

HIGH PRECISION GEARED
SERVO MOTORS
RDrive 60

USER MANUAL



INTRODUCTION

Rozum Robotics has designed its RDrive servomotors to enable precision motion control in industrial and commercial applications.

This manual is intended for technicians and engineers who design, build, and operate systems and machinery that use RDrive servomotors for actuation. In this document, you will find the following information:

- components of RDrive servomotors
- specifications of the RDrive 60 servomotor, as well as its allowable radial and axial loads
- requirements and instructions on mechanical and electrical integration of the RDrive 60 servomotor
- instructions and recommendations on enabling motion control of RDrive servos
- maintenance, transportation, and storage recommendations

WARNING SIGNS AND THEIR MEANINGS

Below, you can see the warning symbols used throughout the manual and their meaning.



The sign denotes important information that is not directly related to safety, but that the user should be aware of.



The sign indicates important safety precautions the user should follow.

TABLE OF CONTENTS

INTRODUCTION	2
WARNING SIGNS AND THEIR MEANINGS	2
1 PRODUCT OVERVIEW	5
1.1 Components.....	5
1.2 Supply package.....	5
1.2.1 Options.....	6
1.2.2 USB-CAN dongle.....	7
1.3 Intended use and operating conditions.....	8
2 WORKING PRINCIPLE	8
2.1 Motion control system.....	8
2.2 Motion and other feedback.....	8
3 PRODUCT SPECIFICATIONS	9
3.1 Specifications of the RDrive 60 servomotor.....	9
3.2 Allowable axial and radial loads.....	10
4 INTEGRATION INTO AN APPLICATION	10
4.1 Installation requirements.....	10
4.2 Installation procedure.....	11
4.2.1 Preparing for servo installation.....	11
4.2.2 Mechanical integration.....	11
4.2.3 CAN connection.....	12
4.2.3.1 General requirements.....	12
4.2.3.2 Connection diagrams.....	12
Connection for control based on CANOpen.....	12
Connection for control via API.....	13
4.2.4 Connecting to a power supply.....	13
4.2.4.1 Electrical connection requirements.....	13
4.2.4.2 Connection diagrams.....	14
4.2.5 Pre-commissioning checks.....	16
5 ENABLING MOTION CONTROL	16
5.1 Enabling CANOpen-based communication.....	16
5.2 Enabling control via the API.....	16
5.2.1 Hardware prerequisites.....	16
5.2.2 Software prerequisites.....	17

5.2.3	Enabling access to the User API	17
	Python, Windows OS	18
	Python, Linux OS	19
	C, Windows (Cygwin) OS	21
	C, Linux OS	29
5.2.4	Setting new CAN IDs when connecting multiple servos	33
6	MAINTENANCE	34
7	TROUBLESHOOTING	34
8	TRANSPORTATION AND STORAGE	35
ANNEX I.	UPDATING THE FIRMWARE OF RDRIVE SERVOS	35
	Linux OS	35
	Windows (Cygwin) OS	36

1 PRODUCT OVERVIEW

1.1 Components

RDrive servomotors are intended for commercial and industrial use to ensure high-precision rotary motion.

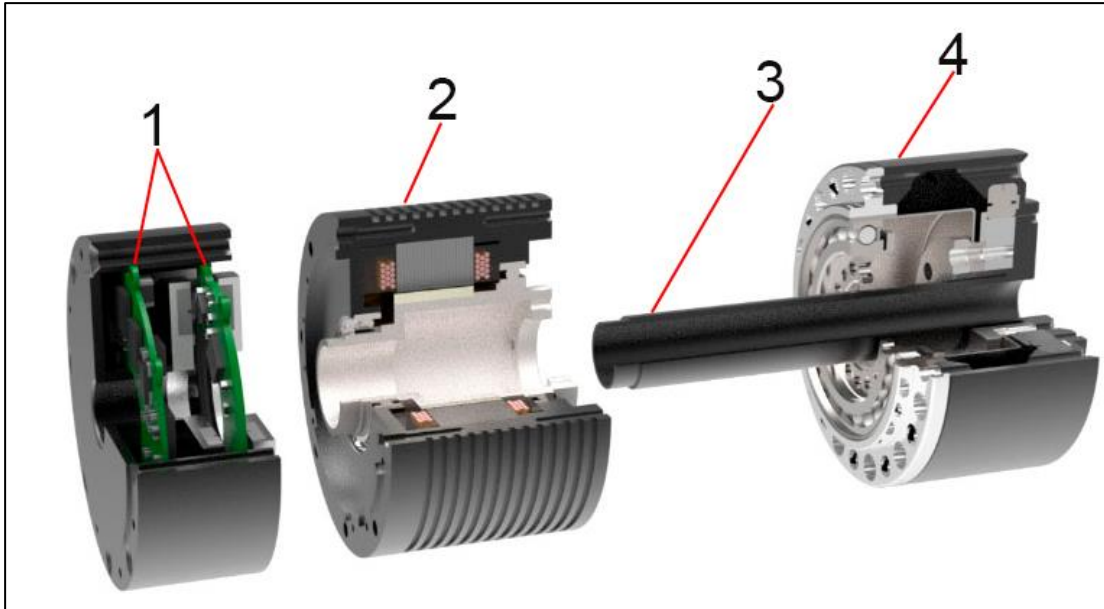


Figure 1-1: The components of an RDrive servomotor

(1)	Two printed circuit boards (PCBs) communicating with each other by means of the BiSS interface. One of the PCBs incorporates a controller and the other—two encoders.
(2)	A frameless brushless alternating current (AC) motor comprising a rotor and a stator, as well as a negative temperature coefficient (NTC) thermistor fitted into the stator winding.
(3)	A hollow shaft, which you can use for laying the cables to connect the servo to a machine.
(4)	A strain-wave gearhead that reduces rotation speed (RPM) and increases motor torque.

By design, the RDrive servomotor also comprises a **cable gland** located on its input flange. The gland includes four wires:

- two wires for connecting the servo to a power supply
- two wires for enabling CAN communication

1.2 Supply package

The basic supply package for an RDrive servomotor comprises the following:

- one or more RDrive servos
- one CAN-USB dongle (see **Section 1.2.2**)



Each supply package includes one USB-CAN dongle only, irrespective of the total quantity of servos in it. However, users can order more dongles for an extra charge.

1.2.1 Options

Apart from the basic supply package, Rozum Robotics offers the following options:

- Servobox
- Quick-start motor mounts

A **servobox** is a set of components intended to ensure safe and correct operation of RDrive servomotors at design loads. The set is customizable and can include the following (see **Figure 1-2**):

- an energy eater complete with a power supply cable
- a capacitor module comprising one or more capacitors (attached to the input flange)
- a USB-CAN adapter
- a USB-A Micro USB cable (1 m long)
- a terminating resistor 120 Ohm
- a quick-start cable set comprising a power supply cable 1 m long and a CAN cable 1 m long (one cable of each type per servomotor)



For detailed information about the servobox solution, see the [Servobox manual](#). The document also contains instructions how to assemble a similar solution on your own.

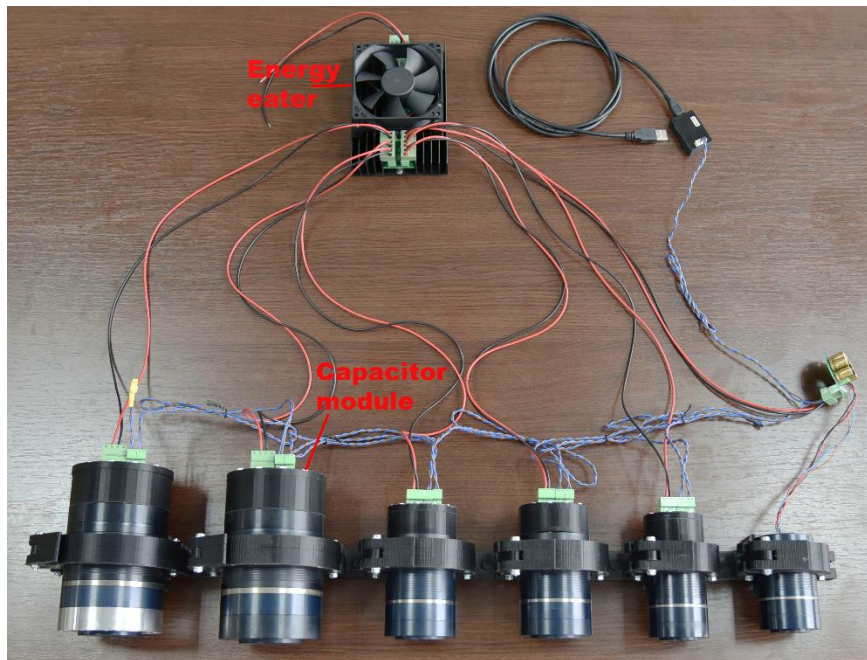


Figure 1-2: A sample servobox with servomotors and motor mounts



A **motor mount** is an optional quick-start accessory, available upon request.

1.2.2 USB-CAN dongle

A USB-CAN dongle is a special adapter coupled to a USB port of a personal computer (PC) to provide CAN connectivity between an RDrive servo and a PC.



If your PC runs Windows 8 or earlier versions of the operating system, you will need to download and install a driver to be able to work with the CAN-USB dongle. For the downloading link and installation instructions, go to the webpage:

<https://www.st.com/en/development-tools/stsw-stm32102.html>

The dongle has two connectors (**Figure 1-3**):

- (1) for connecting the device to a CAN bus (Molex 0022035045 Connector)
- (2) for connecting a Micro USB-USB A cable

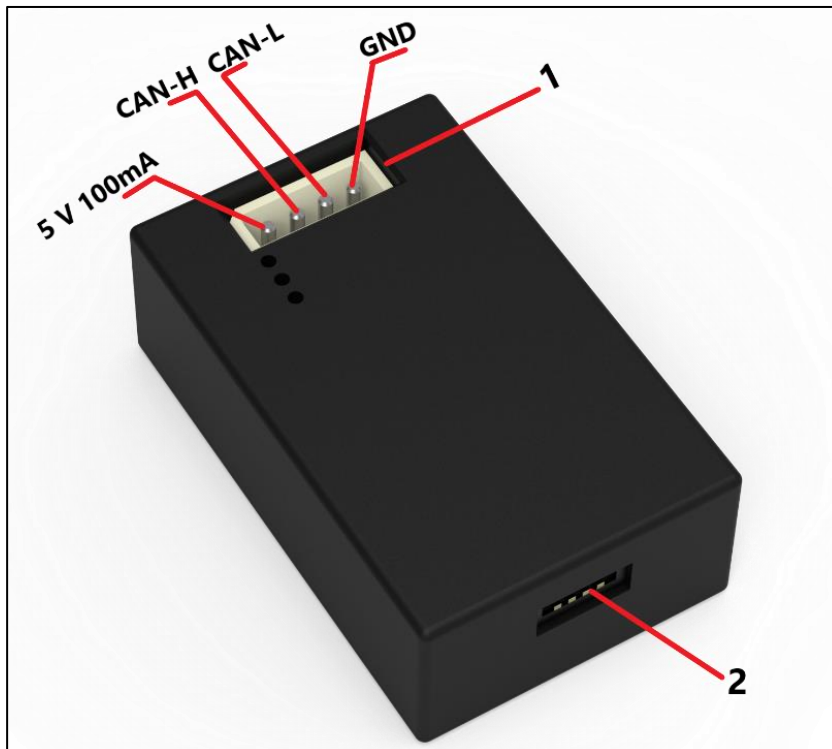


Figure 1-3: The CAN-USB dongle in the RDrive supply package

The CAN-USB dongle is supplied complete with the following:

- a Micro USB-USB A cable
- a Molex 0050375043 connector housing and Molex 08-70-1039 pins for providing CAN bus connection

1.3 Intended use and operating conditions

RDrive motors are designed for industrial and commercial use. **Inadmissible applications** include explosive or otherwise hazardous areas, as well as locations with highly corrosive atmospheres.

Table 1-1: Operating conditions of RDrive servos

Parameter	Value
Altitude	Not higher than 1,000 m above the sea
Operating temperature	0°C to +35°C
Operating humidity	80% max at 25°C (90% at 20°C)
IP protection	IP20



Avoid exposing servomotors to any operating conditions outside of the above specifications. This can damage their components and/or reduce their service life.

2 WORKING PRINCIPLE

2.1 Motion control system

The RDrive motion control system is of the closed-loop type. RDrive servomotors not only receive motion control commands from a control device (e.g., a PC or a CAN master), but also provide feedback on their execution.

The system consists of a self-designed controller and two encoders (see **Section 2.2**). During servo operation, the components of the system interact as described below:

- The controller receives a control command from a control device and applies current to a frameless AC motor to produce motion with required parameters.
- Two encoders monitor the absolute positions of the rotor and output shaft and send feedback to the controller.
- The controller processes the feedback, compares the resulting values with the control command, and sends a signal to adjust the motor position, if needed.
- The controller transmits the command execution results to the control device. Communication between the servomotor and the control device is based either on the CANOpen protocol stack (see **Section 5.1**) or on the Application Programming Interface (API) (see **Section 5.2**).

2.2 Motion and other feedback

RDrive servomotors comprise two integrated feedback devices—absolute magnetic encoders. The two encoders are mounted on a separate PCB connected with the controller PCB by means of a flat cable (a BiSS line). One of the encoders delivers information about the absolute position of the output shaft and the other—about that of the rotor.

Table 3-1: Encoder data

Type	rotary magnetic single-turn
Output	absolute position
Resolution	19-bit

In addition, the controller receives and processes feedback signals from the NTC thermistor in the motor winding. When the value based on the thermistor reading exceeds the maximum temperature limit, the controller cuts the servo off.

3 PRODUCT SPECIFICATIONS

3.1 Specifications of the RDrive 60 servomotor

Table 3-1 contains performance, electrical, and mechanical specifications, as well as dimensions of the RDrive 60 servomotor.

Table 3-1: Specifications of the RDrive 60 servomotor

PERFORMANCE DATA		
Rated rotation speed	55	RPM
RMS current	3.1	A
Rated torque	39	N·m
Peak torque	54	N·m
Torque constant (Kt) at 20°C	86	mN·m/A
Service life	35,000	hours
ELECTRICAL DATA		
Rated power	225	W
Supply voltage	48	V
MECHANICAL DATA		
Motor inertia	0.05	kg·cm ²
Weight	1.28	kg
IP rating	IP21*	
DIMENSIONS		
Length (L):	101.2	mm
Diameter (D):	63	mm
Hollow shaft diameter (d):	9	mm

*Upon request, RDrive 60 servos can be supplied with a higher IP rating.

In **Table 3-2**, you will find basic specifications of the gearhead integrated into the RDrive servo.

Table 3-2: The RDrive gearhead data

GEARHEAD DATA		
Gearhead type	Strain-wave	
Gearhead ratio	1:100	
Gearhead backlash	0.3	arcmin

3.2 Allowable axial and radial loads

The gearhead in the RDrive servomotor incorporates a high-rigidity crossed roller bearing to support output loads. The bearing can withstand high axial and radial forces, as well as high tilting moments. It keeps the gearhead protected from external loads, which guarantees long life and consistent performance of the gearhead.

Table 3-3 lists allowable axial and radial loads for RDrive 60 servomotors.

Table 3-3: RDrive 50 allowable axial and radial loads

Allowable axial load, F_a [N] ¹⁾²⁾	780
Allowable radial load, F_r [N] ¹⁾²⁾	520

1) These data are valid for $n = 15$ RPM and $L_{10} = 25,000$ h, where L_{10} is the operating life of the output bearing.

2) These data are valid, only provided the following conditions are met:

For:

- **Fa**: $M = 0$; **Fr** = 0;

- **Fr**: $M = 0$; **Fa** = 0,

where **M** is the tilting moment.

4 INTEGRATION INTO AN APPLICATION

4.1 Installation requirements



Magnetic-sensitive objects, such as banking cards, pacemakers, or other magnetic information carriers, should be kept away at a distance of 1 m from the motor.

RDrive servomotors are intended for installation as part of a motion system or a machine. You can mount the actuators in any required position — vertical, horizontal, or at an angle. The installation site should meet the following requirements:

- Well-ventilated and free from dust, moisture, and vibration
- Ambient temperature, altitude, and other environmental conditions as specified in **Table 1-1**
- Easy access for inspection and dismantling

4.2 Installation procedure

The procedure for installing an RDrive servomotor includes the following steps:

1. Preparing for installation (**Section 4.2.1**Ошибка! Источник ссылки не найден.)
2. Mechanical integration (**Section 4.2.2**)
3. Connecting to a CAN bus (**Section 4.2.3**)
4. Electrical connection (**Section 4.2.4**)
5. Pre-commissioning checks (**Section 4.2.5**)

4.2.1 Preparing for servo installation

The preparation sequence is as follows:

1. Unpack the servomotor.



Never lift or pull servomotors by cables!



Make sure to avoid shocks as this can damage high-precision encoders inside servos.

2. Check the fitting surfaces of both the servo and the machine for visible damages.



Using damaged servos is forbidden because it can result in unintended operation of the machine and endanger the operator.

3. Clean the fitting surfaces with a lint-free cloth and a suitable cleaning agent, if needed, and degrease them.

4.2.2 Mechanical integration



For mating dimensions and surfaces, see the confirmation drawings available for downloading at <https://rozum.com/documentation/servomotors/rdrive-60/drawing/>.

For mechanical integration, follow the instructions below:

1. Mount the servo into the equipment. To do this, insert twelve M3 screws into the holes on the input flange and tighten them to the torque of 2.6 N·m max.



Allow for sufficient clearance around the servo for proper heat dissipation.

2. Screw a load down to the output flange using fourteen M2.5 holes and applying the torque of 1.4 N·m max.



Make sure not to apply excessive impact or force to the output flange.

4.2.3 CAN connection

4.2.3.1 General requirements

To provide a CAN connection, use the brown and blue wires on the input flange of the RDrive servomotor. The brown is for CAN_{HIGH}, and the blue one—for CAN_{LOW}.

Two connection types are possible, depending on the preferred servo control method:

- In case servo control is based on the CANOpen protocol stack, provide a CAN bus connection of the configuration as shown in **Figure 4-1**.
- In case servo control is via API, provide a CAN-to-PC connection as shown in **Figure 4-2**.

Irrespective of the connection type, make sure to comply with the following requirements:

- Terminate CAN bus lines of less than 40 m long with 120 Ohm resistors at both ends. For bus lines of over 40 m long, use 150-300 Ohm resistors.



In a CAN-to-PC connection, you have to provide only one resistor because one is already integrated into the USB-CAN dongle included in the RDrive supply package.

- The bus line cable must be a twisted pair cable with the lay length of 2 to 4 cm.
- For the cross section of the bus line, see **Table 4-1**.
- To ensure the baud rate required for your application, L Σ should meet the specific values as indicated in **Table 4-1**.

Table 4-1: CAN line parameters

Baud Rate	50 kbit/s	100 kbit/s	250 kbit/s	500 kbit/s	1 Mbit/s
Total line length, L Σ , m	< 1000	< 500	< 200	< 100	< 40
Cross-section, mm ²	0.75 to 0.8	0.5 to 0.6	0.34 to 0.6	0.34 to 0.6	0.25 to 0.34

4.2.3.2 Connection diagrams

Connection for control based on CANOpen

The connection is in accordance with the diagram as shown in **Figure 4-1**. Connect the CAN cables on the RDrive input flange to the corresponding twisted pair wires: the brown one to CAN_{HIGH}, and the blue one — to CAN_{LOW}.

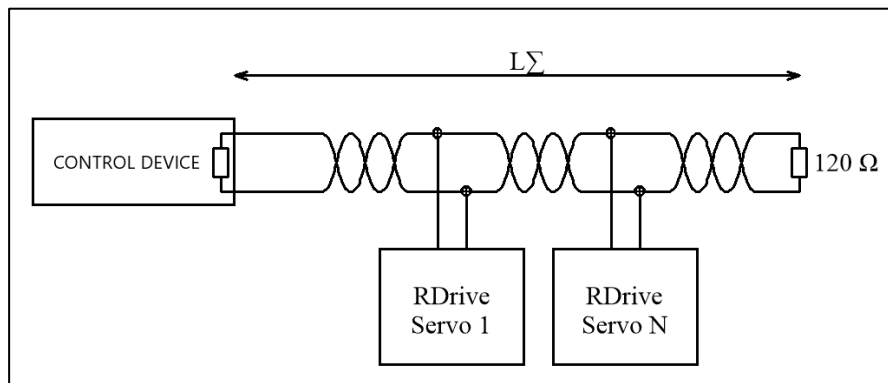


Figure 4-1: CAN connection to control servos based on CANOpen

Connection for control via API

The connection is in accordance with the diagram as shown in **Figure 4-2**, taking into consideration the arrangement of connectors on the USB-CAN dongle (see **Figure 4-3**). To provide the connection, use the cable on the input flange of the servomotor: the brown one corresponds to CAN_{HIGH}, and the blue one — to CAN_{LOW}.

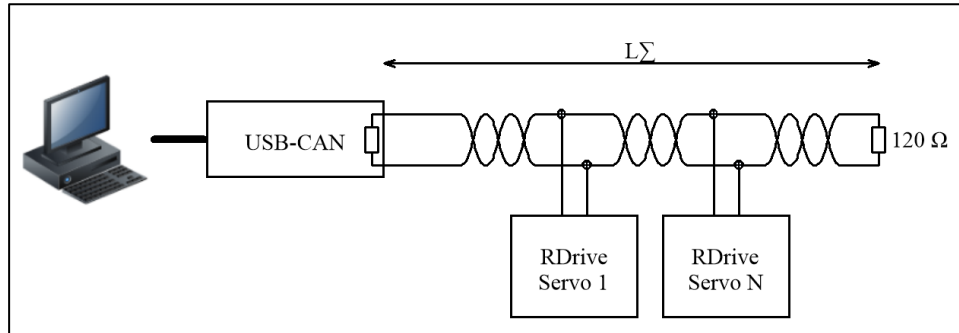


Figure 4-2: CAN-to-PC connection to control servos via API

Before connecting, make sure to solder the CAN cable leads into the Molex 0050375043 housing (included in the USB-CAN dongle supply package), using the Molex 08-70-1039 pins (also included in the supply package).

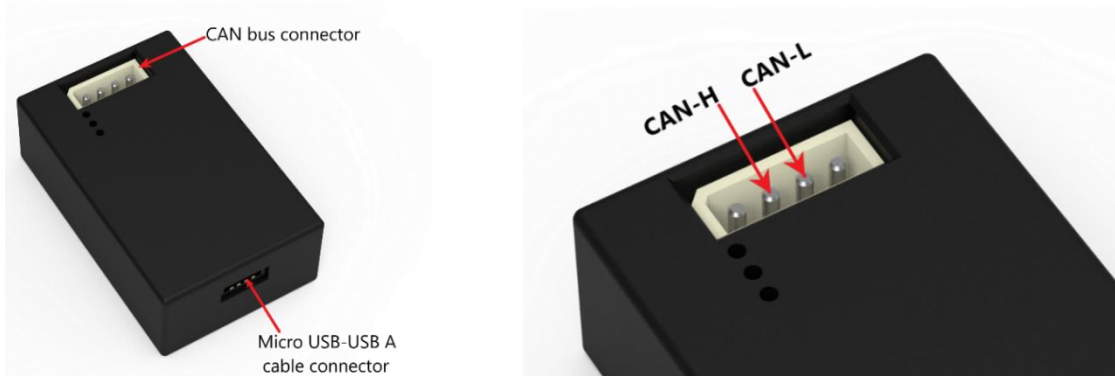


Figure 4-3: Connectors on the USB-CAN dongle

4.2.4 Connecting to a power supply



Before starting any wiring works, make sure that no power is supplied to the circuit you are assembling.

4.2.4.1 Electrical connection requirements

To connect RDrive servomotors to a power supply unit, use the red and black wires on its input flange:

- red for +
- black for –



For cable arrangement and earthing requirements, refer to the installation drawing available at: <https://rozum.com/documentation/servomotors/rdrive-60/drawing/>

However, connecting a servo directly to a power supply is **ONLY POSSIBLE FOR UNLOADED OPERATION** (such as for a test run).

To ensure correct and safe operation of RDrive servomotors CONNECTED TO A LOAD, a power supply circuit for an RDrive servo **must** include:

- at least one **energy eater** to dissipate dynamic braking energy that can cause servos to generate voltages in excess of the power supply voltage
- one or more **capacitor** to accumulate electric energy and supply to the servo, compensating for short-term consumption peaks due to inductive resistance



Total capacitance requirement for a power supply circuit is $\geq 5 \mu\text{F}$ per 1 W of connected servo power

To meet the safe and correct operation requirement, you have **two ways** to proceed:

- to [order a servobox](#) offered by Rozum Robotics as an option (see **Section 1.2.1**), customized to your specific supply circuit requirements

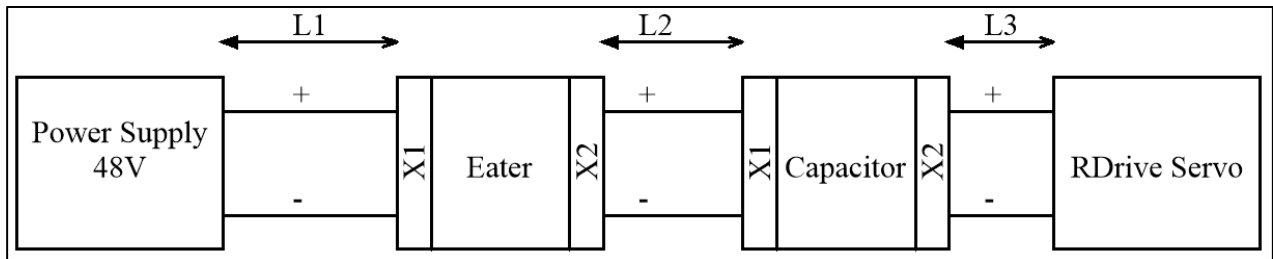
or

- to provide an energy eater(s) and a capacitor(s) on your own, following the instructions and requirements in [the Servobox manual](#).

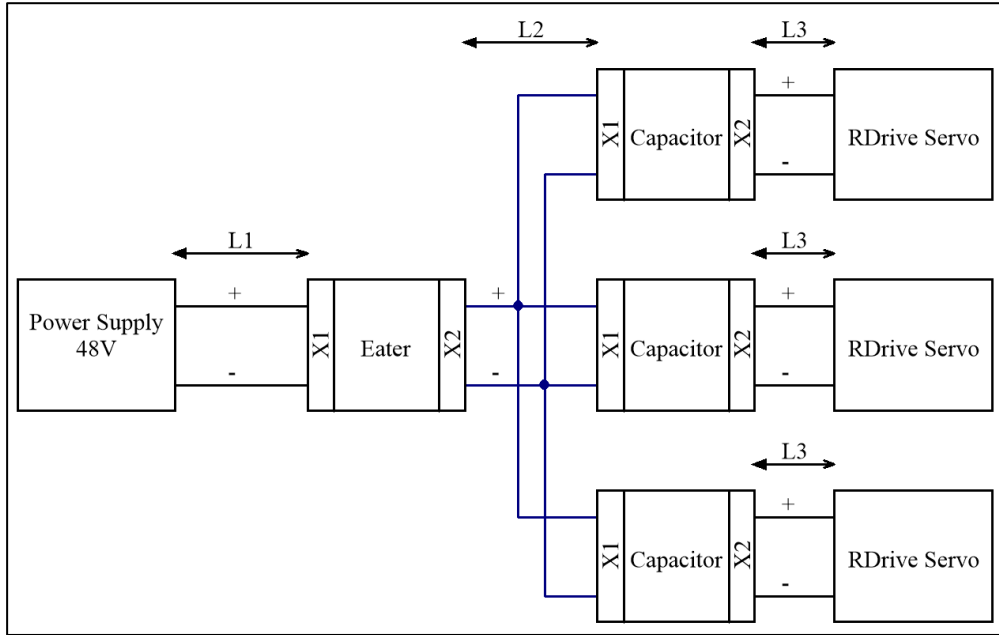
4.2.4.2 Connection diagrams

The connection diagram with an eater and a capacitor included must have either of the following configurations:

- to connect **a single servo**



- to connect **multiple servos**



Legend:

X1	Input connector (power source)
X2	Output connector (from the power consumer to the servo)
L1	Wiring segment from the power supply to the eater
L2	Wiring segment from the eater to the capacitor
L3	Wiring segment from the capacitor to any servo

Whichever of the configurations you are using, make sure to meet the following electrical connection requirements:

- The total circuit length from the power supply unit to any servomotor must not exceed 10 meters.
- The L1 length must not be longer than 10 meters.
 - When the total connected motor power is **less than 250 W**, the cable cross-section within the segment must be at least 1.00 mm².
 - When the total connected motor power is **less than 500 W**, the cable cross-section within the segment must be at least 2.00 mm².
- The L2 length must not exceed the values from **Table 4-2**.
- The L3 length must not exceed the values from **Table 4-2**

Table 4-2: Lengths vs. cross-sections of wiring segments for RD 50

L2						L3								
0.75 mm ²	1.0 mm ²	1.5 mm ²	2.5 mm ²	4.0 mm ²	6.0 mm ²	0.25 mm ²	0.5 mm ²	0.75 mm ²	1.0 mm ²	1.5 mm ²	2.5 mm ²	4.0 mm ²	6.0 mm ²	
2 m	3 m	5 m	9 m	10 m	10 m	-	0.1 m	0.1 m	0.1 m	0.2 m	0.4 m	1.0 m	1.0 m	

4.2.5 Pre-commissioning checks

Before commissioning a servo, it is advisable to check whether the following is up to requirements:

- Operating conditions (refer to **Table 1-1** and **Section 1.3**)
- Mechanical integration (refer to **Section 4.2.2**)
- Electrical integration (see **Section 4.2.4**):
 - protective earthing
 - tight connection and integrity of power supply cables
 - at least one energy eater and one capacitor integrated into the supply circuit of the servo (see **Section 1.2.1** and the [Servobox Manual](#))
- CAN connection, tight connection and integrity of communication cables (see **Section 4.2.3**)

As soon as the connections are provided and the checks are completed as appropriate, you can supply power to servomotors and enable motion control.



Before you proceed to enable motion control and operate a servo connected to a load, it is advisable to perform its test run in accordance with instructions in Section 5. For a testing circuit, you do not have to use capacitors or eaters.

5 ENABLING MOTION CONTROL

You can implement motion control of RDrive servomotors in either of the two ways:

- based on CANOpen communication
- via the Application Programming Interface (API)

5.1 Enabling CANOpen-based communication

CANOpen communication implemented for RDrive servos relies on the Controller Area Network (CAN) for its physical infrastructure. CAN is a two-wire bus line that transmits differential signals—CAN_{HIGH} and CAN_{LOW}.

On a higher level, CANOpen communication for RDrive servos is implemented as a stack of CANOpen protocols. The protocols monitor the network states and transmit and/or read CAN data frames containing various types of data (e.g., commands, parameters, servo telemetry) in the binary format.

To enable control based on CANOpen, connect RDrive servomotors in accordance with **Figure 4-1**. For a detailed description of the CANOpen communication interface implemented for RDrive servos and related application cases (including servo initialization), refer to [“CANOpen Communication Guide.”](#)

5.2 Enabling control via the API

5.2.1 Hardware prerequisites

For enabling motion control of RDrive servos via API, you need the following hardware:

- an electrical connection in accordance with **Section 4.2.4**
- a CAN-PC connection in accordance with **Section 4.2.3.2**



*Since RDrive servos are supplied with default CAN IDs from 32 to 37, there's a risk of collision when you connect more than one motor to the same CAN bus. Accordingly, before you proceed to enable control via API, make sure to change the default CAN IDs of connected servos as described in **Section 5.2.4**.*

5.2.2 Software prerequisites

Software prerequisites depend on the preferred coding technology and operating system:

		C			
			Windows OS		
Linux OS					
<ul style="list-style-type: none"> • gcc MinGW compiler • libpthread library • make package 			<ul style="list-style-type: none"> • Cygwin • gcc MinGW compiler • libpthread library • make package 		
					<p>Additionally, for Windows 8 and earlier versions, you need a driver to be able to work with the USB-CAN dongle (download here).</p>
		Python			
Linux OS			Windows OS		
<ul style="list-style-type: none"> • Python (3.5 or a later version) 			<ul style="list-style-type: none"> • Python (3.5 or a later version) 		
<p>To install Python, run one of the following commands in the console:</p>			<p>To install Python, navigate to Python, download, and complete the setup process.</p>		
<ul style="list-style-type: none"> ○ Ubuntu/Debian: sudo apt install python3 python3-pip ○ Fedora version 22 and higher: sudo dnf install python3 python3-pip ○ RedHat and Fedora before version 22: sudo yum install python3 python3-pip 			<p>Important! At the installation setup screen, make sure to check Add Python v.X to PATH (where v.X. is the downloaded Python version).</p>		
<p>In case you have a Linux distribution other than specified below, proceed in the usual order of running the installation procedure in your system.</p>			<p>Additionally, for Windows 8 and earlier versions, you need a driver to be able to work with the USB-CAN dongle (download here).</p>		
<ul style="list-style-type: none"> • pip 3 package (installed together with Python as described above) • gcc MinGW compiler • make package 					

5.2.3 Enabling access to the User API

Enabling access to servomotors via API is in accordance with the instructions as detailed below, depending on the programming technology (C, Python) and the operating system in use (Linux, Windows).



The instructions below are to set one servo and one USB-CAN adapter.

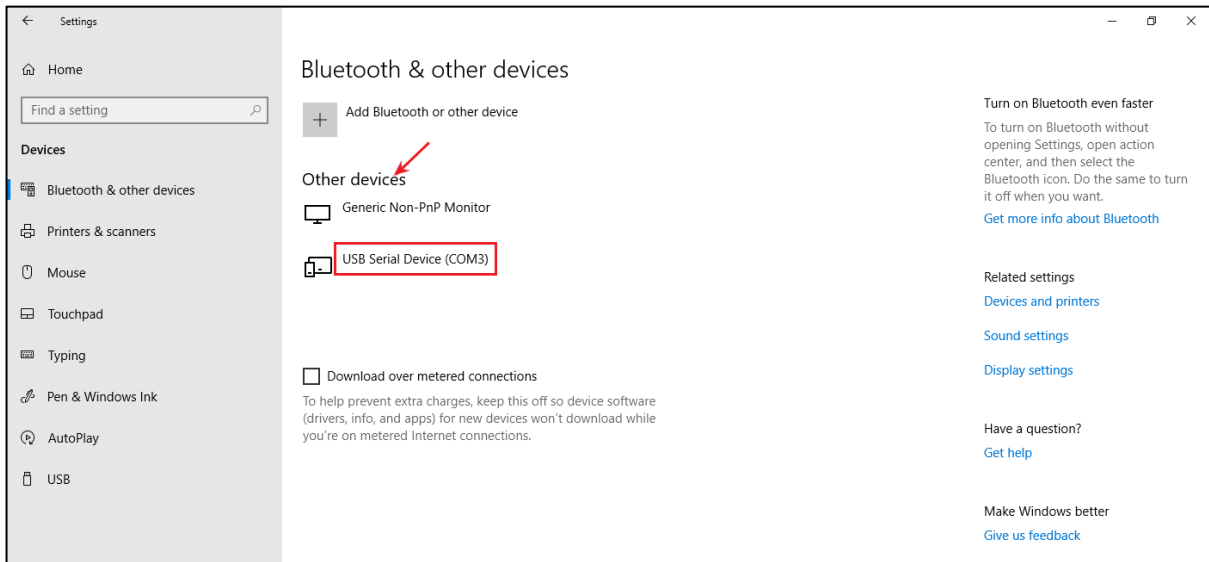
Python, Windows OS

1. Install the [Servo API library](#) by running one of the following commands.
 - To install the latest version, use:


```
pip install rdrive -i https://pip.rozum.com/simple
```
 - To install a specific version of your choice, use:


```
pip install rdrive==v1.v2.v3 -i https://pip.rozum.com/simple
```

 where **v1**, **v2**, and **v3** (e.g., `rdrive==1.0.31`) are version numbers.
2. Open the Command Line Interface (CLI) via the **Start menu**.
3. Find out the CAN interface name—the ID of the CAN-USB dongle in the user environment:
 - a. On your computer, open **Start menu** → **System Settings** → **Devices** → **Bluetooth & other devices**.
 - b. On the **Other devices** list, find a serial USB device with a COM port. The COM port number (e.g., COM 3) is the CAN interface ID you need.



4. Find out the CAN ID—the identifier of the connected servo:
 - a. In the command console, run a tutorial from the examples folder by executing the following command with a specified CAN Interface ID and CAN Servo ID.


```
python ..\userapi\python\examples\read_servo_max_velocity.py --interface COM3 --servo_1_id 32
```

 where `read_servo_max_velocity.py` is the tutorial name;
`interface COM3` is the parameter specifying the CAN Interface ID;
`servo_1_id 37` is the parameter specifying an arbitrary CAN ID for the connected servo.

- b. In the command output, go to the INFO lines (see the example below). Look for IDs within the range between 32 to 37 — default servo IDs (37 in the example below).

Example:

```
INFO: ID: 50 Device is in operational mode
INFO: ID: 37 Device is in pre-operational mode
```

5. Now, you can run any of the tutorials from the examples folder to move your RDrive servo or read parameters from it.

To do this, run the following command in the cmd console:

```
python ..\userapi\python\examples\read_servo_max_velocity.py --
interface COM3 --servo_1_id 37
```

where `read_servo_max_velocity.py` is the tutorial name (replace the name from the current example with any other tutorial name as needed);

`interface COM3` is the parameter specifying the CAN Interface ID (the one we got at Step 3);

`servo_1_id 37` is the parameter specifying an arbitrary CAN ID for the connected servo (the one we got at Step 4).

If the command is successfully executed, the connected servo will behave as commanded — return parameters or move.

Note: Some of the tutorials may require specifying some other parameters in addition to the CAN Interface ID and a CAN Servo ID.

For details, refer to the description of available functions and tutorials in the [GitHub repository](#).

Python, Linux OS

1. Install the Python API library by running one of the following commands.

- To install the latest version, use:

```
pip install rdrive -i https://pip.rozum.com/simple
```

- To install a specific version of your choice, use:

```
pip install rdrive==v1.v2.v3 -i https://pip.rozum.com/simple
```

where **v1**, **v2**, and **v3** (e.g., `rdrive==1.0.31`) are version numbers.

2. Find out the **CAN** interface name (the ID of the CAN-USB dongle in the user environment). To do this, open the console and type in the following command:

```
ls /dev/serial/by-id/.
```

In the output, look for something like: `usb-Rozum_Robotics_USB-CAN_Interface_301-if00`. It is the CAN interface name you need.



On MacOS, you can find out the CAN Interface name by executing the `ls /dev/ | grep cu.usb` command in the console. The output will contain something like: `/dev/cu.usbmodem301`.

3. Find out the **CAN ID of the connected servomotor**. To do this, complete the following steps:

- In the console, run a tutorial from the `examples` folder by executing the following command with a specified CAN Interface ID and a CAN Servo ID.

```
python3 ..\userapi\python\examples\read_servo_max_velocity.py --  
interface /dev/serial/by-id/usb-Rozum_Robotics_USB-  
CAN_Interface_301-if00 --servo_1_id 32
```

where `read_servo_max_velocity.py` is the tutorial name (replace the name from the current example with any other tutorial name as needed);

`interface /dev/serial/by-id/usb-Rozum_Robotics_USB-CAN_Interface_301-if00` is the parameter specifying the CAN Interface ID (the one we got at Step 1);

`servo_1_id 37` is the parameter specifying an arbitrary CAN ID for the connected servo (the one we got at Step 2).

- In the command output, go to the `INFO` lines (see the example below). Look for IDs within the range between 32 to 37 — default servo IDs (37 in the example below).

Example:

```
INFO: ID: 50 Device is in operational mode  
INFO: ID: 37 Device is in pre-operational mode
```

4. Now, you can run any of the tutorials from the `examples` folder to move your RDrive servo or read parameters from it.

To do this, run the following command in the console:

```
python3 ..\userapi\python\examples\read_servo_max_velocity.py --  
interface /dev/serial/by-id/usb-Rozum_Robotics_USB-CAN_Interface_301-  
if00 --servo_1_id 37
```

where `read_servo_max_velocity.py` is the tutorial name (replace the name from the current example with any other tutorial name as needed);

`interface /dev/serial/by-id/usb-Rozum_Robotics_USB-CAN_Interface_301-if00` is the parameter specifying the CAN Interface ID (the one we got at Step 2);

`servo_1_id 37` is the parameter specifying an arbitrary CAN ID for the connected servo (the one we got at Step 3).

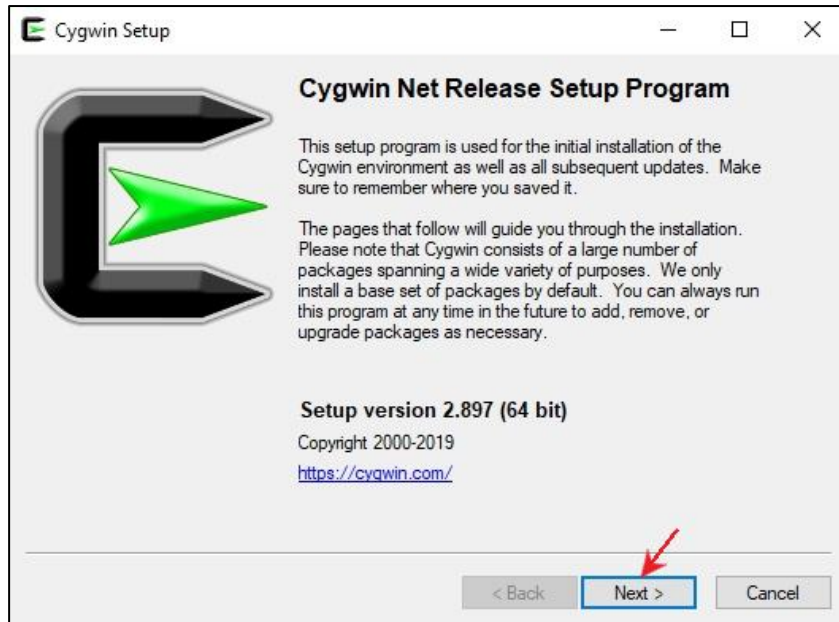
If the command is successfully executed, the connected servo will behave as commanded—return parameters or move.

Note: Some of the tutorials may require specifying some other parameters in addition to the CAN Interface ID and a CAN Servo ID.

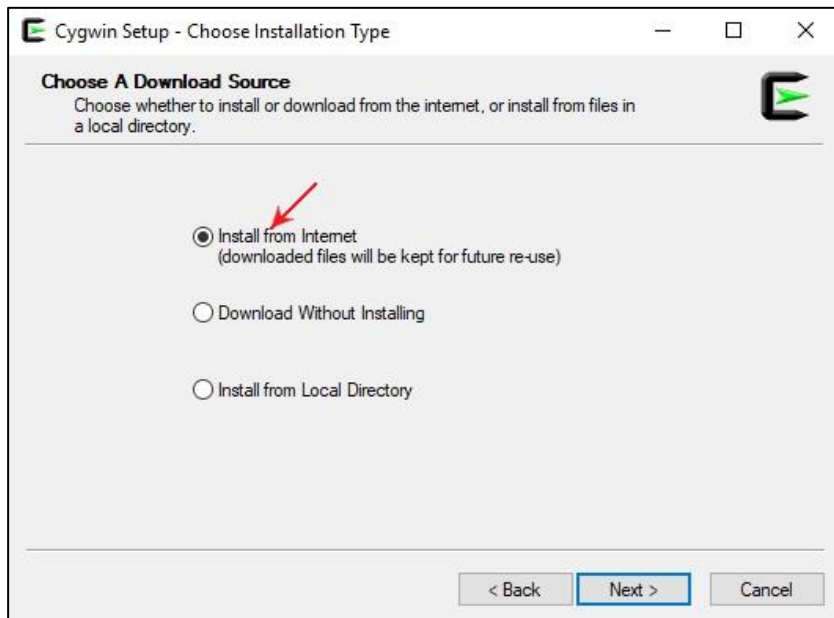
For details, refer to the description of available functions and tutorials in the [GitHub repository](#).

C, Windows (Cygwin) OS

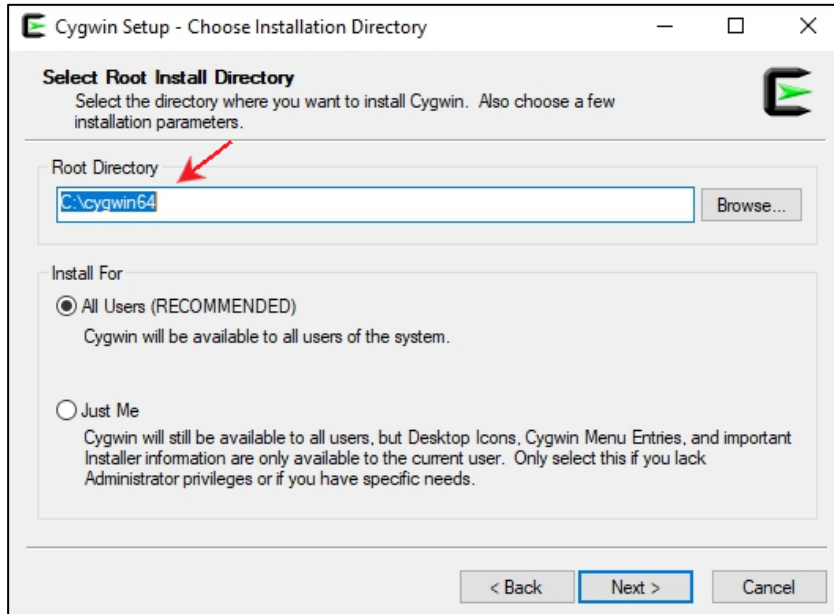
1. Follow the [link](#) to download a proper Cygwin version and install Cygwin following the instructions below:
 - Run the executable file. In the setup screen, click **Next** to start installation.



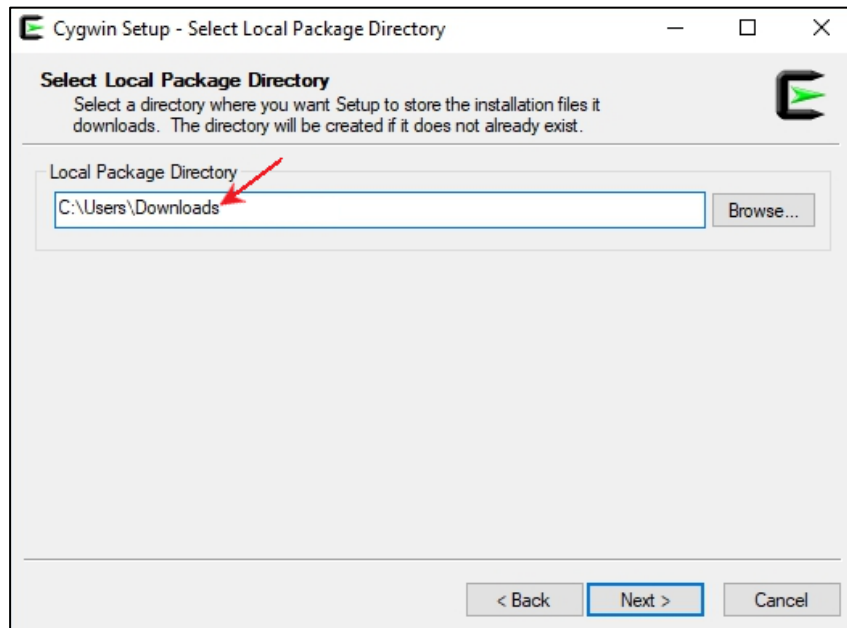
- Choose a source for downloading and click **Next**.



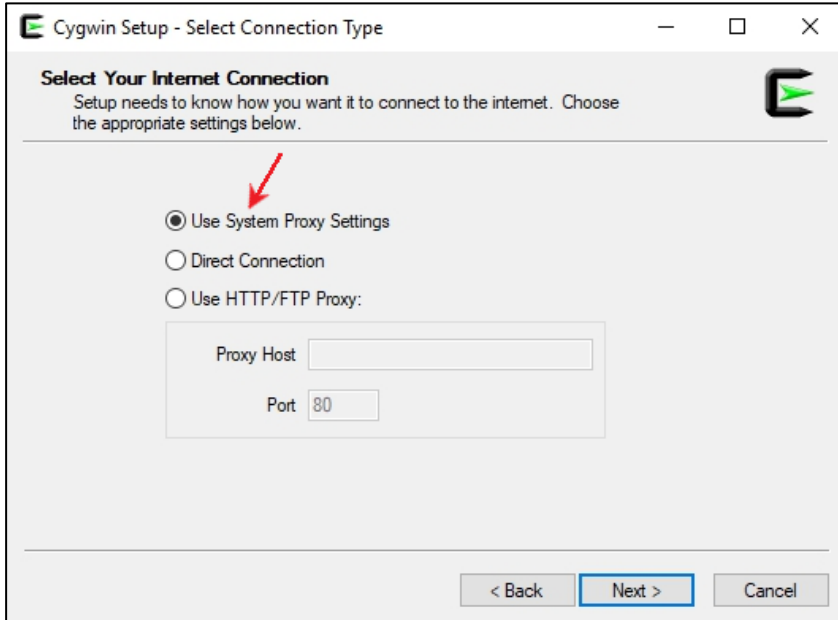
- Select a directory for installing the files. Leave the default one or browse to set up another installation path. Click **Next**.



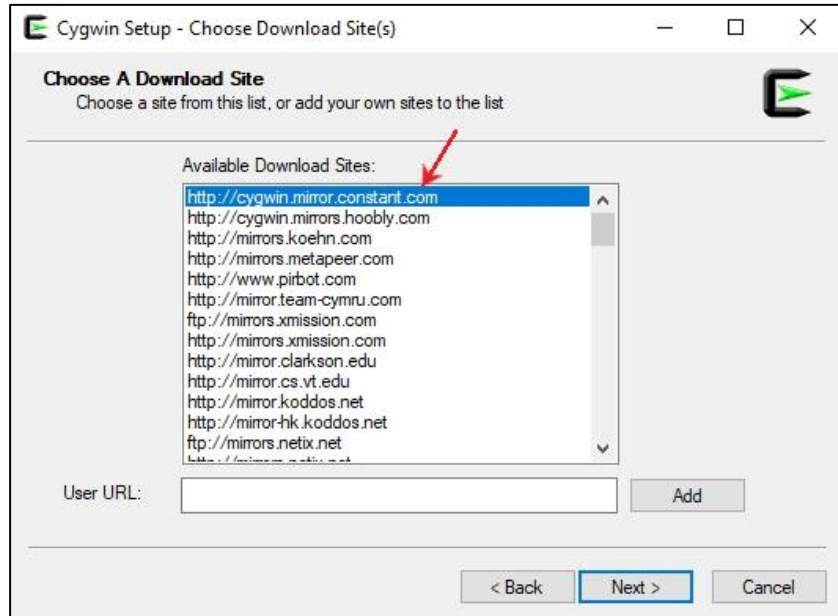
- Select a local directory and click **Next**.



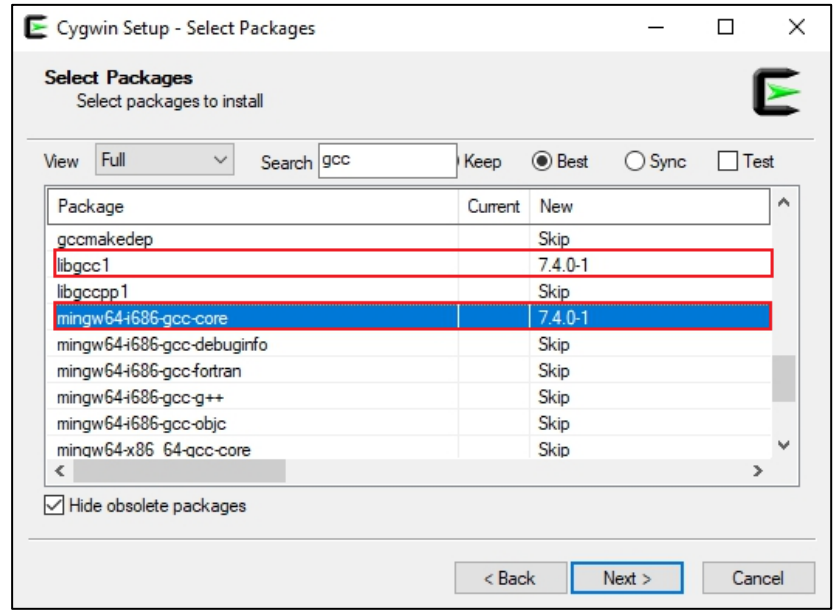
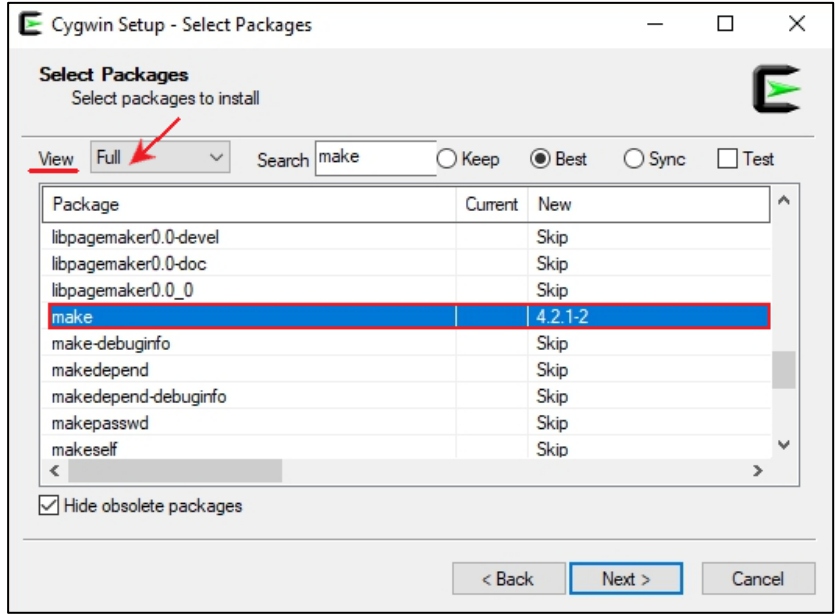
- Select a connection type and click **Next**.

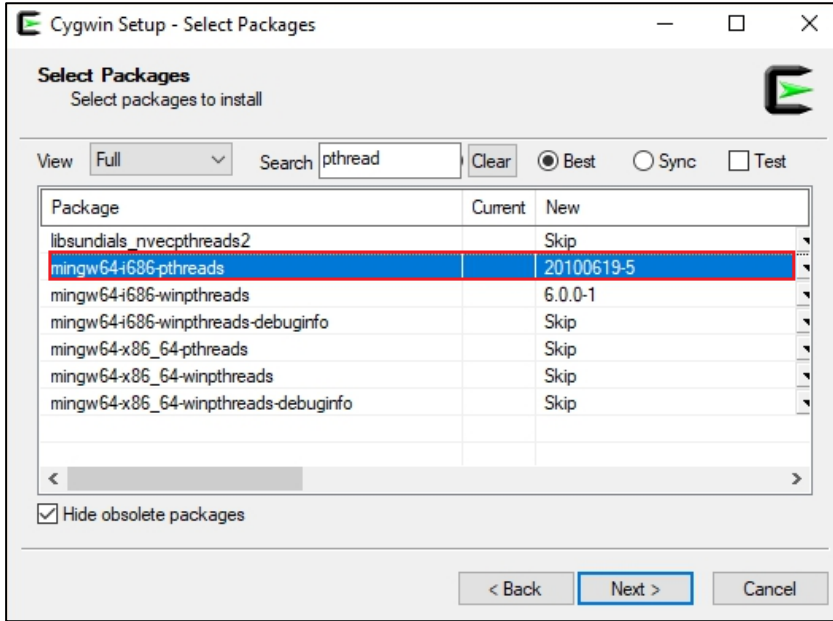


- Choose a source site for downloading. Click **Next**.

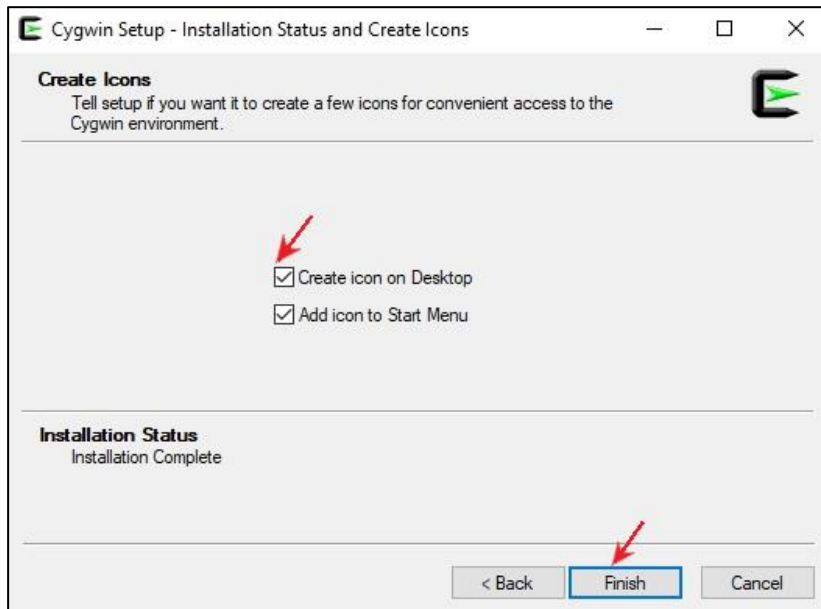


- In the **View** field, select **Full**. Then, select **MinGW**, **make**, and **gcc** packages to install as shown in the figures below:

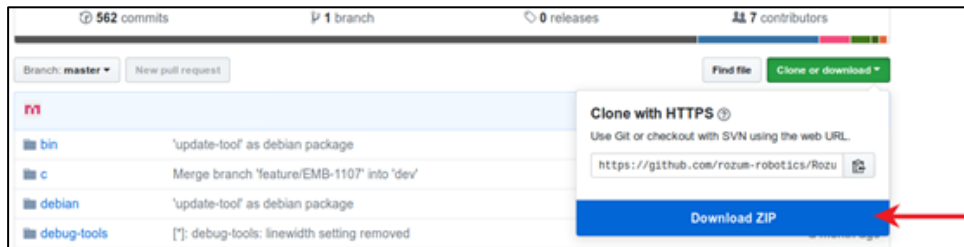




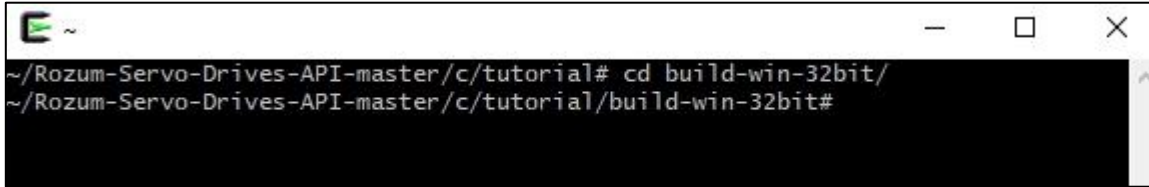
- Wait for the setup to complete and, on the final setup screen, select the checkboxes to create desktop and **Start Menu** icons. Click **Finish** to quit setup.



2. Download **Rozum-Servo-Drives-API** from [the GitHub repository](#).



- Unzip the downloaded file to the folder `C/Cygwin/home/user name`.
- Double-click the **Cygwin** icon to start the console.
- In the console, run the `cd c` command to navigate to the `c` folder.

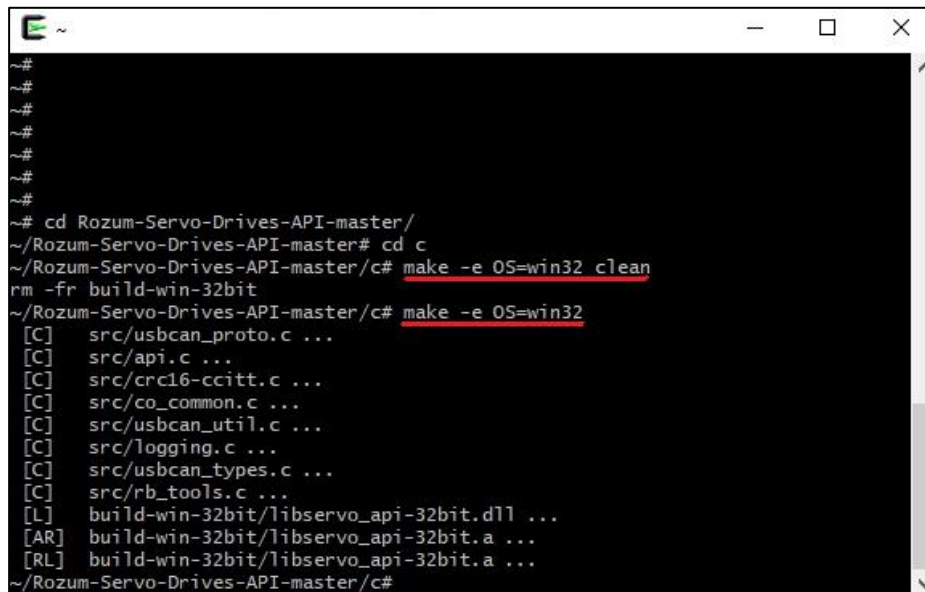


```
~/Rozum-Servo-Drives-API-master/c/tutorial# cd build-win-32bit/
~/Rozum-Servo-Drives-API-master/c/tutorial/build-win-32bit#
```

- In the console, execute the `make clean; make` commands to compile the library. Make sure to add `-e OS = win32`.

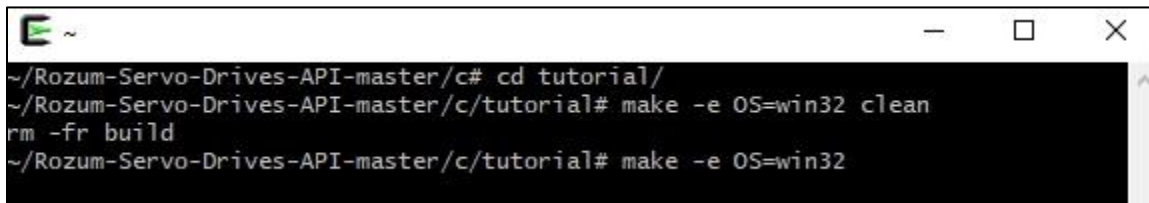
As a result, apart from the rest, you get the two files:

- `build/libservo_api.dll`
- `build/libservo_api.a`



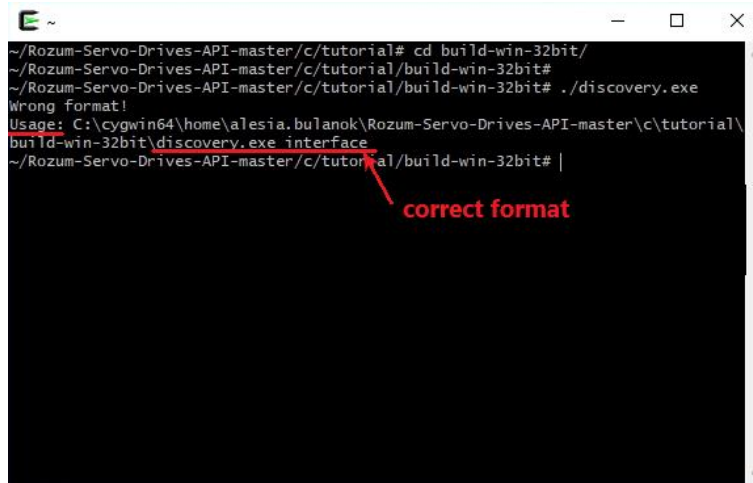
```
~#
~#
~#
~#
~#
~#
~# cd Rozum-Servo-Drives-API-master/
~/Rozum-Servo-Drives-API-master# cd c
~/Rozum-Servo-Drives-API-master/c# make -e OS=win32 clean
rm -fr build-win-32bit
~/Rozum-Servo-Drives-API-master/c# make -e OS=win32
[C] src/usbcan_proto.c ...
[C] src/api.c ...
[C] src/crc16-ccitt.c ...
[C] src/co_common.c ...
[C] src/usbcan_util.c ...
[C] src/logging.c ...
[C] src/usbcan_types.c ...
[C] src/rb_tools.c ...
[L] build-win-32bit/libservo_api-32bit.dll ...
[AR] build-win-32bit/libservo_api-32bit.a ...
[RL] build-win-32bit/libservo_api-32bit.a ...
~/Rozum-Servo-Drives-API-master/c#
```

- Run the `cd tutorial` command to navigate to the `tutorial` folder.
- Run `make`. As a result, you get the build folder with executable tutorial files.



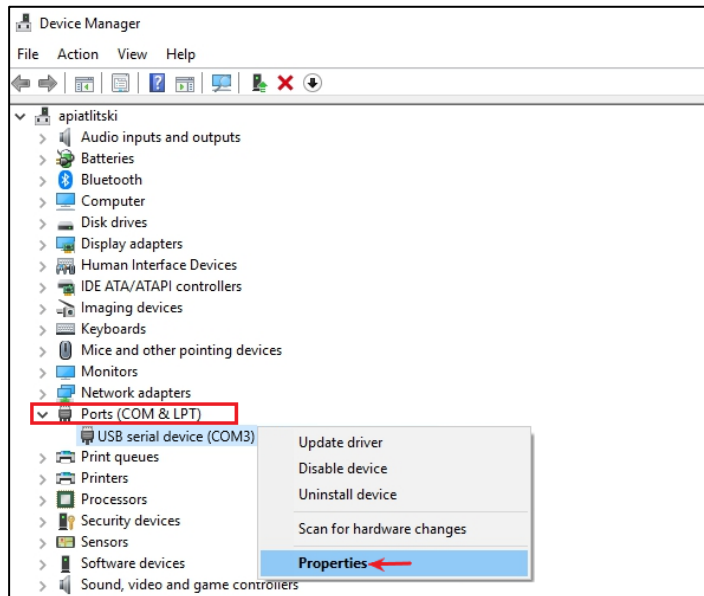
```
~/Rozum-Servo-Drives-API-master/c# cd tutorial/
~/Rozum-Servo-Drives-API-master/c/tutorial# make -e OS=win32 clean
rm -fr build
~/Rozum-Servo-Drives-API-master/c/tutorial# make -e OS=win32
```

9. Navigate to the `build` folder and run an executable file of any tutorial without setting command parameters.

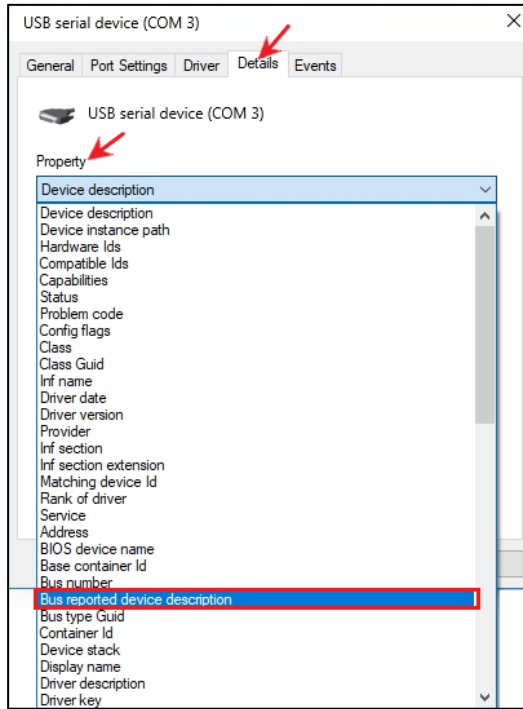


Since you have set no specific parameters, you will get a response with a 'Wrong format' warning. However, the Usage section of the same response will illustrate the correct format for setting up the parameters for a specific tutorial.

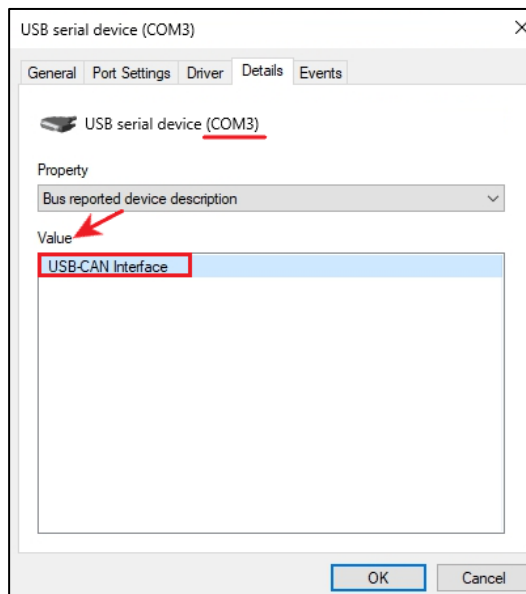
10. Find out the CAN Interface ID — the identifier of the used USB-CAN dongle.
 - a. Open the **Device Manager**. In the device list, open the **Ports (COM & LPT)** dropdown list. Right-click any device on the list and select **Properties** in the context menu.



- b. In the property window, go to the **Details** tab, open the dropdown list of the **Property** field, and select **Bus reported device description**.



- c. On the same tab, look at the **Value** field.
 - If it contains **USB-CAN Interface**, this means the selected device is the USB-CAN dongle you are using. Use the port number of the device (in this case, **COM 3**) as the CAN Interface ID.
 - If the field contains some other value, go back to **step a** and repeat the same sequence for another device on the Device Manager list.

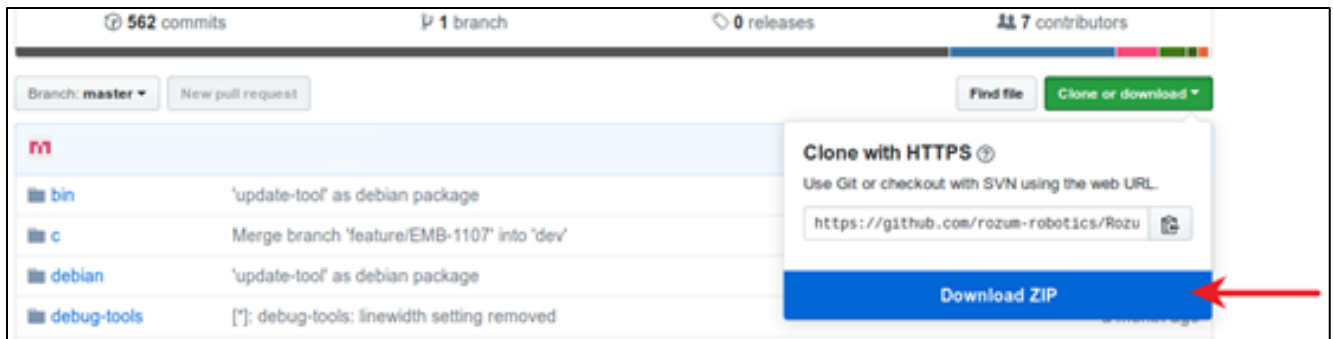


11. Find out the CAN ID of the connected servo. To do this, run the `discovery` tutorial from the `build` folder. In the output of the tutorial, you will find the CAN ID you need.
12. Select a tutorial from the tutorial folder. Before running it, make sure to replace the tutorial variables with the CAN Interface ID from **Step 10** and the CAN ID from **Step 11**.

For details, refer to the description of available functions and tutorials in the [GitHub repository](#).

C, Linux OS

1. Download **Rozum-Servo-Drives-API** from [the GitHub repository](#).



2. Unzip the downloaded file.
3. In the console, run the `cd c` command to navigate to the `c` folder.
4. In the console, execute the `make clean; make` command to compile the library. In the output, apart from the rest, you get the two files:

- a. `build/libservo_api.so`
- b. `build/libservo_api.a`

```
~/work/test/Rozum-Servo-Drives-API-master # cd c
~/work/test/Rozum-Servo-Drives-API-master/c #

~/work/test/Rozum-Servo-Drives-API-master/c # make clean; make
rm -fr build
[C] src/rb_tools.c ...
[C] src/usbcn_types.c ...
[C] src/logging.c ...
[C] src/usbcn_util.c ...
[C] src/co_common.c ...
[C] src/crc16-ccitt.c ...
[C] src/api.c ...
[C] src/usbcn_proto.c ...
[L] build/libservo_api.so ...
[AR] build/libservo_api.a ...
[RL] build/libservo_api.a ...
~/work/test/Rozum-Servo-Drives-API-master/c #
```

5. Run the `cd tutorial` command to navigate to the tutorial folder.

```
~/work/test/Rozum-Servo-Drives-API-master/c # cd tutorial/
~/work/test/Rozum-Servo-Drives-API-master/c/tutorial #
```

6. Run make. As a result, you get the build folder with executable tutorial files.

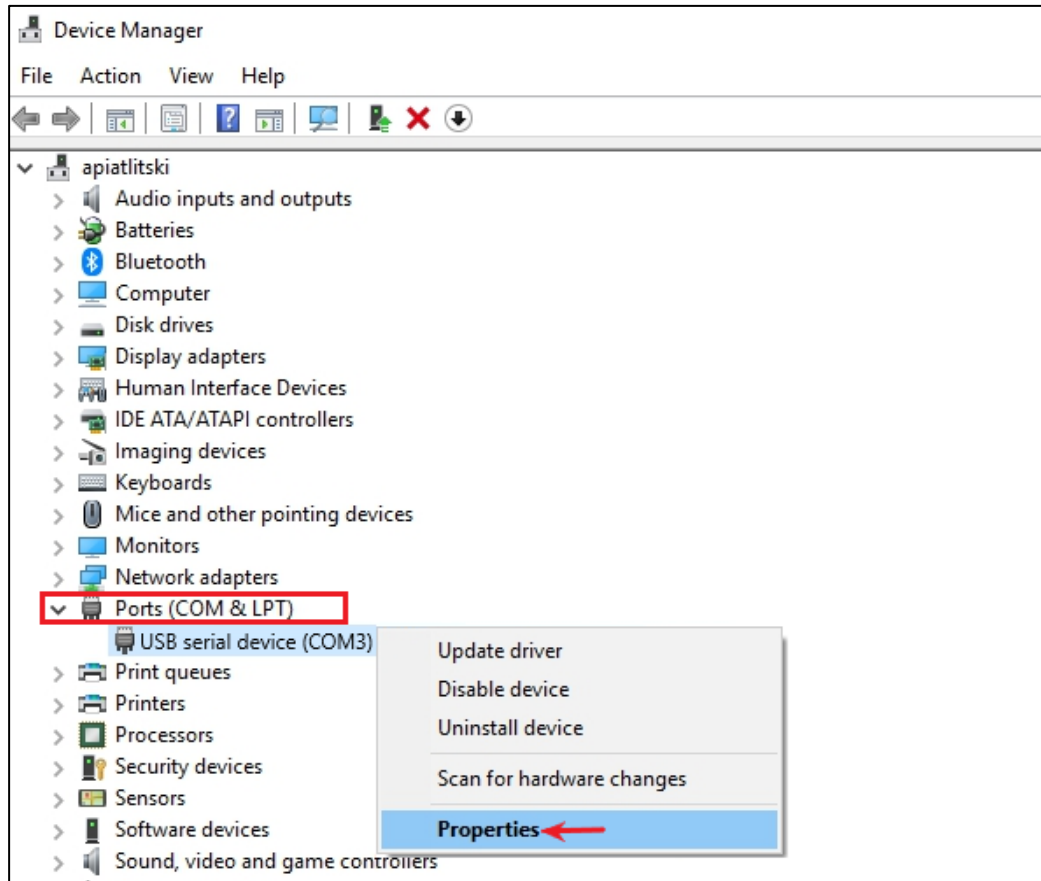
```
~/work/test/Rozum-Servo-Drives-API-master/c/tutorial # make
make -C ..
make[1]: Entering directory '/mnt/hdd/home/work/test/Rozum-ServoDrives-API-master/c'
make[1]: Leaving directory '/mnt/hdd/home/work/test/Rozum-Servo-Drives-API-master/c'
make -f tutorial.mk -C .
make[1]: Entering directory '/mnt/hdd/home/work/test/Rozum-Servo-Drives-API-master/c/tutorial'
make -C ..
make[2]: Entering directory '/mnt/hdd/home/work/test/Rozum-Servo-Drives-API-master/c'
make[2]: Leaving directory '/mnt/hdd/home/work/test/Rozum-Servo-Drives-API-master/c'
mkdir -p build
gcc change_servo_id.c -g2 -I../include -o build/change_servo_id
../build/libservo_api.a -static -lpthread -lm
gcc read_errors.c -g2 -I../include -o build/read_errors
../build/libservo_api.a -static -lpthread -lm
gcc read_servo_trajectory_time.c -g2 -I../include -o
build/read_servo_trajectory_time ../build/libservo_api.a -static -lpthread -
lm
gcc read_any_param.c -g2 -I../include -o build/read_any_param
../build/libservo_api.a -static -lpthread -lm
gcc control_servo_traj_1.c -g2 -I../include -o
build/control_servo_traj_1../build/libservo_api.a -static -lpthread -lm
gcc check_motion_points.c -g2 -I../include -o
build/check_motion_points../build/libservo_api.a -static -lpthread -lm
gcc discovery.c -g2 -I../include -o build/discovery ../build/libservo_api.a
-static -lpthread -lm
gcc time_optimal_movement.c -g2 -I../include -o build/time_optimal_movement
../build/libservo_api.a -static -lpthread -lm
gcc read_servo_max_velocity.c -g2 -I../include -o
build/read_servo_max_velocity ../build/libservo_api.a -static -lpthread -lm
gcc read_emcy_log.c -g2 -I../include -o build/read_emcy_log
../build/libservo_api.a -static -lpthread -lm
gcc calibrate_cogging.c -g2 -I../include -o build/calibrate_cogging
../build/libservo_api.a -static -lpthread -lm
gcc hb_timings.c -g2 -I../include -o build/hb_timings
../build/libservo_api.a -static -lpthread -lm
gcc read_any_param_cache.c -g2 -I../include -o build/read_any_param_cache
../build/libservo_api.a -static -lpthread -lm
gcc calibration_quality.c -g2 -I../include -o build/calibration_quality
../build/libservo_api.a -static -lpthread -lm
gcc control_servo_traj_2.c -g2 -I../include -o build/control_servo_traj_2
../build/libservo_api.a -static -lpthread -lm
gcc control_servo_traj_3.c -g2 -I../include -o build/control_servo_traj_3
../build/libservo_api.a -static -lpthread -lm
gcc read_servo_motion_queue.c -g2 -I../include -o
build/read_servo_motion_queue ../build/libservo_api.a -static -lpthread -lm
make[1]: Leaving directory '/mnt/hdd/home/maksimatkou/work/test/Rozum-
Servo-Drives-API-master/c/tutorial'
~/work/test/Rozum-Servo-Drives-API-master/c/tutorial #
```

7. Navigate to the build folder and run an executable file of any tutorial.

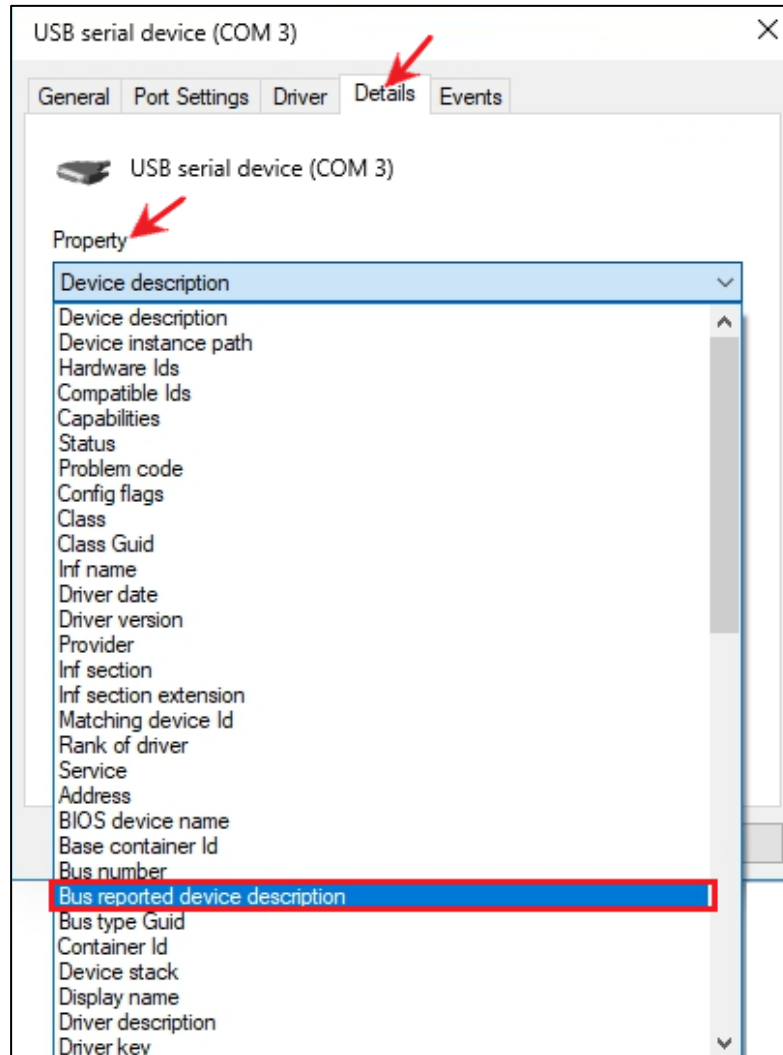
```
~/work/test/Rozum-Servo-Drives-API-master/c/tutorial # cd build/
~/work/test/Rozum-Servo-Drives-API-master/c/tutorial/build #
```

Since you have set no specific parameters for the executable file, you will get a response with a ‘Wrong format’ warning. However, the USAGE section of the same response will illustrate the correct format for setting up the parameters for the specific tutorial.

8. Find out the CAN Interface ID — the identifier of the used USB-CAN dongle:
 - a. Open the **Device Manager**. In the device list, open the **Ports (COM & LPT)** dropdown list. Right-click any device on the list and select **Properties** in the context menu.

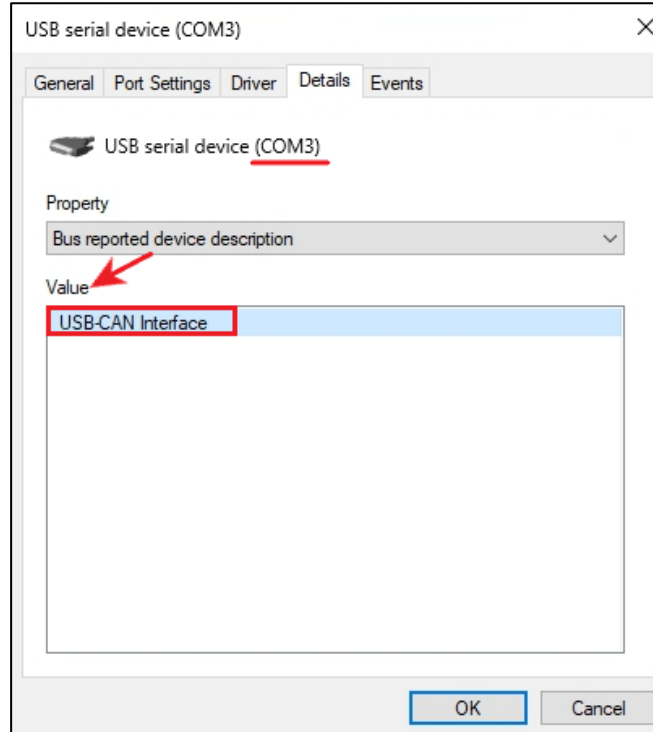


- b. In the **Properties** window, go to the **Details** tab, open the dropdown list of the **Property** field, and select **Bus reported device description**.



- c. On the same tab, look at the **Value** field.

- If it contains **USB-CAN Interface**, this means the selected device is the USB-CAN Interface you need. Use the port number of the device (in this case, **COM 3**) as the CAN Interface ID.
- If the field contains some other value, go back to **step a** and repeat the same sequence for another device on the Device Manager list.



- Find out the CAN ID of your servo. To do this, run the `discovery` tutorial from the `build` folder. In the output of the tutorial, you will find the CAN ID you need.

Note: You will need the CAN ID of the connected servo to set up the parameters for running tutorials on the connected servo (Step 10).

- Run any of the tutorials with preset parameters — the CAN Interface ID from **Step 8**, the CAN ID from **Step 9**, as well as any other parameters as needed.

```
~/work/test/Rozum-Servo-Drives-API-master/c/tutorial/build #
./read_any_param_cache /dev/ttyACM0 123
WARN: Emergency frame received: id(123) code(0x0) reg(0x80) bits(0x45)
info(0x0) :
'Error Reset or No Error, Heartbeat consumer timeout'
INFO: ID: 123 Device is in pre-operational mode
===== Tutorial of programming and reading the device parameter cache =====
APP_PARAM_POSITION_ROTOR value: 137.289
APP_PARAM_VELOCITY_ROTOR value: 0.069
APP_PARAM_VOLTAGE_INPUT value: 47.327
APP_PARAM_CURRENT_INPUT value: 0.000
~/work/test/Rozum-Servo-Drives-API-master/c/tutorial
```

For details, refer to the description of available functions and tutorials in the [GitHub repository](#).

5.2.4 Setting new CAN IDs when connecting multiple servos

Since all RDrive servos are supplied with default CAN IDs ranging from 32 to 37, there is a risk of collision in case you connect more than one servo to the same CAN bus. To avoid collisions, you need to set a unique CAN ID to replace the default one for each of the servos. To do this, follow the instructions below:

1. Take servo 1 and connect it **unloaded** to the CAN bus and the power supply as described in **Section 4.2.3** and **Section 4.2.4** accordingly.
2. Enable API control by completing all the steps, except for the last one, from the instructions in **Section 5.2.3**.

Python, Windows OS

Python, Linux OS

C, Windows (Cygwin) OS

C, Linux OS

3. Run the special tutorials to change the CAN ID
 - **C** — [change_servo_id.c](#)
 - **Python** — [change_servo_id.py](#)
4. Remember or write down the new CAN ID and disconnect the servo.
5. Repeat steps 1-4 to change the IDs of the other servos.

6 MAINTENANCE

RDrive servomotors contain no parts that users need to service. Therefore, the maintenance scope is limited to the following:

- Visual inspection of electrical connections for damages
- Checking screw connections and tightening the loosened ones
- Monitoring vibration and noise levels, as well as heating during daily operation



The above-listed maintenance procedures are not mandatory. It is up to the user to decide to follow the recommendations or not, depending on the application-specific conditions (e.g., whether a servo is accessible for visual inspection).

7 TROUBLESHOOTING

Table 7-1 describes the most common problems that can occur during operation of RDrive servomotors, their possible causes, and recommended user actions.

Table 7-1: Possible servo problems, their causes, and recommended actions

Problem	Possible cause	Recommended action
The motor will not start.	Improper connection.	Check the motor connections.
	Incorrect motor control settings.	Check the motor control settings or contact our service department.
The motor is overheating.	Excessive accumulation of dirt.	Clean the outside of the motor.
	Ambient temperature is too high.	Provide adequate cooling.
	Air pressure is too low because of the altitude.	Provide adequate cooling.

	The motor is too hot.	Check the power supply is up to the specifications (Table 3-1) or contact our service department.
Increased noise during operation.	Damaged bearing.	Contact our service department.
Sporadic failure.	Damaged cable.	Contact our service department.



Only qualified service personnel can perform repair works.

8 TRANSPORTATION AND STORAGE

For transporting the RDrive servomotor, always use the original packaging. In case you have no plans to put the servos into operation immediately upon delivery, make sure to meet the following storage requirements:

- Store servomotors in a dry, dust- and vibration-free location.
- The admissible storage temperature range—from +5°C to +40°C.
- The storage humidity should be 80% max at 25°C (90% at 20°C).

ANNEX I. UPDATING THE FIRMWARE OF RDRIVE SERVOS

Linux OS

1. Download the firmware file to update the RDrive servo.
2. Download the Servo API source code from the [GitHub repository](#).
3. Navigate to the folder with firmware update tools by running the following command in the console:

```
cd/full path to the folder with firmware update tools
```

Example: `cd ~/Sources/userapi/c/tools`

4. Run the `make` command to assemble an up-to-date build of the firmware update tool.
5. Update the servo firmware by running the following command in the console:

```
~/Sources/userapi/c/tools/fw-update-tool/build/rr-fw-update
/dev/serial/by-id/CAN Interface ID CAN ID ~/path to the firmware
update binary file
```

Example: `~/Sources/userapi/userapi/c/tools/fw-update-tool/build/rr-fw-update /dev/serial/by-id/usb-Rozum_Robotics_USB-CAN_Interface_301-if00 37 ~/Downloads/Servo\Firmware\dev-c/firmware/servo_v4_50.bin`

where:

- `usb-Rozum_Robotics_USB-CAN_Interface_301-if00` — name of the CAN interface (bus) where the firmware is to be updated
- `37` — identifier of the servo where the firmware is to be updated
- `~/Downloads/Servo\Firmware/dev-c/firmware/servo_v4_50.bin` — path to the firmware update binary file

When the firmware update is completed successfully, the command output will contain **INFO: FLASH OK**. When any problems occur during the firmware update, the command returns an error.

6. Verify the configuration of servo parameters by running the following command in the console:

```
~/Sources/userapi/c/tools/cfg-update-tool/build/rr-cfg-update
/dev/serial/by-id/CAN_Interface_ID CAN_ID ~/path to the configuration
binary file
```

Example: `~/Sources/userapi/c/tools/cfg-update-tool/build/rr-cfg-update
/dev/serial/by-id/usb-Rozum_Robotics_USB-CAN_Interface_301-if00 37
~/Downloads/Servo\Firmware/dev-c/settings/50.jsonconfig`

where:

- `usb-Rozum_Robotics_USB-CAN_Interface_301-if00` — name of the CAN interface (bus) where the firmware is to be updated
- `37` — identifier of the servo where the firmware is to be updated
- `~/Downloads/Servo\Firmware/dev-c/firmware/servo_v4_50.bin` — path to the configuration binary file

Windows (Cygwin) OS

1. Download the firmware file to update the RDrive servo.
2. Download the Servo API source code from the [GitHub repository](#).
3. Navigate to the folder with firmware update tools by running the following command in the console:

```
cd/ full path to the folder with firmware update tools
```

Example: `cd ~/Sources/userapi/c/tools`

4. Run the `make` command to assemble an up-to-date build of the firmware update tool.
5. Update the servo firmware by running the following command in the console:

```
~/Sources/userapi/c/tools/fw-update-tool/build/rr-fw-update
/dev/ttyS3 CAN_ID ~/ path to the firmware update binary file
```

Example: `~/Sources/userapi/userapi/c/tools/fw-update-tool/build/rr-fw-update /dev/ttyS3 37 ~/Downloads/Servo\ Firmware/dev-c/firmware/servo_v4_50.bin`

where:

- `/dev/ttyS3*` — name of the CAN interface (bus) where the firmware is to be updated
- `37` — identifier of the servo where the firmware is to be updated
- `~/Downloads/Servo\ Firmware/dev-c/firmware/servo_v4_50.bin` — path to the configuration binary file

When the firmware update is completed successfully, the command output will contain `INFO: FLASH OK`. When any problems occur during the firmware update, the command returns an error.

6. Verify the configuration of servo parameters by running the following command in the console:

```
~/Sources/userapi/c/tools/cfg-update-tool/build/rr-cfg-update /dev/ttyS3 CAN_ID ~/ path to the firmware update binary file
```

Example: `~/Sources/userapi/c/tools/cfg-update-tool/build/rr-cfg-update /dev/ttyS3 37 ~/Downloads/Servo\ Firmware/dev-c/settings/50.jsonconfig`

where:

- `/dev/ttyS3*` — name of the CAN interface (bus) where the firmware is to be updated
- `37` — identifier of the servo where the firmware is to be updated
- `~/Downloads/Servo\ Firmware/dev-c/firmware/servo_v4_50.bin` — path to the configuration binary file

**To look through the interface list, run the [Cygwin] command: `ls /dev/tty*`*