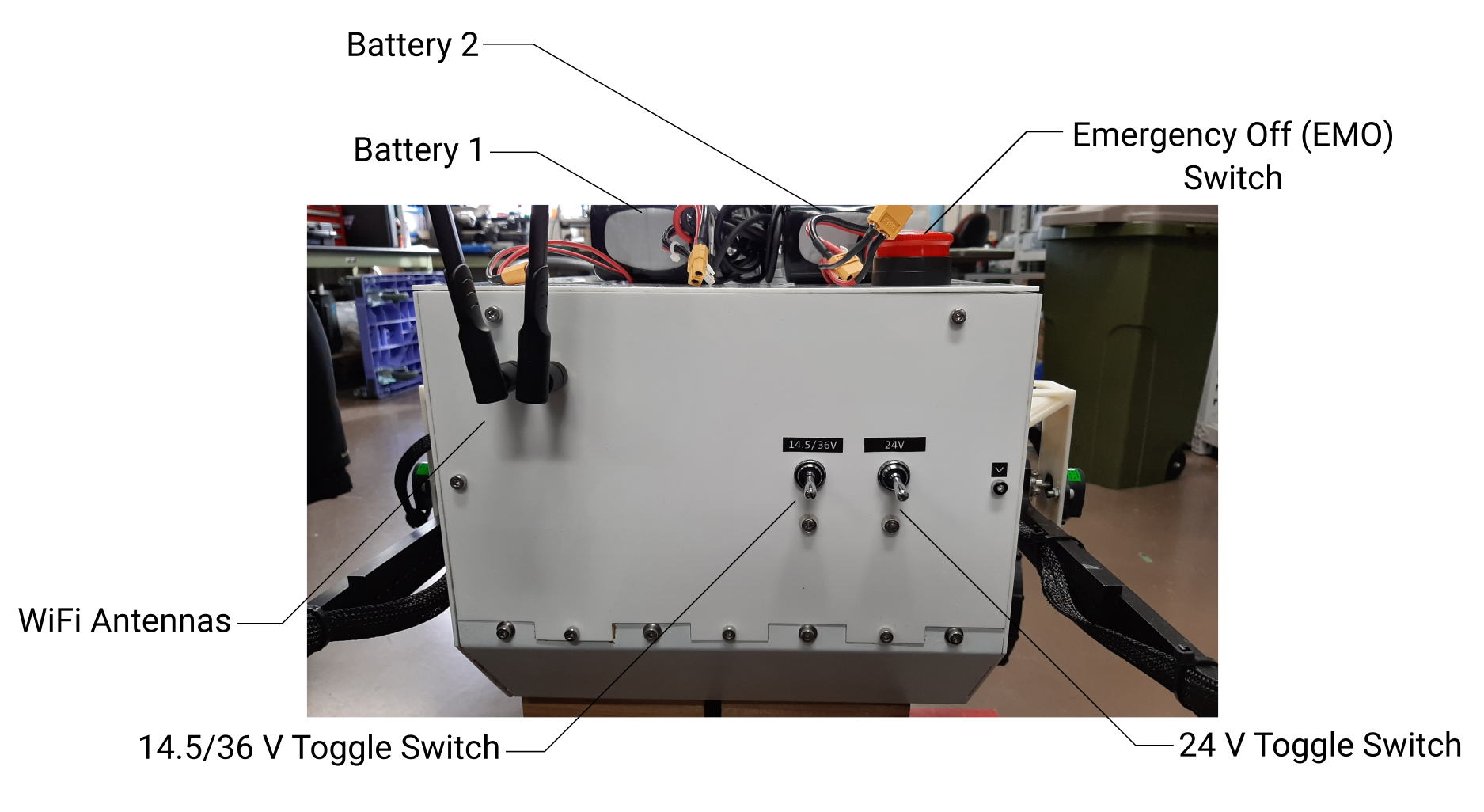
# How to.

## Powering up the rover.

The following procedure needs to be followed every time the rover is to be used.

1. Make sure both batteries are charged and properly connected.
2. Release the EMO switch. This will allow power to flow through all systems requiring 12 V or less, that includes: LCB (lights will automatically turn on), motor controllers and drivers, rocker potentiometers, and the IMU.
3. Allow some time for the capacitors in the system to fully charge before proceeding to the next step (10 seconds should be enough).
4. On the back chassis panel, turn the 24 V switch ON first. This will provide power to the steering motors. The red LED below the switch should be ON when power is provided. As before, allow time for the capacitors on the converter boxes to fully charge.
5. Turn the 14.5/36 V switch ON. This will provide power to the P-OBC and the driving motors. The red LED below the switch should also be ON.

When switching off the rover, follow this same procedure in reverse order.



***Fig. 8-1.*** *Reference image of EX1 back panel.*

## Start-up sequence.

The following procedure describes how to start and operate the EX1 rover from the GCS Main Computer. The following procedure assumes that the rover has already been powered up as indicated in Section 8.1.

### Remote access to the P-OBC.

First step will be to connect the GCS to the P-OBC via Secure Shell (SSH). Make sure both the GCS Main Computer and the P-OBC share the same network.

#### Important Information

The current SSH configuration has been defined using a key-based authorization instead of a username-password based authorization method. The public key can be found under *authorized\_keys* in the *~/.ssh* directory of the P-OBC.

The following information should be noted and kept private:

P-OBC Private IP Address: 10.240.20.170 (RANDOMWALK3\_2.4GHz WLAN Network)

P-OBC Username: roverTX2

P-OBC Password: rovertx2

SSH Port: 2020

Public/private SSH Key Pair:

Key Comment: [davidrm@dc.tohoku.ac.jp](mailto:davidrm@dc.tohoku.ac.jp)

Key Passphrase: her0\_EX1\_r0ver

*Note: The private IP address of the P-OBC may change based on the network used. If the same network is to be used often, it is highly recommended to set up a static IP address for the P-OBC.*

To connect to the P-OBC, open the SSH client application called **PuTTY**. The necessary configuration has been already saved. Under Load, save or delete a stored session, load the following sesion*: EX1 Rover P-OBC SSH Auth 10.240.20.170.*

Make sure to change the IP address under *Hostname* if a new or different network is used. It is highly recommended to change and save the new session settings so as to avoid having to make changes every time a new connection is established.

*Note: The session called EX1 Rover P-OBC 10.240.20.170 is a backup of the initial SSH configuration under a username-password-based authorization. In the event that the Key Pair fails, load this configuration instead and proceed to connect with the P-OBC username and password as indicated above.*

Once the proper configuration has been defined, click Open and a new SSH connection will automatically start.

Input the *Key Passphrase* as previously defined and you will be finally connected to EX1 P-OBC.

### Waking up EX1

Since multiple virtual terminals are necessary to start up EX1, we need to use the Screen package functionality (see Learning Resources for more information).

Start a new screen session:

$ screen

Open a new virtual terminal by clicking Ctrl+a c. Note: Check all the Screen commands with Ctrl+a ?.

First, in the new virtual terminal run the following command in order to connect the PS4 Dualshock joystick.

$ sudo ds4drv

(input P-OBC user password if required)

Connect the joystick following the instructions in Section 8.3.

Next, switch to a different virtual terminal with Ctrl+a space and run the following set of commands to access superuser (root) mode:

$ sudo su

(input P-OBC user password if required)

We need to source the setup.bash file in the root environment by:

$ source HERO\_EX1/devel/setup.bash

*Note: Superuser mode is required in order for the control signals sent when turning the lights on/off to properly work. Future students may look deeper into other ways to define how to access and control GPIO pins from the ROS environment without requiring root privileges. The solution presented here is meant to work but is not perfect.*

And finally start the rover up with:

$ roslaunch ex1\_startup wake\_up.launch

***Note: If we would like to record telemetry data, the command should be instead*** $ roslaunch ex1\_startup wake\_up.launch telemetry:=1

### Learning resources

Learn more about SSH:

1. <https://hostpresto.com/community/tutorials/how-to-connect-to-a-linux-server-using-secure-shell-ssh/>
2. <https://www.codeproject.com/Articles/497728/HowplustoplusUseplusSSHplustoplusAccessplusaplusLi>

Learn more about Screen Terminal-multiplexer.

1. <https://linuxize.com/post/how-to-use-linux-screen/>

## Connecting a PS4 Dualshock controller to the P-OBC.

### Preparation.

Make sure that PIP3 (python 3 package manager) is installed. *Note: the same would apply if you need pip for python 2. Just substitute pip3 for pip.*

$ pip3 --version

If pip3 is not installed, start by updating the package list using the following:

$ sudo apt update

$ sudo apt upgrade

Use the following command to install pip3:

$ sudo apt install python3-pip

All the dependencies will be also installed.

*Note: When installing python modules globally, it is highly recommended to install distribution-provided python modules using the apt package manager since they are tested to work properly on Ubuntu systems. You should install python modules globally using pip only if there is no package available through the package manager.*

### Installation

If not yet installed, make sure to install ds4drv, a Sony DualShock 4 userspace driver for linux (available at <http://github.com/chrippa/ds4drv>).

$ sudo pip3 install ds4drv

### Connection

#### Option 1: Connecting via bluetooth.

Run ds4drv.

$ sudo ds4drv

Hold SHARE and PS buttons in the PS4 controller until its indicator flashes. Ds4drv will automatically start searching for bluetooth devices and establish a connection.

Once ds4drv finds your controller, it will be assigned to /dev/input/js~.

#### Option 2: Connecting via USB.

Run ds4drv in hidraw mode.

$ sudo ds4drv --hidraw

Connect the PS4 controller via usb to the Jetson. Ds4drv will automatically start searching for bluetooth devices and establish a connection.

Once ds4drv finds your controller, it will be assigned to */dev/input/js~.*

### Testing

#### Option 1: Without ROS

Install and run jstest:

$ sudo apt-get install jstest-gtk

$ jstest-gtk

*Note: the -gtk option is only required if you want to see the output displayed on an independent GUI.*

#### Option 2: With ROS

First, make sure you have the joy package installed.

$ rospack find joy

If not installed, install the package. Note: make sure you use your own ROS distribution:

$ sudo apt install ros-melodic-joy

In a new terminal, launch ROS:

$ roscore

In a new terminal, tell the joy node which joystick device to use. Note: the default is js2.

$ rosparam set joy\_node/dev “/dev/input/js~”

Start the joy node

$ rosrun joy joy\_node

In a new terminal, run the following command to see the data published on the joy topic:

$ rostopic echo joy

### Reference.

#### DualShock PS4 Controller buttons and axes mapping

Table of index number of /joy.buttons:

***Table 8-1.*** *Index number of joy.buttons*

|  |  |
| --- | --- |
| Index | Button name on the controller |
| 0 | Square |
| 1 | Cross |
| 2 | Circle |
| 3 | Triangle |
| 4 | L1 |
| 5 | R1 |
| 6 | L2 |
| 7 | R2 |
| 8 | Share |
| 9 | Options |
| 10 | Left joystick press |
| 11 | Right joystick press |
| 12 | PS |
| 13 | Touchpad press |

Table of index number of /joy.axes:

***Table 8-2.*** *Index number of joy.axes.*

|  |  |
| --- | --- |
| Index | Axis name on the controller |
| 0 | Left/Right axis (left stick) |
| 1 | Up/Down axis (left stick) |
| 2 | Left/Right axis (right stick) |
| 3 | L2 |
| 4 | R2 |
| 5 | Up/Down axis (right stick) |
| 9 | Left/Right arrow buttons (+: left, -: right) |
| 10 | Up/Down arrow buttons (+: up, -: down) |
| Others | Accelerometers/Gyroscope |

### Troubleshooting

#### Running ds4drv without sudo permissions.

If you want to use ds4drv as a normal user, you need to make sure ds4drv has permission to use certain features of your system. Ds4drv uses the kernel module uinput to create input devices in userland and the module hidraw to communicate with DualShock 4 controller (when using --hidraw), but this usually requires root permissions.

You can change the permissions by copying the udev rules file to */etc/udev/rules.d* (called “*50-ds4drv.rules*”, check <https://github.com/chrippa/ds4drv/blob/master/udev/50-ds4drv.rules>):

File *50-ds4drv.rules* should contain the following lines

|  |  |
| --- | --- |
|  | KERNEL=="uinput", MODE="0666" |
|  | KERNEL=="hidraw\*", SUBSYSTEM=="hidraw", ATTRS{idVendor}=="054c", ATTRS{idProduct}=="05c4", MODE="0666" |
|  | KERNEL=="hidraw\*", SUBSYSTEM=="hidraw", KERNELS=="0005:054C:05C4.\*", MODE="0666" |
|  | KERNEL=="hidraw\*", SUBSYSTEM=="hidraw", ATTRS{idVendor}=="054c", ATTRS{idProduct}=="09cc", MODE="0666" |
|  | KERNEL=="hidraw\*", SUBSYSTEM=="hidraw", KERNELS=="0005:054C:09CC.\*", MODE="0666" |

You may have to reload your udev rules after this with:

$ sudo udevadm control --reload-rules

$ sudo udevadm trigger

#### Check if linux recognize your PS4 controller

Run the following command:

$ ls /dev/input

You will see a listing of all of your input devices. The joystick devices are referred to as js~.

#### PS4 controller is not accessible for the ROS joy node

First, list the permissions of the PS4 controller:

$ ls -l /dev/input/js0

You will see something similar to:

$ crw-rw-XX- 1 root dialout 188, 9 2019-11-01 12:04 /dev/input/js~

If XX is rw, the js device is configured properly. If XX is not rw, you need to run the following:

$ sudo chmod a+rw /dev/input/js~

#### Ds4drv hidraw mode not detecting your controller

This usually happens when the name associated with your controller is slightly different than the name defined in hidraw.py.

First, let’s check which names the controller has associated. Run the following command in a terminal. Make sure the controller is connected via USB.

$ dmesg

You will be able to read something like the following

[18798.634308] input: Sony Interactive Entertainment Wireless Controller as /devices/virtual/input/input21

Note: if this line does not appear in your terminal, make sure your controller is listed as one of the usb inputs. $ lsusb

In this case, the name associated with the device is “Sony Interactive Entertainment Wireless Controller”. Then, go to /usr/local/lib/python3.6/dist-packages/ds4drv/backends and open the file hidraw.py. Add the following line to the script (highlighted line only).

$ HID\_DEVICES = {

“Sony Interactive Entertainment Wireless Controller”: HidrawUSBDS4Device

“Sony Computer Entertainment Wireless Controller”: HidrawUSBDS4Device

“Wireless Controller”: HidrawBluetoothDS4Device}

## Accessing the P-OBC GPIO pins.

### Basic information.

There are many ways to access GPIO hardware from programs. Linux kernel uses a sysfs interface for basic GPIO control. Sysfs is a pseudo filesystem provided by the linux kernel that makes information about various kernel subsystems, hardware devices, and device drivers available in user space through virtual files. GPIO devices appear as part of sysfs.

As indicated in Section 4.4., EX1 rover LCB is defined in the following way:

* ON : 0 V (low)
* OFF: > 2.9 V (high)

The following pins from the P-OBC J21 Header Pinout are used for the LCB.

***Table 8-3****. P-OBC GPIO reference for controlling the lights.*

|  |  |  |  |
| --- | --- | --- | --- |
| Function in LCB P-OBC CTRL | P-OBC Pin | Connector Label | Sysfs GPIO Ref. |
| GND (Pin 1) | 39 | GND | n.a. |
| ON/OFF LEDs right eye (Pin 2) | 37 | GPIO26 (3.3V) | gpio388 |
| ON/OFF LEDs left eye (Pin 3) | 35 | GPIO19 | gpio395 |
| ON/OFF LED Bar (Pin 4) | 33 | GPIO13 | gpio389 |

### Quick testing of GPIO pins from userspace.

The following instructions describe how to manually activate and control the output of the GPIO pins. First of all, open a new terminal and enter root user mode.

$ sudo su

In root mode, go to the following directory:

$ cd /sys/class/gpio

If you were to check the contents of this directory, you will notice folders with names such as gpiochip216. These gpiochip directories are device names for the GPIO controllers and we won’t use them directly.

First thing we need to do to activate certain pins is to export them using the following command:

$ echo sysfs\_gpio\_number > export

Make sure to substitute sysfs\_gpio\_number for the GPIO number of the pin you would like to activate. For instance, if we would like to activate pin 37 (i.e., gpio388 in sysfs) in the J21 pinout of the P-OBC, the command would be:

$ echo 388 > export

Once exported, a new folder called gpio388 was created inside the gpio directory. From this new folder, we will be able to read and modify the functionality and properties of this GPIO pin.

Two properties are relevant to us: direction (in or out) and value (high or low, 1 or 0).

To read them use the following command within the general gpio directory:

$ cat gpio388/direction

$ cat gpio388/value

To modify its properties, use the following command:

$ echo out > gpio388/direction

$ echo 1 > gpio388/value

Once you are done testing unexport the pin:

$ echo 388 > unexport

And exit root mode:

$ exit

*Notes: In the actual code used to operate EX1, we used a C++ library containing functions that call the export, direction, and value operations of the sysfs interface. The header file can be found under the /src/ folder of the teleop package.*

## Sending data through P-OBC USB port using ROS serial library.

### Preparation.

Connect the device to the USB port or USB Hub port on the P-OBC. We need to know what the /dev path to the usb has been assigned. Note: usually it will be assigned to /dev/ttyUSB0.

There are many ways to do this. A simple way is to create and run the following bash shell file.

#!/bin/bash

#Simple file to find the dev path of peripherals.

for sysdevpath in $(find /sys/bus/usb/devices/usb\*/ -name dev); do

(

syspath="${sysdevpath%/dev}"

devname="$(udevadm info -q name -p $syspath)"

[[ "$devname" == "bus/"\* ]] && continue

eval "$(udevadm info -q property --export -p $syspath)"

[[ -z "$ID\_SERIAL" ]] && continue

echo "/dev/$devname - $ID\_SERIAL"

)

Save this file as find\_dev.sh and run it.

$ bash find\_dev.sh

Next thing we need to do is to install a serial port library called serial.

*Note: change melodic for whatever ROS distribution you are using. Documentation about the serial library can be found in* [*http://wjwwood.io/serial/doc/1.1.0/index.html*](http://wjwwood.io/serial/doc/1.1.0/index.html)

$ sudo apt install ros-melodic-serial

### Operation

The following is an example code used to send a single byte through USB.

// send\_byte.cpp program

#include <ros/ros.h>

#include <serial/serial.h>

#include <stdio.h>

#include <string>

#include <sstream>

#include <time.h>

#include <stdlib.h>

std::string port = "/dev/ttyUSB0"; //Include here the dev path of your USB device!!

uint32\_t baud = 115200; //setup baud

serial::Serial serial(port, baud, serial::Timeout::simpleTimeout(200));

int main(int argc, char\*\* argv)

{

char cmd;

ros::init(argc, argv, "send\_byte"); // ノードの宣言

while(ros::ok())

{

ros::spinOnce();

std::cout << "cmd >> ";

std::cin >> cmd;

serial.write((uint8\_t\*)(&cmd), 1); // データを送信する

ROS\_INFO("%s", &cmd); // デバッグ用

}

return 0;

}

Make sure the variable *port* contains the right */dev* path. After declaring all dependencies and executable files and building the workspace, we need to assign to the USB port the right permissions to write and read. Run the following command.

*Note: make sure you use your own* dev *path.*

$ sudo chmod a+rwx /dev/ttyUSB0

Finally, run roscore and the send\_byte node. One byte will be sent through every time you input a character into the command line.

$ cmd >> A

## Mounting/dismounting chassis panels.

In order to mount EX1 chassis panels always follow this procedure:

1. Start by mounting the side panels.
   1. Attached the rocker sensor support structure to the side panel.
   2. Pass all the cables through the support structure first and then through the center hole in the panel.
   3. Mount the panel to the chassis.
   4. Connect cables to the Rocker Interface Unit (RIU). You may help yourself with a pair of tweezers.
2. Next, pick up the top panel and without fully attaching it to the chassis, connect the EMO switch and the battery connectors.
3. Without tightening any of the screws that attach the top panel to the chassis, attach the NAVMAST to the top panel and pass all the cables through the front side opening. Connect the cables to their respective boards.
4. Bring the front panel closer to the chassis so that the LED Bar cable can be also passed through and connected to the LCB. Do not screw the front panel to the chassis yet. Leave the front panel placed somewhere securely.
5. Next, tighten the nuts of the front and back screws of the top panel.
6. Fasten the screws of the front panel.
7. Connect the switches on the back panel to their respective connectors as well as both red LED lights to the PSU.
8. Last, fasten the screws of the back panel to the chassis.

*NOTE: For the dismounting operation simply follow the previous steps in the reverse order.*

**Important!** Make sure not to attempt to loosen or tighten the screws marked with the symbol shown in Fig. 8-2. These screws are “fake”. Fake screws were glued to the chassis panels as a result of the impossibility to tighten them (the chassis frame was not properly threaded and post-assembly insertion nuts sold by MISUMI for this specific aluminum extrusion frame are currently discontinued).



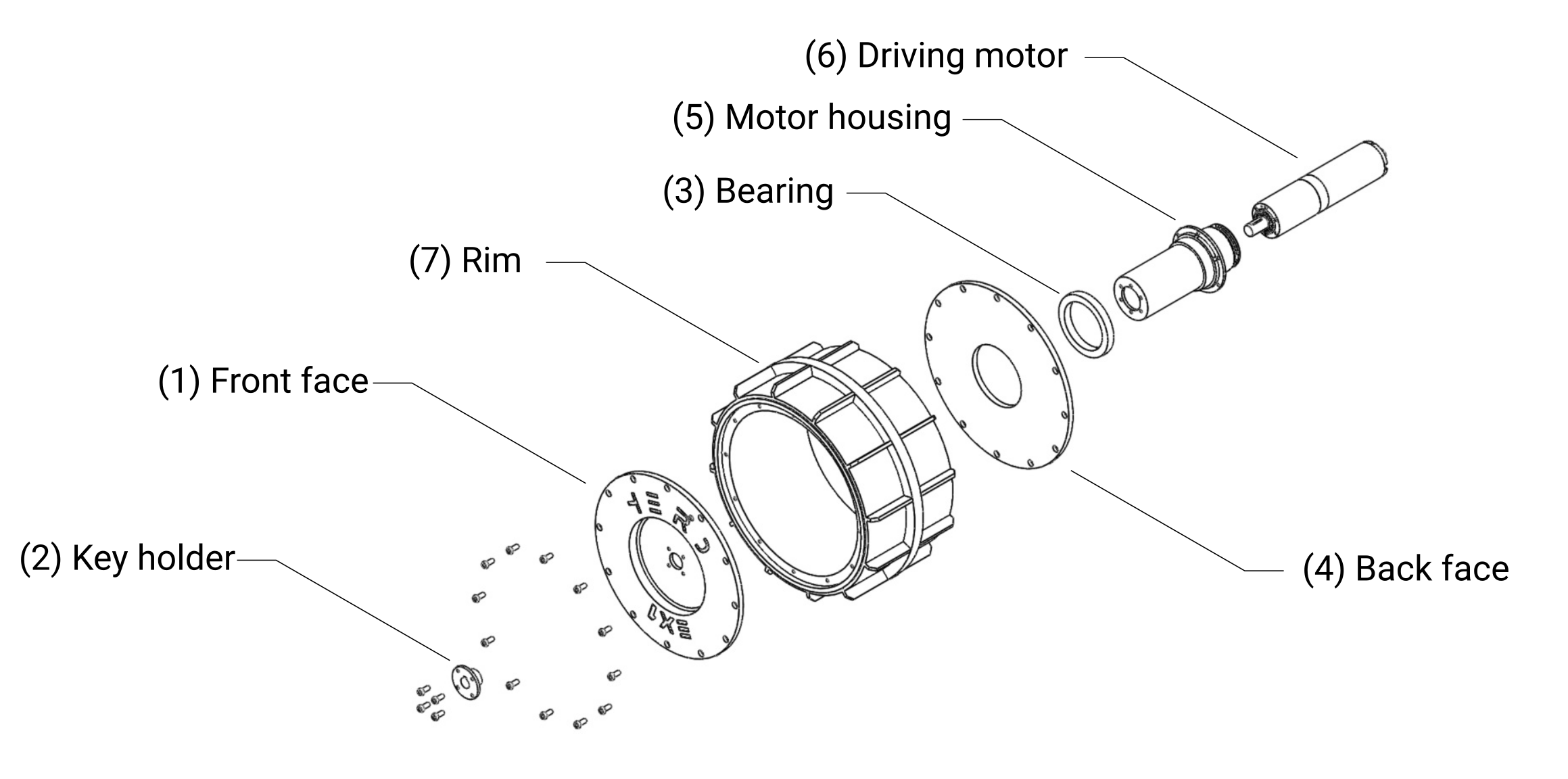
***Fig. 8-2.*** *Fake screw sign.*

## Low-Travel Suspension and wheel assembly process.

The following step-by-step guide describes the best procedure to assemble and disassemble the LTS subsystem of the EX1 rover.

#### Wheel

1. Get the wheel front face (see (1) in Fig. 8-3 for reference) mounted onto the wheel rim. [ 12 x M3x10 screws] (*NOTE: Make sure not to use a very high torque when fastening and loosening screws in order not to detach the nuts from the inside of the rim.*)
2. Attach the small metal key holder (Fig. 8-3 (2)) to the front face of the wheel. [ 4 x M3x10, 4x M3 Nuts].
3. Attach the bearing (Fig. 8-3 (3)) to the back face of the wheel (Fig. 8-3 (4)). [ Hammer needed] (*NOTE: Make sure both the back face of the wheel outer surface and the bearing outer plane are flat.*)
4. Attach the motor housing (Fig. 8-3 (5)) to the wheel knuckle (Fig. 8-4 (7)). [ Hammer may be needed, 6 x M3x12] (*NOTE: Make sure to tighten the safety nut at the other end of the housing. Make sure to use threadlocker on screws.*)
5. Attach the back face of the wheel with the bearing to the motor housing. [ Hammer needed] (*NOTE: Make sure you only hit the metal parts of the bearing with the hammer in order not to damage the plastic parts of the wheel.*)
6. Attach the driving motor (Fig. 8-3 (6)) into the housing. [6 x M3x8] (*NOTE: Make sure motor cables are pointing up or slightly to one side to facilitate wiring afterwards*.)
7. Attach the front face and rim of the wheel (Fig. 8-3 (7)) to its back face. [12 x M3x10] (*NOTE: Make sure not to use a very high torque when fastening and loosening screws in order not to detach the nuts from the inside of the rim.*)



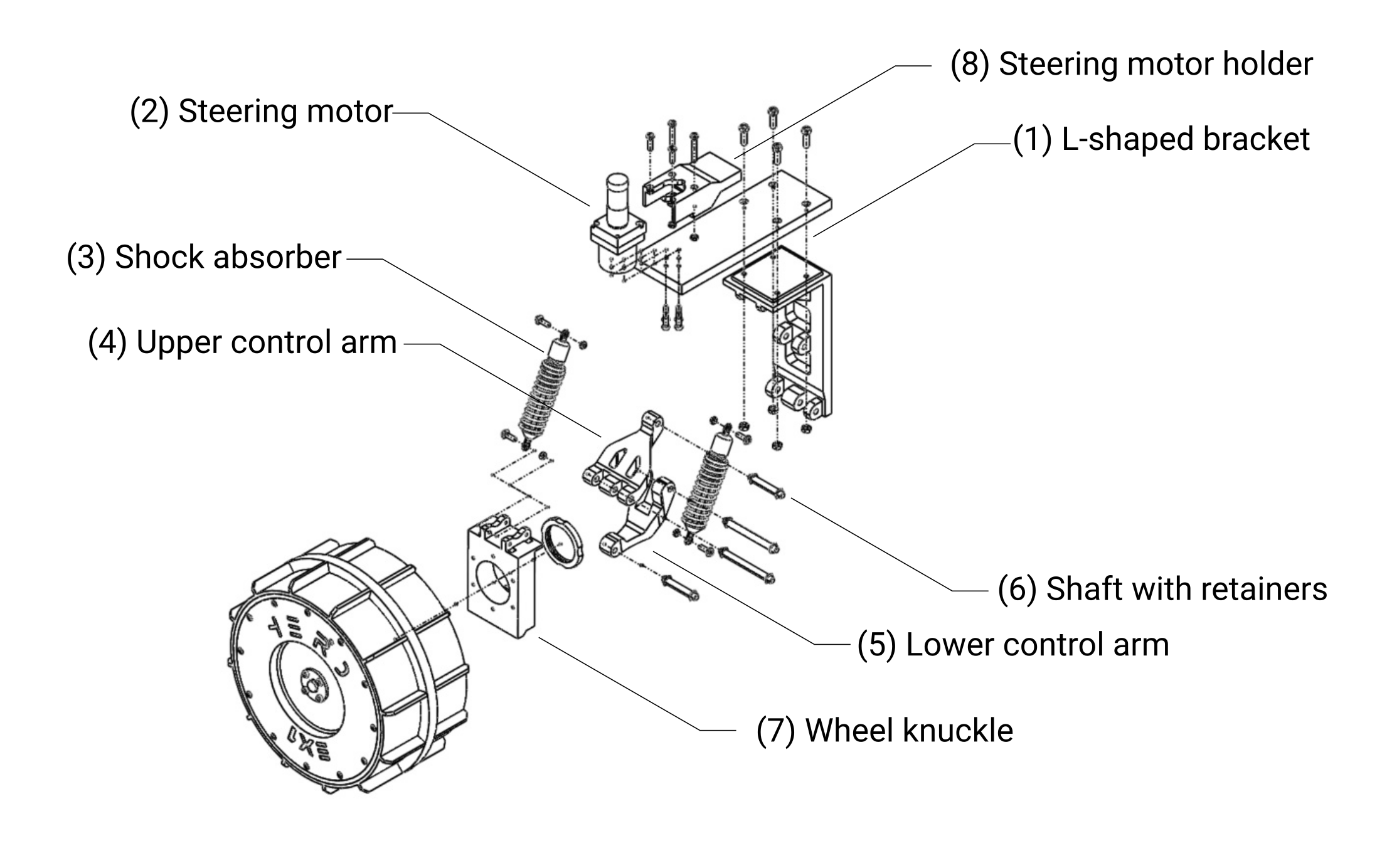
***Fig. 8-3.*** *EX1 wheel exploded view.*

#### Low-travel suspension

1. Assembly the L-shaped bracket (Fig. 8-4 (1)) of the low-travel suspension, i.e., the top part that connects to the geardhead of the steering motor (Fig. 8-4 (2)) and the attachment part for the shock absorbers (Fig. 8-4 (3)) and the upper and lower control arms (Fig. 8-4 (4) and (5), respectively).
2. Add one single retainer to all the shafts (Fig. 8-4 (6)).
3. Mount both upper and lower control arms to the L-shaped part of the suspension. (*NOTE: To add the retainers make sure you always press downwards (take advantage of gravity) and the best tool to use is a hammer with a rubber head. Do not hit the retainer but press against it.*).
4. Connect upper and lower control arms to the wheel knuckle (Fig. 8-4 (7)).
5. Attach the two shock absorbers to the L-shaped part of the suspension and the wheel knuckle. [4x M3x18/15] (*NOTE: Recommended to first attach shocks to L-shaped part.*)
6. Pre-load the springs.

#### Integration on the chassis

1. Mount steering motor holder (i.e., part that connects the low-travel suspension to the rocker)(Fig. 8-4 (8)) onto the rocker. [2 x M4x30]
2. Attach the low-travel suspension to the steering motor gear head. [ 4 x M3x12]
3. Attach the whole assembly to the motor holder. [2 x M3x25, 2 x M3 Nuts, 2x M3x18].



***Fig. 8-4.*** *EX1 Low-travel suspension (LTS) exploded view.*

## EX1 Teleoperation

The following table describes the different button combinations currently supported by EX1 functionality. A visual reference is provided in Fig. 8-5.

***Table 8-3.*** *EX1 Joystick button combination.*

|  |  |
| --- | --- |
| Functionality | Button combination |
| Drive forward | Left joystick UP |
| Drive backward | Left joystick DOWN |
| Steer left | Right joystick LEFT |
| Steer right | Right joystick RIGHT |
| Max. driving speed selection | L1 |
| Point-turn-mode ON/OFF | Triangle + R1 |
| Point-turn left | L2 (full throttle) |
| Point-turn right | R2 (full throttle) |
| Lights ON/OFF | Cross |

**Important notes** regarding teleoperation:

* Maximum driving speed selection loops from 0.1 to 1 m/s in steps of 0.1 m/s. Initially speed is dissable as a safety measure. L1 has to be pressed once before the rover can initiate movement.
* When point-turn mode is active, normal driving and steering capabilities are dissabled.
* Point-turn is currently performed at a fixed speed of 0.2 m/s limited by software. This limit can be changed at any given time from the P-OBC main OS.



***Fig. 8-5.*** *EX1 joystick reference for teleoperation.*