# **RDRIVE SERVOMOTOR**



# QUICK-START GUIDE

## **Getting Started with RDrive**

The document describes how to conduct *an unloaded test run* to get started with an RDrive motor, using **Python or C Servo API** by Rozum Robotics and a PC running under one of the operating systems:

- Linux OS
- Windows OS

Whether with Python or C, Linux or Windows, getting started with an RDrive takes three easy steps:

#### Step 1. Preparing

- Step 2. Connecting the hardware
- Step 3. Enabling servo control via API

Linux OS + Python Windows OS + Python Linux OS + C Windows OS (Cygwin) + C

# Step 1. Preparing

The step is the same for both operating systems, as well as for Python and C languages.

1. Unpack your servo supply package. It should contain:

- an RDrive servo motor
- a CAN-USB dongle
- a Micro USB-USB A cable

For the test run, provide additionally:

- a 48 V power supply unit
- a motor mount to hold the servo stationary

The equipment and hardware connection instructions are FOR UNLOADED TEST RUNS ONLY!



For loaded servo operation, you need to add at least one energy eater and one capacitor to the supply circuit, or use the servobox solution by Rozum Robotics. For details, refer to the <u>Servobox documentation</u>.

## Step 2. Connecting the hardware

Just as the preceding one, the step is the same for both operating systems, as well as C and Python languages.

1. Fix the servo in place in a motor mount (e. g., as shown below) or otherwise as appropriate for your purposes.



- 2. Connect the servo to the power supply unit, using the red and black wires. The red one is for "+," and the black one is for "-."
- 3. Connect the servo to the CAN bus connector on the CAN-USB dongle, using the blue and brown wires. The brown one is for CAN<sub>HIGH</sub>, and the blue one is for CAN<sub>LOW</sub>.



- 4. Connect the Micro USB-USB A cable to the Micro USB-USB A cable connector on the CAN-USB dongle and the USB port on your computer.
- 5. Plug the power supply connector into a power supply socket.

Now, the servo is ready for Step 3.

### Step 3. Enabling servo control via API

### Linux OS + Python

#### Prerequisites

• Python (3.5 or a later version)

To install Python, run one of the following commands in the console:

#### - 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
```



In case you have a Linux distribution other than specified above, proceed in the usual order of running the installation procedure in your system.

- pip 3 package (installed together with Python as described above)
- gcc MinGW compiler
- make package

#### Installation and setup sequence

- 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 identifier of the CAN-USB dongle in the current 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.

- 3. Find out the CAN ID of the connected servo motor. To do this, complete the following steps:
  - In the console, run any tutorial from the examples folder. To do this, execute the below command, specifying the CAN Interface ID from Step 2 and a random 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;



If needed, replace the tutorial name from the current example with any other from the list of available <u>servo API tutorials</u>.

```
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 a random CAN ID for the servo.

• 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 tutorial 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



If needed, replace the tutorial name from the current example with any other from the list of available servo API tutorials.

interface/dev/serial/by-id/usb-Rozum Robotics USB-

<code>CAN\_Interface\_301-if00</code> is the parameter specifying the CAN Interface ID (the one we got at Step 2)

servo\_1\_id 37 is the parameter specifying the CAN ID of the connected servo (the one we got at Step 3).

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



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

Now that you know how to start the servo and use the tutorials, you can create your own code. For more information, refer to the <u>Python API documentation</u>.

### Windows OS + Python

#### **Prerequisites**

• Python (3.5 or a later version)

To install Python, navigate to Python, download, and complete the setup process.



At the installation setup screen, make sure to check Add Python v.X to PATH (where v.X. is the downloaded Python version).



If your PC runs Windows 8 or earlier versions of the operating system, download and install a driver for the CAN-USB dongle from <u>the web page</u>.

#### Installation and setup sequence

- 1. Install the <u>Servo API library</u> 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 via the **Start menu**.
- 3. Find out the CAN interface name the identifier of the CAN-USB dongle in the user environment:
  - a. On your computer, open Start menu  $\rightarrow$  System Settings  $\rightarrow$  Devices  $\rightarrow$  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.

← Settings		– ¤ ×
û Home	Bluetooth & other devices	
Find a setting        Devices	+ Add Bluetooth or other device Other devices	Turn on Bluetooth even faster To turn on Bluetooth without opening Settings, open action center, and then select the Bluetooth icon. Do the same to turn it off when you want.
Printers & scanners	Generic Non-PnP Monitor	Get more info about Bluetooth
() Mouse	USB Serial Device (COM3)	Related settings
Touchpad		Devices and printers
Typing	Download over metered connections	Sound settings Display settings
🖉 Pen & Windows Ink	To help prevent extra charges, keep this off so device software	
AutoPlay     AutoPlay	(drivers, info, and apps) for new devices won't download while you're on metered Internet connections.	Have a question? Get help
🖞 USB		
		Make Windows better Give us feedback

- 4. Find out the CAN ID of the connected servo:
  - a. In the command console, run any tutorial from the examples folder. To do this, execute the below command, specifying the CAN Interface ID from Step 3 and a random 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.



If needed, replace the tutorial name from the current example with any other from the list of available <u>servo API tutorials</u>.

interface COM3 is the parameter specifying the CAN Interface ID from Step 3

servo 1 id 37 is the parameter specifying a random CAN ID for the 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 command line interface:

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



If needed, replace the tutorial name from the current example with any other from the list of available servo API tutorials.

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 the CAN ID of 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.



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

Now that you know how to start the servo and use the tutorials, you can create your own code. For more information, refer to the <u>Python API documentation</u>.

### Linux OS + C

#### **Prerequisites**

- gcc compiler
- libpthread library
- make package
- 1. Download Rozum-Servo-Drives-API from the GitHub repository.

rozum-robotics /	Rozum-Servo-Drives-	API			• Watch 2	★ Sta	ar 4	¥ Fork 1
<>Code ① Issues	0 17 Pull requests 0	Projects 0	C Security	Insights				
		ome to over 40		<b>today</b> ers working toge I build software to				Dismiss
lo description, websit	te, or topics provided. <b>¥ 1</b> branch		🗊 <b>0</b> packag	es	<b>♡ 0</b> releases	2	<b>1</b> contri	butors
Branch: master - New	w pull request					Find file	Clone o	or download <del>-</del>
measles Merge brand	ch 'dev' into 'master'				Clone with HTTPS	-		
bin	[*] Add empty line at the e	end of build_an	d_deploy_pythc	n.sh	Use Git or checkout wit			
i c	Resolve MRKT-101 "Featu	ire/"			https://github.com/	rozum-rob	otics/Roz	tum-
🖿 debian	dependency to build-esse	ential			Open in Desktop		AK	вация ad ZIPивир

- 2. Unzip the downloaded file.
- 3. In the console, run the cd c command to navigate to the c folder.

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

- 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/c # make clean; make
rm -fr build
[C] src/rb tools.c ...
[C]
     src/usbcan types.c ...
[C] src/logging.c ...
[C] src/usbcan_util.c ...
[C]
     src/co common.c ...
[C] src/crc16-ccitt.c ...
[C] src/api.c ...
     src/usbcan proto.c ...
[C]
     build/libservo api.so ...
[L]
[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/maksimkatkou/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.

```
~/work/test/Rozum-Servo-Drives-API-master/c/tutorial/build #
./read_any_param_cache
Wrong format!
Usage: ./read_any_param_cache interface id
~/work/test/Rozum-Servo-Drives-API-master/c/tutorial/build #
```

- 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.

🗂 Device Manager	
File Action View Help	
	٠
🗸 🖁 apiatlitski	
> 🐗 Audio inputs and outputs	
> 🍃 Batteries	
> 🚯 Bluetooth	
> 💻 Computer	
> 👝 Disk drives	
> 🏣 Display adapters	
> 🎮 Human Interface Devices	
> 📷 IDE ATA/ATAPI controllers	
> 🚠 Imaging devices	
> Keyboards	
> II Mice and other pointing devices	
> 📃 Monitors	
> 🚽 Network adapters	
V 💭 Ports (COM & LPT)	
USB serial device (COM3)	Update driver
> 🚍 Print queues	Disable device
> 🚍 Printers	
> Processors	Uninstall device
> P Security devices	Scan for hardware changes
> 🔚 Sensors	-
Software devices	Properties
> Sound, video and game controlle	rs

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

USB serial device (COM 3)	×
General Port Settings Driver Details Events	
USB serial device (COM 3)	
Device description	$\sim$
Device description Device instance path Hardware Ids Compatible Ids Capabilities Status Problem code Corfig flags Class Class Class Guid Inf name Driver date Driver version Provider Inf section Inf section extension Matching device Id Rank of driver Service Address BIOS device name Base container Id Bus number	
Bus reported device description Bus type Guid	
Container Id Device stack Display name Driver description	
Driver key	× 1

- 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 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.

USB serial device (COM3)	×
General Port Settings Driver Details Events	
S USB serial device (COM3)	
Property	
Bus reported device description	$\sim$
Value	
USB-CAN Interface	
ОК	Cancel

- 9. 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 of the connected servo.
- 10. Run any of the tutorials with preset parameters the CAN Interface ID from **Step 8**, the CAN Servo ID from **Step 9**, and other parameters in the format as prompted in **Step 7**.

```
~/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
```

Now that you know how to start the servo and use the tutorials, you can create your own code. For more information, refer to the <u>API documentation</u>.

### Windows OS (Cygwin) + C

#### **Prerequisites**

- Cygwin
- gcc compiler
- libpthread library
- make package



If your PC runs Windows 8 or earlier versions of the operating system, download and install a driver for the CAN-USB dongle from <u>the web page</u>.

- 1. Follow the link to download Cygwin.
- 2. Run the executable file. In the setup screen, click Next to start installation.



3. Choose a source for downloading and click Next.

E Cygwin Setup - Choose Installation Type			×
Choose A Download Source Choose whether to install or download from the internet, or install from files in a local directory.		1	
<ul> <li>Install from Internet (downloaded files will be kept for future re-use)</li> <li>Download Without Installing</li> </ul>			
O Install from Local Directory			
< Back Nex	t>	Car	ncel

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

E Cygwin Setup - Choose Installation Directory -		Х
Select Root Install Directory Select the directory where you want to install Cygwin. Also choose a few installation parameters.	l	
Root Directory	Browse	ə
Ill Users (RECOMMENDED) Cygwin will be available to all users of the system.		
⊖ Just Me		
Cygwin will still be available to all users, but Desktop Icons, Cygwin Menu Entries, Installer information are only available to the current user. Only select this if you law Administrator privileges or if you have specific needs.		ant .
< Back Next >	Car	ncel

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

Cygwin Setup - Select Local Package Directory		-	Ш	×
Select Local Package Directory Select a directory where you want Setup to store the downloads. The directory will be created if it does n				
Local Package Directory				
C:\Users\Downloads			Browse	

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

E Cygwin Setup - Select Connection Type	_		×
Select Your Internet Connection Setup needs to know how you want it to connect to the internet. Choose the appropriate settings below.		1	Å
● Use System Proxy Settings			
O Direct Connection			
O Use HTTP/FTP Proxy:			
Proxy Host			
Port 80			
< Back Ne	xt >	Can	icel

7. Choose a source site for downloading. Click Next.

Choose a si	wnload Site ite from this list, or add your own sites to the list			l
	Available Download Sites:			
	http://cygwin.mirror.constant.com http://cygwin.mirrors.hoobly.com http://mirrors.koehn.com	^		
	http://mirors.metapeer.com http://www.pirbot.com http://wirors.metapeer.com http://mirors.eam.com http://mirors.smission.com http://mirors.smission.com http://miror.cs.vt.edu http://miror.cs.vt.edu http://mirors.net/suddos.net http://mirors.net/suddos.net http://mirors.net/suddos.net	~		
User URL:			Add	

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

/iew Full 🖌 🗸	Search make	О Кеер	Best      Synce	: Test
Package	L	Current	New	
libpagemaker0.0-devel			Skip	
libpagemaker0.0-doc			Skip	
libpagemaker0.0_0			Skip	
make			4.2.1-2	
make-debuginfo			Skip	_
makedepend			Skip	
makedepend-debuginfo			Skip	
makepasswd			Skip	
makeself			Skip	`
<				>

Cygwin Setup - Select Packages					
Select Packages Select packages to install					
View Full View Search gcc	Keep	Best	◯ Sync	Test	_
	TROOP	0.000	Objile		
Package	Current	New			^
gccmakedep		Skip			
libgcc1		7.4.0-1			
libgccpp1		Skip			
mingw64-i686-gcc-core		7.4.0-1			
mingw64i686-gcc-debuginfo		Skip			
mingw64i686-gcc-fortran		Skip			
mingw64i686-gcc-g++		Skip			
mingw64i686-gcc-objc mingw64:x86 64-gcc-core		Skip Skip			5
<		SKIP		>	
Hide obsolete packages					
	< Bac	k I	Next >	Cance	el
Cygwin Setup - Select Packages	< Bac	k I	Next >	Cance	
	< Bac	k I	Next >		
Select Packages Select packages to install	< Bac	Best	Next > 		
Select Packages Select packages to install			_		
Select Packages Select packages to install	Clear	Best	_		
Select Packages Select packages to install View Full ✓ Search pthread Package	Clear	Best     New	_ O Sync		
Select Packages Select packages to install View Full Very Search pthread Package Ibsundials_nvecpthreads2	Clear	<ul> <li>Best</li> <li>New</li> <li>Skip</li> </ul>	_ O Sync		
Select Packages Select packages to install View Full Search pthread Package Ibsundials_nvecpthreads2 mingw644686pthreads	Clear	Best     New     Skip     2010061	_ O Sync		
Select Packages Select packages to install View Full ✓ Search pthread Package libsundials_nvecpthreads2 mingw644686-winpthreads mingw644686-winpthreads	Clear	Best     New     Skip     2010061     6.0.0-1	_ O Sync		;
Select Packages Select packages to install View Full Search pthread Package libsundials_nvecpthreads2 mingw641686.pthreads mingw644686.winpthreads mingw644686.ethreads mingw644686.ethreads	Clear	Best     New     Skip     2010061     6.0.0-1     Skip	_ O Sync		;
Select Packages Select packages to install View Full Search pthread Package libsundials_nvecpthreads2 mingw644686-minpthreads mingw644686-minpthreads mingw644686-minpthreads.debuginfo mingw644586_64-pthreads	Clear	Best     New     Skip     2010061     6.0.0-1     Skip     Skip	_ O Sync		
Select Packages Select packages to install View Full Search pthread Package libsundials_nvecpthreads2 mingw641686.pthreads mingw644686.winpthreads mingw644686.ethreads mingw644686.ethreads	Clear	Best     New     Skip     2010061     6.0.0-1     Skip     Skip     Skip	_ O Sync		t
Select Packages Select packages to install View Full Search pthread Package libsundials_nvecpthreads2 mingw644686.pthreads mingw644686.winpthreads-debuginfo mingw64x86_64.pthreads mingw64x86_64.winpthreads mingw64x86_64.winpthreads.debuginfo	Clear	Best     New     Skip     2010061     6.0.0-1     Skip     Skip     Skip	_ O Sync		;
Select Packages Select packages to install View Full Search pthread Package libsundials_nvecpthreads2 mingw6446866-winpthreads mingw6446866-winpthreads mingw64486_64-winpthreads mingw64-x86_64-winpthreads mingw64-x86_64-winpthreads mingw64-x86_64-winpthreads debuginfo	Clear	Best     New     Skip     2010061     6.0.0-1     Skip     Skip     Skip	_ O Sync		t

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



@ 562 con	mits 🖗 1 branch	○ 0 releases	11 7 contributors	
Branch: master •	ew pull request		Find file Clone or download *	
M katkou Merge branch 'dev' into 'master'		Clone with	HTTPS 💿	
ille bin	'update-tool' as debian package	Use Git or che	Use Git or checkout with SVN using the web URL.	
ille c	Merge branch 'feature/EMB-1107' into 'dev'	https://git	thub.com/rozum-robotics/Rozu	
ile debian	'update-tool' as debian package		Download ZIP	
debug-tools	[*]: debug-tools: linewidth setting removed			
ilis java	[*]: high level API template		8 months ago	
python	Add get. state() method to Servo class		3 days ago	

10. Download Rozum-Servo-Drives-API from the GitHub repository.

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



14. In the console, execute the make clean; make commands to compile the library. Make sure to add -e OS = win32 (see figure below).

<pre>~# ~# ~# ~# ~# ~# ~# ~# ~# ~# ~# ~# ~# ~</pre>	<b>E</b> ~	-	×
<pre>~/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/cl6-ccitt.c [C] src/co_common.c [C] src/usbcan_util.c [C] src/usbcan_util.c [C] src/usbcan_types.c [C] src/usbcan_types.c [C] src/b_tools.c [L] build-win-32bit/libservo_api-32bit.dll [RL] build-win-32bit/libservo_api-32bit.a</pre>	~#		
<pre>~/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/cl6-ccitt.c [C] src/co_common.c [C] src/usbcan_util.c [C] src/usbcan_util.c [C] src/usbcan_types.c [C] src/usbcan_types.c [C] src/b_tools.c [L] build-win-32bit/libservo_api-32bit.dll [RL] build-win-32bit/libservo_api-32bit.a</pre>	~#		
<pre>~/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/cl6-ccitt.c [C] src/co_common.c [C] src/usbcan_util.c [C] src/usbcan_util.c [C] src/usbcan_types.c [C] src/usbcan_types.c [C] src/b_tools.c [L] build-win-32bit/libservo_api-32bit.dll [RL] build-win-32bit/libservo_api-32bit.a</pre>	~#		
<pre>~/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/cl6-ccitt.c [C] src/co_common.c [C] src/usbcan_util.c [C] src/usbcan_util.c [C] src/usbcan_types.c [C] src/usbcan_types.c [C] src/b_tools.c [L] build-win-32bit/libservo_api-32bit.dll [RL] build-win-32bit/libservo_api-32bit.a</pre>	~#		
<pre>~/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/cl6-ccitt.c [C] src/co_common.c [C] src/usbcan_util.c [C] src/usbcan_util.c [C] src/usbcan_types.c [C] src/usbcan_types.c [C] src/b_tools.c [L] build-win-32bit/libservo_api-32bit.dll [RL] build-win-32bit/libservo_api-32bit.a</pre>	~#		
<pre>~/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/cl6-ccitt.c [C] src/co_common.c [C] src/usbcan_util.c [C] src/usbcan_util.c [C] src/usbcan_types.c [C] src/usbcan_types.c [C] src/b_tools.c [L] build-win-32bit/libservo_api-32bit.dll [RL] build-win-32bit/libservo_api-32bit.a</pre>	~#		
<pre>~/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/cl6-ccitt.c [C] src/co_common.c [C] src/usbcan_util.c [C] src/usbcan_util.c [C] src/usbcan_types.c [C] src/usbcan_types.c [C] src/b_tools.c [L] build-win-32bit/libservo_api-32bit.dll [RL] build-win-32bit/libservo_api-32bit.a</pre>	~#		
<pre>~/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/cl6-ccitt.c [C] src/co_common.c [C] src/usbcan_util.c [C] src/usbcan_util.c [C] src/usbcan_types.c [C] src/usbcan_types.c [C] src/b_tools.c [L] build-win-32bit/libservo_api-32bit.dll [RL] build-win-32bit/libservo_api-32bit.a</pre>	~# cd Rozum-Servo-Drives-APT-master/		
<pre>~/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/col6-ccitt.c [C] src/cocommon.c [C] src/usbcan_util.c [C] src/logging.c [C] src/usbcan_types.c [C] src/b_tools.c [L] build-win-32bit/libservo_api-32bit.dll [RL] build-win-32bit/libservo_api-32bit.a</pre>			
<pre>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/co_common.c [C] src/usbcan_util.c [C] src/usbcan_util.c [C] src/usbcan_types.c [C] src/b_tools.c [L] build-win-32bit/libservo_api-32bit.dll [RL] build-win-32bit/libservo_api-32bit.a [RL] build-win-32bit/libservo_api-32bit.a</pre>			
<pre>~/Rozum-Servo-Drives-API-master/c# make -e OS=win32 [C] src/usbcan_proto.c [C] src/api.c [C] src/co_common.c [C] src/usbcan_util.c [C] src/logging.c [C] src/usbcan_types.c [C] src/b_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</pre>			
<pre>[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/usbcan_util.c [C] src/usbcan_types.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</pre>			
<pre>[C] src/api.c [C] src/crc16-ccitt.c [C] src/co_common.c [C] src/usbcan_util.c [C] src/usbcan_types.c [C] src/usbcan_types.c [C] src/b_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</pre>			
<pre>[C] src/crc16-ccitt.c [C] src/co_common.c [C] src/usbcan_util.c [C] src/usbcan_types.c [C] src/usbcan_types.c [C] src/b_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</pre>			
<pre>[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</pre>			
<pre>[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</pre>			
<pre>[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</pre>			
<pre>[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</pre>			
<pre>[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</pre>			
<pre>[L] build-win-32bit/libservo_api-32bit.dll [AR] build-win-32bit/libservo_api-32bit.a [RL] build-win-32bit/libservo_api-32bit.a</pre>			
[AR] build-win-32bit/libservo_api-32bit.a [RL] build-win-32bit/libservo_api-32bit.a			
[RL] build-win-32bit/libservo_api-32bit.a			
	~/Rozum-Servo-Drives-API-master/c#		

Apart from the rest, the command output will contain the following two files:

- a. build/libservo\_api.dll
- b. build/libservo api.a

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



17. 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 of the same response will illustrate the correct format for setting up the parameters in the specific tutorial.

18. Find out the CAN Interface ID — the identifier of the used USB-CAN dongle.

a. Open **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.

USB serial device (COM3)	×
General Port Settings Driver Details Events	
USB serial device (COM3)	
Property	
Bus reported device description	$\sim$
Value	
USB-CAN Interface	
ОК	Cancel

- 19. 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.
- 20. Select a tutorial from the tutorial folder. Before running it, make sure to replace the tutorial variables with the CAN Interface ID from **Step 18** and the CAN ID from **Step 19**.

Now that you know how to start the servo and use the tutorials, you can create your own code. For more information, refer to the <u>API documentation</u>.