Recon Drone Interface Control Document

Last Modified On:
March 29, 2021

Recon Drone Interface Control Document

Document Description

This document specifies the binary format for the socket interface between the Recon GCS and the mobile companion App used for interacting with DJI drones.

Contents

1 Introduction

Recon is a multi-vehicle Ground Control Station (GCS) for interacting with and commanding DJI drones. It includes modules for identifying and tracking cloud shadows on the terrain based on live fisheye imagery streamed from one of the connected drones, and for predicting the evolution of cloud shadows into the future. A Guidance module can exploit these predictions to task drones in a way that avoids shadowed areas. Finally, a DJI Drone Interface Module provides the interface between the GCS and the drones themselves. Through this module, the GCS runs a TCP server that mobile devices on the same local network can connect to as clients. A TCP socket connection is established to each client over which commands, telemetry, and other data are streamed back and forth. The purpose of this document is to define the structure of communications over this socket connection.

Recon is developed as part of a USDA/NIFA research project: “NRI: FND: COLLAB: Multi-Vehicle Systems for Collecting Shadow-Free Imagery in Precision Agriculture” (NIFA Award # 2020-67021-30758)

2 Data Types

Communications over this connection will be packetized. Packets will be made up of fields. Each field will be an item with a specified type. We use a collection of primitive data types (ints, floats, etc.) and compound data types, which are themselves made up of primitive types. Before diving into packet structure we define our data types. Table 1 lists primitive types and Table 2 lists compound types.

Type	Description
uint8	Unsigned 8-bit Integer
uint16	Unsigned 16-bit Integer
uint24	Unsigned 24-bit Integer
uint32	Unsigned 32-bit Integer
uint64	Unsigned 64-bit Integer
int8	Signed 8-bit Integer (2's compliment)
int16	Signed 16-bit Integer (2's compliment)
int24	Signed 24-bit Integer (2's compliment)
int32	Signed 32-bit Integer (2's compliment)
int64	Signed 64-bit Integer (2's compliment)
float	IEEE 754 floating point number: 32-bits
double	IEEE 754 floating point number: 64-bits

Table 1: Primitive Types

Type	Description
String	UTF-8 encoded string
Image	RGB Image (8-bits per channel)

Table 2: Compound Types

Primitive types are transmitted most-significant byte first. The compound types are formed from primitive types and we must specify their structures. These are specified in the following tables.

StrLength	B0	B1	B2	...
4	1	1	1	...

String Serialization: This shows the serialization of a string. The "StrLength" sub-field is of type Uint32 and holds the number of bytes that follow in the string. A full string field consists of $4 +$ StrLength bytes. The bytes that follow the "StrLength" field should be interpreted as a UTF-8 byte stream. The top row provides field names and the bottom row provides field widths (in bytes). If a field width is missing, it is variable and must be deduced from other information in the packet.

Rows	Cols	$R_{0,0}$	$G_{0,0}$	$B_{0,0}$	$R_{0,1}$	$G_{0,1}$	$B_{0,1}$	...
2	2	1	1	1	1	1	1	...

Image Serialization: This shows the serialization of an RGB Image field. Sub-fields "Rows" and "Cols" are of type Uint16. All other sub-fields are of type Uint8. The total number of bytes in an Image field is $(4 + \text{Rows} * \text{Cols} * 3)$. Pixel data is organized row-by-row, starting with the upper-left corner of the image. There is no padding at the end of rows or at the end of the array. The top row provides field names and the bottom row provides field widths (in bytes). If a field width is missing, it is variable and must be deduced from other information in the packet.

3 Data Packets

This section defines each of the packets that is supported by the drone socket interface. When reading packet definitions in this document, packets are sent left-to-right. That is, the *sync* field is always sent first, moving towards the right, with the *hash* field sent last. Multi-byte primitive fields are sent most-significant byte first. Compound fields are sent left-to-right based on their structures in the previous section, with each sub-field being sent most-significant byte first.

We provide the general packet structure for all packets, including a packet-dependent *Payload* field. The structure of this field is then defined separately for each packet type.

sync	size	PID	Payload	hash
2	4	1		2

Drone Interface Packet structure: The top row provides field names and the bottom row provides field widths (in bytes). The structure of the payload field depends on the packet type. The Payload field size is *size* - 9 bytes.

field	type	description
sync	uint16	A field used to help delimit packets (Value = 55975U)
size	uint32	Total number of bytes in packet (including sync and hash fields)
PID	uint8	Packet ID: Code used to indicate packet type
Payload		Content of packet - Structure depends on packet type
hash		2-byte hash of packet (*sync* through *Payload*) (see 3.3.1)

In the following subsections we define each specific packet type sent to or from the drone companion App. These packets are identified by their *PID*.

Contents

3.1 Packets From Client (Mobile Device) To Server (Computer)

3.1.1 Core Telemetry Packet [*PID*=0U]

This packet encapsulates core telemetry information: position, velocity, orientation, & height. Field definitions are designed to match the DJI SDK API as closely as possible, with the expectation that necessary conversions will take place on the server side. Most, if not all, of this data can be accessed via the "DJIFlightControllerState" in the SDK (referred to as FCS in tables).

Field	Type	Size	Offset in Payload
IsFlying	uint8	1	0
Latitude	double	8	1
Longitude	double	8	9
Altitude	double	8	17
HAG	double	8	25
V_N	float	4	33
V_E	float	4	37
V_D	float	4	41
Yaw	double	8	45
Pitch	double	8	53
Roll	double	8	61

Field	Type	Description
IsFlying	uint8	=1 if FCS.isFlying is true, =0 if false
Latitude	double	WGS84 Latitude (Degrees). Taken from FCS.aircraftLocation
Longitude	double	WGS84 Longitude (Degrees). Taken from FCS.aircraftLocation
Altitude	double	WGS84 Altitude (m). Taken from FCS.aircraftLocation if populated, or computed as FCS.takeoffLocationAltitude + FCS.altitude
HAG	double	Height above takeoff location. Equal to FCS.altitude
V_N	float	North velocity (m/s). Equals FCS.velocityX
V_E	float	East velocity (m/s). Equals FCS.velocityY
V_D	float	Down velocity (m/s). Equals FCS.velocityZ
Yaw	double	Vehicle Yaw (Degrees). DJI def. Taken from FCS.attitude
Pitch	double	Vehicle Pitch (Degrees). DJI def. Taken from FCS.attitude
Roll	double	Vehicle Roll (Degrees). DJI def. Taken from FCS.attitude

3.1.2 Extended Telemetry Packet [*PID*=1U]

This packet encapsulates extended telemetry information that isn't included in the core telemetry packet. This data is generally less critical and may be sent at a lower rate than core data. Most of this data can be accessed via the "DJIFlightControllerState" in the SDK ("FCS" in tables).

Field	Type	Size	Offset in Payload
GNSSSatCount	uint16	2	0
GNSSSignal	uint8	1	2
MaxHeight	uint8	1	3
MaxDist	uint8	1	4
BatLevel	uint8	1	5
BatWarning	uint8	1	6
WindLevel	uint8	1	7
DJICam	uint8	1	8
FlightMode	uint8	1	9
MissionID	uint16	2	10
DroneSerial	String		12

Field	Type	Description
GNSSSatCount	uint16	Equals FCS.satelliteCount
GNSSSignal	int8	From FCS.GPSSignalLevel (-1: None, ≥ 0: equals signal level)
MaxHeight	uint8	=1 if FCS.hasReachedMaxFlightHeight is true. =0 if false
MaxDist	uint8	=1 if FCS.hasReachedMaxFlightRadius if true. =0 if false
BatLevel	uint8	Equals DJIBatteryState.chargeRemainingInPercent
BatWarning	uint8	Let X=FCS.isLowerThanBatteryWarningThreshold and Y=FCS.isLowerThanSeriousBatteryWarningThreshold. Value: 0 if X and Y are false. 1 if X is true and Y is false. 2 if Y is true.
WindLevel	int8	Wind level. (-1: FCS.windWarning = Unknown, ≥ 0: equals warning level)
DJICam	uint8	=0 if no DJI cam connected. =1 if connected but live feed off. =2 if connected and live feed is on
FlightMode	uint8	Based on FCS.flightMode. See Table 3.
MissionID	uint16	"missionID" for waypoint mission being flown (0 if in other mode)
DroneSerial	String	Serial for drone

FCS.flightMode	"FlightMode" field
DJIFlightModeManual	0
DJIFlightModeAtti	1
DJIFlightModeAttiCourseLock	2
DJIFlightModeGPSAtti	3
DJIFlightModeGPSCourseLock	4
DJIFlightModeGPSHomeLock	5
DJIFlightModeGPSHotPoint	6
DJIFlightModeAssistedTakeoff	7
DJIFlightModeAutoTakeoff	8
DJIFlightModeAutoLanding	9
DJIFlightModeGPSWaypoint	10
DJIFlightModeGoHome	11
DJIFlightModeJoystick	12
DJIFlightModeGPSAttiWristband	13
DJIFlightModeDraw	14
DJIFlightModeGPSFollowMe	15
DJIFlightModeActiveTrack	16
DJIFlightModeTapFly	17
DJIFlightModeGPSSport	18
DJIFlightModeGPSNovice	19
DJIFlightModeUnknown	20
DJIFlightModeConfirmLanding	21
DJIFlightModeTerrainFollow	22
DJIFlightModeTripod	23
DJIFlightModeActiveTrackSpotlight	24
DJIFlightModeMotorsJustStarted	25
Unrecognized/Other Mode	255

Table 3: Values for "FlightMode" field

3.1.3 Image Packet [*PID*=2U]

This packet encapsulates a single image from the drone video feed, if connected and enabled.

Field	Type	Size	Offset in Payload
TargetFPS	float	4	0
Frame	Image		4

Field	Type	Description
TargetFPS	float	The current target frame rate (frames per second)
Frame	Image	A single RGB image

3.1.4 Acknowledgment Packet [*PID*=3U]

This packet is used to reply to the server to acknowledge receipt of a command packet.

Field	Type	Size	Offset in Payload
Positive	uint8	1	0
SourcePID	uint8	1	1

Field	Type	Description
Positive	uint8	1 for positive acknowledgment (will execute command), 0 for negative acknowledgment (won't execute command)
SourcePID	uint8	The PID of the packet being acknowledged

3.1.5 Message String Packet [*PID*=4U]

This packet is used to send a message string to the server containing human-readable text. The big use case for this is development and debugging, but it can also be used to issue warnings and error messages to the server origination from the client App.

Field	Type	Size	Offset in Payload
Type	uint8	1	0
Message	String		1

Field	Type	Description
Type	uint8	0: Debug, 1: Info, 2: Warning, 3: Error It should be assumed that Info, Warning, and Error messages contain information that a user might want to see during normal use (and so the UI may make these available) and it should be expected that warning and error messages will be presented with appropriate urgency. Thus, these message types should only be used when it's appropriate. For general messages needed during development and testing use type Debug (Type = 0).
Message	String	The human-readable message

3.2 Packets From Server (Computer) To Client (Mobile Device)

3.2.1 Emergency Command Packet [*PID*=255U]

This packet instructs the App and drone to stop whatever mission may be running and immediately execute a given command.

Field	Type	Size	Offset in Payload
Action	uint8	1	0

Field	Type	Description
Action	uint8	0: Hover (switch to Position mode), 1: Land Now, 2: Return home and land (This is called RTL by DJI, for ”Return To Launch”)

3.2.2 Camera Control Packet [*PID*=254U]

This packet tells the App and drone to start or stop sending images from the drones live video feed from a connected DJI camera.

Field	Type	Size	Offset in Payload
Action	uint8	1	0
TargetFPS	float	4	1

Field	Type	Description
Action	uint8	0: Stop video feed. 1: Start video feed
TargetFPS	float	The frame rate (frames per second) at which the App/drone should send images to the server. This will usually be lower than the framerate at which the drone is sending images to the mobile device so typically many frames received by the App will be discarded in between the frames that are forwarded to the server. Ignored if Action = 0.

3.2.3 Execute Waypoint Mission Packet [*PID*=253U]

This packet tells the App and drone to start flying a waypoint mission. If the drone is currently executing another mission it should be canceled and if a mode switch is necessary it should be made. When this packet is received the client should respond with an Acknowledgment packet.

Field	Type	Size	Offset in Payload
LandAtEnd	uint8	1	0
CurvedFlight	uint8	1	1
Start of repeated section			
Latitude	double	8	$2 + 40N$
Longitude	double	8	$10 + 40N$
Altitude	double	8	$18 + 40N$
CornerRadius	float	4	$26 + 40N$
Speed	float	4	$30 + 40N$
LoiterTime	float	4	$34 + 40N$
GimbalPitch	float	4	$38 + 40N$

Field	Type	Description
LandAtEnd	uint8	0: Hover after last waypoint. 1: Land after last waypoint.
CurvedFlight	uint8	0: Point-to-point flight, stopping at each waypoint. 1: curved flight that doesn't completely stop at waypoints.
Latitude	double	WGS84 Latitude of waypoint (degrees)
Longitude	double	WGS84 Longitude of waypoint (degrees)
Altitude	double	WGS84 altitude of waypoint (meters). This is not height above the home point, which is what DJI uses for waypoints. When creating a waypoint for the SDK, use altitude - home altitude.
CornerRadius	float	Ignored if CurvedFlight = 0. If CurvedFlight = 1, this is the radius of the arc (in meters) made when cutting the corner at this waypoint.
Speed	float	Speed of the drone between this waypoint and the next (m/s)
LoiterTime	float	Time to hover at this waypoint before moving to the next (s), implemented by adding a DJIWaypointActionTypeStay action to the waypoint. If NaN, no action should be added.
GimbalPitch	float	Pitch of DJI camera gimbal at this waypoint (degrees), implemented by adding a DJIWaypointActionTypeRotateGimbalPitch action to the waypoint. If NaN, no action should be added.

3.2.4 VirtualStick Command Packet [*PID*=252U]

This packet tells the App to issue a virtual stick command. If the drone is currently executing another mission it should be canceled and if a mode switch is necessary it should be made. When this packet is received the client should respond with an Acknowledgment packet.

Field	Type	Size	Offset in Payload
Mode	uint8	1	0
Yaw	float	4	1
V_x	float	4	5
V_y	float	4	9
HAG	float	4	13
timeout	float	4	17

Field	Type	Description
Mode	uint8	0: Mode A. 1: Mode B (See below)
Yaw	float	Vehicle yaw (degrees). 0=North, clockwise is positive
V_x	float	Speed (m/s) in North (Mode A) or forward (Mode B) direction
V_y	float	Speed (m/s) in East (Mode A) or vehicle-right (Mode B) direction
HAG	float	Height above ground (m)
timeout	float	Amount of time (s) after which, if another virtualstick command packet is not received and a mode switch is not issued, another command should be issued with 0 for both V_x and V_y (but other fields the same as this command).

Mode Definitions:

DJI Parameter	Mode A	Mode B
DJIVirtualStickVerticalControlMode	DJIVirtualStickVerticalControlModePosition	
DJIVirtualStickRollPitchControlMode	DJIVirtualStickRollPitchControlModeVelocity	
DJIVirtualStickYawControlMode	DJIVirtualStickYawControlModeAngle	
DJIVirtualStickFlightCoordinateSystem	...SystemGround	...SystemBody

3.3 Miscellaneous Notes

3.3.1 Hash Function

All packets sent over the socket include a 2-byte hash field. Any packet with an invalid hash is to be discarded.

The hash is computed starting with the *sync* field and covers all bytes up to (but not including) the *hash* field.

It is important to note that the value in a packets *size* field includes the *hash* field for that packet. That is, the size field is computed first, as if the hash field has already been added to the packet. After the *size* field is set, the hash for the packet is evaluated.

The hash field has structure:

hashA	hashB
1	1

Hash Field Structure The top row provides field names and the bottom row provides field widths (in bytes). If a field width is missing, it is variable and must be deduced from other information in the packet.

The two components of the hash field are computed as follows:

```
//Hash computation for byte sequence in "buffer", with length "N"
unsigned char hashA = 0U;
unsigned char hashB = 0U;
for (int i=0; i<N; i++) {
      hashA += buffer[i];
      hashB += hashA;
}
```