##### Table of Contents

[Introduction 3](#_Toc62733497)

[SDK Documentation Home 3](#_Toc62733498)

[MAVProxy 3](#_Toc62733499)

[Works Modes 3](#_Toc62733500)

[Moving the Gimbal 3](#_Toc62733501)

[Body Coordinate System 4](#_Toc62733502)

[General Protocol Overview 4](#_Toc62733503)

[Enable SDK API 5](#_Toc62733504)

[Baud & Data Transmission Rates 5](#_Toc62733505)

[Gimbal Control Messages 7](#_Toc62733506)

[What It Does 10](#_Toc62733507)

[Development Workflow 12](#_Toc62733508)

[Prerequisites 12](#_Toc62733509)

[Hardware setup guide 12](#_Toc62733510)

[Data 12](#_Toc62733511)

[Power 12](#_Toc62733512)

[UART 12](#_Toc62733513)

[Software](#_Toc62733514)

[Download the gSDK and Required tools 14](#_Toc62733515)

[Update Firmware 14](#_Toc62733516)

[Configure](#_Toc62733517)

[Setting up samples 17](#_Toc62733518)

[Before you start 17](#_Toc62733519)

[Run The Sample On The Linux 18](#_Toc62733520)

[Building the gSDK and running example 18](#_Toc62733521)

[Execution 18](#_Toc62733522)

##### C:\Users\vanch\Desktop\Dev_Camera\footer-logo (1).png

##### GREMSY SDK

##### INTEGRATION GUIDE

Release date: 28 Jan 2021

Revision Number: 2.4.0

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)](https://ardupilot.org/mavproxy/docs/modules/gimbal.html)

MAVSDK

Link: [Introduction · MAVSDK Guide (mavlink.io)](https://mavsdk.mavlink.io/main/en/)

Coordinate System

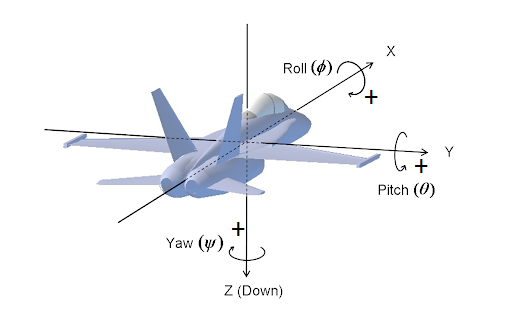
Earth Frame

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, th](https://en.wikipedia.org/wiki/Right-hand_rule)e **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: Meaning 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](https://mavlink.io/en/). 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.

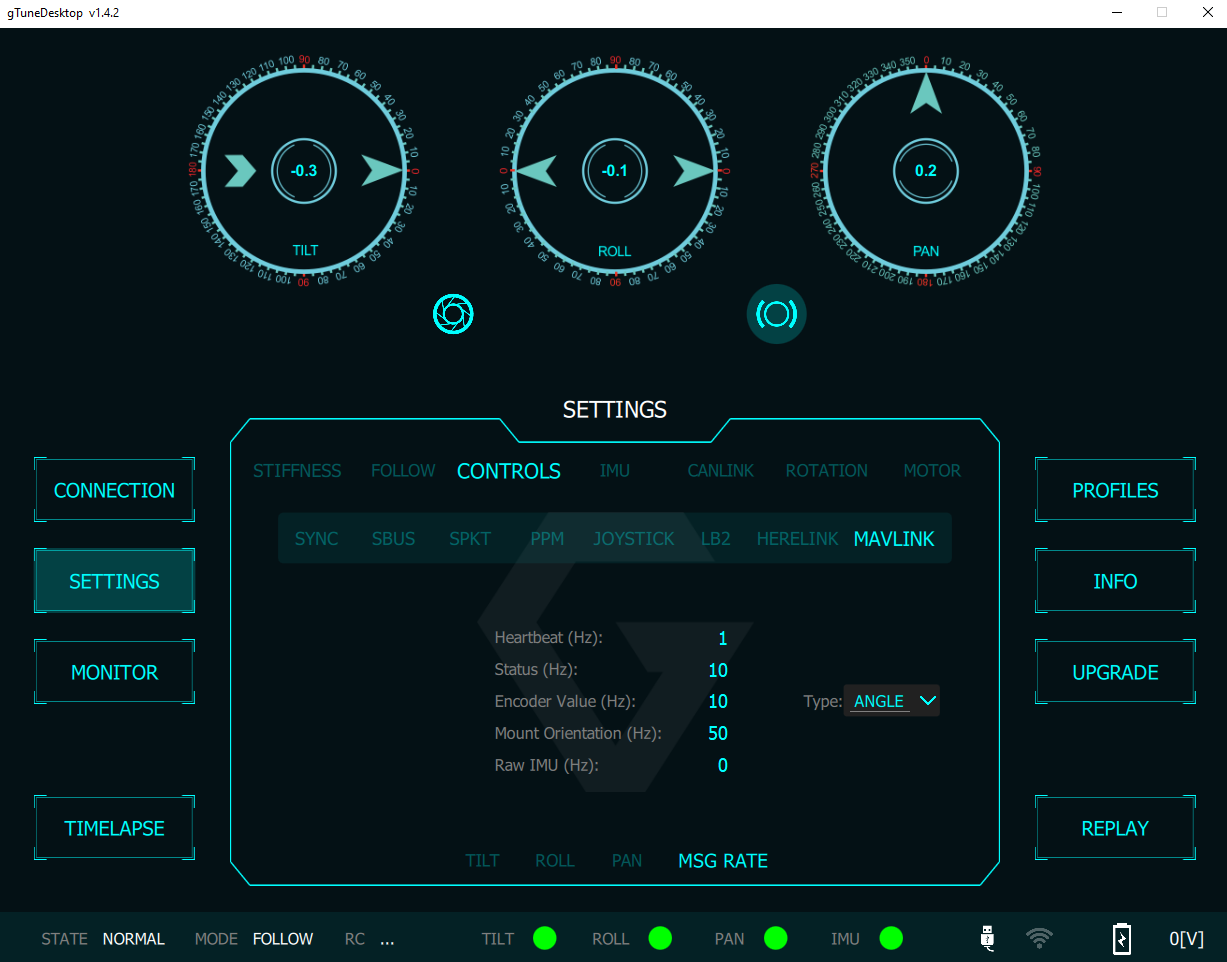
**NOTE:**

* SDK\_V2 has just been released for the gimbal’s firmware and gun desktop app.
* **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 transmission 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 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: 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] | Custom name 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 |

**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**

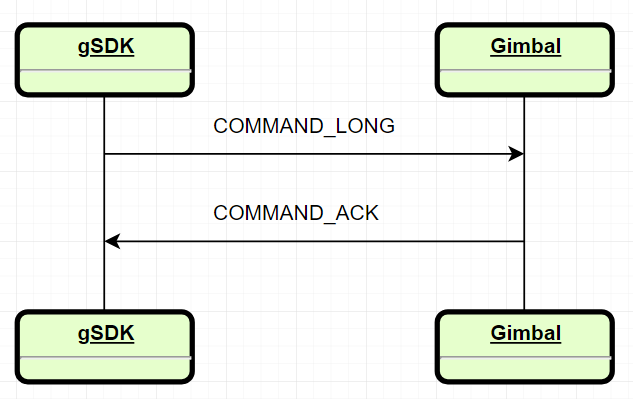
This command is used to request and set the interval between messages for a particular MAVLink message ID.

|  |  |  |
| --- | --- | --- |
| 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 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 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 containing autopilot state for gimbal deivce.

|  |  |  |
| --- | --- | --- |
| 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 | Feed forward 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 |

What It Does

**Gimbal supports the following functions.**

* **Gimbal status (****see gimbal\_status\_t structure)**. Gimbal’s status. Gimbal sends status messages at approximately 1Hz.
* **The gimbal attitude (see attitude<float> structure)**. Gimbal sends the attitude messages at requested rate.
* **The gimbal encoder values (see attitude<int16\_t> structure)**. There is raw encoder values or angle after calculating. It depends on your setting of the Data Transmission above. Gimbal sends encoder values at requested rate.
* **The gimbal raw imu values (see imu\_t structure)**. Gimbal sends the raw imu values at requested rate.
* **The HEARTBEAT message** can ensure that the Gimbal is booted and ready to receive Mavlink Packets. Gimbal will send a heartbeat message approximately once per second by default.
* **The COMMAND\_ACK message** the feedback from the gimbal to indicate command arrival.

**SDK\_v2 supports the following functions.**

* **Control gimbal for each axis mode (see structure control\_gimbal\_axis\_mode\_t)**
* **Control gimbal motor (ON/OFF).**
* **Set gimbal move**. This function will be combined with control gimbal for each axis mode. Allow control gimbal move in angle body or angular rate.
* **Set gimbal reboot**. This function will allow the reboot gimbal.

**These functions are supported by the firmware version from v7.5.0 or above**

* **Set gimbal motor control**: For a gimbal with adjustable payload, finer level motor control is useful to compensate for the payload with different masses. See **set\_gimbal\_****motor\_control() function:**

|  |  |  |  |
| --- | --- | --- | --- |
| Field Name | Type | Description | Range |
| stiffness | uint8\_t | This setting adjusts the degrees to which the gimbal tries to correct for unwanted camera movement and hold the camera stable | It depends on your payload |
| Hold strength | uint8\_t | The power level provided to the corresponding axis | Default: 40 |
| gyro\_filter | uint8\_t | Defines the strength of the filter applied to Gyro sensor output. | Default: 2 |
| output\_filter | uint8\_t | Defines the strength of the filter applied to motors output. | Default: 3 |
| gain | uint8\_t | Defines how fast each axis will return to the commanded position. | Default: 120 |

**NOTE: See** [**user\_manual**](https://gremsy.com/gremsy-t3-manual/) **to tune some parameters for the best performance. Using these parameters for your payload profile.**

* **Set gimbal config axis:** User can adjust some parameters related to control in lock or follow.

(**Note**: See **gimbal\_config\_axis\_t** structure or **set\_gimbal\_config\_pan\_axis(), set\_gimbal\_config\_roll\_axis(), set\_gimbal\_config\_tilt\_axis().**

* **Set gimbal config mavlink message rate:**  Configuration the message mavink with rate.

**(NOTE:** see **set\_gimbal\_config\_mavlink\_msg() function)**

|  |  |  |  |
| --- | --- | --- | --- |
| Field Name | Type | Description | Range |
| emit\_heatbeat | int8\_t | Enable or disable emit heartbeat from the gimbal to the COM2 port. | 0: Disable and 1Hz  Default: 1Hz |
| status\_rate | uint8\_t | Providing gimbal status | [0: 100] Hz.  Default 10Hz |
| henc\_value\_rate | uint8\_t | The rate for sending the encoder values. The type of message depends on the enc\_type\_send (Raw/Angle) | [0: 100] Hz.  Default 50Hz |
| enc\_type\_send | uint8\_t | 0: Encoder angle  1: Encoder value (Resolution 2^16) | [0:1]  Default: 0 |
| orientation\_rate | uint8\_t | The rate for sending the orientation of the gimbal | [0: 100] Hz  Default: 50Hz |
| imu\_rate | uint8\_t | The rate for sending the raw imu values | [0:100] Hz  Default: 0 (Disable) |

Development Workflow

Prerequisites

To build an SDK based application the following are required

* Programming experience C/C++
* A compatible with gimbal (T3, S1, Pixy)
* Your Onboard Computer with an available TTL UART port
* Software tool to build 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

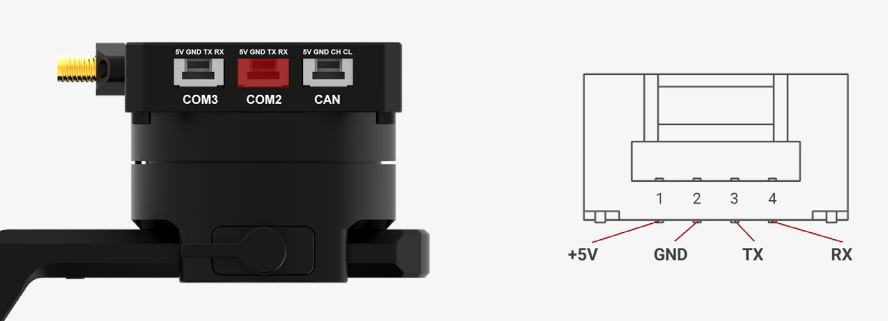
Power can be drawn directly from the COM2 port (1A max@5V).

UART

**Interface Details**

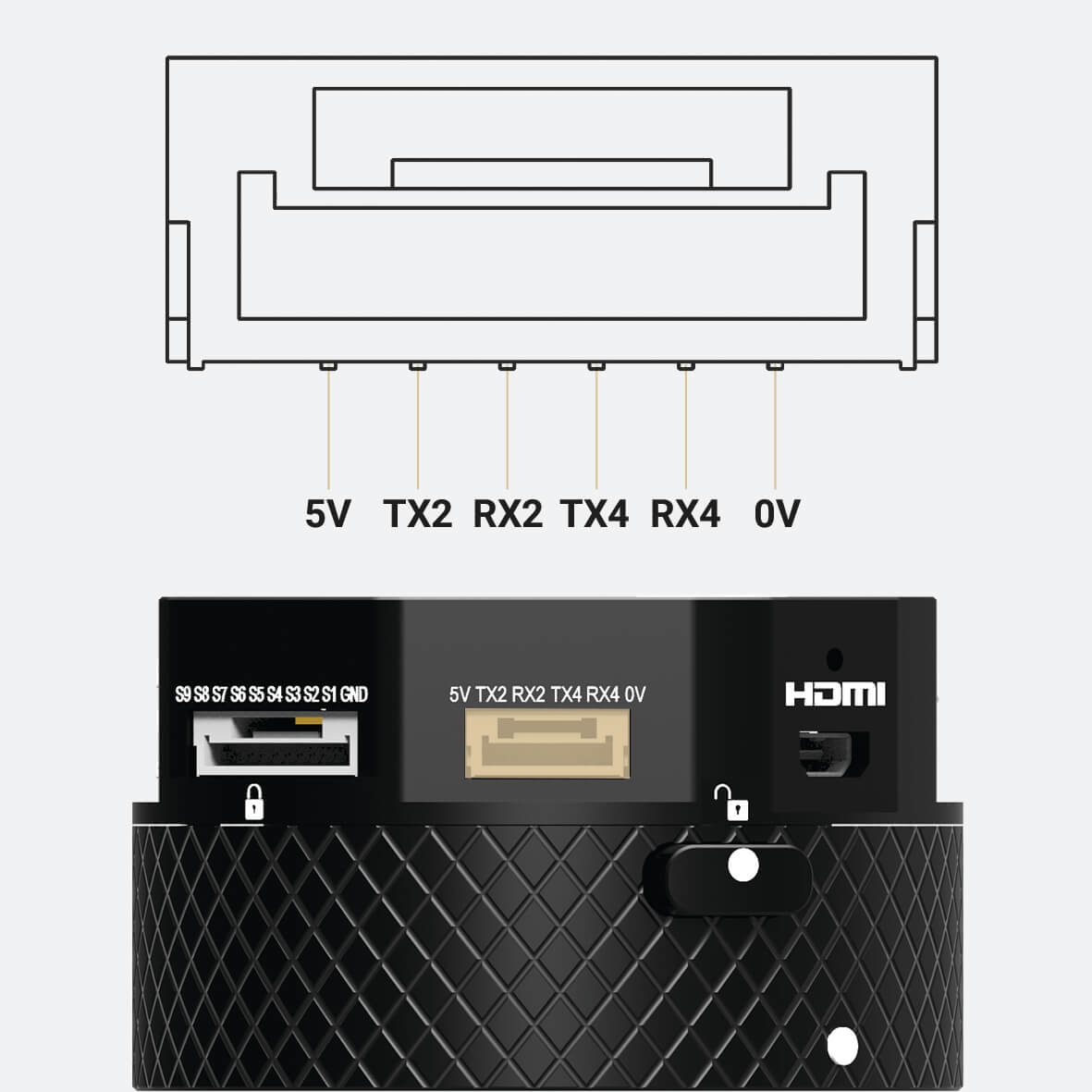
* The UART electrical interface for all SDK compatible Gremsy Gimbal is 3.3 volt TTL
* You must ensure that your onboard computer UART port operates at the same voltage to avoid damaging the Gimbal Controller. For example, RS-232 ports will need a level-shifting circuit
* The UART interface does not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

**Connector Pinout S1**



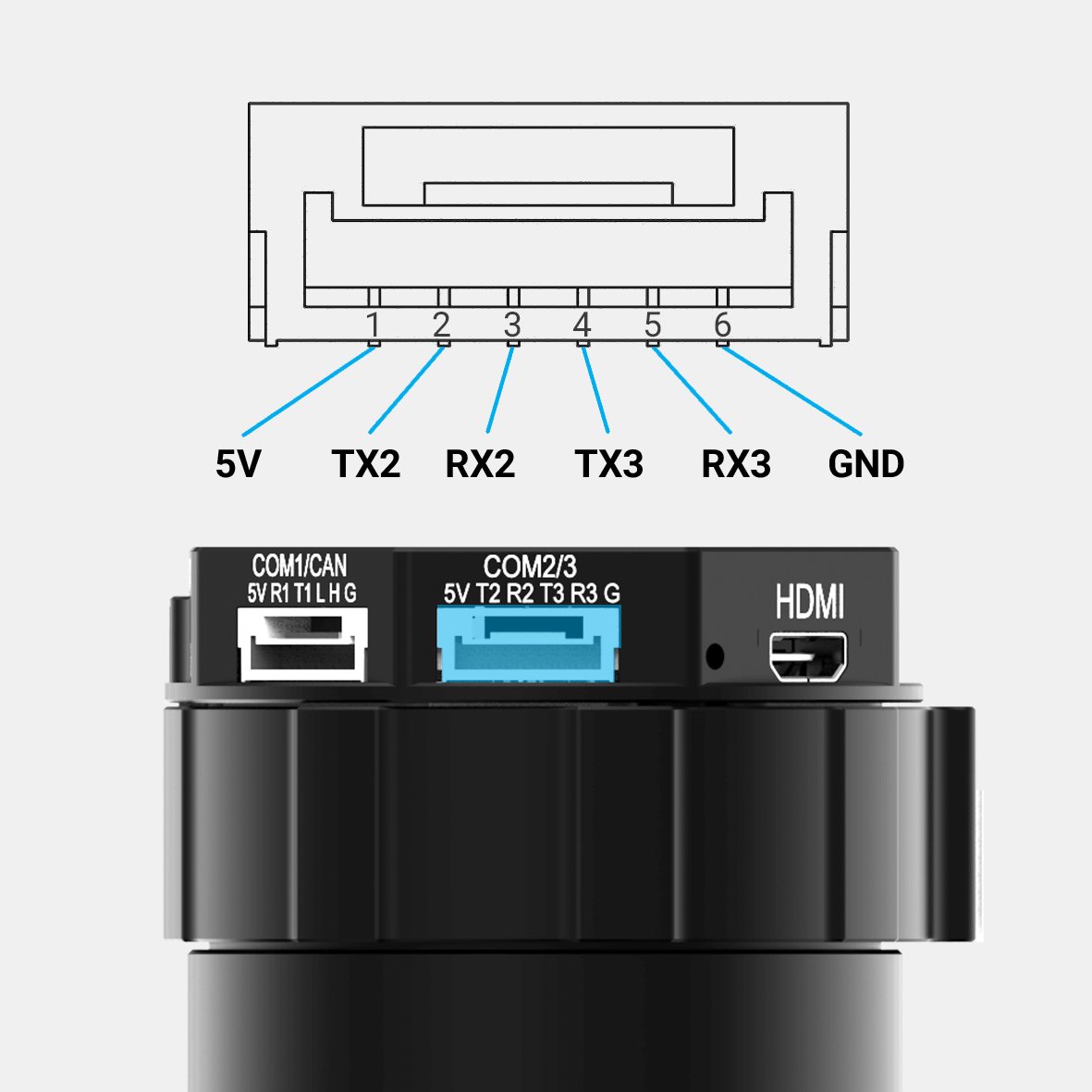
**Connector Pinout T3**





T3V2\_QR T3V3\_QR

Connector Pinout PIXY

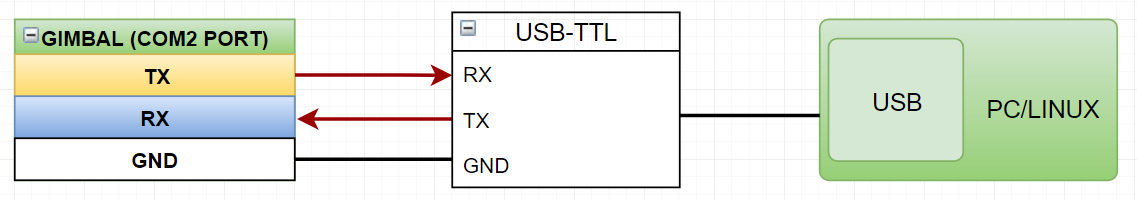


**Connecting to your Onboard Computer**

* **Gimbal + PC/LINUX machine**

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.
* Baurate: 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 [www.gremsy.com](http://www.gremsy.com) 🡪 Support 🡪 Product Support 🡪 GremsyS1 (S1, T3, Pixy)

Update Firmware

Here’s what you need to update your gimbal PixyF, S1V2, and T3V2 from firmware version 6.x.x to 7.x.x.

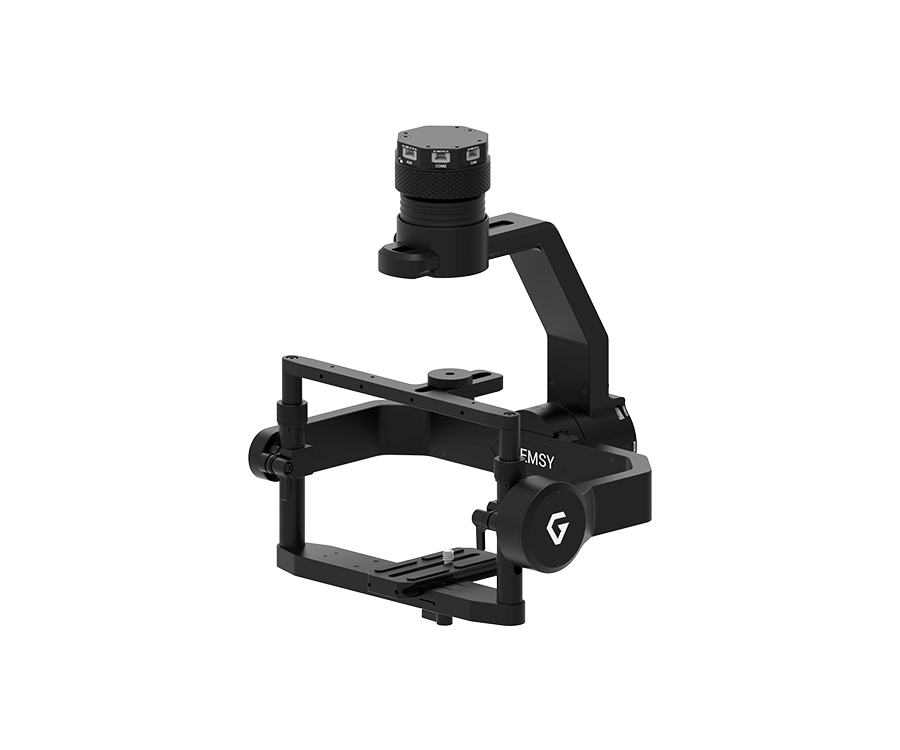
**NOTE:**

* With T3V2 is slightly different. You need to remove your camera and balance without payload. But with PixyF and S1 must be conducted with camera and balanced carefully.
* Firmware V7.x.x or above must be flashed with the gTuneDesktop V1.4.0 or above.

**Prerequisites**

* **Remove your payload on the T3V2 and balance the Roll Axis**

When the proper left-right roll balanced is achieved, the roll axis will stay level when you remove your hands.



* **Balancing Gimbal S1V2 and PixyF**

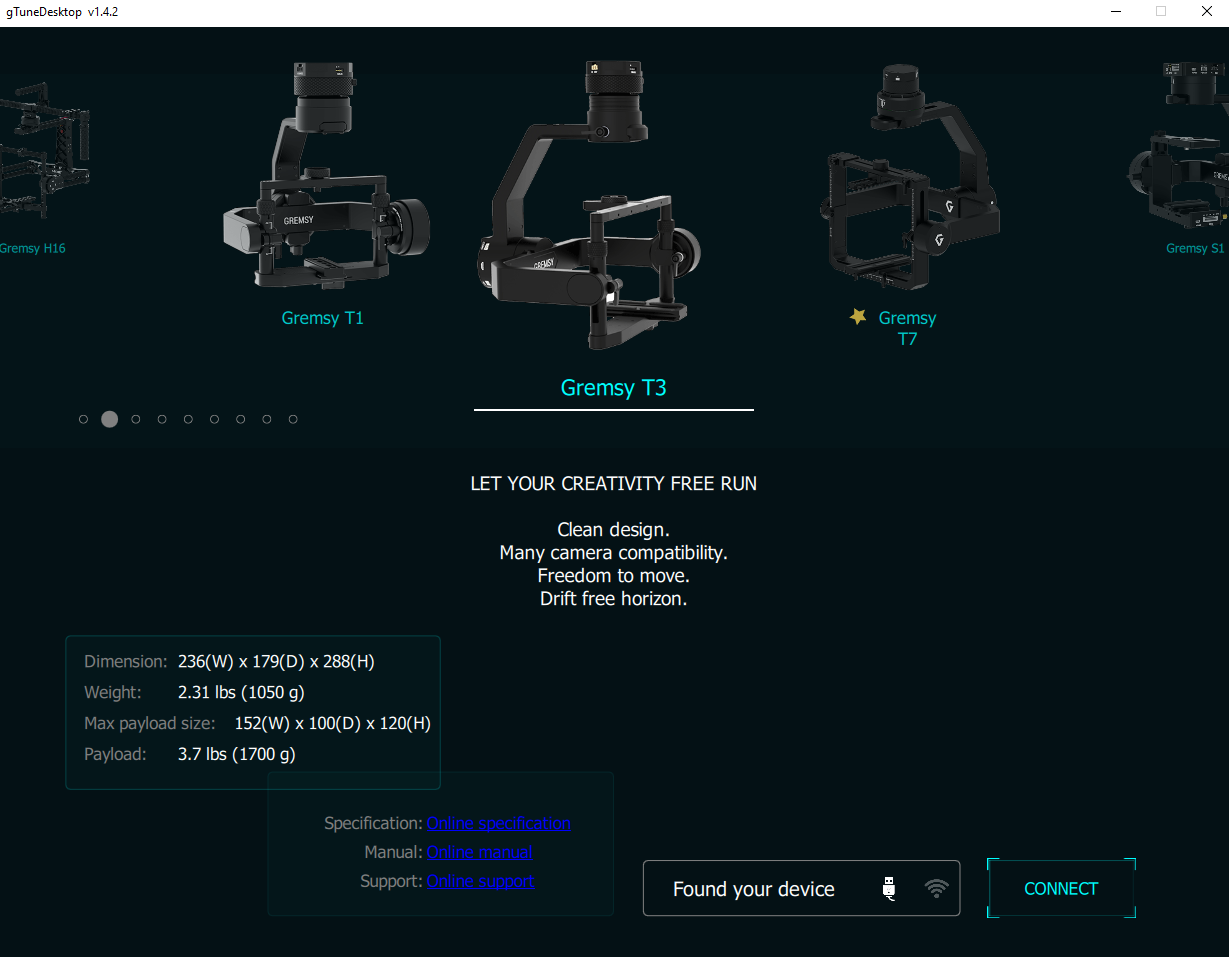
A payload needs to be properly balanced before starting gimbal calibration. For gimbal without adjustable payload like PixyF, then move to the next step.

Refer to S1 manual 🡪 balancing: <https://gremsy.com/gremsy-s1-manual/>

**Upgrading Firmware**

* **Uninstall the old gtune on your PC/MAC and install the new gTuneDesktop v1.4.2**

Link: <https://github.com/Gremsy/gTuneDesktop/releases>



* **Download the firmware v7.5.0 or above**

Link: <https://github.com/Gremsy>

* **Open gTuneDesktop v1.4.2 and update the firmware.**

01 - Power on the GIMBAL.

02 - Connect the USB cable from the GIMBAL controller to Mac/PC.

03 - Run the gTune Desktop v1.4.2 or above software.

04 - In the software, select the “Serial” option on the “connection” tab.

05 - Select the port in the list.

06 - Click on the “Connect” button.

07 - Go to the “Upgrade” tab.

08 - “Browse” to firmware file from your computer.

09 - Make sure the RF receiver (if available) is already removed.

10 - Click the “Upgrade” button. The process will take about 2 minutes. When the firmware is upgraded successfully, the Gimbal will be restarted automatically.

* **Calibration.**

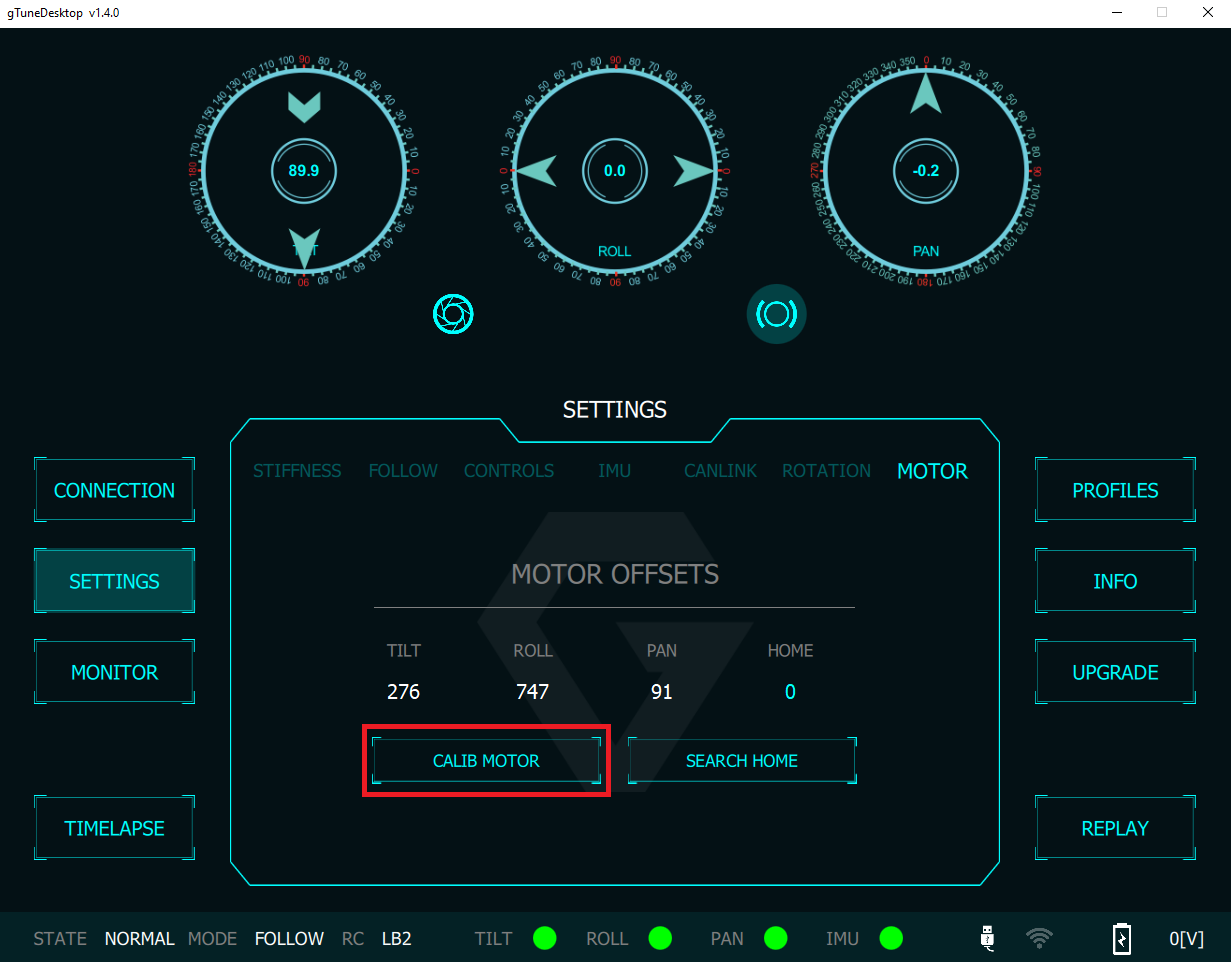
Gimbal will be automatically calibrated on power-up at the first time upgrading from the firmware version less than v7.2.0. Users will see the camera rotate and shake around all three axes for several minutes when the gimbal is calibrating.

The gimbal should be stationary (not flying, or being held) and horizontal during calibration. For a gimbal with an adjustable payload, the payload should be present and [balanced](https://gremsy.com/gremsy-t3-manual/) before doing a calibration (**NOTE: Exception for Gimbal T3 needs to calibrate without Payload**).

In case, gTuneDesktop pops up a motor status like below. Please, close it and go to **SETTINGS 🡪 MOTOR 🡪 CALIB MOTOR.**



**Press CALIB MOTOR and wait for a second**



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**](https://github.com/Gremsy/gSDK.git)

1. 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

