# Table of Contents

GREMSY

# GREMSY SDK
# INTEGRATION GUIDE

Release date: 8 May 2022

Revision Number: 3.0.0

# Introduction

## *SDK Documentation Home*

This document helps you get started with the various aspects of building an SDK application and describes the protocol which can be used by software to control Gremsy's gimbal. The gimbal can be controlled using serial via the COM2 connector.

Besides, Gremsy gimbal also supports MAVProxy with command gimbal point and MAVSDK to control the gimbal and autopilot system.

### MAVProxy

Link: Gimbal Management — MAVProxy documentation (ardupilot.org)

### MAVSDK

Link: Introduction · MAVSDK Guide (mavlink.io)

## *Coordinate System*

### Earth Frame

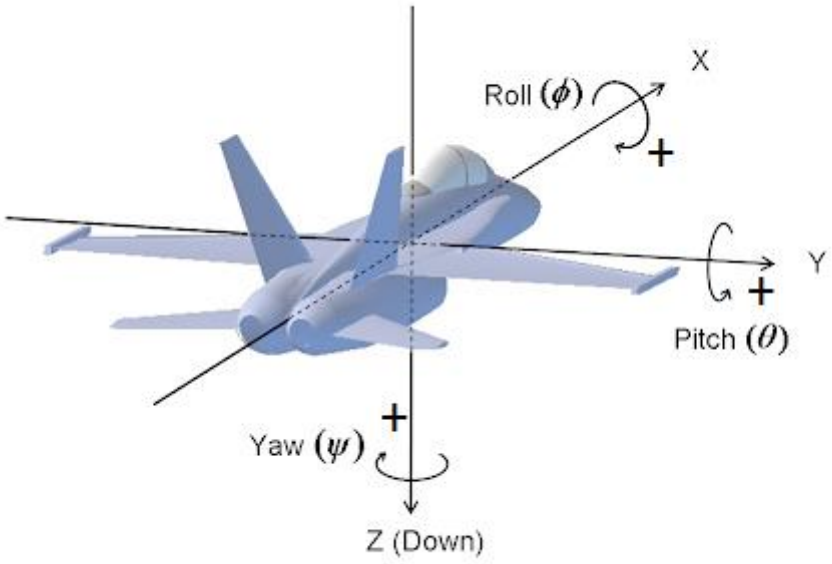
The Earth frames are aircraft's location dependent, the frames are defined tangent to the lines of geographical coordinates. The convention used in SDK is Local North, East, Down (NED) coordinates. Three perpendicular axes are defined such that the origin is the center of

mass, the **X-axis** is pointing North, and the **Y-axis** is pointing East. Using the coordinate right-hand rule, the **Z-axis** is pointing Down.

### Body Frame

The body frame is in the Earth frame rotated so that the **X-axis** is pointing forward the aircraft heading, the **Y-axis** is pointing to the right, and the **Z-axis** is pointing down.

Gimbal rotation is also described in these frames with Euler Roll, Pitch, and Yaw angles rotating around X, Y, and Z axes.




## *Works Modes*

The gimbal has several work modes that define how the gimbal follows aircraft movement, and how many axes are available for control.

- Follow Mode: Yaw will follow the aircraft heading.
- Lock Mode:  The gimbal can move independently from the aircraft.

## *Moving the Gimbal*

The gimbal can be controlled in two input modes:

- Angle Mode: Move to a target attitude
- Speed Mode: Move at a target rate for the individual axis.

When using Angle Mode, the gimbal moves to target attitude in Earth frame if operating in Lock mode. Otherwise, the gimbal moves to the target attitude in the Body frame if operating in Follow mode.

When using Speed Mode, the gimbal moves at the absolute rate in the Earth frame.

# *General Protocol Overview*

The API is implemented based on the MAVLink protocol. Mavlink provides an open data format for interaction as well as a suite of tools to assist the programmer in developing and testing the interface.

Gimbal uses Mavlink v2.0 message and communicates with other components at 115200 baudrate and 8N1.

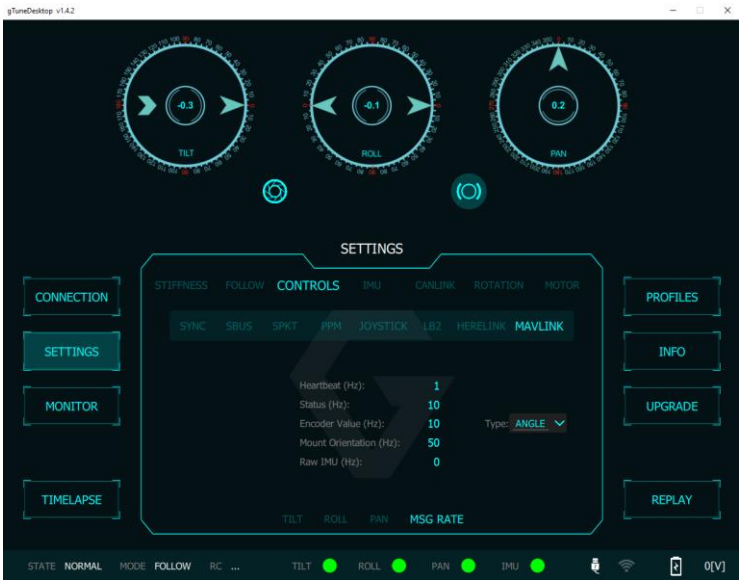
**NOTE:**

- **Firmware:**
  - Pixy Series: v756_Official and above.
  - T3 Series: v756_Official and above.
  - T7V1: v756_Official and above.
  - S1V3: v756_Official and above.
  - Mio: v756_Official and above.
- **gTune Desktop**
  - gTuneDesktop_v1.4.2 or above.
- **Link:** https://github.com/Gremsy

# *Enable SDK API*

## Gimbal Data Tranmissions over MAVLink

- Gimbal sends status data through MAVLink messages. Other components can configure the gimbal to transmit data at the desired rate via MAVLink commands.
- The SDK supports reading data and configuring the gimbal with open APIs.




## HEARTBEAT (Message ID: #0)

The Gimbal sends HEARTBEAT messages at approximately 1Hz. It sends through COM2 and COM4 on the QR after receiving any HEARTBEAT messages from other components. Take advantage of this message to check the connection between your devices and the gimbal.

## SYS_STATUS (Message ID: #1)

The gimbal pushes status information with SYS_STATUS messages at approximately 1Hz. Status information includes the working modes, sensor states, and motor states.

See Gimbal_Interface::gimbal_status_t structure for more details.

## RAW_IMU (Message ID: #27)

The gimbal provides raw imu data for applications that want to inspect data and develop algorithms.

Acceleration (raw): range 4g = 8192.

Gyro(raw): range 1000dps = 32768.

| Field | Type | Description | Gimbal Implementation |
|---|---|---|---|
| time_usec | uint64_t | Timestamp (us) | Same as official |
| xacc | int16_t | X acceleration (raw) | Same as official |
| yacc | int16_t | Y acceleration (raw) | Same as official |
| zacc | int16_t | Z acceleration (raw) | Same as official |
| xgyro | int16_t | Angular speed around the X-axis | Same as official |
| ygyro | int16_t | Angular speed around the Y-axis | Same as official |
| zgyro | int16_t | Angular speed around the Z-axis | Same as official |
| xmag | int16_t | X Magnetic field | Ignored |
| ymag | int16_t | Y Magnetic field | Ignored |
| zmag | int16_t | Z Magnetic field | Ignored |

## MOUNT_STATUS (Message ID: #158)

This message provides raw encoder values or encoder angle values of the gimbal.

The encoder resolution: $2^{16}$ bits.

| Field Name | Type | Description | Gimbal Implementation |
|---|---|---|---|
| pointing_a | int32_t | Pitch (cdeg) | Tilt encoder value |
| pointing_b | int32_t | Roll (cdeg) | Roll encoder value |
| pointing_c | int32_t | Yaw (cdeg) | Pan encoder value |
| target_system | uint8_t | System ID | Target system ID |
| target_component | uint8_t | Component ID | Target component ID |

## MOUNT_ORIENTATION (Message ID: #265)

This message contains information about gimbal attitude. Use this message as feedback when controlling the gimbal.

| Field Name | Type | Description | Gimbal Implementation |
|---|---|---|---|
| time_boot_ms | int32_t | Timestamp (ms) | Same as official |
| roll | float | Roll in the Earth frame (deg) | Same as official |
| pitch | float | Pitch in the Earth frame (deg) | Same as official |
| yaw | float | Yaw in the Body frame (deg) | Same as official |
| yaw_absolute** | float | Yaw in the Earth frame | Same as official |

## GIMBAL_DEVICE_INFORMATION

This message contains gimbal information.

| Field | Type | Description | Gimbal Implementation |
|---|---|---|---|
| time_boot_ms | uint32_t | Timestamp (ms) | Same as official |
| vendor_name | char[32] | Vendor's name | Same as official |
| model_name | char[32] | Model's name | Same as official |
| custom_name | char[32] | A custom name is given by the user | Same as official |
| firmware_version | uint32_t | Firmware version | Same as official |
| hardware_version | uint32_t | Hardwar version | Ignored |
| uid | uint64_t | Hardware uid | Ignored |
| cap_flags | uint16_t | Capability flags | Same as official |
| custom_cap_flags | uint16_t | Specific capability flags | Ignored |
| roll_min | float | Minimum hardware roll angle | Same as official |
| roll_max | float | Maximum hardware roll angle | Same as official |
| pitch_min | float | Minimum hardware pitch angle | Same as official |
| pitch_max | float | Maximum hardware pitch angle | Same as official |
| yaw_min | float | Minimum hardware yaw angle | Same as official |
| yaw_max | float | Maximum hardware yaw angle | Same as official |

### GIMBAL_DEVICE_ATTITUDE_STATUS

The gimbal broadcast this message to report status.

| Field | Type | Description | Gimbal Implementation |
|---|---|---|---|
| target_system | uint8_t | System ID | Same as official |
| target_component | uint8_t | Component ID | Same as official |
| time_boot_ms | uint32_t | Timestamp (ms) | Same as official |
| flags | uint16_t | Gimbal flags | Same as official |
| q | float[4] | Gimbal attitude | Same as official |
| angular_velocity_x | float | X component of angular velocity (rad/s) | Same as official |
| angular_velocity_y | float | Y component of angular velocity (rad/s) | Same as official |
| angular_velocity_z | float | Z component of angular velocity (rad/s) | Same as official |
| failure_flags | uint32_t | Failure flags | Same as official |

## Gimbal Control Messages

### Common Messages

Common messages are those used for gimbal control and are independent of the MAVLink Gimbal Protocol version.

### COMMAND_LONG (Message ID: #76)

| Field | Type | Description |
|---|---|---|
| target_system | uint8_t | Gimbal System ID |
| target_component | uint8_t | Gimbal Component ID |
| command | uint16_t | Command ID |
| confirmation | uint8_t | 0 |
| param1 | float | Parameter 1. Default 0 |
| param2 | float | Parameter 2. Default 0 |
| param3 | float | Parameter 3. Default 0 |
| param4 | float | Parameter 4. Default 0 |
| param5 | float | Parameter 5. Default 0 |
| param6 | float | Parameter 6. Default 0 |
| param7 | float | Parameter 7. Default 0 |

**command: MAV_CMD_USER_1**

This command is used to turn the gimbal motor on or off.

| Param (: Label) | Description | Values |
|---|---|---|
| 7: | ON/OFF | 0: OFF, 1: ON |

**command: MAV_CMD_DO_MOUNT_CONFIGURE**

This command is used to configure gimbal mount mode. Detail implementation is in SDK.

| Param (: Label) | Description | Values |
|---|---|---|
| 1: Mode | Mount operation mode | See MAV_MOUNT_MODE and APIs |

**command: MAV_CMD_PREFLIGHT_REBOOT_SHUTDOWN**

This command is used to reboot the gimbal.

| Param (: Label) | Description | Values |
|---|---|---|
| 4: Flag | Flag to check reboot command | 1 |


Gremsy Co., Ltd
No. 22 Street 18, Linh Dong Ward, Thu Duc Dist., HCM City , VN https://gremsy.com Tel: 028-3720 5221 / Fax: 028-3720 5231

**command: MAV_CMD_REQUEST_MESSAGE**

This command is used to request gimbal emit a single instance of a specified message.

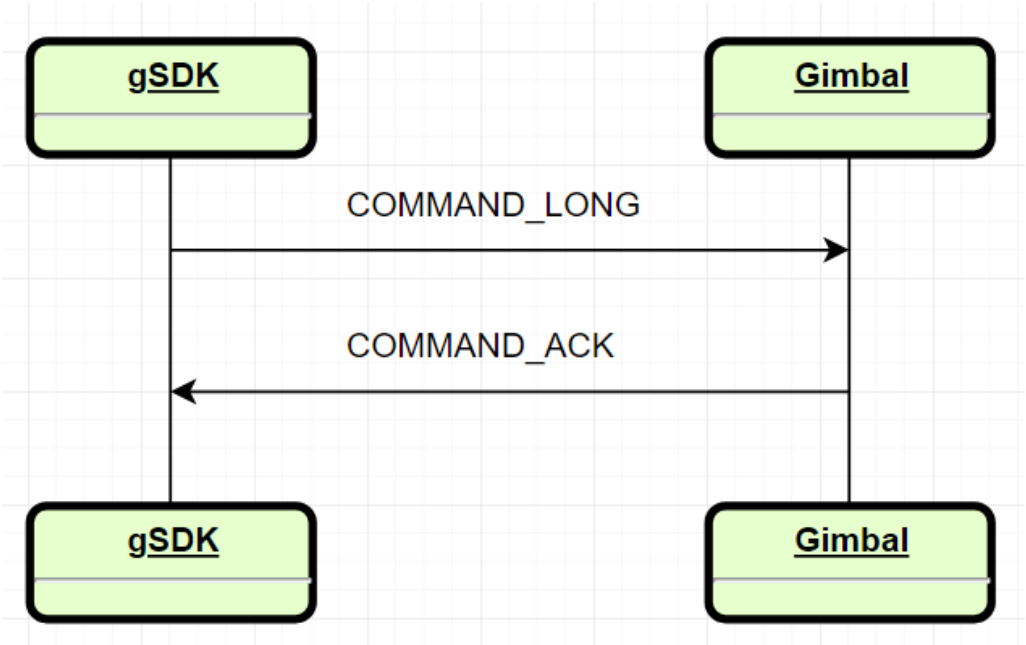
| Param (: Label) | Description | Values |
|---|---|---|
| 1: Message ID | The MAVLink message ID of the requested message | Message ID |

**command: MAV_CMD_SET_MESSAGE_INTERVAL**

This command is used to request and set the interval between messages for a particular MAVLink message ID.

| Param (: Label) | Description | Values |
|---|---|---|
| 1: Message ID | The MAVLink message ID of the requested message | Message ID |
| 2: Interval | The interval between 2 messages. Set to -1 to disable and 0 to request the default rate | Interval (us) |

## COMMAND_ACK (Message ID: #77)



The MAVLink command protocol allows for guaranteed delivery of MAVLink commands. The gimbal responds to commands from other components with COMMAND_ACK. Additionally, the gimbal waits for a COMMAND_ACK response from other components when sending commands.

| Field | Type | Description |
|---|---|---|
| commad | uint16_t | Command ID |
| result | uint8_t | Result of command. |
| progress ** | uint8_t | Ignored |
| result_param2 ** | int32_t | Ignored |
| target_system ** | uint8_t | Target System ID |
| target_component ** | uint8_t | Target Component ID |

## MAVLink Gimbal Protocol V1

The messages below are specific messages implemented for gimbal control in MAVLink Gimbal Protocol V1.

### COMMAND_LONG (Message ID: #76)

**command: MAV_CMD_DO_MOUNT_CONTROL**

This command is used to control the gimbal to move to the target attitude or at the target speed.

| Param (: Label) | Description | Values |
|---|---|---|
| 1: Pitch | Pitch control value | Depends on the input mode (deg or deg/s) |
| 2: Roll | Roll control value | Depends on the input mode (deg or deg/s) |
| 3: Yaw | Yaw control value | Depends on the input mode (deg or deg/s) |
| 6: Input mode | Input mode | See Gimbal_Protocol::input_mode_t |
| 7: Mount mode | Mount mode | MAV_MOUNT_MODE_MAVLINK_TARGETTING |

**command: MAV_CMD_USER_2**

This command is used to change the gimbal control mode Lock/Follow.

| Param (: Label) | Description | Values |
|---|---|---|
| 6: Reset mode | Reset mode | See Gimbal_Protocol::gimbal_reset_mode_t |
| 7: Control mode | Control mode | See Gimbal_Protocol::control_mode_t |

## MAVLink Gimbal Protocol V2

### GIMBAL_DEVICE_SET_ATTITUDE

This message is used to control the gimbal device.

| Field | Type | Description |
|---|---|---|
| target_system | uint8_t | Gimbal System ID |
| target_component | uint8_t | Gimbal Component ID |
| flags | uint16_t | Low level gimbal flags. |
| q | float[4] | Target attitude in quaternion form |
| angular_velocity_x | float | X component of angular velocity (rad/s) |
| angular_velocity_y | float | Y component of angular velocity (rad/s) |
| angular_velocity_z | float | Z component of angular velocity (rad/s) |

### AUTOPILOT_STATE_FOR_GIMBAL_DEVICE

This message contains the autopilot state for the gimbal device.

| Field | Type | Description |
|---|---|---|
| target_system | uint8_t | Gimbal System ID |
| target_component | uint8_t | Gimbal Component ID |
| time_boot_us | uint64_t | Timestamp (us) |
| q | float[4] | Autopilot attitude in quaternion form |
| q_estimated_delay_us | uint32_t | Estimated delay of the attitude data |

| | | |
|---|---|---|
| v_x | float | X Speed in NED (m/s) |
| v_y | float | Y Speed in NED (m/s) |
| v_z | float | Z Speed in NED (m/s) |
| v_estimated_delay_us | uint32_t | Estimated delay of the speed data |
| feed_forward_angular_velocity_z | float | Feedforward Z component of angular velocity (rad/s) |
| estimator_status | uint16_t | Bitmap indicating which estimator outputs are valid |
| landed_state | uint8_t | The landed state |

# APIs Reference

## Public Types

| Type | Description |
|---|---|
| struct Gimbal_Interface::time_stamps_t | Gimbal messages timestamps |
| enum Gimbal_Interface::control_direction_t | Gimbal axis control direction |
| enum Gimbal_Interface::control_motor_t | Gimbal motor mode ON/OFF |
| enum Gimbal_Interface::operation_state_t | Gimbal operation state |
| struct Gimbal_Interface::fw_version_t | Gimbal firmware version |
| struct Gimbal_Interface::gimbal_status_t | Gimbal status |
| struct Gimbal_Interface::gimbal_config_axis_t | Gimbal axis motion parameters |
| struct Gimbal_Interface::gimbal_motor_control_t | Gimbal axis stiffness parameters |
| enum Gimbal_Interface::MAVLINK_PROTO | MAVLink gimbal protocol version |
| struct Gimbal_Interface::limit_angle_t | Gimbal axis software limit |
| struct Gimbal_Interface::imu_t | Gimbal imu data type |
| enum Gimbal_Protocol::result_t | Enum described function return status |
| enum Gimbal_Protocol::control_mode_t | Gimbal control mode |
| enum Gimbal_Protocol::gimbal_reset_mode_t | Gimbal reset mode |

## Public Member Functions

| Type | Name | Description |
|---|---|---|
| | Gimbal_Interface(Serial_Port *serial_port, uint8_t sysid, uint8_t compid, MAVLINK_PROTO proto) | Constructor. Creates a gimbal interface for a specific system and MAVLink Gimbal Protocol version. |
| | ~Gimbal_Interface() | Destructor. |
| void | start() | Starts the gimbal interface. |
| void | stop() | Stops the gimbal interface. |
| void | start_read_thread() | Internal use only. |
| void | start_write_thread() | Internal use only. |
| void | handle_quit() | Quit handler. |
| bool | get_flag_exit() | Checks if the interface has stopped. |
| bool | get_connection() | Check if the interface has connected to the gimbal device. |
| bool | present() | Check if the gimbal is ready to control. |
| Gimbal_Protocol:: result_t | set_gimbal_reboot() | Reboots gimbal. |
| Gimbal_Protocol:: result_t | set_gimbal_rc_input_sync() | Sets the gimbal to RC Input mode, will return when the command is responded to. This function is blocking. |
| Gimbal_Protocol:: result_t | set_gimbal_motor( control_motor_t type) | Sets the gimbal motor ON/OFF. |
| Gimbal_Protocol:: control_mode_t | get_gimbal_mode() | Gets the gimbal control mode. |

| | | |
|---|---|---|
| Gimbal_Protocol:: result_t | set_gimbal_reset_mode( gimbal_reset_mode_t reset_mode) | Resets the gimbal with a specific mode. |
| Gimbal_Protocol:: result_t | set_gimbal_lock_mode_sync() | Sets the gimbal to Lock mode, will return when the command is responded to. This function is blocking. |
| Gimbal_Protocol:: result_t | set_gimbal_follow_mode_sync() | Sets the gimbal to Follow mode, will return when the command is responded to. This function is blocking. |
| Gimbal_Protocol:: result_t | set_gimbal_rotation_sync(float pitch, float roll, float yaw) | Rotates the gimbal to target attitude, will return when the command is responded to. This function is blocking. |
| Gimbal_Protocol:: result_t | set_gimbal_rotation_rate_sync( float pitch, float roll, float yaw) | Rotates the gimbal at target rate, and will return when the command is responded to. This function is blocking. |
| Gimbal_Interface:: gimbal_status_t | get_gimbal_status() | Gets the gimbal status. |
| Gimbal_Interface:: imu_t | get_gimbal_raw_imu() | Gets the gimbal raw imu data. |
| attitude<float> | get_gimbal_attitude() | Gets the gimbal attitude. |
| attitude<int16_t> | get_gimbal_encoder() | Gets the gimbal encoder value, depending on the encoder type send. |
| Gimbal_Interface:: timestamps_t | get_gimbal_timestamps() | Gets the gimbal timestamps. |
| Gimbal_Interface:: fw_version_t | get_gimbal_version() | Gets the gimbal firmware version. |

| | | |
|---|---|---|
| Gimbal_Protocol:: result_t | set_gimbal_config_tilt_axis(const gimbal_config_axis_t &config) | Configures the gimbal tilt axis motion parameters. |
| Gimbal_Interface:: gimbal_config_axis_t | get_gimbal_config_tilt_axis() | Gets the gimbal tilt axis motion parameters |
| Gimbal_Protocol:: result_t | set_gimbal_config_roll_axis(const gimbal_config_axis_t &config) | Configures the gimbal roll axis motion parameters. |
| Gimbal_Interface:: gimbal_config_axis_t | get_gimbal_config_roll_axis() | Gets the gimbal roll axis motion parameters |
| Gimbal_Protocol:: result_t | set_gimbal_config_pan_axis(const gimbal_config_axis_t &config) | Configures the gimbal pan axis motion parameters. |
| Gimbal_Interface:: gimbal_config_axis_t | get_gimbal_config_pan_axis() | Gets the gimbal pans axis motion parameters |
| Gimbal_Protocol:: result_t | get_gimbal_motor_control( gimbal_motor_control_t &tilt, gimbal_motor_control_t &roll, gimbal_motor_control_t &pan, uint8_t &gyro_filter, uint8_t &output_filter, uint8_t &gain) | Get the gimbal motor configuration. |
| Gimbal_Protocol:: result_t | set_gimbal_motor_control(const gimbal_motor_control_t &tilt, const gimbal_motor_control_t &roll, const gimbal_motor_control_t &pan, uint8_t gyro_filter, uint8_t output_filter, uint8_t gain) | Configures the gimbal motor parameters. |
| Gimbal_Protocol:: result_t | set_msg_encoder_rate(uint8_t rate) | Sets the gimbal encoder messages rate.s |
| Gimbal_Protocol:: result_t | set_msg_mnt_orient_rate(uint8_t rate) | Sets the gimbal mount orientation messages rate. |
| Gimbal_Protocol:: result_t | set_msg_attitude_status_rate(uint8_t rate) | Sets the gimbal attitude status messages rate. |

| | | |
|---|---|---|
| Gimbal_Protocol:: result_t | set_msg_raw_imu_rate(uint8_t rate) | Sets the gimbal raw imu messages rate. |
| Gimbal_Protocol:: result_t | set_gimbal_encoder_type_send(bool type) | Sets the gimbal encoder type send. |
| Gimbal_Protocol:: result_t | request_gimbal_device_info() | Requests the gimbal device information. |
| bool | get_gimbal_encoder_type_send() | Gets the gimbal encoder type send. |
| Gimbal_Protocol:: result_t | set_gimbal_combine_attitude(bool flag) | Sets the gimbal enabling reduced yaw drifting. |
| Gimbal_Protocol:: result_t | set_limit_angle_pitch(const limit_angle_t &limit_angle) | Sets the gimbal pitch axis limit. |
| Gimbal_Interface:: limit_angle_t | get_limit_angle_pitch() | Gets the gimbal pitch axis limit. |
| Gimbal_Protocol:: result_t | set_limit_angle_yaw(const limit_angle_t &limit_angle) | Sets the gimbal yaw axis limit. |
| Gimbal_Interface:: limit_angle_t | get_limit_angle_yaw() | Gets the gimbal yaw axis limit. |
| Gimbal_Protocol:: result_t | set_limit_angle_roll(const limit_angle_t &limit_angle) | Sets the gimbal roll axis limit. |
| Gimbal_Interface:: limit_angle_t | get_limit_angle_roll() | Gets the gimbal roll axis limit. |

**NOTE: Please refer to the User Manual to configure gimbal parameters for the best performance.**

# *Development Workflow*

## Prerequisites

To build an SDK based application the following are required

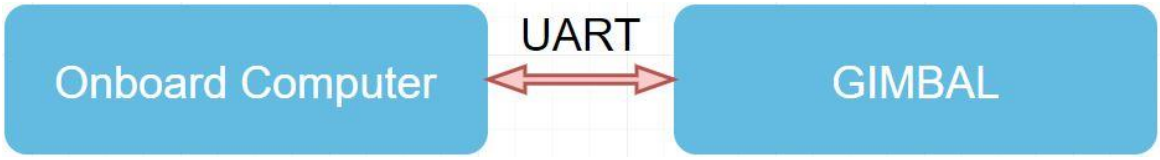
- Programming experience C/C++.
- A compatible gimbal.
- Your Onboard Computer with an available TTL UART port.
- Software tool to build the SDK .
- PC to run the required software tool.

## Hardware setup guide

This guide will help you connect your Onboard Computer with the Gimbal (T3, S1, PixyF, PixyU).

### Data

The onboard computer communicates to the Gimbal through a UART interface.




### Power

Power can be drawn directly from the COM2 port (1A max@5V).

### UART

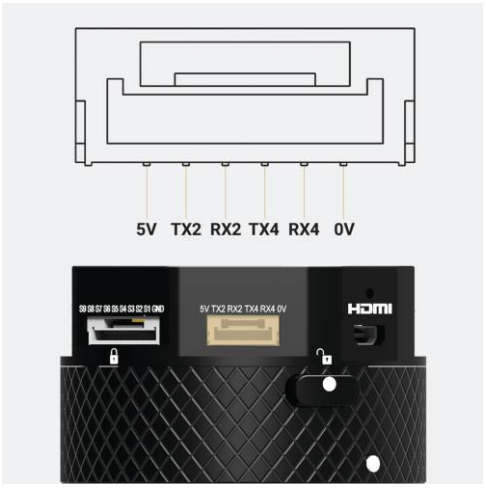
**Interface Details**

- The UART electrical interface for all SDK compatible Gremsy Gimbal is 3.3 volt TTL.
- You must ensure that your onboard computer UART port operates at the same voltage to avoid damaging the Gimbal Controller. For example, RS-232 ports will need a level-shifting circuit.
- The UART interface does not require power from the onboard computer.

## Connector Pinout S1

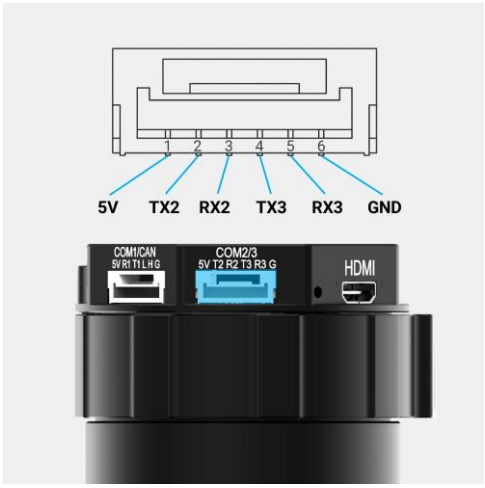

## Connector Pinout T3


T3V2_QR




T3V3_QR

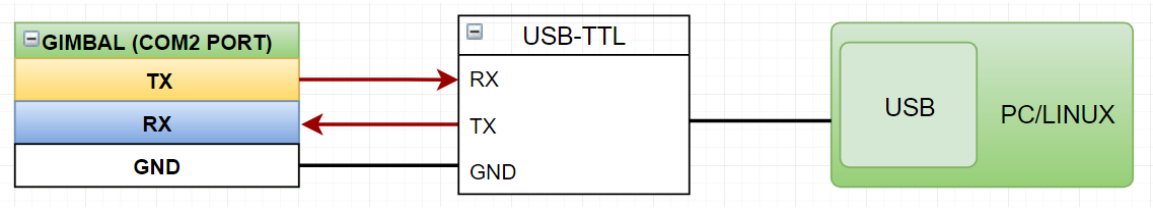
## Connector Pinout PIXY



**Connecting to your Onboard Computer**

The diagram below shows the hardware connection between a GIMBAL and a PC or LINUX machine. Note that:

- The recommended choice of USB to TTL cable is the FT232 module.
- Baudrate: 115200
- Bit data: 8 bits
- Parity: No Parity




# Software Environment Setup Guide

This guide details the software environment needed to work with the SDK

## Download the gSDK and Required tools

❖ Installing gTune app Gremsy Store → Support → Product Support → Choose your gimbal → Download

## Update Firmware

Please follow the gimbal **User Manual** to update the new firmware for the gimbal.

## Configure Linux Development Environment

### Install Development Tools

To build standalone Linux applications based on the gSDK, you need:

- A supported C++ compiler.
- A bash shell.
- A modern Linux distribution.

### Add UART Permissions

Please follow the steps below to add UART read and write permissions for users specified in Linux:

- Use the `sudo usermod -a -G dialout $ USER` command to add the user to the `dialout` group.
- After logging in to the added account again, the account can obtain UART read and write permissions.

## Setting up samples

### Before you start

1. Make sure you have followed the steps in the **Hardware Setup Guide** to get your connection right.
2. Follow the steps in the Environment Setup guide to get your software ready to run samples.

### Run The Sample On The Linux

This is a simple MAVLink to UART interface example for Linux systems that can allow communication between a gimbal and an Onboard Computer.

This example will receive Mavlink messages from the gimbal and send MAVLink messages for controlling and setting the gimbal.

### Building the gSDK and running the example

1. Clone (or download as a zip) the gSDK

   **https://github.com/Gremsy/gSDK.git**
2. Open a terminal, cd into the gSDK folder, and follow these steps to build the gSDK:

   *$ cd gSDK*

   *$ make*

### Execution

You have to pick a port name, try searching for it with

*$ ls /dev/ttyACM**

*$ ls /dev/ttyUSB**

Run the example executable on the host shell:

*$ cd gSDK*

*$ ./gSDK –d /dev/ttyUSB0*

To stop the program, use the key sequence Ctrl-C

```
OPEN PORT
Connected to /dev/ttyUSB0 with 115200 baud, 8 data bits, no parity, 1 stop bit (8N1)

START READ THREAD

 Lost Connection!
 Lost Connection!
Found

GOT GIMBAL SYSTEM ID: 4
START WRITE THREAD


Got message gimbal status
Gimbal is operating
Got message RAW IMU.
    raw imu: time: 1591149964954814, xacc:20, yacc:-60, zacc:8534, xgyro:1287, xgyro:110, xgyro:124(raw)
Got message Mount orientation.
    orientation: time: 2020804268, p:-0.002313, r:0.058161, y:-0.032959 (degree)
Got message Mount status
    Encoder Angle: time: 1591149964954833, p:0, r:0, y:0 (Degree)
    SETTING TILT: dir 1, speed_follow: 65, speed_control: 50
    MOTOR_CONTROL: GYRO: 2, OUT 3, GAIN 120
    TILT  stiff 80, hold: 40
    ROLL  stiff 90, hold: 40
    PAN   stiff 100, hold: 40

FW Version: 7.5.0.OFFICIAL
READ SOME MESSAGES

Got message gimbal status
Gimbal is operating
Got message RAW IMU.
    raw imu: time: 1591149965057013, xacc:103, yacc:-168, zacc:8289, xgyro:502, xgyro:-43, xgyro:-14(raw)
Got message Mount orientation.
    orientation: time: 2020874504, p:-0.049958, r:0.079474, y:-0.032959 (degree)
Got message Mount status
    Encoder Angle: time: 1591149965057037, p:0, r:0, y:0 (Degree)
    SETTING TILT: dir 1, speed_follow: 65, speed_control: 50
    MOTOR_CONTROL: GYRO: 2, OUT 3, GAIN 120
    TILT  stiff 80, hold: 40
    ROLL  stiff 90, hold: 40
    PAN   stiff 100, hold: 40

^C
TERMINATING AT USER REQUEST

CLOSE THREADS

CLOSE PORT
```