Hula Drone Expansion Interface Instructions

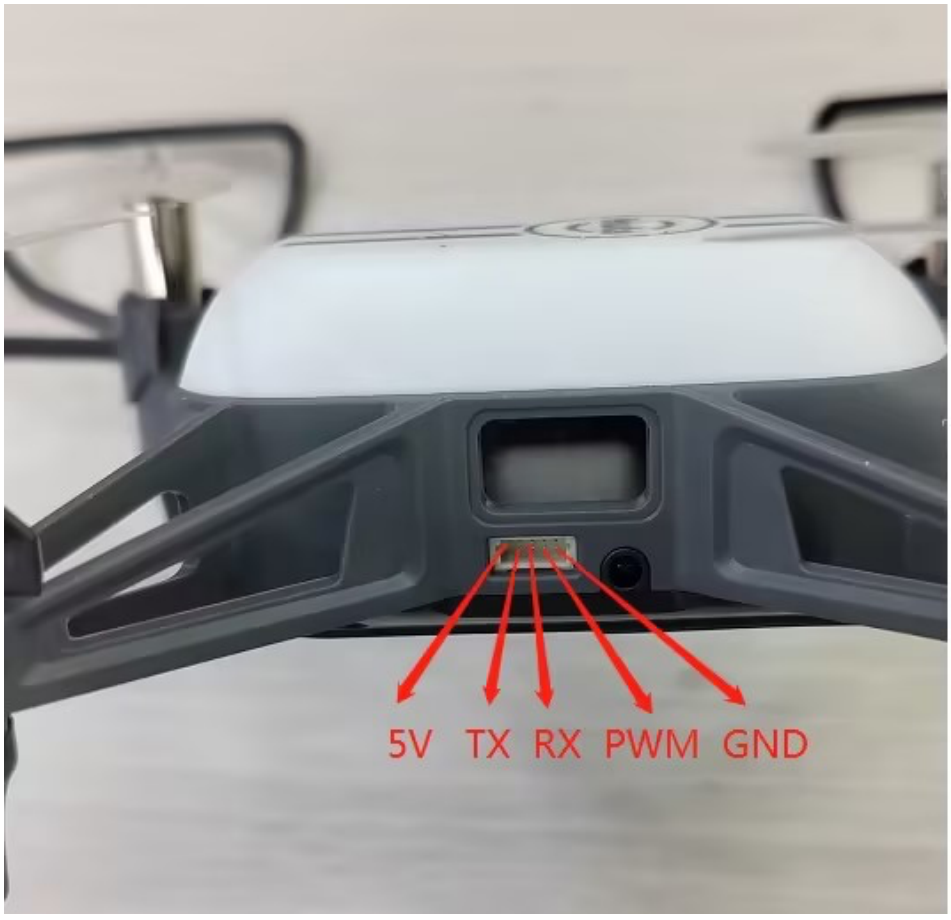
目录 1

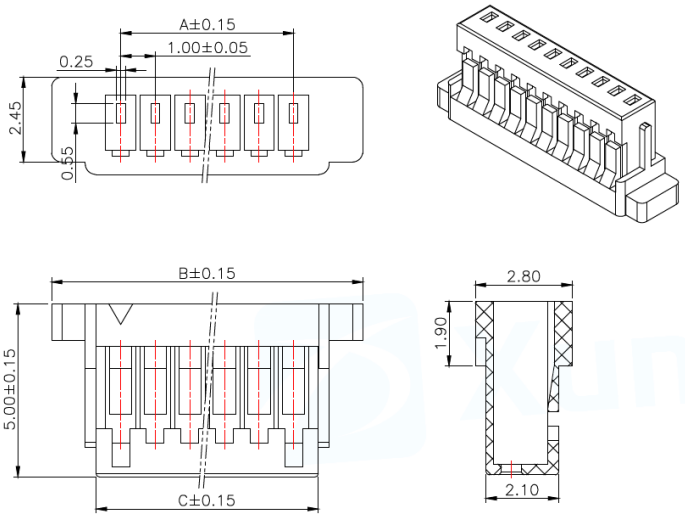
1. Expansion Connection

Hula has a reserved hardware expansion interface, its interface definition is shown in the following figure. The expansion interface can communicate with ESP32 module (or other microcontrollers), but can also directly drive the servo and solenoid and other devices.

Attention is required when expanding peripherals:

The maximum output of the UAV expansion interface is 5V 0.5A, and the maximum takeoff weight of Hula is 120g, the above limitations need to be taken into consideration when designing.





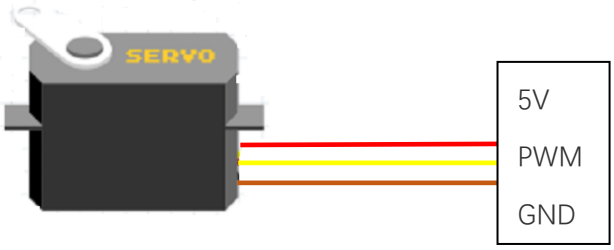
Part No	Pin	A	B	C
WAFER-SH1.0-2PJK	2	1.00	5.00	2.40
WAFER-SH1.0-3PJK	3	2.00	6.00	3.40
WAFER-SH1.0-4PJK	4	3.00	7.00	4.40
WAFER-SH1.0-5PJK	5	4.00	8.00	5.40
WAFER-SH1.0-6PJK	6	5.00	9.00	6.40
WAFER-SH1.0-7PJK	7	6.00	10.00	7.40
WAFER-SH1.0-8PJK	8	7.00	11.00	8.40
WAFER-SH1.0-9PJK	9	8.00	12.00	9.40
WAFER-SH1.0-10PJK	10	9.00	13.00	10.40
WAFER-SH1.0-11PJK	11	10.00	14.00	11.40
WAFER-SH1.0-12PJK	12	11.00	15.00	12.40
WAFER-SH1.0-13PJK	13	12.00	16.00	13.40
WAFER-SH1.0-14PJK	14	13.00	17.00	14.40
WAFER-SH1.0-15PJK	15	14.00	18.00	15.40
WAFER-SH1.0-16PJK	16	15.00	19.00	16.40

(Connector Specifications)

1.1 Servo Connection

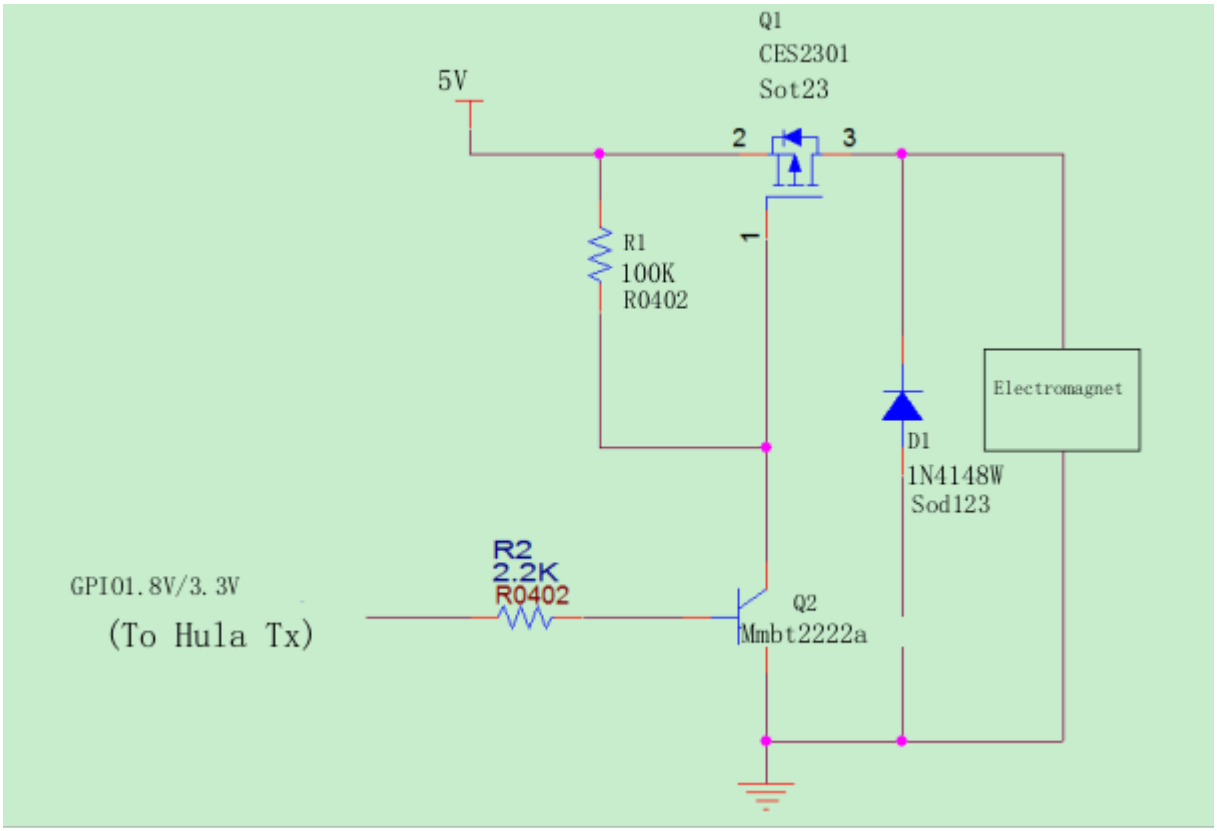
Direct drive servo connection schematic

:



1.2 Solenoid connection:

Electromagnet for connection schematic: (At present, the electromagnet circuit needs to be designed by the user with reference to the following diagram. The electromagnet needs to be selected 5V working voltage. The electromagnet reference link: https://detail.1688.com/offer/637062459601.html?spm=a360q.8274423.0.0. 7b854c9aLXedQ4)



1.3 ESP32 connection

Users can customize the development of extended functions through the ESP32 module, which can be connected to the aircraft through the expansion of the serial port or WIFI for communication.

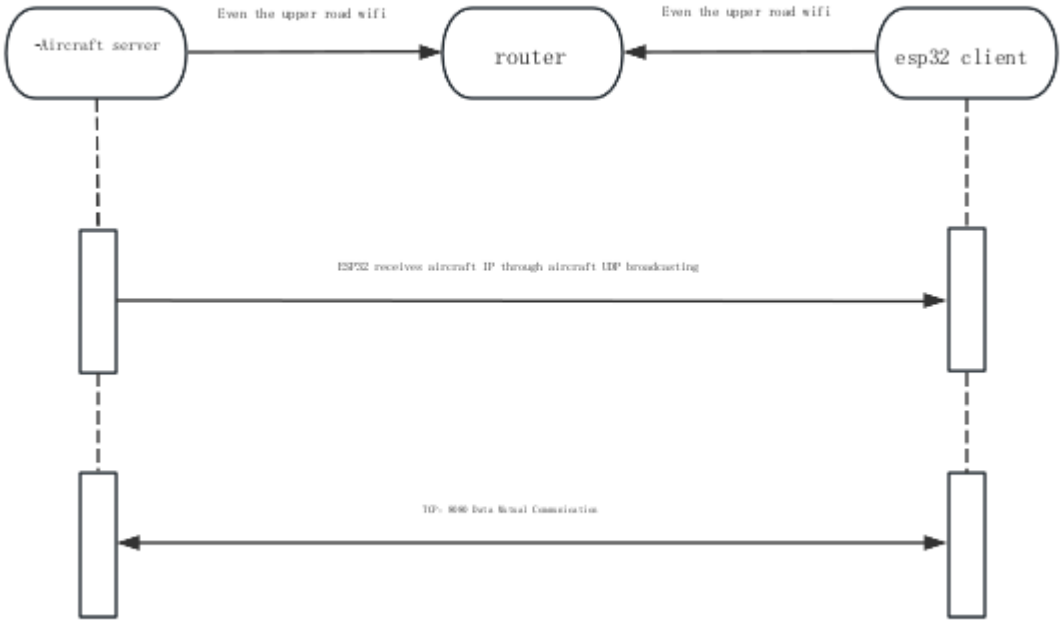
When using the serial port, the ESP32 is directly connected to the aircraft through the data cable according to the interface definition.

There are two ways to use WIFI communication. Since the ESP32 does not support 5G WIFI, the airplane or the AP of the router must use 2.4G WIFI. For the airplane, it uses Hula APP software settings - system - WiFi band settings 2.4G band.

Direct Connect Mode

For direct connection mode, the ESP32 connects to the airplane ap and establishes tcp directly. Implementation function: wifi_connect_sta; build_aptcpsocket.

Routing Mode



For routing mode, the ESP32 and the airplane both connect to the router. Then, the ESP32 receives ip from the airplane's udp broadcast and establishes tcp communication. Implementation function: wifi_connect_sta; build_routertcpsocket

2. Protocol Summary of ESP32

2.1 Sending Protocol

First ESP32, send buf is 32 bytes, the protocol is described in the following table:

Domain	Offset position	Size (bytes)	Detailed description
SOF	0	1	Start of frame header, fixed at 0x88.
MSG_ID	1	1	Message ID
PAYLOAD	2	28	Message data, depending on the message ID
CheckSum [0-7].	30	1	Byte 0 to byte 28
CheckSum 【8-15】.	31	1	Word 0 to byte 28

After the end of sending, receive the return answer buf is 1 byte, buf[0] is equal to the MSG_ID of the sending request, indicating that the reception is successful

Implemented by SerialProtocolSend function.

2.2 Asking for running status

Send two bytes of buf, buf[0] fixed equal to 0x66, buf[1] equal to MSG_ID

Return four bytes buf, buf[3] fixed equal to 0x05, buf[0] equal to MSG_ID, buf[1] equal to run result

Implemented through the WaitForRunToEnd function, the function 100ms interval cycle to ask the state, but the corresponding MSG_ID function returns after the end of the execution

2.3 Receiving protocol

Send two bytes of buf, buf[0] fixed equal to 0x66, buf[1] equal to MSG_ID

The received buf is 32 bytes, the specific content of the following table:

Domain	**Offset position**	**Size (bytes)**	**Detailed description**
SOF	0	1	Start of frame header, fixed at 0x88.
MSG_ID	1	1	Message ID
PAYLOAD	2	28	Message data, depending on the message ID
CheckSum [0-7].	30	1	Byte 0 to byte 28
CheckSum 【8-15】.	31	1	Word 0 through byte 28

Realized through the GetDataProtocol function

3. Basic function interface

Set PAYLOAD content control function according to MSG_ID

3.1 Tripod Head Angle

3.1.1 Control Command

Control the angle of the Tripod head, MSG_ID is equal to 0x01.

Need to set the content as follows:

```
Protocol[1] = 0x01
Protocol[2] = Val           //Turn to the angle value
Protocol[3] = Direction     //Direction: 0 positive 1 negative
```

3.1.2 Get command

Get current Tripod head angle, MSG_ID is equal to 0x21

Get the content in the following way:

```
temp = self.GetDataProtocol(0x21)
angle = self.byte4float([temp[1],temp[2],temp[3],temp[4]])
```

3.2 Tripod Head Angle Calibration

3.2.1 Control Commands

Restore the head to the horizontal position 0 degrees, MSG_ID is equal to 0x02

The following settings are required:

```
Protocol[1] = 0x02
```

3.3 Graphic Transmission

3.3.1 Control Command

MSG_ID is equal to 0x03.

The following settings are required:

```
Protocol[1] = 0x03
Protocol[2] = Switch     //0 on 1 off
```

3.4 Laser

3.4.1 Control commands

MSG_ID is equal to 0x04.

The following settings are required:

```
Protocol[1] = 0x04
Protocol[2] = Val    //Continuous shot volume
Protocol[3] = Mode  //0 single shot 1 continuous shot (with bullet limit) 2 turn on laser reception 3 turn off laser reception 4 continuous shot all the time (no bullet limit ) 5 turn off laser
```

3.4.2 Get Command

MSG_ID is equal to 0x22.

Get the content in the following way:

```
temp = self.GetDataProtocol(0x22) #GetDataProtocolonly returns if SN code is detected
print(temp.decode("UTF-8") )     #SN code as string
```

3.5 Patrol

3.5.1 Control Commands

MSG_ID is equal to 0x05.

The content to be set is as follows:

```
#Mode 0:go forward, ignore intersection; 1: go forward, ends
moving when encountering intersection; 2: hover at intersection;
3: hover on track; 10: stop patrol
#dist:distance(cm)  tv: patrol time(s)  color: patrol color gamut
Protocol[1] = 0x05
Protocol[2] = Mode & 0x00ff
Protocol[3] = (Mode >> 8) & 0xff
Protocol[4] = (Mode >> 16) & 0xff
Protocol[5] = (Mode >> 24) & 0xff
Protocol[6] = dist & 0x00ff
Protocol[7] = (dist >> 8) & 0xff
Protocol[8] = (dist >> 16) & 0xff
Protocol[9] = (dist >> 24) & 0xff
Protocol[10] = tv & 0x00ff
Protocol[11] = (tv >> 8) & 0xff
Protocol[12] = (tv >> 16) & 0xff
Protocol[13] = (tv >> 24) & 0xff
Protocol[14] = color & 0x00ff
Protocol[15] = (color >> 8) & 0xff
Protocol[16] = (color >> 16) & 0xff
Protocol[17] = (color >> 24) & 0xff
self.SerialProtocolSend(Protocol)
if(Mode ! = Protocol) if(Mode !
  self.WaitForRunToEnd(0x05) // wait for the end of the moving  execution
```

3.6 QR code recognition

3.6.1 Control Instruction

MSG_ID is equal to 0x06.

The content needs to be set as follows:

```
Protocol = bytearray(10)
Protocol[1] = 0x06
Protocol[2] = Run_rate & 0x00ff
Protocol[3] = (Run_rate >> 8) & 0xff
Protocol[4] = (Run_rate >> 16) & 0xff
Protocol[5] = (Run_rate >> 24) & 0xff
Protocol[6] = Qr_Id
Protocol[7] = Qr_background_grayscale
Protocol[8] = Mode
Protocol[9] = 0x00
data = struct.pack("d",Time_duration)
data1 = struct.pack("f",Search_radius)
data2 = struct.pack("f",Qr_size)
data3 = struct.pack("d",0)
Protocol.extend(data)
Protocol.extend(data1)
Protocol.extend(data2)
Protocol.extend(data2)
```

3.6.2 Getting Status Instructions

MSG_ID is equal to 0x23

Get the status information in the following way:

```
#float x_com;body coordinate system, x direction compensation, x_com=0.5, means the aircraft needs to fly forward 0.5m
#float y_com;Compensation for y direction in fuselage coordinate system, y_com=0.5, means the airplane needs to fly to the right by 0.5m.
#float z_com;Compensation for z-direction in fuselage coordinate system, z_com=0.5, means the airplane needs to fly down 0.5m.
#float yaw_com;body coordinate system, yaw angle compensation, yaw_com=0.5, means the airplane turns clockwise
#int8_t status;the execution status of the task,0 execution in the detection of QR code success, 1 execution in the detection of QR code failure
temp = self.GetDataProtocol(0x23)
status = temp[22]
x_com = f09_api.byte4float([temp[2], temp[3], temp[4], temp[5]])
y_com = f09_api.byte4float([temp[6], temp[7], temp[8], temp[9]])
z_com = f09_api.byte4float([temp[10], temp[11],temp[12],temp[13]])
yaw_com = f09_api.byte4float([temp[14], temp[15], temp[16],temp[17]])
```

3.7 Color Recognition

3.7.1 Control Instructions

MSG_ID is equal to 0x07

Need to set the content as follows:

```
Protocol[1] = 0x07
Protocol[2] = Mode #Mode:1 start, run one frame
```

3.7.2 Get command

MSG_ID equals 0x24

Get the content in the following way:

```
temp = self.GetDataProtocol(0x24) #Get the result of color
recognition, no data over will block and wait
state = temp[2] #state:0fail 1success
r = temp[3]    #r,g,b: color gamut
g = temp[4]
b = temp[5]
```

3.8 Takeoff and landing control

3.8.1 Control Instructions

MSG_ID is equal to 0x81

Need to set the content as follows:

```
Protocol[1] = 0x81
Protocol[2] = Switch //0 takeoff 1 landing
self.SerialProtocolSend(Protocol)
self.WaitForRunToEnd(0x81) //wait for execution to end
```

3.9 Unlocking and Locking

3.9.1 Control Instruction

MSG_ID is equal to 0x82

Need to set the content as follows:

```
Protocol[1] = 0x82
```

3.10 Flight Speed Distance Control

3.10.1 Control Instruction

MSG_ID equals 0x83

Need to set the content as follows:

```
Protocol[1] = 0x83
Protocol[2] = Dist #Dist:Flight Distance (cm)
Protocol[3] = Speed #Speed:Flight speed (0.01m/s) *Do not exceed the maximum flight speed of the drone
Protocol[4] = Switch #Switch:0 up 1 down 2 front 3 back 4 left 5 right
self.SerialProtocolSend(Protocol)
self.WaitForRunToEnd(0x83) #Wait for execution to end
```

3.10.2 Getting Instructions

MSG_ID is equal to 0xf1

Need to set the content as follows:

```
    temp = self.GetDataProtocol(0xf1)
    #accx:x-axis acceleration (unit: 0.01m/s2), plus or minus
    represents the direction.
    #vel_x: x-axis velocity (unit: 0.01m/s), plus or minus represents
    the direction.
    accx = (temp[3] << 8) | temp[2]
    accy = (temp[5] << 8) | temp[4]
    accz = (temp[7] << 8) | temp[6]
    vel_x = (temp[9] << 8) | temp[8]
    vel_y = (temp[11] << 8) | temp[10]
    vel_z = (temp[13] << 8) | temp[12]
    if(accx & 0x8000):
        accx = -(0xffff - accx + 1)
    if(accy & 0x8000):
        accy = -(0xffff - accy + 1)
    if(accz & 0x8000):
        accz = -(0xffff - accz + 1)
    if(vel_x & 0x8000):
        vel_x = -(0xffff - vel_x + 1)
    if(vel_y & 0x8000):
        vel_y = -(0xffff - vel_y + 1)
    if(vel_z & 0x8000):
      vel_z = -(0xffff - vel_z + 1)
```

3.11 Rollover

3.11.1 Control Commands

MSG_ID is equal to 0x84

Need to set the content as follows:

```
Protocol[1] = 0x84
Protocol[2] = Switch #0 before 1 after
self.SerialProtocolSend(Protocol)
self.WaitForRunToEnd(0x84) # wait for execution to end
```

3.11 Heading Angle

3.11.1 Control Instruction

MSG_ID is equal to 0x85

Need to set the content as follows:

```
Protocol[1] = 0x85
Protocol[2] = Speed #Speed:Angle
Protocol[3] = Switch #Switch:0 counterclockwise 1 clockwise
self.SerialProtocolSend(Protocol)
self.WaitForRunToEnd(0x85) #Wait for execution to end
```

3.11 Light Control

3.11.1 Control Instruction

MSG_ID is equal to 0x87, set RGB lighting

Need to set the content as follows:

```
Protocol[1] = 0x87
Protocol[2] = R
Protocol[3] = G
Protocol[4] = B        #RGB:RGB VALUE
Protocol[5] = Mode        #Mode:mode 0x01:constant light 0x04:RGB
3-color cycle  0x8:running light 0x20:flash
Protocol[6] = Time   #Time:Duration (sec) 0 all the time
```

MSG_ID equals 0x86, cancel rgb

Need to set the content as follows:

```
Protocol[1] = 0x86
```

3.12 Obstacle avoidance

3.12.1 Receiving commands

MSG_ID equals 0xf2

Need to set the content as follows:

```
temp = self.GetDataProtocol(0xf2)
barrier = temp[14]
```

3.13 Altitude

3.13.1 Receive instruction

MSG_ID is equal to 0xf3

Need to set the content as follows:

```
temp = self.GetDataProtocol(0xf3)
tof = temp[15]  #TOF altitude
baro = self.byte4float([temp[16], temp[17], temp[18], temp[19]])
#barometer height
```