=0.0
velocityY=0.0
velocityZ=0.0
speed=0.0
altitude=0.0
pitch=0.0
roll=0.0
yaw=0.0
remainPower=0.0
remainFlyTime=0.0
```

Main Controller Error:
MCU NO ERROR

Fig.4 DJI-SDK-DEMO Transparent Data Transmission demo

The relevant sample codes are shown below:

1) iOS:

```
// Setting Delegation
inspireMC.mcDelegate = self;

//The legation function is called when receiving data
```

```
(void)mainController:(DJIMainController*)mc
didReceivedDataFromExternalDevice:(NSData*)data{
        //Here is the receiving data
        NSLog(@"%@",data);
}
```

2) Android:

```
//Receiving the data callback interface sent from UAV
DJIMainControllerExternalDeviceRecvDataCallBack mExtDevReceiveDataCallBack = null;

//Instantiate callback interface
mExtDevReceiveDataCallBack = new DJIMainControllerExternalDeviceRecvDataCallBack() {
        @Override
        public void onResult(byte[] data)
        {
                //Here is the receiving data
        }
};
//Setting callback interface
DJIDrone.getDjiMC().setExternalDeviceRecvDataCallBack(mExtDevReceiveDataCallBack);
```

3 Transparent Data Transmission (Mobile to Onboard)

3.1 Mobile Device to UAV

The relevant sample codes are shown below:

1) iOS：

1. Initialization.

```
//Create DJI Drone object according to relative UAV type.
DJIDrone* drone = [DJIDrone droneWithType:DJIDrone_Inspire];
//Obtain Main controller object from DJI Drone object.
DJIInspireMainController* inspireMC = (DJIInspireMainController*)drone.mainController;
//Start data connection.
[drone connectToDrone];
```

2. Sending data.

```
//Please note that data size should be no larger than 100 bytes.
  NSData* data = [NSData dataWithByte:"…"];
// Sending data to peripheral and check the sending status through callback function.
```

```
[inspireMC sendDataToExternalDevice:data withResult:(^(DJIError* error)){
        if (error.errorCode == ERR_Successed)
        {
                //Data sent successfully.
        }
        Else if(error.errorCode == ERR_InvalidParam)
        {
                //Data size is null or larger than 100 bytes.
        }
        else
        {
                //Data sent failed.
        }
}];
```

2) Android:

```
//Data needs to be sent, please note the data size should be no larger than 100 bytes.
byte[] data = {0};
//Sending data to UAV
DJIDrone.getDjiMC().sendDataToExternalDevice(data,new DJIExecuteResultCallback(){
        @Override
        public void onResult(DJIError result)
        {
                //result is the callback status after sending data:
                // 1 . result == DJIError.ERR_PARAM_IILEGAL ,    Data size is null or larger than 100 bytes.
                // 2 . result == DJIError.ERR_TIMEOUT ,          Data sent failed.
                // 3 . result == DJIError.RESULT_OK,             Data sent successfully.

        }
});
```

3.2 UAV to Onboard Device

Use this function with the same method mentioned previously. The communication protocol is described below:

Command set: 0X02

Command ID: 0X02

Sample code：

	Offset	*Size*	*Description*
Request data	**0**	1~100	User defined data
Return data	**0**	0	No return data