



GAL2
2-Finger Angular Lead-Screw Gripper
(Beta v1.1)

Installation and Operation Manual

Document #: GAL2_B1.1_MAN_JUN26

June, 2026

Foreword

Information contained in this document is the property of Contactile Pty Ltd. and shall not be reproduced in whole or in part without prior written approval of Contactile Pty Ltd. The information herein is subject to change without notice and should not be construed as a commitment on Contactile Pty Ltd. This manual is periodically revised to reflect and incorporate changes made to the GAL2 2-Finger Angular Lead-Screw Gripper.

Contactile Pty Ltd assumes no responsibility for any errors or omissions in this document. Users' critical evaluation is welcome to assist in the preparation of future documentation.

Copyright © by Contactile Pty Ltd, Sydney, Australia. All Rights Reserved.
Published in Australia.

All trademarks belong to their respective owners.

Conditions of Sale

Contactile's conditions of sale apply to all products sold by Contactile to the Distributor under this Agreement. The conditions of sale that apply are provided on the USB flash drive shipped with the product in the folder 'LEGAL' in the root directory.

End User Licence Agreement

Contactile's end user license agreement applies to all software and algorithms included with the products sold by Contactile. The end user license agreement that applies is provided on the USB flash drive shipped with the product in the folder 'LEGAL' in the root directory.

Compliance

The devices are sold as is.

The devices are specifically designed solely for the purposes of research and development only made available on a business-to-business basis.

The devices are not for resale.

Table of Contents

1	Introduction	7
2	Safety	8
2.1	General	8
2.2	Explanation of warnings	8
2.3	Precautions	8
3	Getting started	9
3.1	Introduction	9
3.2	Scope of delivery	9
3.2.1	GAL2 2-Finger Angular Lead-Screw Gripper (Beta v1.1)	9
3.2.2	Gripper Mounting Adaptor and Screws (optional).....	10
3.2.3	Gripper Power Pack.....	10
3.2.4	USB-C Cable	11
3.2.5	M8 Cable (optional)	11
3.2.6	USB Flash Drive	11
4	Installation	14
4.1	Precautions.....	14
4.2	Mounting.....	14
4.3	Connecting the Gripper and Sensors	14
4.4	Powering up the Gripper and Sensors	14
4.5	Interfacing the Gripper and Sensors to a PC.....	14
4.6	Normal operation start-up sequence	14
5	Gripper Specifications.....	15
5.1	Physical specifications	15
5.1.1	Mechanical drawings	15
5.1.2	Gripper construction	16
5.2	Technical specifications	16
5.3	Temperature effects on accuracy	17
5.4	Power	17
5.4.1	Current draw profile	17
6	Gripping Modes	19
6.1	Position control	19
6.2	Fixed grip force control.....	19
6.3	Dynamic grip force control.....	19
6.3.1	Overview.....	19
6.3.2	Tips.....	19
7	Gripper interfaces	20

7.1	Overview.....	20
7.2	LED Indicator	20
7.3	Manual Control Buttons.....	21
7.4	DC Power Jack	21
7.5	USB-C Port #1	21
7.6	USB-C Port #2	21
7.6.1	Gripper serial communications.....	21
7.6.2	Data streaming	21
7.7	M8 Port.....	22
8	Maintenance	23
8.1	General.....	23
8.2	Cleaning.....	23
8.3	PapillArray Tactile Sensor silicone integrity.....	23
8.4	Cabling and connectors	23
8.5	Periodic calibration.....	23
9	Communication protocol	24
9.1	Command template.....	24
9.1.1	Sending a command.....	24
9.1.2	Receiving gripper responses.....	24
9.2	List of commands.....	25
9.3	Command subsets available for different ports.....	33
10	Gripper State Machine	34
10.1	States	34
10.2	Errors.....	36
11	Visualisation Software.....	37
11.1	Connecting the gripper to a PC	37
11.2	Launching the Visualisation Software.....	37
11.2.1	Windows	37
11.2.2	Linux.....	37
11.3	Connecting to the COM Port	37
11.3.1	Windows	38
11.3.2	Linux.....	38
11.4	Visualisation.....	39
11.4.1	Gripper width	39
11.4.2	Tactile sensor data	40
11.4.3	IMU data, State and Last Error	40
11.5	Logging data	41

11.6	Generating a usage report	41
11.7	Gripper control	43
12	C++ Library	44
12.1	Folder structure	44
12.2	Library installation	44
12.3	Class and function documentation	45
12.3.1	Constants	45
12.3.2	Structures	46
12.3.3	GripperClass function list	46
12.4	Example using the C++ library	54
13	Python Library	55
13.1	Folder structure	55
13.2	Library installation	55
13.3	Class and function documentation	55
13.3.1	Constants	55
13.3.2	Structures	55
13.3.3	GripperClass function list	55
13.4	Example using the Python library	56
14	ROS	57
14.1	Minimum requirements	57
14.2	Folder structure	57
14.3	Package summary	57
14.4	Installing the package	58
14.5	Starting the node	58
14.6	contactile_gripper_ros1 node	58
14.6.1	Topics	58
14.6.2	Services	58
14.7	Example using ROS node	58
15	ROS 2	59
15.1	Minimum requirements	59
15.2	Folder structure	59
15.3	Package summary	59
15.4	Installing the package	60
15.5	Starting the node	60
15.6	contactile_gripper_ros2 node	60
15.6.1	Topics	60
15.6.2	Services	60

15.7 Example using ROS2 node	60
------------------------------------	----

1 Introduction

The GAL2 2-Finger Angular Lead-Screw Gripper (Beta v1.1) is an electric 2-finger linear gripper with PapillArray Tactile Sensor Array integrated into each of the two gripper fingertips. The PapillArray Tactile Sensor Arrays measure real-time tactile data that is used for real-time closed-loop feedback control of the grip force applied by the Gripper. The Gripper autonomously and dynamically applies the optimal grip force, calculated in real-time, to prevent translational slip of the grasped object, based on the measured coefficient of static friction between the Sensors and the grasped object and the measured shear force acting on the Sensors.

The GAL2 2-Finger Angular Lead-Screw Gripper (Beta v1.1) is shown in Figure 1.1.



Figure 1.1 – The GAL2 2-Finger Angular Lead-Screw Gripper (Beta v1.1).

This document contains the installation and operation manual for the GAL2 2-Finger Angular Lead-Screw Gripper (Beta v1.1).

2 Safety

2.1 General

The customer should verify that the maximum loads and moments expected during operation fall within the sensing range of the sensors (see the document PTS_2.0_SPEC_DEC21 in the MANUALS folder of the supplied USB Flash Drive) and the operating range of the gripper (see section 5.2 Technical specifications). Outside these ranges, sensor reading accuracy is not guaranteed, and safe operation of the gripper is not guaranteed. Particular attention should be paid to dynamic loads caused by robot acceleration and deceleration if the gripper is mounted on robotic equipment. These forces can be many multiples of the value of static forces in high acceleration or deceleration situations.

2.2 Explanation of warnings

The warnings included here are specific to the product(s) covered by this manual. It is expected that the user heeds all warnings from the manufacturers of other components used in the installation.



Danger indicates that a situation could result in potentially serious injury or damage to equipment.



Caution indicates that a situation could result in damage to the product and/or the other system components.

2.3 Precautions



DANGER: Do not attempt to disassemble the gripper or sensors. This could damage the gripper or sensors and will invalidate the sensor calibration.



DANGER: Do not attempt to drill, tap, machine, or otherwise modify the gripper or sensor casing. This could damage the gripper or sensor and will void any warranty.



DANGER: Do not use the gripper on abrasive surfaces or surfaces with sharp points/edges. This could damage the silicone surface of the sensors.



CAUTION: Sensors may exhibit a small offset in readings when exposed to intense light sources.



CAUTION: Exceptionally strong and changing electromagnetic fields, such as those produced by magnetic resonance imaging (MRI) machines, constitute a possible source of interference with the operation of the gripper and sensors.



CAUTION: Temperature variations can cause drift in sensor readings. Some temperature compensation is included. However, bias removal in software prior to operation may be necessary, and it is recommended that biasing is performed each time the sensor is known to be unloaded.

3 Getting started

3.1 Introduction

This section describes the contents of the GAL2 Gripper Kit (Beta v1.1). Installation is covered in section 4.

3.2 Scope of delivery

Check the shipping box and components for damage during shipping. Any damage should be reported to Contactile Pty Ltd.

Standard components are:

- 1x GAL2 2-Finger Angular Lead-Screw Gripper (Beta v1.1)
- 2x PapillArray Tactile Sensor (v2.0) – integrated on the fingertips of the GAL2 Gripper
- 1x Gripper Mounting Adaptor and Screws (optional)
- 1x Gripper Power Pack
- 1x M8 Cable (optional)
- 1x USB-C cable – 1 m length
- 1x Contactile USB flash drive

Each of these components are described in the sections below.

3.2.1 GAL2 2-Finger Angular Lead-Screw Gripper (Beta v1.1)

Figure 3.1 shows the GAL2 2-Finger Angular Lead-Screw Gripper (Beta v1.1) with integrated PapillArray Tactile Sensors.



Figure 3.1 – The GAL2 2-Finger Angular Lead-Screw Gripper (Beta v1.1) with integrated PapillArray Tactile Sensors.

For detailed specifications of the PapillArray Tactile Sensors, refer to the document PTS_2.0_SPEC_MMMYY in the MANUALS folder of the Contactile USB Flash Drive.

3.2.2 Gripper Mounting Adaptor and Screws (optional)

The Gripper Mounting Adaptor can be used to mount the GAL2 Gripper to the wrist of a robot arm with an ISO 9409-1-50-4-M6, ISO 9409-1-40-4-M6 or ISO 9409-1-31,5-4-M5 compatible flange. The Gripper Mounting Adaptor is shown in Figure 3.2. Sixteen M6x8 screws are also provided.

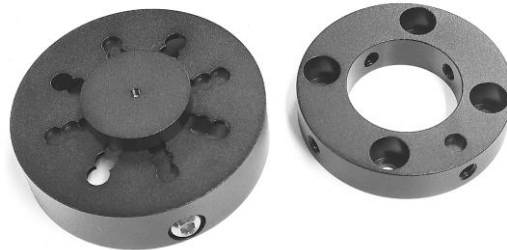


Figure 3.2 – The Gripper Mounting Adaptor

3.2.3 Gripper Power Pack

The Gripper Power Pack can power the GAL2 Gripper via the DC Power Jack (see section 7.4 DC Power Jack). The Gripper Power Pack is shown in Figure 3.3.



Figure 3.3 – The Gripper Power Pack

The specifications of the Gripper Power Pack are described in Table 3.1.

Table 3.1 – Specification of Gripper Power Pack*

	Measurement	Value	Units
Input (AC)	Voltage	100 – 240	V
	Current	1.2	A
	Frequency	50/60	Hz
Output (DC)	Voltage	24	V
	Current	2	A

* If an alternative power pack is used, a Class I power pack is recommended

3.2.4 USB-C Cable

The Development Kit comes with a standard 1 m long USB-C to USB-A cable for connecting the Gripper to a PC (for power and communications). The USB cable is shown in Figure 3.4.



Figure 3.4 – The USB-C cable

NOTE: The USB-C cable comprises part of the calibrated PapillArray Tactile Sensors. Changing the length or type of the cable can affect the calibration. Check with Contactile when making cabling changes to ensure your system's calibration will not be affected.

3.2.5 M8 Cable (optional)

18/06/2026The M8 cable can be used to connect the Gripper to power via a suitable robot arm tool interface (e.g., Universal Robots UR5e tool interface). The M8 cable also provides for pins for RS485 communications to the GAL2 Gripper. The M8 cable is shown in Figure 3.5.



Figure 3.5 – The M8 cable

The M8 cable is intended to fit the M8 port on the Gripper (refer to section 7.7 M8 Port) and the tool connector from Universal Robots. The output tool connector on the Gripper shares the same connections as the input cable described in Figure 7.2 and Table 7.3.

3.2.6 USB Flash Drive

The GAL2 Gripper comes with a Contactile USB flash drive containing manuals, software, and legal documents. The USB flash drive is shown in Figure 3.6.



Figure 3.6 – The Contactile USB Flash Drive

3.2.6.1 Root folder

The contents of the USB flash drive are described in Table 3.2.

Table 3.2 – Files and folders in the root folder of the Contactile USB flash drive

Name	Type	Description
LEGAL	Folder	Conditions of Sale and End User Licence Agreement
MANUALS	Folder	Manuals and specifications for the PapillArray Tactile Sensors and GAL2 Gripper (this document).
MODELS	Folder	Models of the gripper and the mounting adaptor
SOFTWARE	Folder	Gripper Visualisation software (executables for Windows and Linux), C++ and Python library and example code, ROS and ROS2 nodes

3.2.6.2 LEGAL folder

The LEGAL folder contains legal agreements including the conditions of sale of all products and the end user licence agreement that applies to all software and algorithms supplied with or on the products. A list of the files in this folder is shown in Table 3.3.

Table 3.3 – Contents of the LEGAL folder

Name	Type	More Information
CONDITIONSOFSALE_20211028.pdf	File	Conditions of Sale for products sold by Contactile
ENDUSERLICENCEAGREEMENT_20211028.pdf	File	End User Licence Agreement for software provided by Contactile

3.2.6.3 MANUALS folder

The MANUALS folder contains all specifications and manuals for the GAL2 2-Finger Angular Lead-Screw Gripper (Beta v1.1) and PapillArray Tactile Sensor (2.0). A list of the files in this folder is shown in Table 3.4.

Table 3.4 – Contents of the MANUALS folder

Name	Type	More Information
GAL2_1.1_MAN_MMMYY.pdf	File	Manual for the GAL2 2-Finger Angular Lead-Screw Gripper (Beta v1.1). This document.
PTS_2.0_SPEC_MMMYY.pdf	File	Specifications for the PapillArray Tactile Sensor (2.0)

3.2.6.4 MODELS folder

The MODELS folder contains STEP files and a URDF file for the GAL2 2-Finger Angular Lead-Screw Gripper (Beta v1.1), and a STEP file for the gripper mounting adaptor. A list of the files in this folder is shown in Table 3.5.

Table 3.5 – Contents of the MODELS folder

Name	Type	More Information
GripperMount.STEP	File	STEP file of the mounting adaptor
GAL2_B1.1_STEP	Folder	STEP files for the gripper – fully closed, half open, fully open
GAL2_B1.1_URDF	Folder	URDF file and associated files for the gripper

3.2.6.5 SOFTWARE folder

The SOFTWARE folder contains all the software for the PapillArray Tactile Sensor (v2.0). A list of the files in this folder is shown in Table 3.6.

Table 3.6 – Contents of the Visualisation Software folder

Name	Type	More Information
VIS	Folder	Executable Gripper Visualiser for use with USB-C Port #1 (Windows and Linux version)
C++_LIBRARY	Folder	C++ library (Windows and Linux version) and example code for communicating via the USB-C Port #1
PYTHON_LIBRARIES	Folder	Python library (Windows and Linux version) and example code for communicating via the USB-C Port #1
ROS	Folder	ROS node and launch file for communicating via the USB-C Port #1
ROS2	Folder	ROS2 node and launch file for communicating via the USB-C Port #1
URCAP	Folder	URCap for Universal Robots and template URP program for communicating with the gripper via the Universal Robot Tool I/O port (RS485)

4 Installation

4.1 Precautions

Before attempting to mount or connect the GAL2 Gripper, the user must ensure they have read the specifications and safety information pertaining to the Gripper (see section 5 Gripper Specifications), and the PapillArray Tactile Sensors (see document PTS_2.0_SPEC_MMMYY in the MANUALS folder of the USB Flash Drive shipped with the Gripper).

4.2 Mounting



DANGER: Do not attempt to drill, tap, machine, or otherwise modify or disassemble the Gripper, or Sensors. This could damage the sensor and will void any warranty.

Mount the Gripper to a structure with sufficient mechanical strength. Not doing so can lead to suboptimal performance. The Gripper can be mounted using the bolt pattern provided – See Figure 5.1.

4.3 Connecting the Gripper and Sensors



CAUTION: Do not stress or over bend the sensor cable.



CAUTION: Do not attempt to disconnect sensor cables.

4.4 Powering up the Gripper and Sensors

The Gripper can be powered via the DC Power Jack (see 7.4 DC Power Jack) or via the power pins on the M8 port (see 7.7 M8 Port).

The Sensors can optionally be powered without powering the Gripper, via UBC-C Port #1 (see 7.5 USB-C Port #1). In this way, it is possible to access the Sensor data (see 7.6.2 Data streaming) without powering the gripper. This can be useful for testing purposes.

4.5 Interfacing the Gripper and Sensors to a PC

After powering up the Gripper, a PC can be connected to read the data being transmitted by the PapillArray Tactile Sensors (refer to section 7.6.2 Data streaming) and to transmit commands to the GAL2 Gripper (refer to section 7.6.1 Gripper serial communications).

4.6 Normal operation start-up sequence

Under normal operation when Gripper is powered on the Gripper LED Indicator will flash white for a couple of seconds while the Gripper is initialised. The LED Indicator will then turn solid white. Once the LED Indicator is solid white, the Gripper is ready to receive commands – via the manual control buttons (refer to section 7.3 Manual Control Buttons), the serial port on USB-C Port #1 (refer to section 7.5 USB-C Port #1), or via the RS485 interface on the M8 connector (refer to section 7.7 M8 Port).

5 Gripper Specifications

5.1 Physical specifications

5.1.1 Mechanical drawings

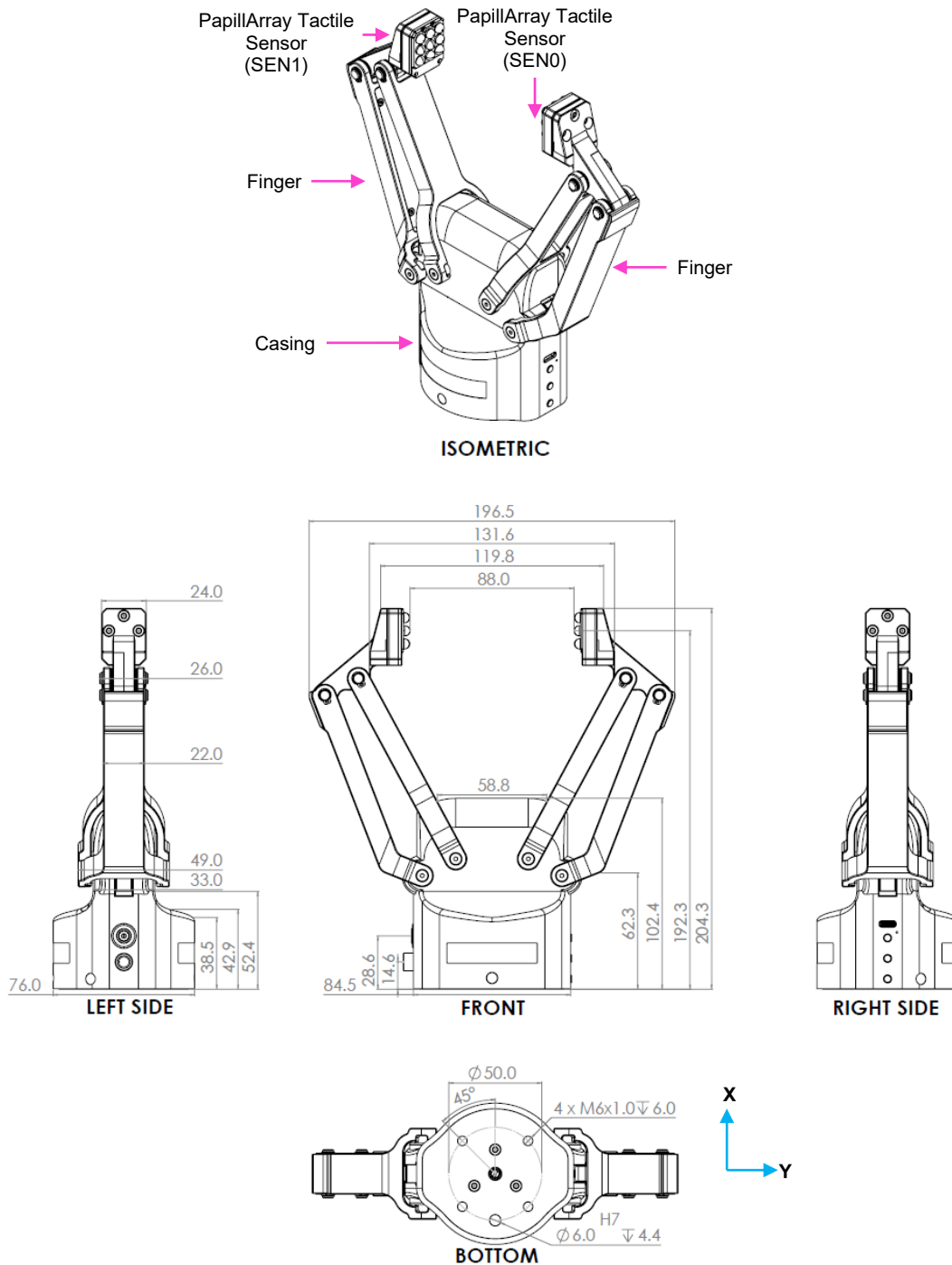


Figure 5.1 – Drawing of GAL2 2-Finger Angular Lead-Screw Gripper (Beta v1.1) with integrated PapillArray Tactile Sensor Arrays (v2.0). All dimensions are in mm. Reference axes of IMU indicated with blue arrows. IMU origin is (-25.6, -12.502, 9.31) mm from centre of base of gripper.

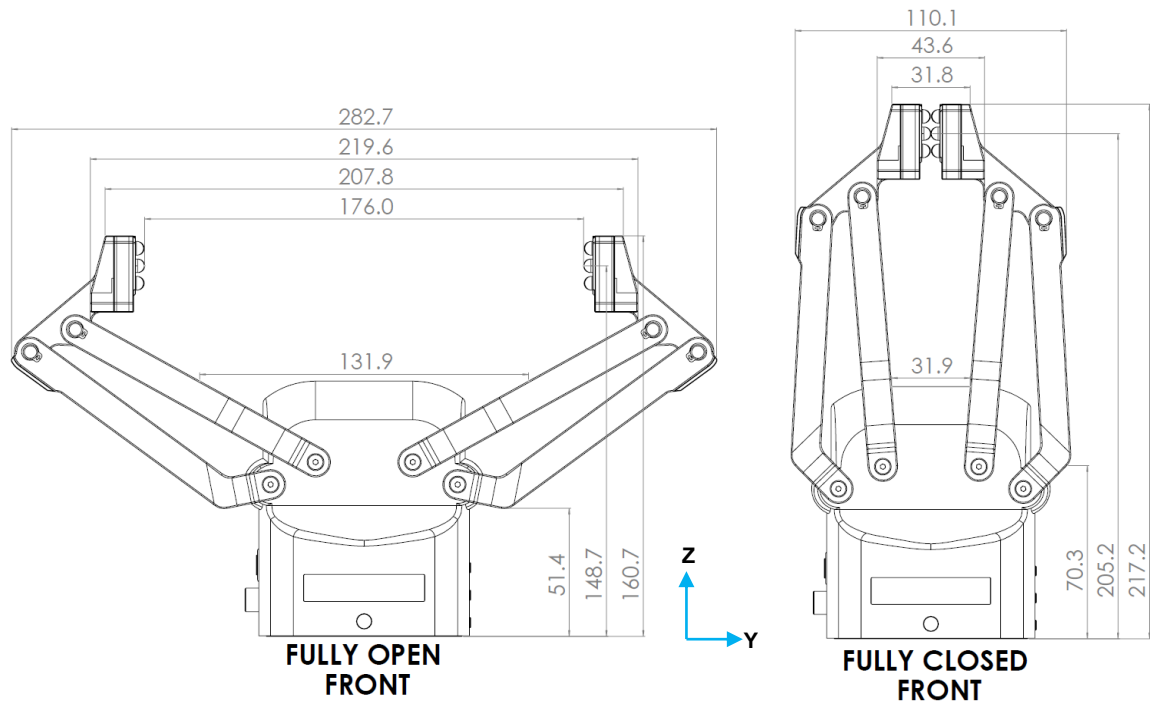


Figure 5.2 – Mechanical drawing of GAL2 2-Finger Angular Lead-Screw Gripper (Beta v1.1) fully open (LEFT) and fully closed (RIGHT). All dimensions are in mm. Reference axes of IMU indicated with blue arrows. IMU origin is (-25.6, -12.502, 9.31) mm from centre of base of gripper.

5.1.2 Gripper construction

The material composition of the Gripper and the material characteristics are shown in Table 5.1.

Table 5.1 – GAL2 Gripper composition and materials physical characteristics.

Gripper Part *	Material	Comments
Fingers	Anodised Aluminium	-
Casing	PA-12 HP Nylon	-

* Refer to Figure 5.1 for reference to gripper parts

5.2 Technical specifications

The Gripper technical specifications are described in Table 5.2.

Table 5.2 – GAL2 Gripper technical specifications

Technical Data	Min	Typical	Max	Units
Stroke	0	-	175.2	mm
Position sensing resolution	-	2.1	-	μm
Position repeatability	-	±0.2	-	mm
Gripping force range	5	-	30	N
Gripping force repeatability	-	0.5	-	N
Gripping speed	-	167.0	167.0	mm/s
Operating voltage	-	24	-	V DC
Current draw	-	-	2	A
Ambient operating temperature	10	-	40	°C
Weight	-	785	-	g

5.3 Temperature effects on accuracy

Temperature variations can cause drift in sensor readings. See document PTS_2.0_SPEC_MMMYY for details of built-in temperature compensation and Sensor biasing.

5.4 Power



CAUTION: Grounding mismatch between the Gripper and the power supply and any other interfaced equipment (e.g., robotic arm via the M8 connector, or laptop via either USB-C Port #1 or USB-C Port #2) could cause issues and unexpected behaviour.

Care should be taken when connecting the Gripper to multiple devices – for example, if the Gripper is connected via both the M8 port and USB, or if a non-isolated power supply is used.

See section 3.2.3 for details of the supplied Power Pack.

See section 3.2.5 for details of powering the Gripper via the M8 port.

5.4.1 Current draw profile

When powered through the M8 connector from the Tool I/O port of a robot arm, the user must ensure that the current supply limits of the robot arm are sufficient for the gripper operations that the user wishes to execute.

The current draw profile for the gripper is shown in Figure 5.3 (for explanations of each of the Gripper commands and parameters, see section 9.2 List of commands).

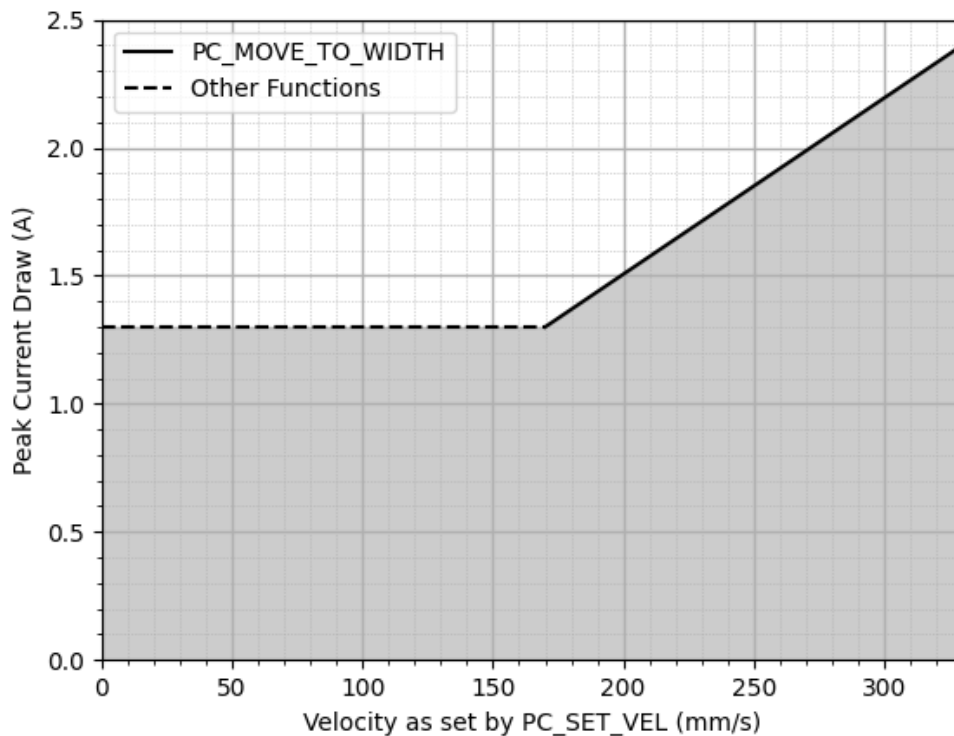


Figure 5.3 – GAL2 Gripper peak current draw. The solid line indicates where the peak current draw is dictated by the `PC_MOVE_TO_WIDTH` command velocity setting (as set by the command `PC_SET_VEL`). The dashed line indicates where the peak current draw is dictated by other Gripper movement commands (e.g. `FF_GRIP`, `DF_GRIP` etc). The grey area indicates the region of operation of the Gripper - if the power supply cannot supply such currents, then it is incompatible with the gripper. For explanations of each of the Gripper commands and parameters, see section 9.2 List of commands.

For example, if the robot arm tool I/O specifies a maximum current of 1.5 A, then the velocity of a `PC_MOVE_TO_WIDTH` should be set to less than 200 mm/s (using the `PC_SET_VEL` command) so as not to trigger a current overdraw fault on the robot arm.

The minimum supply current to support execution of all gripper movement commands up to a maximum velocity of 170 mm/s is 1.3 A.

6 Gripping Modes

The GAL2 2-Finger Angular Lead-Screw Gripper (Beta v1.1) can be operated in 3 gripping modes:

- Position control
- Fixed force-controlled grip
- Dynamic force-controlled grip

6.1 Position control

In this mode, the gripper finger target position is specified by the operator. The gripper fingers will move to the commanded target position.

Currently, position control is not intended to be used to make contact with or grip an object.

In future, position control may also be used for gripping.

6.2 Fixed grip force control

In this mode, the target grip force at the fingertips (i.e., the grip force to be exerted on the PapillArray Tactile Sensors) is specified by the operator.

6.3 Dynamic grip force control

6.3.1 Overview

In this mode, the target grip force at the fingertips is determined in real-time autonomously by the PapillArray Tactile Sensors. The target grip force is dynamic and is calculated as:

$$F_{target} = \max(F_{exploratory}, F_{target,0}, F_{target,1})$$
$$F_{target,n} = k_{shear} \times \frac{F_{shear,n}}{\mu_n} + k_{torque} \times \frac{T_{z,n}}{\mu_n}$$

where F_{target} is the target grip force (N), $F_{exploratory}$ is the exploration force (N), $F_{target,n}$ is the target grip force (N) calculated from sensor n (where n is 0 or 1), $F_{shear,n}$ is the global shear force (N) on sensor n , μ_n is the coefficient of static friction measured by sensor n , $T_{z,n}$ is the Z-torque (Nmm) measured by sensor n , k_{shear} is the shear gain (default value is 1.2 [unitless]), and k_{torque} is the torque gain (default value is 0.0 m⁻¹). SEN0 and SEN1 are the PapillArray Tactile Sensors as labelled in Figure 5.1.

6.3.2 Tips

The exploration force is a small grip force (default 5 N) which is applied by the gripper (such that a few pillars of the PapillArray Sensors are in contact with the target object). The user should aim to position the gripper such that the exploratory force can be applied to the target object without generating any tangential force at the sensors.

Once the exploratory force is applied by the gripper, the sensors begin performing the slip detection algorithm and friction estimation. The user should aim to only begin manipulating the target object (e.g., lifting, pulling, pushing) once the exploratory force has been applied.

7 Gripper interfaces

7.1 Overview

There are multiple interfaces to the Gripper around the main casing including an LED Indicator, Manual Control Buttons, a DC Power Jack, and 2 USB-C ports. These can be seen in Figure 7.1. Each component is described below.



Figure 7.1 – Gripper interfaces

7.2 LED Indicator

Once the Gripper is powered, the LED Indicator indicates the status of the Controller. The LED Indicator can be in one of four states which are described in Table 7.1.

Table 7.1 – States of the LED Indicator

State	Description
Off	There is no power to the Controller
Solid yellow	The Gripper is initialising
Solid white	The Gripper is functioning normally, sampling data from the sensors and ready to receive commands.
Solid green	The Gripper is actively moving in position control mode (including via the open/close manual control buttons).
Solid blue	The Gripper is actively gripping in the fixed force gripping mode.
Solid pink	The Gripper is actively gripping in the dynamic force gripping mode.
Solid orange	The Gripper has not received valid data from the sensor within a timeout and cannot receive sensor-dependant commands – e.g., <code>FF_GRIP</code> , <code>DF_GRIP</code> , <code>GET_TACTILE</code>
Solid red	The Gripper is in an error state and cannot receive any commands

7.3 Manual Control Buttons

There are three manual control buttons. The functions of each button are described in Table 7.2.

Table 7.2 – Functions of the Manual Control Buttons

Button	Description
CLOSE	Manually close the gripper while button is held down (speed 112.0 mm/s per finger).
OPEN	Manually open the gripper while button is held down (speed 112.0 mm/s per finger).
DF_GRIP	Manually initiate dynamic grip force control

7.4 DC Power Jack

Care should be taken when connecting the Gripper to power via the male 5.5 mm DC Power Jack. Only use a power pack matching the specifications described in section 3.2.3 Gripper Power Pack and refer to section 5.4 Power for warnings and cautions.

7.5 USB-C Port #1

USB-C Port #1 has High Speed capabilities and is used for streaming sensor data and sending/receiving commands/responses to/from the gripper.

7.6 USB-C Port #2

USB-C Port #2 has Full Speed capabilities and is used for firmware updates.

7.6.1 Gripper serial communications

The Gripper serial communications interface can be accessed through the COM Port of USB-C Port #1.

The Gripper communications interface allows the user to send commands to the gripper to bias the PapillArray Tactile Sensor arrays, to retrieve gripper-specific data, and the set parameters for gripping, and of course initiating movements and gripping in various grip modes.

For more information, see section 9 Communication protocol.

7.6.2 Data streaming

Currently, the fully resolved PapillArray Tactile Sensor data, gripper width, and IMU data are available through USB-C Port #1 at a rate of 500 Hz per second.

In future, the user will be able to configure the data streaming to select which data to stream.

For more information, see section 9 Communication protocol.

7.7 M8 Port

The M8 port is intended to be compatible with the tool connector from Universal Robots. The connections are described below. The output tool connector on the Gripper shares the same connections as the input cable described in Figure 7.2 and Table 7.3.



Figure 7.2 – M8 port pin locations

Table 7.3 – M8 cable pin descriptions

Pin	Function
1	RS485 +
2	RS485 -
3	Unused
4	Unused
5	24 V DC
6	Unused
7	Unused
8	GND (0 V DC)

The connection parameters for RS485 communications are described in Table 7.4.

Table 7.4 – RS485 parameters

Parameter	Value	Units
Baud Rate	115200	Bits/s
Word length (incl. Parity)	8	Bits
Parity	None	-
Stop Bits	1	Bits

RS485 can be used to send/receive commands/responses to/from the gripper as described in 9 Communication protocol. RS485 cannot be used for streaming data from the gripper.

When powered through the M8 connector from the Tool I/O port of a robot arm, the user must ensure that the current limits of the robot arm are sufficient for the gripper operations that the user wishes to execute. For more information, see section 5.4.1 Current draw profile.

8 Maintenance

8.1 General

For most applications, there are no parts that need to be replaced during normal operation.

8.2 Cleaning



CAUTION: The Sensor casing, gripper fingers and Gripper casing end-cap material is anodised aluminium. Do not clean with strong alkaline or acidic substances which can cause corrosion. Isopropanol is a suitable cleaning agent; however, care must be taken to avoid liquid ingress.



CAUTION: The Sensor pillar material is silicone. Do not clean with strong alkaline or acidic substances. Isopropanol is a suitable cleaning agent; however, care must be taken to avoid liquid ingress.



CAUTION: The Gripper casing material is nylon. Clean using soap and water; however care must be taken to avoid liquid ingress.

The Gripper and Sensors must be kept free of excessive dust, debris, and moisture. Debris and dust should be kept from accumulating on or in the Gripper and Sensors.

8.3 PapillArray Tactile Sensor silicone integrity

Periodic inspection of the condition of the silicone of the PapillArray Tactile Sensors is recommended. During normal use, the silicone surface finish may become dull - this is normal.

If the silicone appears worn or there are signs of damage, the silicone may need to be replaced and the sensor recalibrated. Contact Contactile for options on replacing the silicone and recalibration.

8.4 Cabling and connectors



CAUTION: Do not disconnect the PapillArray Tactile Sensors from the Gripper. The Gripper is not intended to operate without the Sensors.

In industrial-like applications that continuously or frequently move the system's cabling, you should periodically check the cables for signs of wear.

8.5 Periodic calibration

Periodic calibration of the sensor and its electronics is required to maintain accuracy and resolution. We recommend annual recalibrations, especially for applications that frequently cycle the loads applied to the sensor. Contact Contactile for options on recalibration.

9 Communication protocol

9.1 Command template

9.1.1 Sending a command

All commands are ASCII text with the following structure:

```
#<CMD_NAME>:<ARG0>,<ARG1>,...,<ARGN>$
```

- The hash (#) character indicates the start of a new command.
- The <CMD_NAME> is the command name.
- If the command requires any arguments, the command name is followed by a colon (:) character and then a comma (,) separated argument list.
- The dollar (\$) character indicates the end of the command.

9.1.2 Receiving gripper responses

If a command is successfully received by the gripper, the gripper will send a response (within 2.5 ms). The response has the following structure:

```
#<CMD_NAME>:<ARG0>,<ARG1>,...,<ARGN>|<RSP_TYPE>,<ID>$
```

- The hash (#) character indicates the start of a new command response.
- The <CMD_NAME> is the command name to which a response is being received.
- If the response returns any arguments, the command name is followed by a colon (:) character and then a comma (,) separated return argument list.
- The pipe (|) character indicates the end of the return argument list.
- The <RSP_TYPE> is the response type. The response types are described in Table 9.1.
- The <ID> is an identifier that is generated by the gripper. The ID is an unformatted integer between 0 and 99,999.
- The dollar (\$) character indicates the end of the command response.

Table 9.1 – Response types

Response Type	Description
ACK	The gripper has received a valid command with valid arguments
INV	The gripper has received a valid command with invalid argument, or the gripper is not in a state to execute the command – e.g., the FF_GRIP and DF_GRIP, GET_TACTILE and GET_TACTILE_GLOBAL commands will return INV if either of the tactile sensors are disconnected.

If a command is sent that is malformed or uninterpretable by the gripper (e.g. due to a typographical error, or the absence of the leading and trailing # and \$, respectively), there will be no response from the gripper.

9.2 List of commands

The commands that have been implemented are described in the following tables:

- Commands for (synchronously) streaming data are in Table 9.2 on USB Port #1
- Commands for the tactile sensors are in Table 9.3
- Commands for (asynchronously) retrieving sensor and gripper data are in Table 9.4
- Commands for position control are in Table 9.5
- Commands for fixed grip force control are in Table 9.6
- Commands for dynamic grip force control are in Table 9.7
- Commands for stopping and releasing grip are in Table 9.8
- Commands for polling the gripper state and errors are in Table 9.9

Table 9.2 – Data streaming commands*

CMD_NAME	Description	Example
DATA_STREAM	Starts a continuous data stream of the global sensor data, global force and global torque Values are reported to 2 decimal places, <TIME_STAMP> is in microseconds. See GET_TACTILE, GET_WIDTH and GET_IMU for units and ranges of respective data. This data stream is started by default on USB-C Port #1 when the gripper is powered on.	Command: #DATA_STREAM\$ Response: #DATA_STREAM ACK,1110\$ Subsequent data stream: #DATA_STREAM:<TIME_STAMP>,<S0_FX>,<S0_FY>,<S0_FZ>,<S0_TX>,<S0_TY>,<S0_TZ>,<S1_FX>,<S1_FY>,<S1_FZ>,<S1_TX>,<S1_TY>,<S1_TZ>,<WIDTH>,<ACC_X>,<ACC_Y>,<ACC_Z>,<GYR_X>,<GYR_Y>,<GYR_Z>,<S0_P0_FX>,<S0_P0_FY>,<S0_P0_FZ>,<S0_P1_FX>,<S0_P1_FY>,<S0_P1_FZ>,...,<S0_P8_FX>,<S0_P8_FY>,<S0_P8_FZ>,<S1_P0_FX>,<S1_P0_FY>,<S1_P0_FZ>,<S1_P1_FX>,<S1_P1_FY>,<S1_P1_FZ>,...,<S1_P8_FX>,<S1_P8_FY>,<S1_P8_FZ>,<S0_FRIC>,<S1_FRIC>,<STATE>,<LAST_ERROR>\$
DATA_STOP	Stops the data stream. It is recommended to flush/reset the input buffer after a DATA_STOP command.	Command: #DATA_STOP\$ Response: #DATA_STOP ACK,1111\$
DATA_STREAM_GLOBAL	Starts a continuous data stream of the global force and global torque Values are reported to 2 decimal places, <TIME_STAMP> is in microseconds. See GET_TACTILE_GLOBAL for units and ranges of respective data.	Command: #DATA_STREAM_GLOBAL\$ Response: #DATA_STREAM_GLOBAL ACK, 1110\$ Subsequent data stream: #DATA_STREAM_GLOBAL: <TIME_STAMP>,<S0_FX>,<S0_FY>,<S0_FZ>,<S0_TX>,<S0_TY>,<S0_TZ>,<S1_FX>,<S1_FY>,<S1_FZ>,<S1_TX>,<S1_TY>,<S1_TZ>,<S0_FRIC>,<S1_FRIC>,<STATE>,<LAST_ERROR>\$
DATA_STOP_GLOBAL	Stops the global data stream. It is recommended to flush/reset the input buffer after a DATA_STOP_GLOBAL command.	Command: #DATA_STOP_GLOBAL\$ Response: #DATA_STOP_GLOBAL ACK,1111\$

* Sending these commands over the RS485 interface will result in a response from the gripper over the RS485 interface and the resulting data stream will start/stop on USB-C Port #1. It is not possible to stream data on the RS485 interface.

Table 9.3 – Tactile sensor commands

CMD_NAME	ARGS	RETURN	Description	Example
BIAS	-	-	Zeros the tactile sensor output. This should be called only when the sensors are known to be unloaded (i.e., not in contact with anything)	Command: #BIAS\$ Response: #BIAS ACK,1112\$

Table 9.4 – Commands for retrieving Gripper sensor data

CMD_NAME	ARGS	RETURN	Description	Example
GET_WIDTH	-	WIDTH: ▪ Type: float ▪ Units: mm Value is reported to 2 decimal places.	Returns the distance between unloaded pillar tips on opposite gripper fingers. For example: pillars that are just touching would return a WIDTH of 0.00 mm. NOTE: A negative width is possible if the silicone pillars of the PapillArray Tactile Sensor arrays are in contact and compressed.	Command: #GET_WIDTH\$ Response: #GET_WIDTH:10.99 ACK,1113\$ The gripper fingers are 10.99 mm apart (from pillar tip to pillar tip on opposite fingers).
GET_VEL	-	VELOCITY: ▪ Type: float ▪ Units: mm/s Value is reported to 2 decimal places.	Returns the approximate velocity between the gripper fingers. As a function of motor RPM, velocity is non-linear across the gripper stroke. For a particular RPM, this value is calculated from the time it takes to move the fingers from fully open to full closed.	Command: #GET_VEL\$ Response: #GET_VEL:10.25 ACK,1114\$ The approximate gripper finger velocity is 10.25 mm/s.
GET_IMU	-	ACC_X ACC_Y ACC_Z: ▪ Type: float ▪ Units: G GYR_X GYR_Y GYR_Z: ▪ Type: float ▪ Units: deg/s	Returns the IMU data, where ACC_X, ACC_Y, and ACC_Z are accelerations in the X, Y and Z axes, respectively, and GYR_X, GYR_Y, and GYR_Z are the angular velocities around the X, Y and Z axes, respectively. Values are reported to 2 decimal places. The X, Y and Z axes of the IMU are defined as shown in Figure 5.1	Command: #GET_IMU\$ Response: #GET_IMU:0.11,-0.12,-1.00,0.24,-0.22,0.17 ACK,1115\$ The accelerations are 0.11, 0.12 and -1.00 G in the X, Y and Z axes, respectively, and the angular velocities are 0.24, 0.22 and 0.17 deg/s around the X, Y and Z axes, respectively.

GET_TACTILE	-	S0_FX S0_FY S0_FZ S0_TX S0_TY S0_TZ S1_FX S1_FY S1_FZ S1_TX S1_TY S1_TZ S0_P0_FX S0_P0_FY S0_P0_FZ ... S0_P8_FX S0_P8_FY S0_P8_FZ S1_P0_FX S1_P0_FY S1_P0_FZ ... S1_P8_FX S1_P8_FY S1_P8_FZ S0_FRIC S1_FRIC	Returns all the tactile sensor data. S0_FX, S0_FY, and S0_FZ are the global forces on S0 in the X, Y and Z axes, respectively, S0_TX, S0_TY and S0_TZ are the global torques on SEN0 around the X, Y and Z axes, respectively, S1_FX, S1_FY and S1_FZ are the global forces on SEN1 in the X, Y and Z axes, respectively, and S1_TX, S1_TY and S1_TZ around the global torques on SEN1 around the X, Y and Z axes, respectively. S0_P0_FX, S0_P0_FY and S0_P0_FZ are the forces on the X, Y and Z forces on Pillar 0 of SEN0, etc., and finally the current estimate of coefficient of static friction from S0 and S1 Values are reported as floats to 6 decimal places. All forces are in newtons (N). All Torques are in newton-millimetres (Nmm). The coefficient of static friction is a unitless value. If a friction has not been measured, the value is -1. The X, Y and Z axes of the sensors is shown in Figure 5.2. The reference point for each sensor is the unloaded tip position of P4. For more information, refer to the doc PTS_2.0_SPEC_MMMYY.pdf	Command: #GET_TACTILE\$ Response: #GET_TACTILE: <68 comma separated float values> ACK,1116\$ NB: Response is #GET_TACTILE INV,1116\$ if either tactile sensor is disconnected
GET_TACTILE_GLOBAL	-	S0_FX S0_FY S0_FZ S0_TX S0_TY S0_TZ S1_FX S1_FY S1_FZ S1_TX S1_TY S1_TZ S0_FRIC S1_FRIC	Returns the tactile sensor data, where S0_FX, S0_FY, and S0_FZ are the global forces on S0 in the X, Y and Z axes, respectively, S0_TX, S0_TY and S0_TZ are the global torques on SEN0 around the X, Y and Z axes, respectively, S1_FX, S1_FY and S1_FZ are the global forces on SEN1 in the X, Y and Z axes, respectively, and S1_TX, S1_TY and S1_TZ around the global torques on SEN1 around the X, Y and Z axes, respectively, and finally the current estimate of coefficient of static friction from S0 and S1. Values are reported as floats to 6 decimal places. All forces are in newtons (N). All Torques are in newton-millimetres (Nmm). The coefficient of static friction is a unitless value. If a friction has not been measured, the value is -1. The X, Y and Z axes of the sensors is shown in Figure 5.2. The reference point for each sensor is the unloaded tip position of P4. For more information, refer to the doc PTS_2.0_SPEC_MMMYY.pdf	Command: #GET_TACTILE_GLOBAL\$ Response: #GET_TACTILE_GLOBAL: <14 comma separated float values> ACK,1116\$ NB: Response is #GET_TACTILE_GLOBAL INV,1116\$ if either tactile sensor is disconnected

Table 9.5 – Position control commands

CMD_NAME	ARGS	RETURN	Description	Example
PC_SET_VEL	VEL: - Type: float - Units: mm/s - Limits: 5.00 to 334.0 - Default 167.0	-	Sets the target velocity for position control mode.	Command: #PC_SET_VEL:200.00\$ Response: #PC_SET_VEL:200.00 ACK,1117\$ Sets the position control target velocity to 200.00 mm/s.
PC_GET_VEL	-	VEL: - Type: float - Units: mm/s	Gets the target velocity for position control mode.	Command: #PC_GET_VEL\$ Response: #PC_GET_VEL:200.00 ACK,1118\$ Returns the value 200.00 mm/s.
PC_MOVE_TO_WIDTH *	WIDTH: - Type: float - Units: mm - Limits: 0.00 to 175.00	-	Moves the gripper fingers to the target WIDTH. Should not be used to terminate FF_GRIP or DF_GRIP.	Command: #PC_MOVE_TO_WIDTH:125.00\$ Response: #PC_MOVE_TO_WIDTH:125.00 ACK,1119\$

* PC_MOVE_TO_WIDTH is not intended to be used to grip an object. It is only intended to move the gripper fingers while unloaded.

Table 9.6 – Fixed grip force control commands

CMD_NAME	ARGS	RETURN	Description	Example
FF_SET_VEL	VEL: - Type: float - Units: mm/s - Limits: 5.0 to 167.0 - Default 167.0	-	Sets the target approach velocity (pre-contact) for fixed force control mode.	Command: #FF_SET_VEL:100.00\$ Response: #FF_SET_VEL:100.00 ACK,1120\$ Returns the value 100.00 mm/s.
FF_GET_VEL	-	VEL: - Type: float - Units: mm/s	Gets the target velocity (pre-contact) for fixed force control mode.	Command: #FF_GET_VEL\$ Response: #FF_GET_VEL:200.00 ACK,1121\$ Returns the value 200.00 mm/s.
FF_SET_FORCE *	FORCE: - Type: float - Units: N - Limits: 1.50 to 30.00 - Default: 5.00	-	Sets the target grip force for fixed force control mode.	Command: #FF_SET_FORCE:20.00\$ Response: #FF_SET_FORCE:20.00 ACK,1122\$ Sets the fixed grip target grip force to 20.00 N.
FF_GET_FORCE	-	FORCE: - Type: float - Units: N	Gets the target grip force for fixed force control mode.	Command: #FF_GET_FORCE\$ Response: #FF_GET_FORCE:20.00 ACK,1123\$ Returns the value 20.00 N.
FF_GRIP	-	-	Starts fixed force control gripping.	Command: #FF_GRIP\$ Response: #FF_GRIP ACK,1124\$ <i>NB:</i> Response is #FF_GRIP INV,1124\$ if either tactile sensor is disconnected

* The command FF_SET_FORCE can be called while an FF_GRIP command is being executed (when the gripper is in the FF_APPROACH or FF_HOLD state - see 10.1 States). This will update the target grip force and the gripper will enter the FF_APPROACH state, without requiring the gripper to stop.

Table 9.7 – Dynamic grip force control commands

CMD_NAME	ARGS	RETURN	Description	Example
DF_SET_VEL	VEL: - Type: float - Units: mm/s - Limits: 5.00 to 167.0 - Default: 167.0	-	Sets the target approach velocity (pre-contact) for dynamic force control mode.	Command: #DF_SET_VEL:100.00\$ Response: #DF_SET_VEL:100.00 ACK,1125\$ Sets the target velocity to 100 mm/s.
DF_GET_VEL		VEL: - Type: float - Units: mm/s	Gets the target approach velocity (pre-contact) for dynamic force control mode.	Command: #DF_GET_VEL\$ Response: #DF_GET_VEL:100.00 ACK,1126\$ Gets the target velocity value 100 mm/s.
DF_SET_EXF	FORCE: - Type: float - Units: N - Limits: 1.50 to 15.00 - Default: 5.00	-	Sets the target exploratory grip force for dynamic force control mode. See section 6.3.1	Command: #DF_SET_EXF:4.00\$ Response: #DF_SET_EXF:4.00 ACK,1127\$ Sets the target exploratory force to 4 N.
DF_GET_EXF		FORCE: - Type: float - Units: N	Gets the target exploratory grip force for dynamic force control mode. Default: 5.00	Command: #DF_GET_EXF\$ Response: #DF_GET_EXF:4.00 ACK,1128\$ Gets the target exploratory force value 4 N.
DF_SET_MXF *	FORCE: - Type: float - Units: N - Limits: 1.50 to 30.00 - Default: 30.00	-	Sets the maximum grip force for dynamic force control mode.	Command: #DF_SET_MXF:4.00\$ Response: #DF_SET_MXF:4.00 ACK,1127\$ Sets the maximum target force to 4 N.
DF_GET_MXF *		FORCE: - Type: float - Units: N	Gets the grip force for dynamic force control mode. Default: 30.00	Command: #DF_GET_MXF\$ Response: #DF_GET_MXF:4.00 ACK,1128\$ Gets the maximum target force value 4 N.
DF_SET_SHEARG	SHEAR_GAIN: - Type: float - Units: - - Limits: 1.10 to 3.00 - Default: 1.20	-	Sets the shear gain for dynamic force control mode. See section 6.3.1	Command: #DF_SET_SHEARG:1.50\$ Response: #DF_SET_SHEARG:1.50 ACK,1127\$ Sets the shear gain to 1.5.
DF_GET_SHEARG		SHEAR_GAIN: - Type: float - Units: -	Gets the shear gain for dynamic force control mode. See section 6.3.1	Command: #DF_GET_SHEARG\$ Response: #DF_GET_SHEARG:1.50 ACK,1128\$ Gets the shear gain value 1.5.

* Not implemented yet

DF_SET_TORQG	TORQUE_GAIN: - Type: float - Units: m^{-1} - Limits: 0.00 to 2000.00 - Default: 0.00	-	Sets the torque gain for dynamic force control mode. A typical value is 500.00 m^{-1} See section 6.3.1	Command: #DF_SET_TORQG:4.00\$ Response: #DF_SET_TORQG:4.00 ACK,1129\$ Sets the torque gain to 4 m^{-1}.
DF_GET_TORQG		TORQUE_GAIN: - Type: float - Units: m^{-1}	Gets the torque gain for dynamic force control mode. See section 6.3.1	Command: #DF_GET_TORQG\$ Response: #DF_GET_TORQG:4.00 ACK,1130\$ Gets the torque gain value 4 m^{-1}.
DF_DYN_EXF_EN	-	-	Enables dynamic exploration force for DF_GRIP mode, based on the measured compliance of object.	Command: #DF_DYN_EXF_EN\$ Response: #DF_DYN_EXF_EN ACK,1131\$ By default, dynamic exploration force is disabled and exploratory force is fixed.
DF_DYN_EXF_DIS	-	-	Disables dynamic exploration force for DF_GRIP mode.	Command: #DF_DYN_EXF_DIS\$ Response: #DF_DYN_EXF_DIS ACK,1132\$ By default, dynamic exploration force is disabled and exploratory force is fixed.
DF_GET_DYN_EXF		DYN_EXF: - Type: int - Units: -	Gets the flag for dynamic exploration force for DF_GRIP mode. 0 is disabled 1 is enabled	Command: #DF_GET_DYN_EXF\$ Response: #DF_GET_DYN_EXF:0 ACK,1133\$ Gets the dynamic exploration force flag value 0 (disabled).
DF_GRIP	-	-	Starts dynamic force control gripping.	Command: #DF_GRIP\$ Response: #DF_GRIP ACK,1134\$ NB: Response is #DF_GRIP INV,1134\$ if either tactile sensor is disconnected

Table 9.8 – Stopping commands

CMD_NAME	ARGS	RETURN	Description	Example
STOP	-	-	Stops the current motion command	Command: #STOP\$ Response: #STOP ACK,1135\$
BRAKE	-	-	Performs a brake that opposes the gripper's current motion	Command: #BRAKE\$ Response: #BRAKE ACK,1136\$
RELEASE	-	-	Releases the object currently being held in FF_GRIP or DF_GRIP modes (moves just out of contact). <i>NB: If STATE is FF_APPROACH or DF_APPROACH, then RELEASE behaves like STOP.</i>	Command: #RELEASE\$ Response: #RELEASE ACK,1137\$

Table 9.9 – State and error commands

CMD_NAME	ARGS	RETURN	Description	Example
GET_STATE	-	STATE ▪ Type: int ▪ Units: -	Gets the current state of the gripper. See 10.1 States for information about the gripper states	Command: #GET_STATES\$ Response: #GET_STATE:0 ACK,1138\$ Gets the current state of 0.
GET_LAST_ERROR	-	ERROR ▪ Type: int ▪ Units: -	Gets the last error experienced by the gripper. See 10.2 Errors for more information about gripper errors.	Command: #GET_LAST_ERROR\$ Response: #GET_LAST_ERROR:1 ACK,1139\$ Gets the last error of 1.
CLEAR_ERROR_STATE <i>(previously called CLEAR_LAST_ERROR)</i>	-	-	Returns STATE to IDLE_READY if error is not persistent. Does not clear the error – i.e. GET_LAST_ERROR will still return the code of the last error encountered. See 10.2 Errors for more information about gripper errors.	Command: #CLEAR_ERROR_STATES\$ Response: #CLEAR_ERROR_STATE ACK,1140\$
RESET_PARAMS	-	-	Resets all parameters back to the default values. Does not clear errors or change the state of the gripper.	Command: #RESET_PARAMS\$ Response: #RESET_PARAMS ACK,1141\$

Table 9.10 – Gripper statistics commands

CMD_NAME	ARGS	RETURN	Description	Example
WHO_AM_I *	-	GRIPPER_SN ▪ Type: String HW_BUILD ▪ Type: uint16 FW_VERSION ▪ Type: String SEN0_SN ▪ Type: String SEN1_SN: ▪ Type: String	Gets the gripper identification values: Gripper serial number, gripper hardware build, gripper firmware version, SEN0 serial number and SEN1 serial number.	Command: #WHO_AM_I\$ Response: #WHO_AM_I: GAL200026,6,1.0.0, TS220247,TS220123 ACK, 1140\$ Gets all the gripper identification values.
GET_STATISTICS*	-	N_PC_MOVE_TO_WIDTH N_PC_MOVE_TO_WIDTH_FAIL N_PC_MOVE_TO_WIDTH_OBSTR N_FF_GRIP N_FF_GRIP_FAIL N_FF_GRIP_OBSTR N_DF_GRIP N_DF_GRIP_FAIL N_DF_GRIP_OBSTR N_RELEASE N_RELEASE_FAIL N_BRAKE N_BRAKE_FAIL N_STOP N_BUTTON_OPEN N_BUTTON_CLOSE N_BUTTON_OPEN_OBSTR N_BUTTON_CLOSE_OBSTR N_FAULT_TEMP TOTAL_GRIP_COUNT TOTAL_UPTIME DIST_PC_MOVE DIST_FF_APPROACH DIST_FF_HOLD DIST_DF_APPROACH DIST_DF_HOLD DIST_RELEASE DIST_BRAKE DIST_BUTTON_OPEN DIST_BUTTON_CLOSE FF_HOLD_TIME_S_0-5N FF_HOLD_TIME_S_5-10N FF_HOLD_TIME_S_10-15N FF_HOLD_TIME_S_15-20N FF_HOLD_TIME_S_20-25N FF_HOLD_TIME_S_25-30N FF_HOLD_TIME_S_30+N DF_HOLD_TIME_S_0-5N DF_HOLD_TIME_S_5-10N DF_HOLD_TIME_S_10-15N DF_HOLD_TIME_S_15-20N DF_HOLD_TIME_S_20-25N DF_HOLD_TIME_S_25-30N DF_HOLD_TIME_S_30+N SEN0_SN SEN0_N_SLIP[0..8] SEN0_MAX_FN[0..8] SEN0_MAX_FT[0..8] SEN1_SN SEN1_N_SLIP[0..8] SEN1_MAX_FN[0..8] SEN1_MAX_FT[0..8]	Gets the usage statistics of the gripper. ▪ N_ and COUNT values are uint16 ▪ All DIST_, TIME, UPTIME, FN and FT values are float ▪ SEN0_SN and SEN1_SN are strings See 11.6 Generating a usage report, for example interpretation of values.	Command: #GET_STATISTICS\$ Response: #GET_STATISTICS:3751, 165,0,14,0,0,147,1,1, 79,0,0,0,0,203,83,1,3, 0,0,182187,77845.59, 64.22,1.17,809.90, 1049.53,101.19,26.59, 1209.99,551.22,92.75, 120.46,69.80,0.23, 7.81,0.41,15.87, 551.43,2204.18, 237.91,699.23,318.27, 89.29,19.83,TS220247, 6,45,11,38,17,51,4,35, 5,9.38,9.83,8.30, 11.60,13.21,8.24, 21.71,8.28,4.92,2.60, 5.75,4.37,3.73,4.39, 3.90,10.06,5.79,2.91, TS220123,4,14,4,18,19, 49,3,34,9,8.40,11.74, 6.87,7.78,9.91,10.92, 11.65,10.25,6.73,4.93, 3.13,4.25,3.28,3.22, 4.10,5.26,4.72,5.41 ACK,1141\$ Gets all the statistics.

* NB. If latest gripper firmware is installed but gripper hardware is not compatible, gripper will respond to these commands with INV. If gripper firmware is not up to date, gripper will not respond to these commands.

9.3 Command subsets available for different ports

- USB-C Port #1: All commands are available through this port
- RS485: Data streaming is not available through this interface. Sending data streaming commands over the RS485 interface will result in a response from the gripper over the RS485 interface and the resulting data stream will start/stop on USB-C Port #1. All other commands are available through this port.

10 Gripper State Machine

10.1 States

During normal operation, the gripper will transition from one state to another. The state machine diagram in Figure 10.1 shows the different states and transition events.

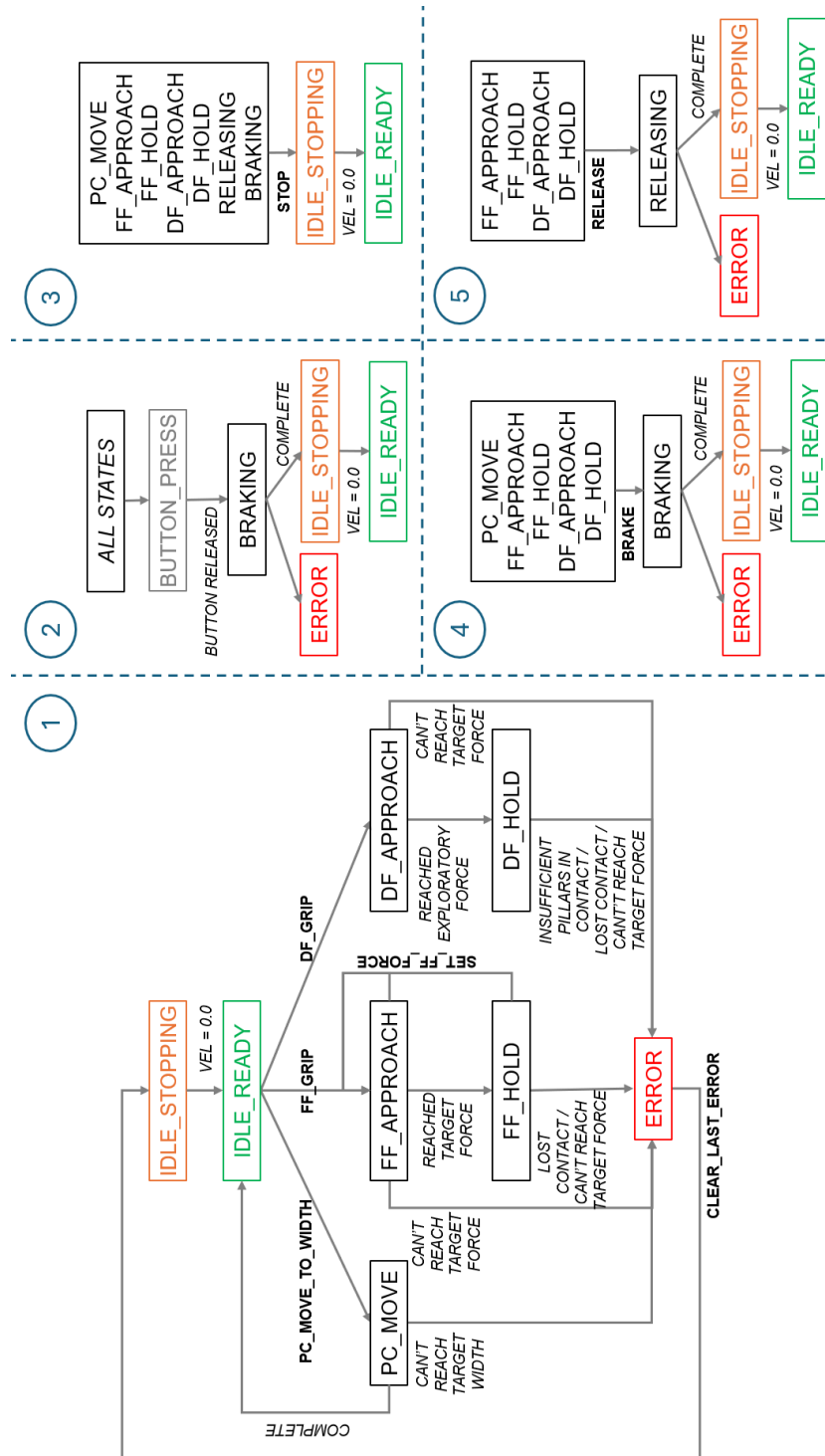


Figure 10.1 – State diagram

The states are described in Table 10.1

Table 10.1 – State descriptions

State	Value	Description	Transitions from	Transitions to
IDLE_READY	0	The gripper is not executing any movements, the motor is off, velocity is 0 mm/s. <i>The gripper is ready for next movement command.</i>	IDLE_STOPPING	PC_MOVE FF_APPROACH DF_APPROACH ERROR
IDLE_STOPPING	1	The gripper is not executing any movements, the motor is off, but velocity is non-zero. <i>The gripper is not ready for next movement command.</i>	ERROR STOPPING RELEASING BRAKING	IDLE_READY ERROR
PC_MOVE	2	The gripper is actively moving to the specified width after issuing a PC_MOVE_TO_WIDTH command	IDLE_READY	IDLE_READY IDLE_STOPPING ERROR
FF_APPROACH	3	The start of a FF_GRIP before the gripper has first applied the target force.	IDLE_READY FF_APPROACH FF_HOLD	FF_HOLD IDLE_STOPPING BRAKING RELEASING ERROR
FF_HOLD	4	The gripper is actively maintaining the target force during the execution of a FF_GRIP command.	FF_APPROACH	FF_APPROACH IDLE_STOPPING BRAKING RELEASING ERROR
DF_APPROACH	5	The start of a DF_GRIP before the gripper has applied the exploratory force.	IDLE_READY	DF_HOLD IDLE_STOPPING BRAKING RELEASING ERROR
DF_HOLD	6	The gripper is actively applying the dynamically calculated grip force during the execution of a DF_GRIP command.	DF_APPROACH	IDLE_STOPPING BRAKING RELEASING ERROR
BRAKING	7	The gripper is actively braking after issuing the BRAKE command.	PC_MOVE FF_APPROACH FF_HOLD DF_APPROACH DF_HOLD	IDLE_STOPPING ERROR
RELEASING	8	The gripper is actively releasing after issuing the RELEASE command.	FF_APPROACH FF_HOLD DF_APPROACH DF_HOLD	IDLE_STOPPING ERROR
BUTTON_PRESS	9	The gripper is moving due to a button press (OPEN and CLOSE buttons) NB – Pressing the GRIP button would cause the gripper to transition to a DF_APPROACH state	Any state	BRAKING ERROR
ERROR	-1	The gripper experienced an error while executing a movement command, or has experienced a hardware fault.	Any state	IDLE_STOPPING

10.2 Errors

The error values returned by the command `GET_LAST_ERROR` are described in Table 10.2.

Table 10.2 – Error descriptions

Error Value	Error Name	Description	States generating this error	Troubleshooting
0	NO_ERROR	No error. The last movement completed successfully.	Any	-
1	OUT_OF_BOUNDS	Unreachable target width while opening the gripper.	PC_MOVE RELEASING	This can potentially occur when opening the gripper using the manual control buttons beyond the max open position. This can potentially occur when gripping an object whose width is close to the stroke of the gripper, then calling <code>RELEASE</code> .
2	ERR_PC_RINGING	Position ringing detected while trying to reach a target width	PC_MOVE	Contact support.
3	MOTION_OBSTRUCTION	An unexpected current in the motor has occurred indicative of an obstacle opposing the gripper arm motion.	PC_MOVE FF_APPROACH DF_APPROACH RELEASE	Check that there are no objects preventing the gripper from moving to the target width/force
4	UNABLE_TO_REACH_TARGET_FORCE	The gripper cannot apply the target force to the object between the sensors.	FF_APPROACH FF_GRIP DF_APPROACH DF_GRIP	This can occur when gripping an extremely deformable object, or when an insufficient number of pillars are in contact with the object being gripped, or if there is an obstacle opposing the gripper arm motion (despite the sensors also being in contact).
100	NO_MOTOR_VOLTAGE	No voltage is detected in the gripper motor.	PC_MOVE FF_APPROACH FF_HOLD DF_APPROACH DF_HOLD	Contact support.
101	OVER_TEMP	The motor temperature is too high.	PC_MOVE FF_APPROACH FF_HOLD DF_APPROACH DF_HOLD	If the gripper has been gripping for a long period of time, stop gripping, and allow the gripper motor temperature to dissipate and return to normal before attempting to grip again

11 Visualisation Software

The visualisation software is a standalone executable for visualising the gripper and all sensor data of the gripper.

The executable is compatible with both Windows and Linux operating systems.

11.1 Connecting the gripper to a PC

Power the gripper using the DC power jack and the power plug pack that was supplied with the gripper.

Connect a USB-C cable to USB Port #1 on the gripper, and the other end of the cable to your PC/laptop.

11.2 Launching the Visualisation Software

11.2.1 Windows

Navigate to the GAL2_VIS_WIN_vX.X.exe file and double click the icon.

11.2.2 Linux

In a command terminal, navigate to the folder containing the GAL2_VIS_LIN_vX.X file. Type the following in the command prompt followed by <Enter>:

```
./GAL2_VIS_LIN_vX.X
```

11.3 Connecting to the COM Port

After launching the visualisation software, a dialog box will appear prompting you to select the COM Port.

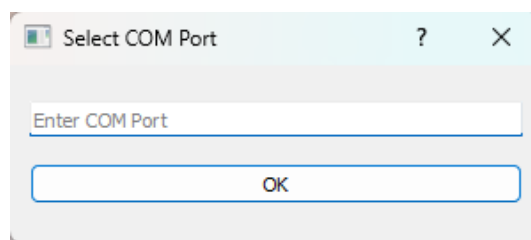


Figure 11.1 – Visualisation Software: COM Port selection

11.3.1 Windows

You can check the COM Port name by opening Device Manager. The Gripper port will appear under the category “Ports (COM & LPT)”:

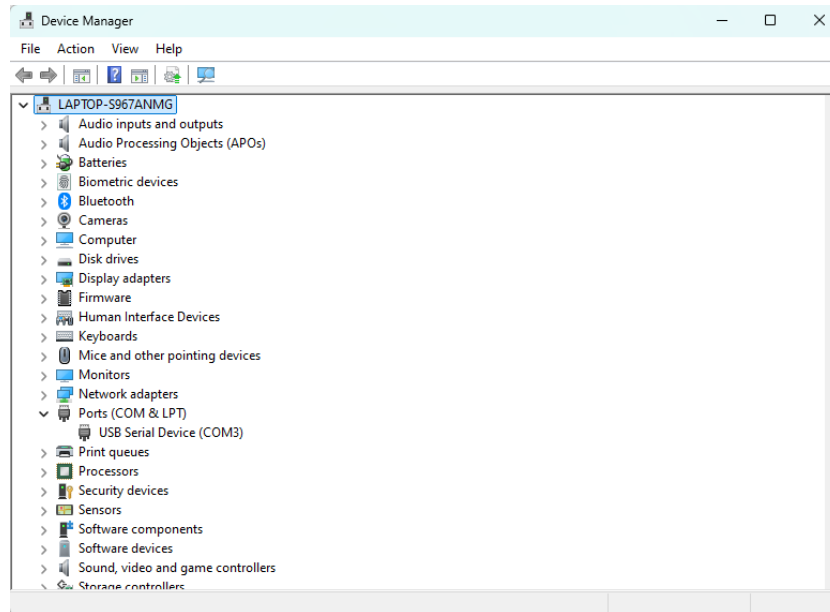


Figure 11.2 – Visualisation Software: COM Port identification in Windows Device Manager

Enter the COM Port name in the text box and select OK. For the example above, enter “COM3” in the text box and select OK.

11.3.2 Linux

You can check the COM Port name by typing the following at the prompt of a command terminal, followed by <Enter>:

```
hwinfo --short
```

This command will display a list of connected devices. For example:

```
modem:
      /dev/ttyACM0          STMicroelectronics Virtual COM Port
```

The Gripper appears as STMicroelectronics Virtual COM Port.

The COM Port name is what follows “/dev/tty” for the Gripper – for the example above, type “ACM0” in the text box and select OK.

NB: Ensure that the user has permissions to read/write to this COM port (by adding the user to the dialout group).

11.4 Visualisation

Once the COM port has been selected, the following window will appear:

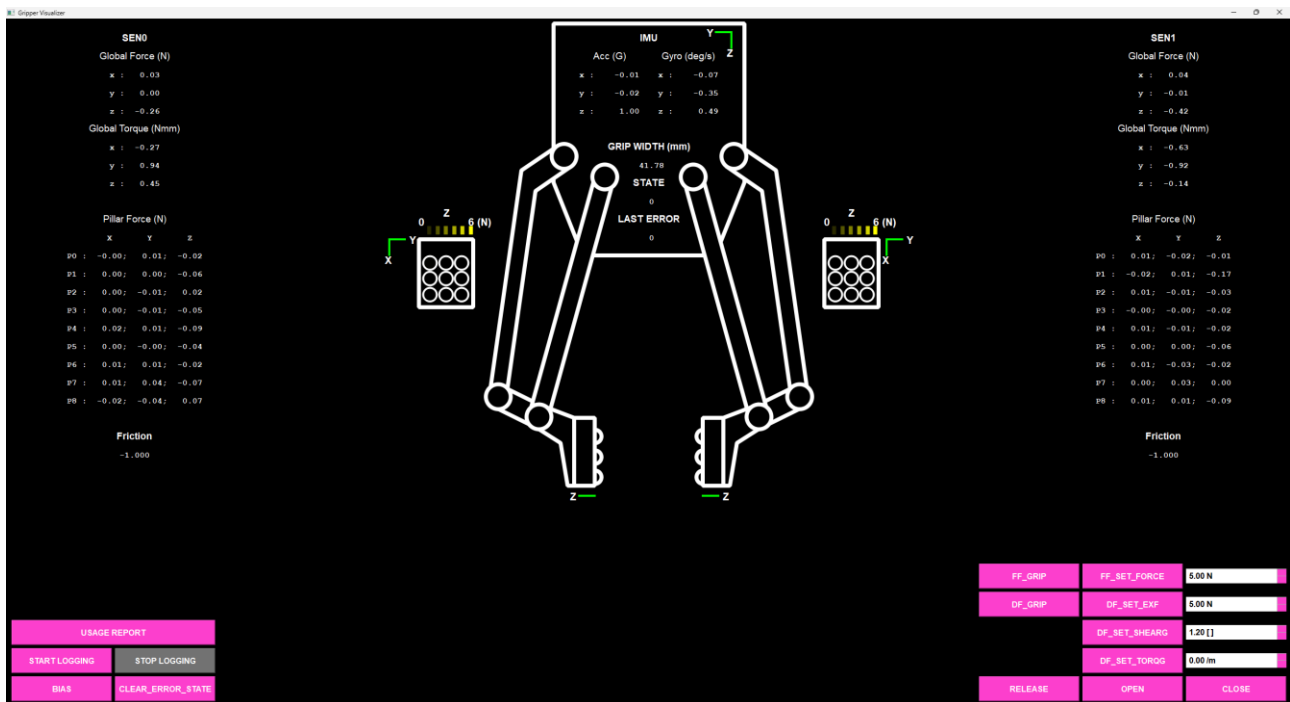


Figure 11.3 – Visualisation Software: Main window

11.4.1 Gripper width

The gripper width is displayed in two ways:

1. The width is displayed as text in between the fingers towards the main body of the gripper
2. The width is displayed graphically as the distance between the gripper fingers which move

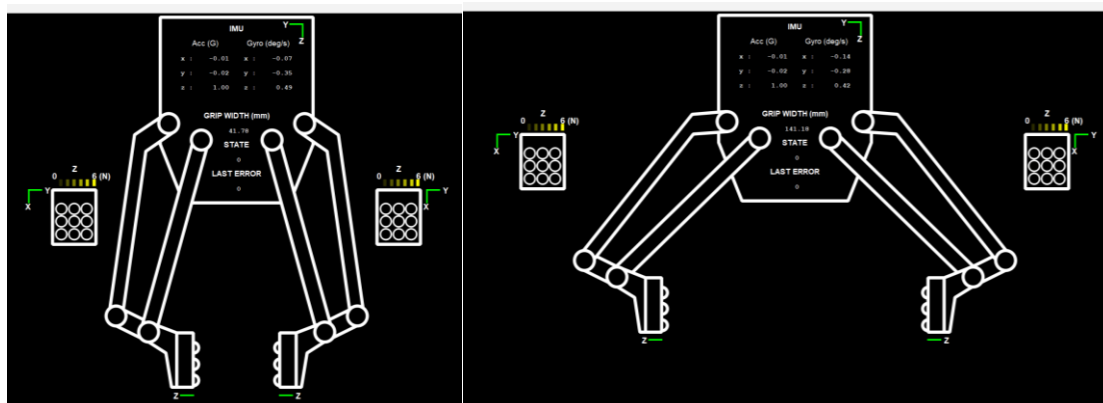


Figure 11.4 – Visualisation Software: Visualisation of gripper width. Left: gripper in an almost fully closed position. Right: gripper in a more open position.

11.4.2 Tactile sensor data

Tactile sensor data is displayed in two ways:

1. The 3D global forces, 3D global torques, 3D pillar forces and coefficient of static friction for each sensor are displayed as text – SEN0 data is displayed on the left side of the window, and SEN1 data is displayed on the right side of the window
2. The global Z-forces are displayed as vectors represented by a pink arrow at the sensor locations on the gripper fingers and the global shear forces are displayed as vectors represented by a pink arrow at a second sensor visualisation showing the top-view of the sensors.
3. The local Z-forces are displayed as a heat map overlaid on the top-view sensor visualisation with a legend above displayed above the top-view sensor visualisation.

The axis coordinate systems of the respective tactile sensors are shown beneath each sensor. When both sensors are in contact, an “object” (cyan rectangle) is drawn between the two sensors.

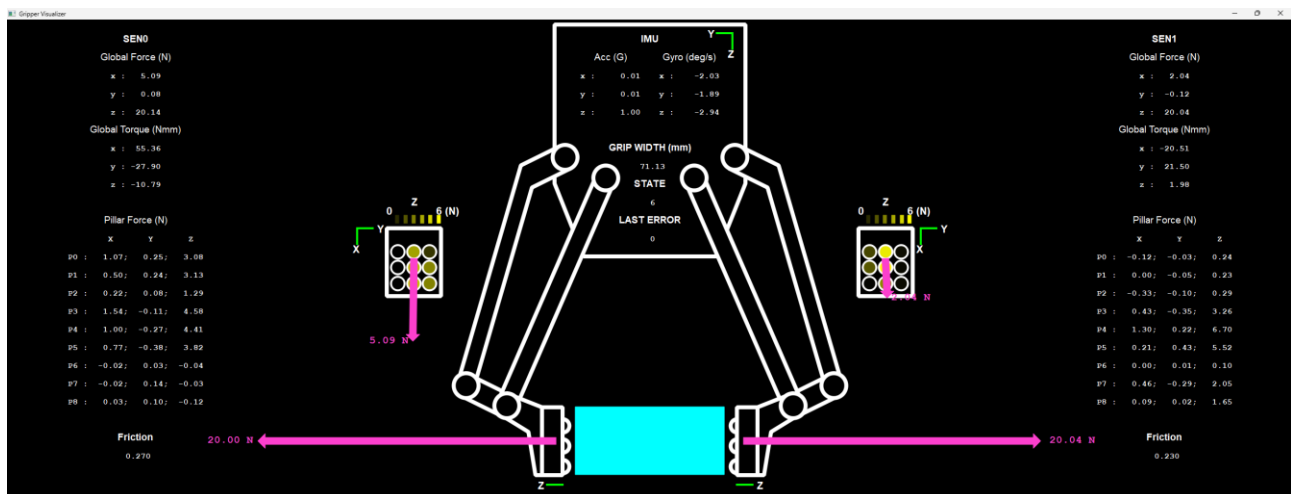


Figure 11.5 – Visualisation Software: Visualisation of tactile sensor data

11.4.3 IMU data, State and Last Error

IMU data is displayed as text in the top centre of the main window. The Accelerometer data is displayed in the left column, and the gyroscope data is displayed in the right column. Axis coordinate system is shown to the top right of the data.

The gripper STATE and LAST_ERROR are displayed below the IMU data and Gripper Width.



Figure 11.6 – Visualisation Software: Visualisation of IMU data, State and Last Error

11.5 Logging data

The gripper data can be logged to a .csv file using the START LOGGING button (in the button group in the bottom left of the visualiser).



Figure 11.7 – Visualisation Software: Data logging buttons; no log started

Once the START LOGGING button is pressed, a log file is created in the Logs folder (in the same location as the visualiser executable) and data is written to the file as it is streamed from the gripper. The STOP LOGGING button is activated and the START LOGGING button is deactivated.

To stop logging and close the log file, press the STOP LOGGING button (in the button group in the bottom left of the visualiser).

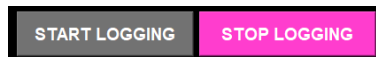


Figure 11.8 – Visualisation Software: Data logging buttons; log started

The log file is a comma-separated vales (CSV) file with a header row identifying each column of data and units and each row representing a sample from the data stream (see description of DATA_STREAM in Table 9.2 for more information about the order of values).

11.6 Generating a usage report

The usage statistics of the gripper can be used to generate a usage report as a .html file. The filename is STAT_YYYYMMDD_hhmmss.html where YYYYMMDD_hhmmss is a timestamp of when the report was generated, and the file is stored in the Logs folder (in the same location as the visualiser executable). Depending on the system settings, the report may automatically be rendered in the default browser.



Figure 11.9 – Visualisation Software: Usage report button

When rendered in a browser, the usage report displays all the statistics and aggregate data from both the WHO_AM_I command and GET_STATISTICS command, and also suggests whether the system may be operating normally, or where there may be cause to suspect wear, damage or improper use.



GAL2 Usage Report

2026-06-18 12:03:28

WHO AM I

Gripper Serial Number: GAL200026
Gripper HW Version: 6
Gripper FW Version: 1.0.0
SEN0 Serial Number: TS220247
SEN1 Serial Number: TS220123

GENERAL

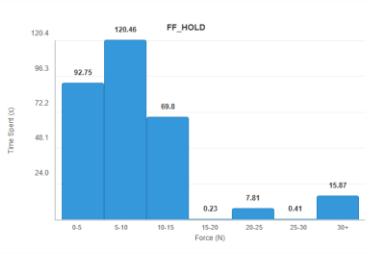
Total Uptime (h): 50.8 Your gripper should be operating optimally
Total Distance Travelled (mm): 81659.4 Your gripper should be operating optimally
Total Grip Count (counts): 101.0 Your gripper should be operating optimally
Total Temperature Faults (counts): 0.0 Your gripper should be operating optimally

PC_MOVE_TO_WIDTH

Total Count (counts): 3916.0 Your gripper should be operating optimally
→ Success (counts): 3751.0
→ Fail (counts): 165.0
→ Obstruction (counts): 0.0
Distance Travelled (mm): 77845.5 Your gripper should be operating optimally

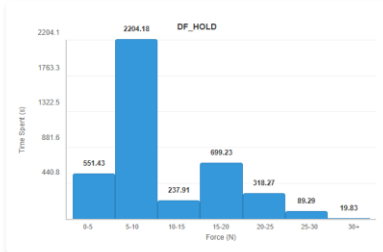
FF_GRIP

Total Count (counts): 14.0 Your gripper should be operating optimally
→ Success (counts): 14.0
→ Fail (counts): 0.0
→ Obstruction (counts): 0.0
Distance Travelled (mm): 65.3 Your gripper should be operating optimally
→ On Approach (mm): 64.2
→ On Hold (mm): 1.1



DF_GRIP

Total Count (counts): 149.0 Your gripper should be operating optimally
→ Success (counts): 147.0
→ Fail (counts): 1.0
→ Obstruction (counts): 1.0
Distance Travelled (mm): 1859.4 Your gripper should be operating optimally
→ On Approach (mm): 809.9
→ On Hold (mm): 1049.5



RELEASE

Total Count (counts): 79.0 Your gripper should be operating optimally
→ Success (counts): 79.0
→ Fail (counts): 0.0
Distance Travelled (mm): 101.1 Your gripper should be operating optimally

BRAKE

Total Count (counts): 0.0 Your gripper should be operating optimally
→ Success (counts): 0.0
→ Fail (counts): 0.0
Distance Travelled (mm): 26.5 Your gripper should be operating optimally

STOP

Total Count (counts): 0.0 Your gripper should be operating optimally

BUTTON_OPEN

Total Count (counts): 204.0 Your gripper should be operating optimally
→ Success (counts): 203.0
→ Obstruction (counts): 1.0
Distance Travelled (mm): 1209.9 Your gripper should be operating optimally

BUTTON_CLOSE

Total Count (counts): 86.0 Your gripper should be operating optimally
→ Success (counts): 83.0
→ Obstruction (counts): 3.0
Distance Travelled (mm): 551.2 Your gripper should be operating optimally

SEN0

Slip Count (counts): 6.0 This sensor pillar should be operating optimally
P 0 Max FN (N): 9.3 This sensor pillar should be operating optimally
Max FT (N): 2.6 This sensor pillar should be operating optimally
Slip Count (counts): 45.0 This sensor pillar should be operating optimally
P 1 Max FN (N): 9.8 This sensor pillar should be operating optimally
Max FT (N): 5.7 Max FT > 5.0 N may indicate that this sensor pillar has sustained damage due to overloading
Slip Count (counts): 11.0 This sensor pillar should be operating optimally
P 2 Max FN (N): 8.3 This sensor pillar should be operating optimally
Max FT (N): 4.3 This sensor pillar should be operating optimally
Slip Count (counts): 38.0 This sensor pillar should be operating optimally
P 3 Max FN (N): 11.6 Max FN > 10.0 N may indicate that this sensor pillar has sustained damage due to overloading
Max FT (N): 3.7 This sensor pillar should be operating optimally
Slip Count (counts): 17.0 This sensor pillar should be operating optimally
P 4 Max FN (N): 13.2 Max FN > 10.0 N may indicate that this sensor pillar has sustained damage due to overloading
Max FT (N): 4.3 This sensor pillar should be operating optimally
Slip Count (counts): 51.0 This sensor pillar should be operating optimally
P 5 Max FN (N): 8.2 This sensor pillar should be operating optimally
Max FT (N): 3.9 This sensor pillar should be operating optimally
Slip Count (counts): 4.0 This sensor pillar should be operating optimally
P 6 Max FN (N): 21.7 Max FN > 20 N may indicate that this sensor pillar has sustained damage due to overloading
Max FT (N): 10.0 Max FT > 10 N may indicate that this sensor pillar has sustained damage due to overloading
Slip Count (counts): 35.0 This sensor pillar should be operating optimally
P 7 Max FN (N): 8.2 This sensor pillar should be operating optimally
Max FT (N): 5.7 Max FT > 5.0 N may indicate that this sensor pillar has sustained damage due to overloading
Slip Count (counts): 5.0 This sensor pillar should be operating optimally
P 8 Max FN (N): 4.9 This sensor pillar should be operating optimally
Max FT (N): 2.9 This sensor pillar should be operating optimally

SEN1

Slip Count (counts): 0 This sensor pillar should be operating optimally
P 0 Max FN (N): 9.0 This sensor pillar should be operating optimally
Max FT (N): 6.7 Max FT > 5.0 N may indicate that this sensor pillar has sustained damage due to overloading
Slip Count (counts): 4.0 This sensor pillar should be operating optimally
P 1 Max FN (N): 8.4 This sensor pillar should be operating optimally
Max FT (N): 4.9 This sensor pillar should be operating optimally
Slip Count (counts): 14.0 This sensor pillar should be operating optimally
P 2 Max FN (N): 11.7 Max FN > 10.0 N may indicate that this sensor pillar has sustained damage due to overloading
Max FT (N): 3.1 This sensor pillar should be operating optimally
Slip Count (counts): 4.0 This sensor pillar should be operating optimally
P 3 Max FN (N): 6.8 This sensor pillar should be operating optimally
Max FT (N): 4.2 This sensor pillar should be operating optimally
Slip Count (counts): 18.0 This sensor pillar should be operating optimally
P 4 Max FN (N): 7.7 This sensor pillar should be operating optimally
Max FT (N): 3.2 This sensor pillar should be operating optimally
Slip Count (counts): 19.0 This sensor pillar should be operating optimally
P 5 Max FN (N): 9.9 This sensor pillar should be operating optimally
Max FT (N): 3.2 This sensor pillar should be operating optimally
Slip Count (counts): 49.0 This sensor pillar should be operating optimally
P 6 Max FN (N): 10.9 Max FN > 10.0 N may indicate that this sensor pillar has sustained damage due to overloading
Max FT (N): 4.1 This sensor pillar should be operating optimally
Slip Count (counts): 3.0 This sensor pillar should be operating optimally
P 7 Max FN (N): 11.6 Max FN > 10.0 N may indicate that this sensor pillar has sustained damage due to overloading
Max FT (N): 5.2 Max FT > 5.0 N may indicate that this sensor pillar has sustained damage due to overloading
Slip Count (counts): 34.0 This sensor pillar should be operating optimally
P 8 Max FN (N): 10.2 Max FN > 10.0 N may indicate that this sensor pillar has sustained damage due to overloading
Max FT (N): 4.7 This sensor pillar should be operating optimally

Figure 11.10 – Visualisation Software: Usage report rendered in a browser

11.7 Gripper control

Basic gripper control can be achieved with the buttons:

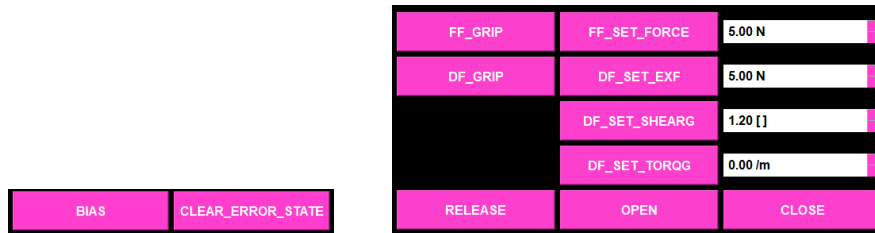


Figure 11.11 – Visualisation Software: Gripper control buttons. Left: in the button group in the bottom left; and Right: in the button group in the bottom right of the visualiser

Table 11.1 – Visualisation Software: Description of gripper control buttons

Button	Description
BIAS	Sends a <code>BIAS</code> command. Biases the PapillArray Tactile Sensors – ensure the sensors are not loaded (not in contact) before biasing.
CLEAR_ERROR_STATE	Sends a <code>CLEAR_ERROR_STATE</code> command.
FF_GRIP	Sends a <code>FF_GRIP</code> command. Uses the fixed force control parameter values currently configured on the gripper.
FF_SET_FORCE	Sends a <code>FF_SET_FORCE</code> command with the value in the adjacent input box. This button can be pressed even after the <code>FF_GRIP</code> button is pressed to adjust the grip force while already gripping.
DF_GRIP	Sends a <code>DF_GRIP</code> command. Uses the dynamic grip force control parameter values currently configured on the gripper.
DF_SET_EXF	Sends a <code>DF_SET_EXF</code> command with the value in the adjacent input box. This button should be used before the <code>DF_GRIP</code> button.
DF_SET_SHEARG	Sends a <code>DF_SET_SHEARG</code> command with the value in the adjacent input box. This button can be pressed even after the <code>DF_GRIP</code> button is pressed to adjust the shear gain while already gripping.
DF_SET_TORQG	Sends a <code>DF_SET_TORQG</code> command with the value in the adjacent input box. This button can be pressed even after the <code>DF_GRIP</code> button is pressed to adjust the torque gain while already gripping.
RELEASE	Sends a <code>RELEASE</code> command. Only relevant if the gripper is in <code>FF_APPROACH</code> , <code>FF_HOLD</code> , <code>DF_APPROACH</code> or <code>DF_HOLD</code> state.
OPEN	Sends a <code>PC_MOVE_TO_WIDTH</code> command with a width value of the current position +1 mm.
CLOSE	Sends a <code>PC_MOVE_TO_WIDTH</code> command with a width value of the current position -1 mm.

12 C++ Library

12.1 Folder structure

The C++ Library is provided on the Contactile USB flash drive that was shipped with the development kit in a folder named SOFTWARE/C++_LIBRARY. Within this folder there are two subfolders: C++LIN and C++WIN, containing the C++ Library for Linux, and the C++ Library for Windows respectively. In Table 12.1 is a brief description of the contents of these folders.

Table 12.1 – C++ Library: Folder contents of C++XXX/ Contactile_Gripper_C++_Lib_XXX where XXX refers to LIN or WIN

Sub Folder	Name	More Information
include	GripperClass.h	The header file for the GripperClass class
	GripperConstants.h	The header file of constants for the Gripper Library
	Structures.h	The header file of structure definitions for the Gripper Library
	SerialManager.h	The header file for a utility class used by GripperClass
	LogManager.h	The header file for a utility class used by Gripper Class
lib	libContactileGripper.a or ContactileGripper.lib	The library file: libContactileGripper.a for LIN ContactileGripper.lib for WIN

12.2 Library installation

Copy the entire contents of the C++LIN or C++WIN folder to a location on a PC running Linux or Windows, respectively.

The library (in the lib subfolder) must be linked in your software project. For detailed information about linking the library to your software project, consult the relevant documentation of the development environment or compiler and linker being used.

12.3 Class and function documentation

12.3.1 Constants

The GripperConstants.h file contains constant definitions that are used within GripperClass and may be used in a user program. Constants for user programming are described in Table 12.2.

Table 12.2 – Constants for user programming defined in GripperConstants.h – all are int types

Name	Value	Description
ERROR_STATE	-1	Refer to Table 10.1 for gripper state descriptions
IDLE_READY_STATE	0	
IDLE_STOPPING_STATE	1	
PC_MOVE_STATE	2	
FF_APPROACH_STATE	3	
FF_HOLD_STATE	4	
DF_APPROACH_STATE	5	
DF_HOLD_STATE	6	
BRAKING_STATE	7	
RELEASING_STATE	8	
BUTTON_PRESS_STATE	9	
ERR_NONE	0	Refer to Table 10.2 for gripper error descriptions
ERR_OUT_OF_BOUNDS	1	
ERR_PC_RINGING	2	
ERR_MOTION OBSTRUCTION	3	
ERR_UNABLE_TO_REACH_TARGET_FORCE	4	
ERR_NO MOTOR VOLTAGE	100	
ERR_OVER_TEMP	101	
IMU_DOF	6	Number of DOF in the IMU data
IMU_NDIM	3	Number of dimensions in the IMU data (X, Y and Z)
IMU_ACC_OFFSET	0	Offset of the accelerometer data in the IMU data array
IMU_GYR_OFFSET	1	Offset of the gyroscope data in the IMU data array
IMU_N	6	Number of elements in the array returned by GET_IMU
TACTILE_GLOBAL_DOF	6	Number of DOF in the global tactile sensor data
TACTILE_NDIM	3	Number of dimensions in the tactile data (X, Y and Z)
TACTILE_NSENSOR	2	Number of tactile sensors
TACTILE_NPILLAR	9	Number of pillars in each tactile sensor
TACTILE_FORCE_OFFSET	0	Offset of the global force data in the tactile sensor data array (per sensor)
TACTILE_TORQUE_OFFSET	3	Offset of the global torque data in the tactile sensor data array (per sensor)
TACTILE_N	66	Number of elements in array returned by GET_TACTILE
X_IND	0	The index of X-axis data
Y_IND	1	The index of Y-axis data
Z_IND	2	The index of Z-axis data

12.3.2 Structures

The CMD_STATUS enum is defined in Structures.h. The CMD_STATUS defines error values that can be generated by the C++ library – these are different to error states of the Gripper.

Table 12.3 – Contents of CMD_STATUS enum defined in Structures.h

Name	Value	Description
COMMAND_SUCCESS	0	Command sent and acknowledgment received successfully.
COMMAND_FAIL	-999999	
ERR_COMMAND_NAME	-1	The command name in the response does not match the command that was sent.
ERR_RESP_TYPE	-2	The response type received was not ACK or INV.
ERR_N_RET	-3	The number of arguments in the gripper response was incorrect.
ERR_SEND	-4	There was an error sending the command via the serial port.
ERR_RECEIVE	-5	There was an error reading the gripper response via the serial port.
ERR_INVALID_ARG	-6	The argument sent to the gripper was invalid.
ERR_BUTTON_INTERRUPT	-7	The gripper was interrupted by a button press.
ERR_TIMEOUT	-8	The library experienced a timeout while waiting to read the gripper response.

12.3.3 GripperClass function list

GripperClass describes an object that connects with the Gripper via a serial connection emulated on the computer's USB port, and reads and processes the data streaming through the serial connection. Commands can be sent and responses received from the Gripper through this class. The public member functions of the GripperClass class for user programming are described below.

12.3.3.1 Setup functions

GripperClass(bool *DEBUG* = false)

Description: Constructs a new GripperClass object

Parameters: [in] *DEBUG* True or false for debug terminal printout. If false the class will output nothing to the terminal.

~GripperClass()

Description: Destroys the Gripper Class object and closes the open threads.

void terminate()

Description: This function is called by the destructor, used to cleanup threads.
This function can be called when wanting to manually manage resources.

bool setComPort (std::string *comPort*)

Description: Sets and connects to the serial port. If this function has not been called the library will wait until it is as it is required to communicate with the gripper.

Parameters: [in] *comPort* The COM port name.
E.g. "COM12" for Windows, "ACM0" for Linux

Returns: True if successfully connected to the COM port, false if unsuccessful.

bool sendReceiveReady()

Description: True if the gripper is ready to send and receive messages, use this function to wait after setting the serial port. After receiving a true response the user may send commands.

Returns: True if the library is ready to read and write to gripper.
False if not ready.

12.3.3.2 Multithreading functions

std::vector<std::string> getCommandQueue()

Description: Returns the Command Queue object. Normally the command queue will be empty as when commands are executed they are removed from the command queue. If many commands are sent at once, then this will fill up.

Returns: Returns a vector which has been converted from an internal queue for compatibility with the python pybind implementation.

std::vector<responseStruct> getResponseQueue()

Description: Returns the Response Queue object, normally the Response queue will be empty as when Response is acted upon it is removed from the queue.

Returns: Returns a vector of responses. The internal queue is converted to a vector for return for compatibility with the python pybind implementation.

12.3.3.3 Gripper data streaming functions

std::vector<std::vector<double>> getDataStream()

Description: Returns data that has been not yet been requested through this function. Data is removed from the data queue when this function is called. Internal queue is converted to a vector for compatibility with pybind python.

Returns: Returns the most recent data will be at the end of the vector.

CMD_STATUS data_stream()

Description: Sends DATA_STREAM, checks acknowledgement.

Returns: CMD_STATUS int representing the result of the command.

CMD_STATUS data_stop()

Description: Sends DATA_STOP, checks acknowledgement.

User program should wait 2 s after calling to ensure the serial connection is ready. After 1.5 s library will stop looking for acknowledgement from the gripper.

Returns: CMD_STATUS int representing the result of the command.

CMD_STATUS data_stream_global()

Description: Sends DATA_STREAM_GLOBAL, checks acknowledgement.

Returns: CMD_STATUS int representing the result of the command.

CMD_STATUS data_stop_global()

Description: Sends DATA_STOP_GLOBAL, checks acknowledgement.

User program should wait 2 s after calling to ensure the serial connection is ready. After 1.5 s library will stop looking for acknowledgement from the gripper.

Returns: CMD_STATUS int representing the result of the command.

12.3.3.4 Gripper state and error functions

int get_state()

Description: Sends GET_STATE, checks acknowledgement, returns result.

Returns: See GripperConstants.h for state meaning.

int get_last_error()

Description: Sends GET_LAST_ERROR, checks acknowledgement, returns result.

Returns: See GripperConstants.h for error meaning.

CMD_STATUS clear_error_state()

Description: Sends CLEAR_ERROR_STATE, checks acknowledgement.

Returns: CMD_STATUS int representing the result of the command.

12.3.3.5 Gripper get sensor data functions

double get_width()

Description: Sends GET_WIDTH, checks acknowledgement, returns result.

Returns: Distance between the tips of the sensor arrays. COMMAND_FAIL if acknowledgment fails.

double get_vel()

Description: Sends GET_VEL, checks acknowledgement, returns result.

Returns: Velocity of gripper fingers. COMMAND_FAIL if acknowledgment fails.

std::vector<double> get_imu()

Description: Sends GET_IMU, checks acknowledgement, returns result.

Returns: Vector of IMU data. Zeros if acknowledgment fails.
See Table 9.4 for description of data.

std::vector<double> get_tactile()

Description: Sends GET_TACTILE, checks acknowledgement, returns result.

Returns: Vector of tactile sensor data. Zeros if acknowledgment fails.
See Table 9.4 for description of data.

std::vector<double> get_tactile_global()

Description: Sends GET_TACTILE_GLOBAL, checks acknowledgement, returns result.

Returns: Vector of tactile sensor global data. Zeros if acknowledgment fails.
See Table 9.4 for description of data.

12.3.3.6 Gripper get parameter functions

double pc_get_vel()

Description: Sends PC_GET_VEL, checks acknowledgement, returns result.

Returns: Velocity setting for PC_MOVE_TO_WIDTH. COMMAND_FAIL if acknowledgment fails.

double ff_get_vel()

Description: Sends FF_GET_VEL, checks acknowledgement, returns result.

Returns: Velocity setting for FF_GRIP. COMMAND_FAIL if acknowledgment fails.

double ff_get_force()

Description: Sends FF_GET_FORCE, checks acknowledgement, returns result.

Returns: Force setting for FF_GRIP. COMMAND_FAIL if acknowledgment fails.

double df_get_vel()

Description: Sends DF_GET_VEL, checks acknowledgement, returns result.

Returns: Velocity setting for DF_GRIP. COMMAND_FAIL if acknowledgment fails.

double df_get_exf()

Description: Sends DF_GET_EXF, checks acknowledgement, returns result.

Returns: Exploratory grip force setting for DF_GRIP. COMMAND_FAIL if acknowledgment fails.

double df_get_mxf()

Description: Sends DF_GET_MXF, checks acknowledgement, returns result.

Returns: Maximum grip force setting for DF_GRIP. COMMAND_FAIL if acknowledgment fails.

int df_get_dyn_exf()

Description: Sends DF_GET_DYN_EXF, checks acknowledgement, returns result.

Returns: Flag for dynamic exploration force for DF_GRIP mode, 0 is disabled, 1 is enabled. COMMAND_FAIL if acknowledgment fails.

double df_get_shearg()

Description: Sends DF_GET_SHEARG, checks acknowledgement, returns result.

Returns: Shear gain setting for DF_GRIP. COMMAND_FAIL if acknowledgment fails.

double df_get_torqg()

Description: Sends DF_GET_TORQG, checks acknowledgement, returns result.

Returns: Torque gain setting for DF_GRIP. COMMAND_FAIL if acknowledgment fails.

12.3.3.7 Gripper set parameter functions

CMD_STATUS reset_params()

Description: Sends RESET_PARAMS, checks acknowledgement.
Resets all parameters back to the default values
Returns: CMD_STATUS int representing the result of the command.

CMD_STATUS pc_set_vel(double vel)

Description: Sends PC_SET_VEL, checks acknowledgement.
Sets the target velocity for PC_MOVE_TO_WIDTH.
Parameters: [in] vel Velocity (mm/s); refer to section 9.2 for min and max values
Returns: CMD_STATUS int representing the result of the command.

CMD_STATUS ff_set_vel(double vel)

Description: Sends FF_SET_VEL, checks acknowledgement.
Sets the target velocity for FF_GRIP.
Parameters: [in] vel Velocity (mm/s); refer to section 9.2 for min and max values
Returns: CMD_STATUS int representing the result of the command.

CMD_STATUS ff_set_force(double force)

Description: Sends FF_SET_FORCE, checks acknowledgement.
Sets the target grip force for FF_GRIP.
Parameters: [in] force Force (N); refer to section 9.2 for min and max values
Returns: CMD_STATUS int representing the result of the command.

CMD_STATUS df_set_vel(double vel)

Description: Sends DF_SET_VEL, checks acknowledgement.
Sets the target velocity for DF_GRIP.
Parameters: [in] vel Velocity (mm/s); refer to section 9.2 for min and max values
Returns: CMD_STATUS int representing the result of the command.

CMD_STATUS df_set_exf(double force)

Description: Sends DF_SET_EXF, checks acknowledgement.
Sets the target exploratory force for DF_GRIP.
Parameters: [in] force Force (N); refer to section 9.2 for min and max values
Returns: CMD_STATUS int representing the result of the command.

CMD_STATUS df_set_mxf(double force)

Description: Sends DF_SET_MXF, checks acknowledgement.
Sets the maximum target force for DF_GRIP.
Parameters: [in] force Force (N); refer to section 9.2 for min and max values
Returns: CMD_STATUS int representing the result of the command.

CMD_STATUS df_dyn_exf_en()

Description: Sends DF_DYN_EXF_EN, checks acknowledgement.
Enables the dynamic exploratory force for DF_GRIP.
Returns: CMD_STATUS int representing the result of the command.

CMD_STATUS df_dyn_exf_dis()

Description: Sends DF_DYN_EXF_DIS, checks acknowledgement.
Disables the dynamic exploratory force for DF_GRIP.
Returns: CMD_STATUS int representing the result of the command.

CMD_STATUS df_set_shearg(double gain)

Description: Sends DF_SET_SHEARG, checks acknowledgement.
Sets the shear gain for DF_GRIP.

Parameters: [in] gain Gain (); refer to section 9.2 for min and max values

Returns: CMD_STATUS int representing the result of the command.

CMD_STATUS df_set_torqg(double gain)

Description: Sends DF_SET_TORQG, checks acknowledgement.
Sets the torque gain for DF_GRIP.

Parameters: [in] gain Gain (m^{-1}); refer to section 9.2 for min and max values

Returns: CMD_STATUS int representing the result of the command.

12.3.3.8 Gripper action functions

CMD_STATUS bias()

Description: Sends BIAS, checks acknowledgement.
Zeros the tactile sensor output.

Returns: CMD_STATUS int representing the result of the command.

CMD_STATUS pc_move_to_width(double width)

Description: Sends PC_MOVE_TO_WIDTH, checks acknowledgement.
Moves the gripper fingers to target width - not intended to be used to grip an object.

Parameters: [in] width Width (mm); refer to section 9.2 for min and max values

Returns: CMD_STATUS int representing the result of the command.

CMD_STATUS ff_grip()

Description: Sends FF_GRIP, checks acknowledgement.
Starts fixed force control gripping.

Returns: CMD_STATUS int representing the result of the command.

CMD_STATUS df_grip()

Description: Sends DF_GRIP, checks acknowledgement.
Starts dynamic force control gripping.

Returns: CMD_STATUS int representing the result of the command.

CMD_STATUS stop()

Description: Sends STOP, checks acknowledgement.
Stops the gripper movement.

Returns: CMD_STATUS int representing the result of the command.

CMD_STATUS brake()

Description: Sends BRAKE, checks acknowledgement.
The gripper applies the brakes to the motor.

Returns: CMD_STATUS int representing the result of the command.

CMD_STATUS release()

Description: Sends RELEASE, checks acknowledgement.
Starts fixed force gripping.

Returns: CMD_STATUS int representing the result of the command.

12.3.3.9 Program flow functions

int waitUnit_idle_ready(double timeout_s)

Description: Blocks until the state of the gripper is 0 i.e. the gripper is in idle state, unless the timeout period is exceeded.

Parameters: [in] timeout_s Max wait time (s) before leaving the function.

Returns: Error code representing why the function returned: any gripper error defined in GripperConstants.h or CMD_STATUS.

int waitUnit_hold(int holdState, double timeout_s)

Description: Blocks until the state of the gripper matches holdState.

Parameters: [in] holdState The hold state to wait for: FF_HOLD_STATE, or DF_HOLD_STATE.

[in] timeout_s Max wait time (s) before leaving the function.

Returns: Error code representing why the function returned: any gripper error defined in GripperConstants.h or CMD_STATUS.

CMD_STATUS flushSerialInputBuffer()

Description: Flushes the serial input buffer - used when the communication with gripper goes out of sync. Use if command recognition is routinely failing, in normal operation users should not need to call this function.

Returns: CMD_STATUS representing the result of the command.

12.3.3.10 Logging functions

void startLogging(const std::string& logsDir)

Description: Opens a log file with the name LOG_YYYYMMDD_hhmmss.csv, where YYYYMMDD_hhmmss is the local time. Starts logging the values from the DATA_STREAM as comma-separated values in the file. DATA_STREAM must be enabled.

Parameters: [in] logsDir The full path of the target log file. If folder does not exist, logging will fail.

void stopLogging()

Description: Stops logging the DATA_STREAM values and flushes and closes the log file.

12.3.3.11 Statistics functions

std::vector<std::string> who_am_i(void)

Description: Opens a log file with the name LOG_YYYYMMDD_hhmmss.csv, where YYYYMMDD_hhmmss is the local time. Starts logging the values from the DATA_STREAM as comma-separated values in the file. DATA_STREAM must be enabled.

Returns: Vector of gripper identification string values. Empty strings if acknowledgment fails. See Table 9.4 for description of data.

std::vector<std::string> get_statistics(void)

Description: Stops logging the DATA_STREAM values and flushes and closes the log file.

Returns: Vector of gripper usage string values. Zeros if acknowledgment fails. See Table 9.4 for description of data.

12.4 Example using the C++ library

There are several examples programs provided for using the C++ library.

For each operating system (Windows and Linux), two example programs are provided:

1. Simple (single-threaded) example
2. Multithreaded example – showing how to create a thread to use the data stream while another thread sends and receives gripper commands

Table 12.4 describes the location of the example C++ programs and each of the files included.

Table 12.4 – C++ Examples: Folder contents of C++XXX/ Contactile_Gripper_C++_Lib_XXX
where XXX refers to LIN or WIN C++

Sub Folder	Name	More Information
Simple example	main.cpp	Example C++ program, connecting to the gripper COM port and performing an FF_GRIP.
	CMakeLists.txt	Use this file to build the simple example using CMake .
	README.md	Instructions for using the simple example code.
Multithreaded example	main.cpp	Example C++ program, connecting to the gripper COM port and performing a DF_GRIP while continuously getting data using the getDataStream library function.
	CMakeLists.txt	Use this file to build the multithreaded example using CMake .
	README.md	Instructions for using the multithreaded example code.

13 Python Library

13.1 Folder structure

The Python Library is provided on the Contactile USB flash drive that was shipped with the development kit in a folder named `SOFTWARE/PYTHON_LIBRARY`. Within this folder there are three subfolders: `Pybind_Lib_LIN_Vx.x`, `Pybind_Lib_WIN_Vx.x` and `Python examples`, containing the Python Library for Linux, the Python Library for Windows, and examples for using the libraries respectively. In Table 13.1 is a brief description of the contents of these folders.

Table 13.1 – Python Library: Folder contents of `Pybind_Lib_XXX_Vx.x` where XXX refers to LIN or WIN

Name	More Information
<code>contractile_gripper_lib-x.x-cpYYY-cpYYY-XXX_ZZZ.whl</code>	The wheel file for installing the python library where x.x is the library version, YYY is the python version, XXX is linux or win, and ZZZ is the target processor
<code>README.md</code>	Instructions for using the python library

13.2 Library installation

Copy the entire contents of the `Pybind_Lib_LIN_Vx.x` or `Pybind_Lib_WIN_Vx.x` folder to a location on a PC running Linux or Windows, respectively.

The library can be installed with pip using:

```
pip install <wheel file name>
```

For more information about pip, see the [pip documentation](#).

13.3 Class and function documentation

13.3.1 Constants

The `GripperConstants.py` file contains the same constant definitions that are defined in the C++ library `GripperConstants.h` file – for more information, refer to 12.3.1 Constants.

13.3.2 Structures

The `CMD_STATUS` enum defines error values that can be generated by the Python library – these are different to error states of the Gripper. The enum is defined as in the C++ library – for more information, refer to 12.3.2 Structures.

13.3.3 GripperClass function list

The `GripperClass` defines all the same functions as in the C++ library – for more information, refer to 12.3.3 `GripperClass` function list.

13.4 Example using the Python library

There are two examples programs provided for using the Python library:

1. Simple (single-threaded) example
2. Multithreaded example – showing how to create a thread to use the data stream while another thread sends and receives gripper commands

Table 13.2 describes the location of the Python example programs and each of the files included.

Table 13.2 – Python Examples: Folder contents of PYTHON_LIBRARY/Python examples

Name	More Information
simple_example.py	Example Python program, connecting to the gripper COM port and performing a PC_MOVE_TO_WIDTH and a DF_GRIP.
multithreaded_example.py	Example Python program, connecting to the gripper COM port and performing a DF_GRIP while continuously getting data using the getDataStream library function.

14 ROS

14.1 Minimum requirements

The ROS node has been tested on a Lenovo ThinkPad X1 Carbon Gen 10 with the following specifications

- **CPU:** Intel Core i5-1235U
- **RAM:** 16 GB RAM
- **Operating System:** Ubuntu 20.04.6 LTS
- **ROS installation:** noetic

14.2 Folder structure

The ROS node is provided on the Contactile USB flash drive that was shipped with the gripper in the folder named SOFTWARE/ROS1. The ROS node is the *contactile_gripper_ros1* subfolder. The files in the *ROS/contactile_gripper_ros1* folder are summarised in Table 14.1.

Table 14.1 – Files in *SOFTWARE/ROS/contactile_gripper_ros1*

Sub Folder	File Name	File Description
.	CMakeLists.txt	Directives and instructions describing the project's source files and targets
.	Doxyfile	Configuration for Doxygen documentation
.	GRIPPER_ROS1.launch	A launch file
.	package.xml	The package manifest
.	README.md	Readme file
.	documentation.html	Doxygen documentation in html format
include/ Contactile_Gripper_ C++_Lib_LIN_Vx.x	include/GripperClass.h include/GripperConstants.h include/Structures.h lib/libContactileGripper.a	The C++ library and associated header files
src	node.cpp	The cpp file for the node
srv	GripperCommand.srv GripperSerialPort.srv	Service definitions of the <i>contactile_gripper_comms</i> service (sends a gripper command and receives a gripper response), and <i>contactile_gripper_port_set</i> service (connects to a COM Port).
html	.	Html elements for documentation.html

14.3 Package summary

This package implements a driver for the GAL2 – 2-Finger Angular Gripper.

Maintainer status: maintained
Maintainer: Heba Khamis (Contactile) <heba DOT khamis AT contractile DOT com>
Author: Charles Fawcett
License: LGPLv3

The *contactile_gripper_ros1* package provides a ROS interface for the following:

1. Connecting to the COM Port, and publishing the data stream of the Gripper
2. Sending commands and receiving responses from the Gripper

14.4 Installing the package

To use this package you should have [ROS up and running](#). To install the package copy the `contractile_gripper_ros1` folder into the `src` folder of your catkin workspace.

14.5 Starting the node

The `GRIPPER_ROS1.launch` launch file is provided to start the `contactile_gripper_ros1` node. To start the node using the launch file:

```
roslaunch contactile_gripper_ros1 GRIPPER_ROS1.launch
```

After launching the node, the `GripperSerialPort` service should be evoked to connect to the COM port. For example, if the COM Port is `ACM0`:

```
rosservice call contactile_gripper_port_set "port: 'ACM0'"
```

14.6 contactile_gripper_ros1 node

14.6.1 Topics

`/contactile_data_stream` – The messages of this topic are float32 arrays of the gripper data as described in Table 9.2.

14.6.2 Services

Table 14.2 – Services of the `contactile_gripper_ros1` node

Name	Description
<code>contactile_gripper_port_set</code>	- Connects to the COM Port - This service must be evoked after launching the node and before attempting to send a command to the gripper. - To set the COM Port and connect: <pre>rosservice call contactile_gripper_port_set "port: 'ACM0'"</pre>
<code>contactile_gripper_comms</code>	- Sends a command to the gripper, and receives the response. - For available commands, see 9.2 List of commands - For example, to send a command <code>#PC_MOVE_TO_WIDTH:100.0\$</code>: <pre>rosservice call /contactile_gripper_comms "{command_name: 'PC_MOVE_TO_WIDTH', command_argument: 100}"</pre>

14.7 Example using ROS node

In the folder named `SOFTWARE/ROS1` a package called `contactile_gripper_example_user_package` is provided that shows an example of how to use the ROS topics and services defined by the `contractile_gripper_ros1` package.

For more information, refer to the `README.md` file within the folder `SOFTWARE/ROS1/contactile_gripper_example_user_package`.

15 ROS 2

15.1 Minimum requirements

The ROS2 node has been tested on a ThinkPad X1 Carbon Gen 10 With the following specifications:

- **CPU:** Intel Core i5-1235U
- **RAM:** 16 GB RAM
- **Operating System:** Ubuntu 22.04.5 LTS
- **ROS2 installation:** jammy

15.2 Folder structure

The ROS2 node is provided on the Contactile USB flash drive that was shipped with the gripper in the folder named SOFTWARE/ROS2. The ROS node is the *contactile_gripper_ros2* subfolder. The files in the *ROS/contactile_gripper_ros2* folder are summarised in Table 15.1.

Table 15.1 – Files in *SOFTWARE/ROS/contactile_gripper_ros2*

Sub Folder	File Name	File Description
.	CMakeLists.txt	Directives and instructions describing the project's source files and targets
.	Doxyfile	Configuration for Doxygen documentation
.	package.xml	The package manifest
.	README.md	Readme file
.	documentation.html	Doxygen documentation in html format
launch	GRIPPER_ROS2.launch.py	A launch file
include/ Contactile_Gripper_ C++_Lib_LIN_Vx.x	include/GripperClass.h include/GripperConstants.h include/Structures.h lib/libContactileGripper.a	The C++ library and associated header files
src	gripper_node.cpp	The cpp file for the node
srv	GripperCommand.srv GripperSerialPort.srv	Service definitions of the <i>contactile_gripper_comms</i> service (sends a gripper command and receives a gripper response), and <i>contactile_gripper_port_set</i> service (connects to a COM Port).
html	.	Html elements for documentation.html

15.3 Package summary

This package implements a driver for the GAL2 – 2-Finger Angular Gripper.

Maintainer status: maintained
Maintainer: Heba Khamis (Contactile) <heba DOT khamis AT contractile DOT com>
Author: Charles Fawcett
License: LGPLv3

The *contactile_gripper_ros2* package provides a ROS2 interface for the following:

1. Connecting to the COM Port, and publishing the data stream of the Gripper
2. Sending commands and receiving responses from the Gripper

15.4 Installing the package

To use this package you should have [ROS2 up and running](#). To install the package copy the `contractile_gripper_ros2` folder into the `src` folder of your ROS2 workspace.

15.5 Starting the node

The `GRIPPER_ROS2.launch.py` launch file is provided to start the `contactile_gripper_ros2` node. To start the node using the launch file:

```
ros2 launch contactile_gripper_ros2 GRIPPER_ROS2.launch.py
```

After launching the node, the `GripperSerialPort` service should be evoked to connect to the COM port. For example, if the COM Port is `ACM0`:

```
ros2 service call /contactile_gripper_port_set
contactile_gripper_ros2/srv/GripperSerialPort "{port: 'ACM0'}"
```

15.6 contactile_gripper_ros2 node

15.6.1 Topics

`/contactile_data_stream` – The messages of this topic are float32 arrays of the gripper data as described in Table 9.2.

15.6.2 Services

Table 15.2 – Services of the contactile gripper ros2 node

Name	Description
<code>contactile_gripper_port_set</code>	- Connects to the COM Port - This service must be evoked after launching the node and before attempting to send a command to the gripper. - To set the COM Port and connect: <pre>ros2 service call /contactile_gripper_port_set contactile_gripper_ros2/srv/GripperSerialPort "{port: 'ACM0'}"</pre>
<code>contactile_gripper_comms</code>	- Sends a command to the gripper, and receives the response. - For available commands, see 9.2 List of commands - For example, to send a command <code>#PC_MOVE_TO_WIDTH:100.0\$</code>: <pre>ros2 service call /contactile_gripper_comms contactile_gripper_ros2/srv/GripperCommand "{command_name: 'PC_MOVE_TO_WIDTH', command_argument: 100}"</pre>

15.7 Example using ROS2 node

In the folder named `SOFTWARE/ROS2` a package called `contactile_gripper_example_user_package` is provided that shows an example of how to use the ROS2 topics and services defined by the `contractile_gripper_ros2` package.

For more information, refer to the `README.md` file within the folder `SOFTWARE/ROS2/contactile_gripper_example_user_package`.