

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!

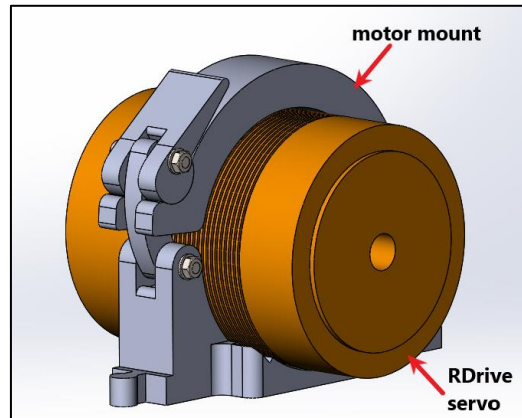


For loaded servo operation, you need to add at least one energy eater and one capacitor to the supply circuit, or use the servobox solution by Rozum Robotics. For details, refer to the [Servobox documentation](#).

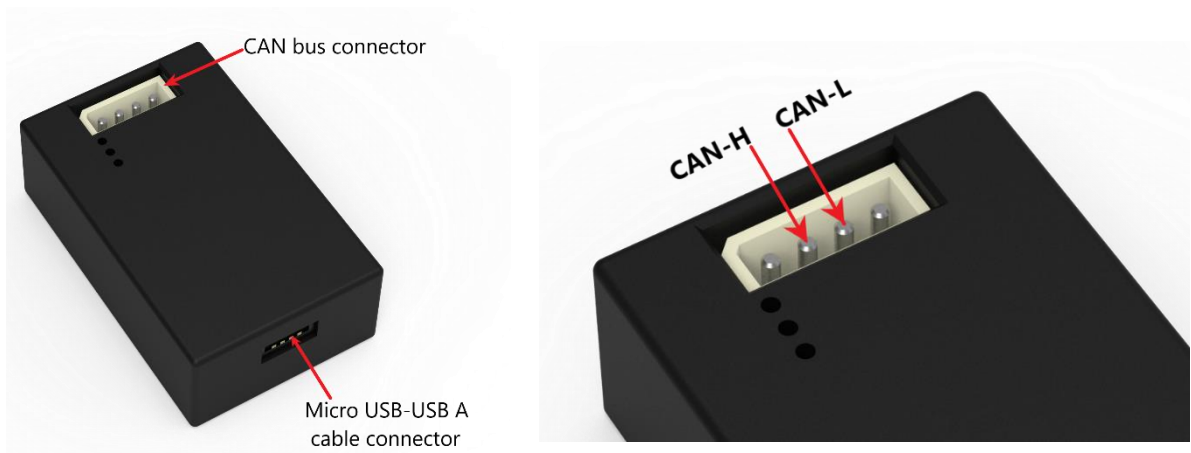
Step 2. Connecting the hardware

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_{HIGH}, and the blue one is for CAN_{LOW}.



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.

- 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 [servo API tutorials](#).

`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
```

- 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-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 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 [Python API documentation](#).

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 [the web page](#).

Installation and setup sequence

1. Install the [Servo API library](#) by running one of the following commands.

- To install the latest version, use:

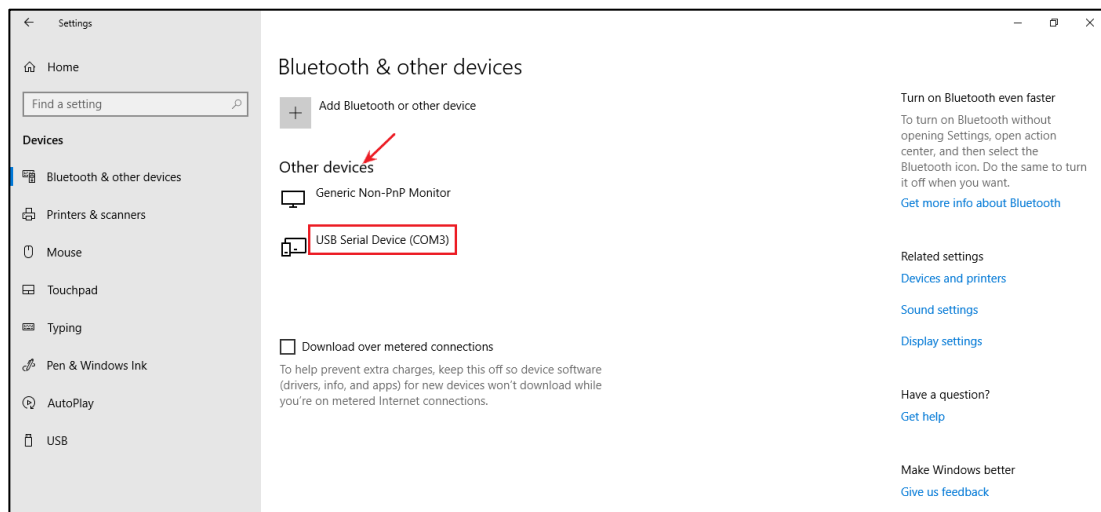
```
pip install rdrive -i https://pip.rozum.com/simple
```

- 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 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** → **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 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 [servo API tutorials](#).

`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.

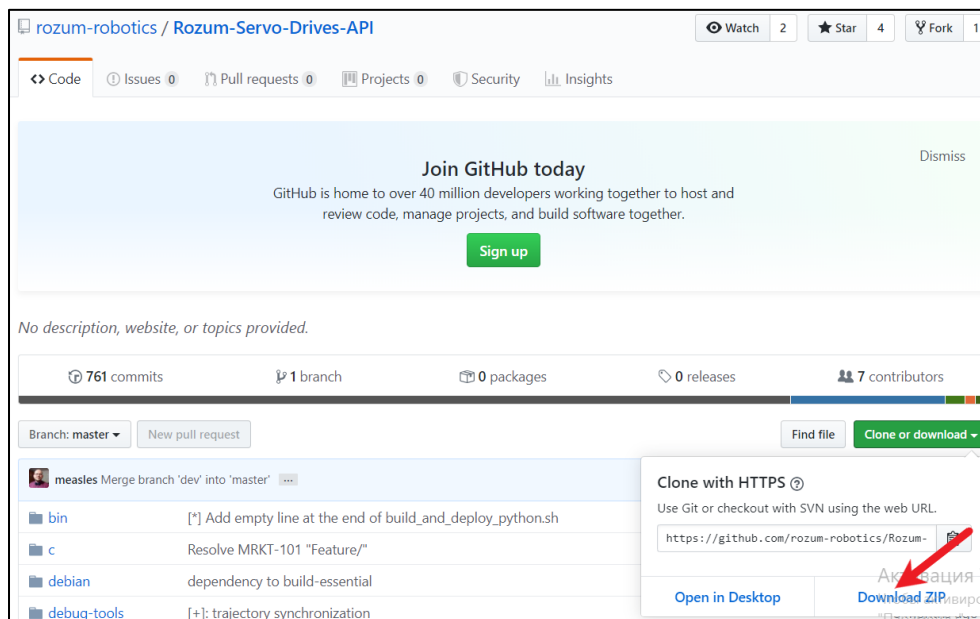
Now that you know how to start the servo and use the tutorials, you can create your own code. For more information, refer to the [Python API documentation](#).

Linux OS + C

Prerequisites

- **gcc** compiler
- **libpthread** library
- **make** package

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/c # make clean; make
rm -fr build
[C] src/rb_tools.c ...
[C] src/usbcam_types.c ...
[C] src/logging.c ...
[C] src/usbcam_util.c ...
[C] src/co_common.c ...
[C] src/crc16-ccitt.c ...
[C] src/api.c ...
[C] src/usbcam_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/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.

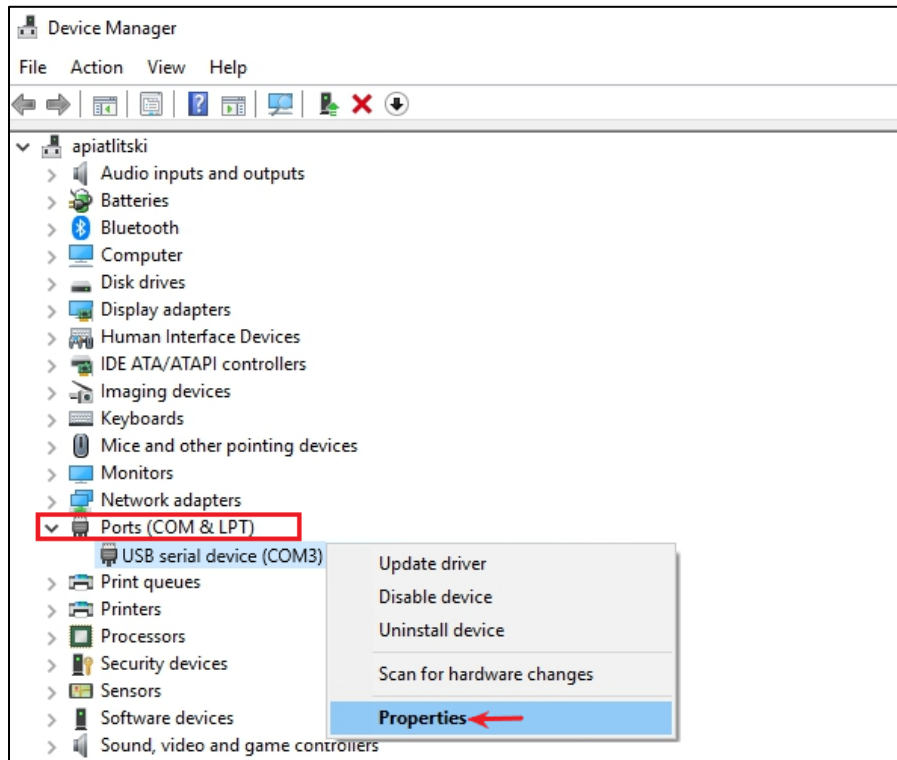
```
~/work/test/Rozum-Servo-Drives-API-master/c/tutorial # cd build/
~/work/test/Rozum-Servo-Drives-API-master/c/tutorial/build #
```

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.

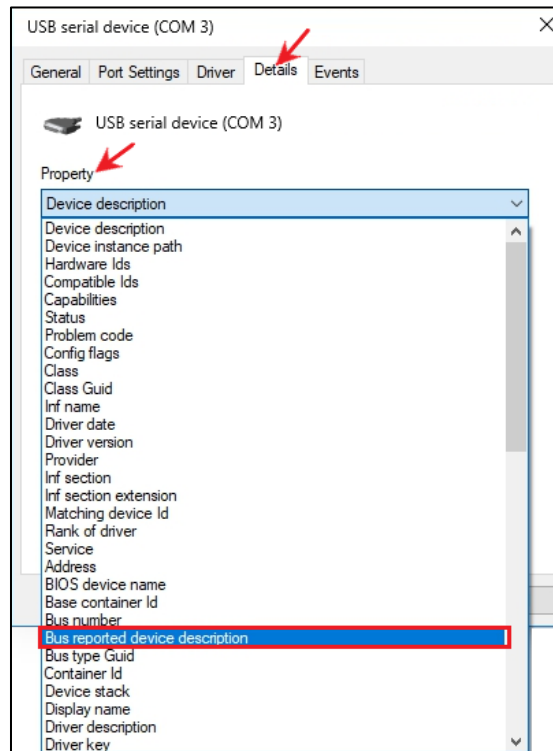
```
~/work/test/Rozum-Servo-Drives-API-master/c/tutorial/build #
./read_any_param_cache
Wrong format!
Usage: ./read_any_param_cache interface id
~/work/test/Rozum-Servo-Drives-API-master/c/tutorial/build #
```

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

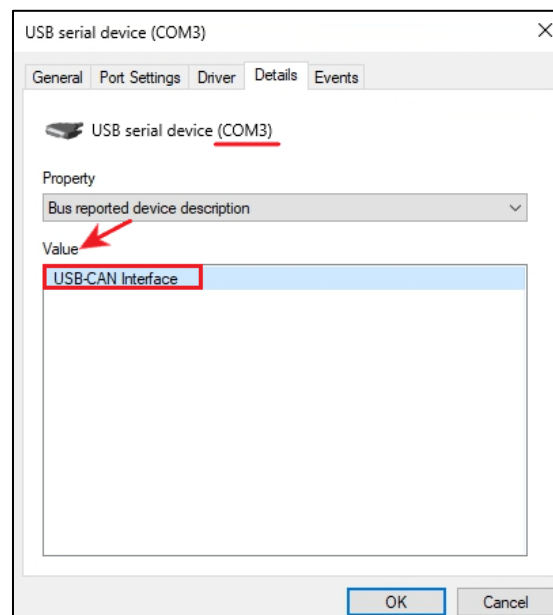
- a. Open the **Device Manager**. In the device list, open the **Ports (COM & LPT)** dropdown list. Right-click any device on the list and select **Properties** in the context menu.



- 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 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 of the connected servo.
- 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 [API documentation](#).

Windows OS (Cygwin) + C

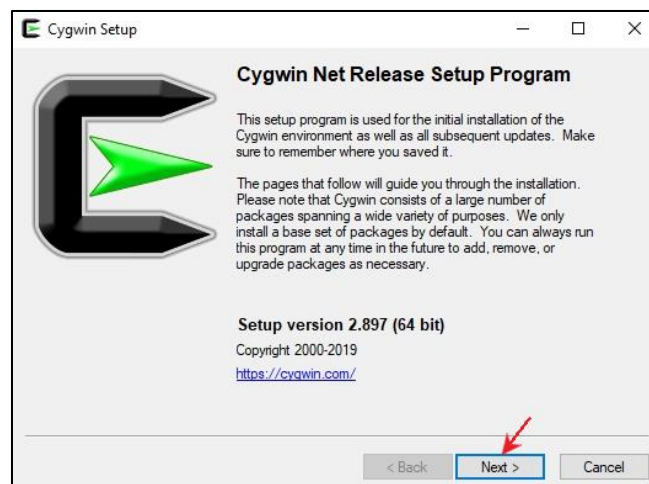
Prerequisites

- Cygwin
- **gcc** compiler
- **libpthread** library
- **make** package

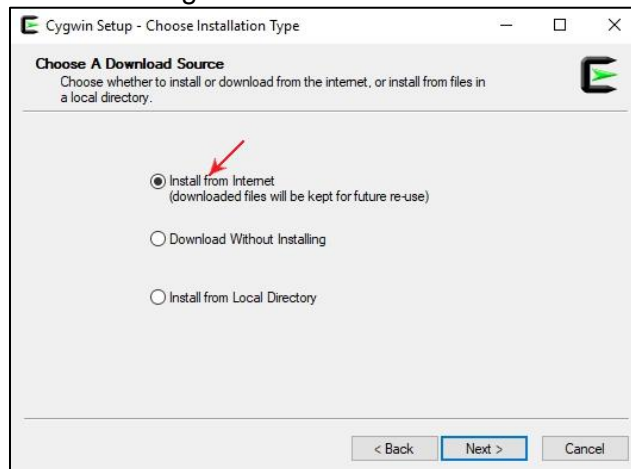


If your PC runs Windows 8 or earlier versions of the operating system, download and install a driver for the CAN-USB dongle from [the web page](#).

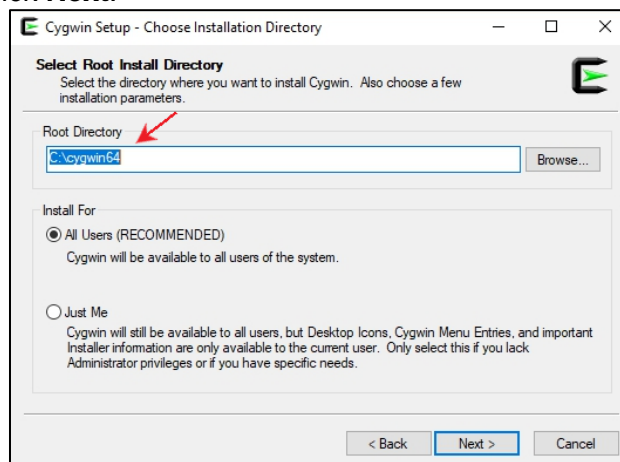
- Follow the [link](#) to download Cygwin.
- Run the executable file. In the setup screen, click **Next** to start installation.



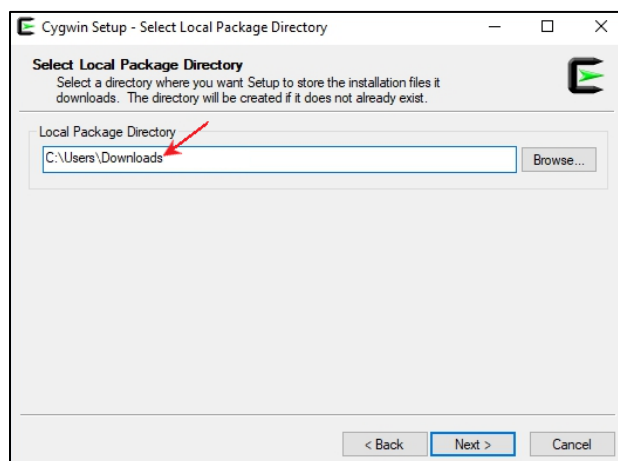
3. Choose a source for downloading and click **Next**.



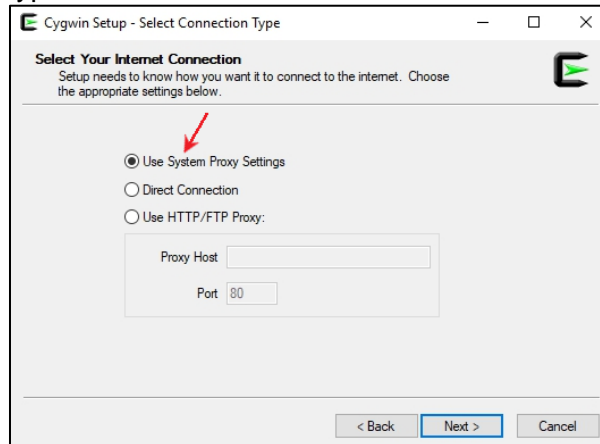
4. Select a directory for installing the files. Leave the default one or browse to set up another installation path. Click **Next**.



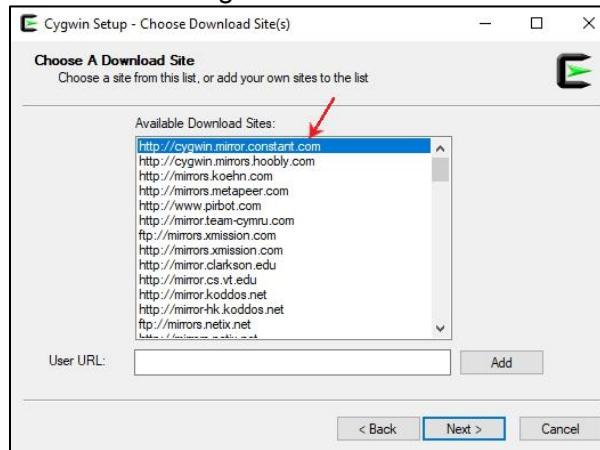
5. Select a local directory and click **Next**.



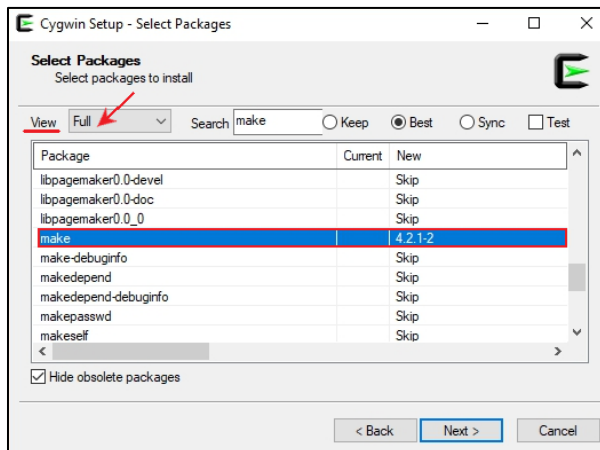
6. Select a connection type and click **Next**.

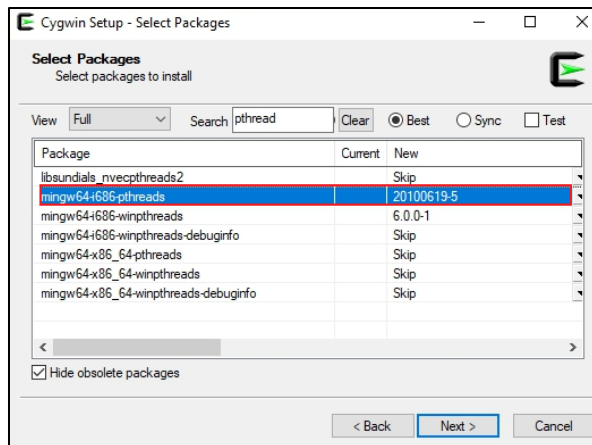
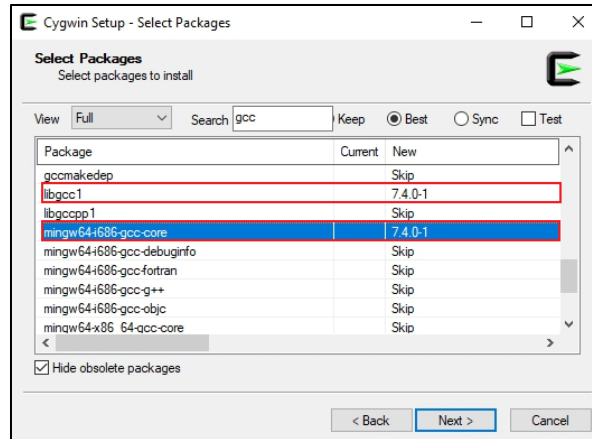


7. Choose a source site for downloading. Click **Next**.

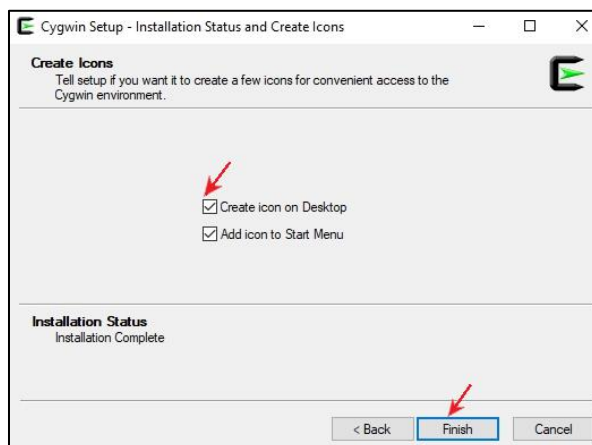


8. In the **View** field, select **Full**. Then, select **MinGW**, **make**, and **gcc** packages to install as shown in the figures below:

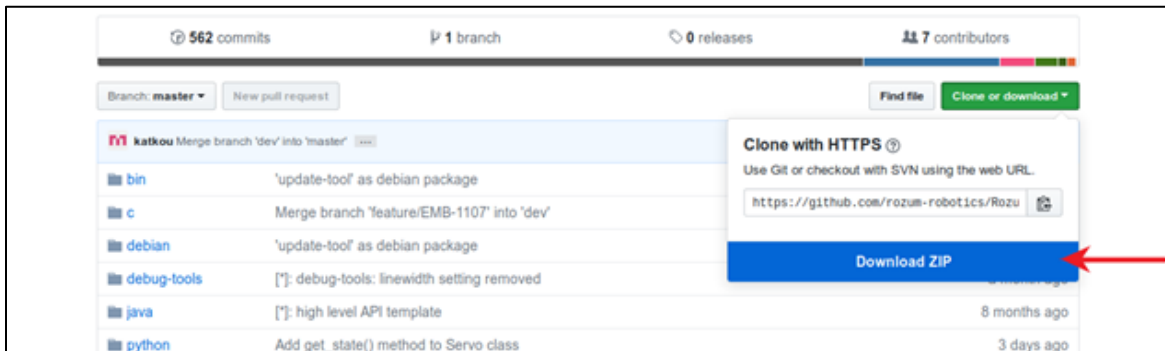




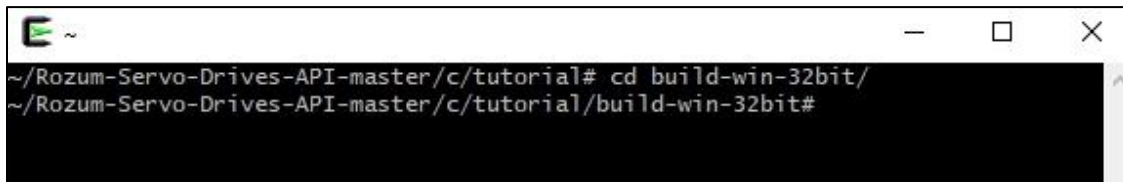
9. Wait for the setup to complete. In the final setup screen, select the checkboxes to create desktop and **Start Menu** icons. Click **Finish** to quit setup.



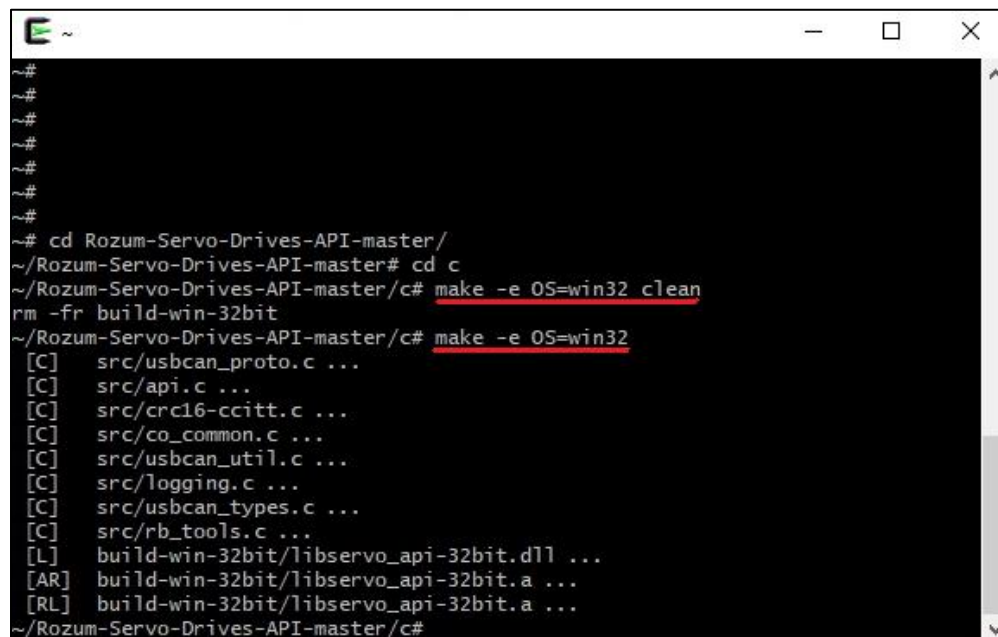
10. Download **Rozum-Servo-Drives-API** from [the GitHub repository](#).



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.



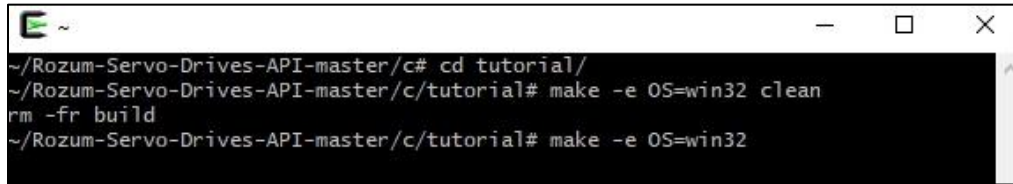
14. In the console, execute the `make clean; make` commands to compile the library. Make sure to add `-e OS = win32` (see figure below).



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

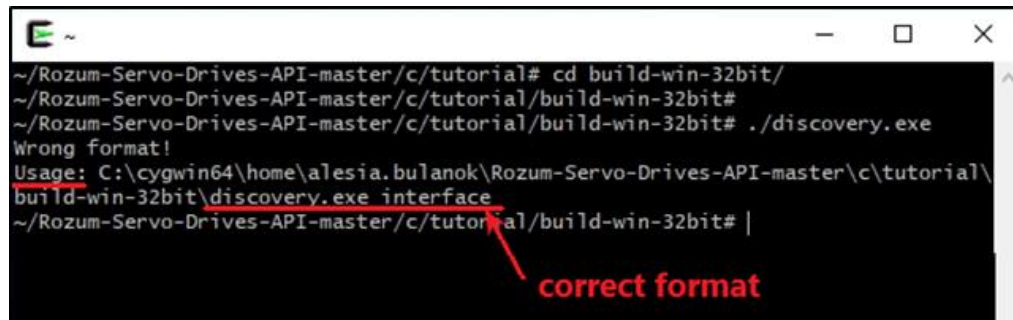
- a. `build/libservo_api.dll`
- b. `build/libservo_api.a`

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.



```
~/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
```

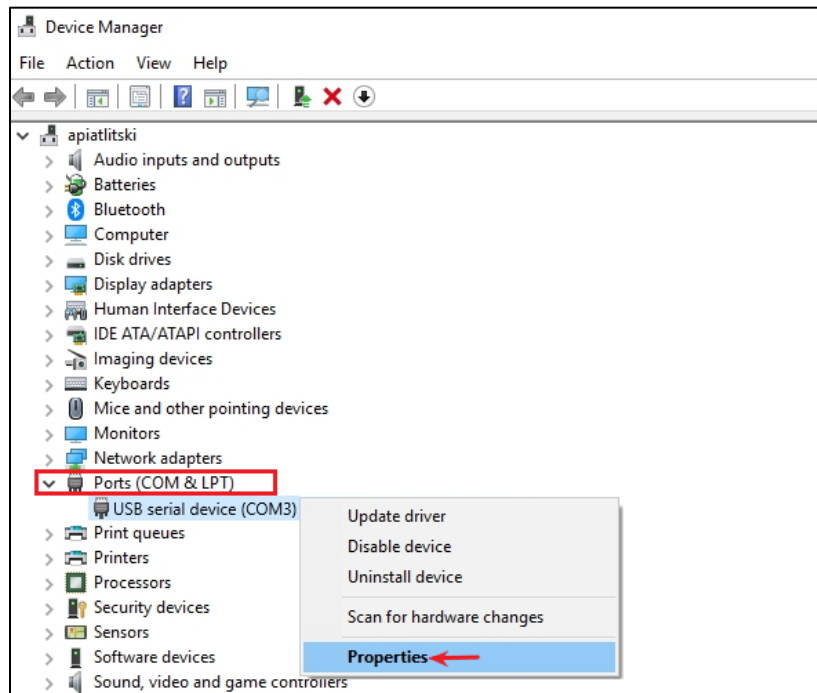
17. Navigate to the `build` folder and run an executable file of any tutorial without setting command parameters.



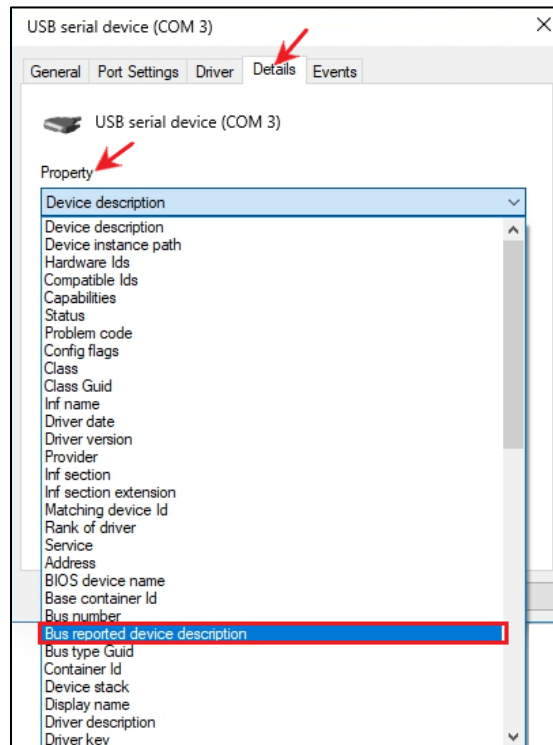
```
~/Rozum-Servo-Drives-API-master/c/tutorial# cd build-win-32bit/  
~/Rozum-Servo-Drives-API-master/c/tutorial/build-win-32bit#  
~/Rozum-Servo-Drives-API-master/c/tutorial/build-win-32bit# ./discovery.exe  
Wrong format!  
Usage: C:\cygwin64\home\alesia.bulanok\Rozum-Servo-Drives-API-master\c\tutorial\  
build-win-32bit\discovery.exe interface  
~/Rozum-Servo-Drives-API-master/c/tutorial/build-win-32bit# |  
correct format
```

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.

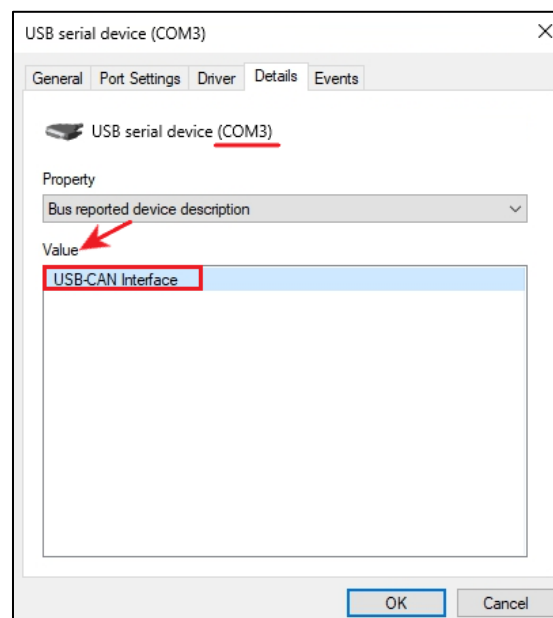
18. Find out the CAN Interface ID — the identifier of the used USB-CAN dongle.
 - a. Open **Device Manager**. In the device list, open the **Ports (COM & LPT)** dropdown list. Right-click any device on the list and select **Properties** in the context menu.



- 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.



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 [API documentation](#).