Introduction

This project contains the core packages for the ROS2-based AGV platform IOTBot: the communication package 'iotbot_shield' for the Siemens SIMATIC IOT2050 expansion board, 'iotbot_interface' for custom msg- and srv-types and 'iotbot_motion' with nodes for calculating the kinematics. Currently differential drive and mecanum drive, each in combination with two motor variants. All four variations can be launched bundled by their respective launch-files, as followed.

Table of Contents

Electrical components and functions

Charging the robot

The round socket intended for charging is located in the rear area of the cover plate. The rubber cover should always be closed when the battery is in use to protect the charging socket from the entry of foreign particles. Note that the robot cannot move during the charging process! A new enable signal is necessary after every charging process.

Battery switch

The switch is a toggle switch with a latching function. It is not used for booting and lowering the platform, but for de-energising the circuit when the platform is switched off or transported for a longer period of time. Even if the robot is left unattended for some time (e.g. lunch break), it is recommended to release it via this switch. The switch is located in the rear area of the cover plate and has a rectangular shape.

On button

This button is used directly for booting the computer. It does not lock when pressed and must be pressed for a few seconds to prevent it from being accidentally switched on. An indicator for sufficiently long pressing is the headlights and the IOT2050. It is located near the charging socket at the rear of the platform as a round black button.

Turning off your robot

To shut down the robot safely, it is necessary to connect to it. Either directly using a screen via DisplayPort and a keyboard or via a remote connection using ssh (see here). There, the computer can be shut down with the Linux command `sudo shutdown now`. After a successful shutdown, the STAT LED of the IOT2050 lights up red. Alternatively, the entire robot can be switched off by hitting the battery switch.

Emergency stop

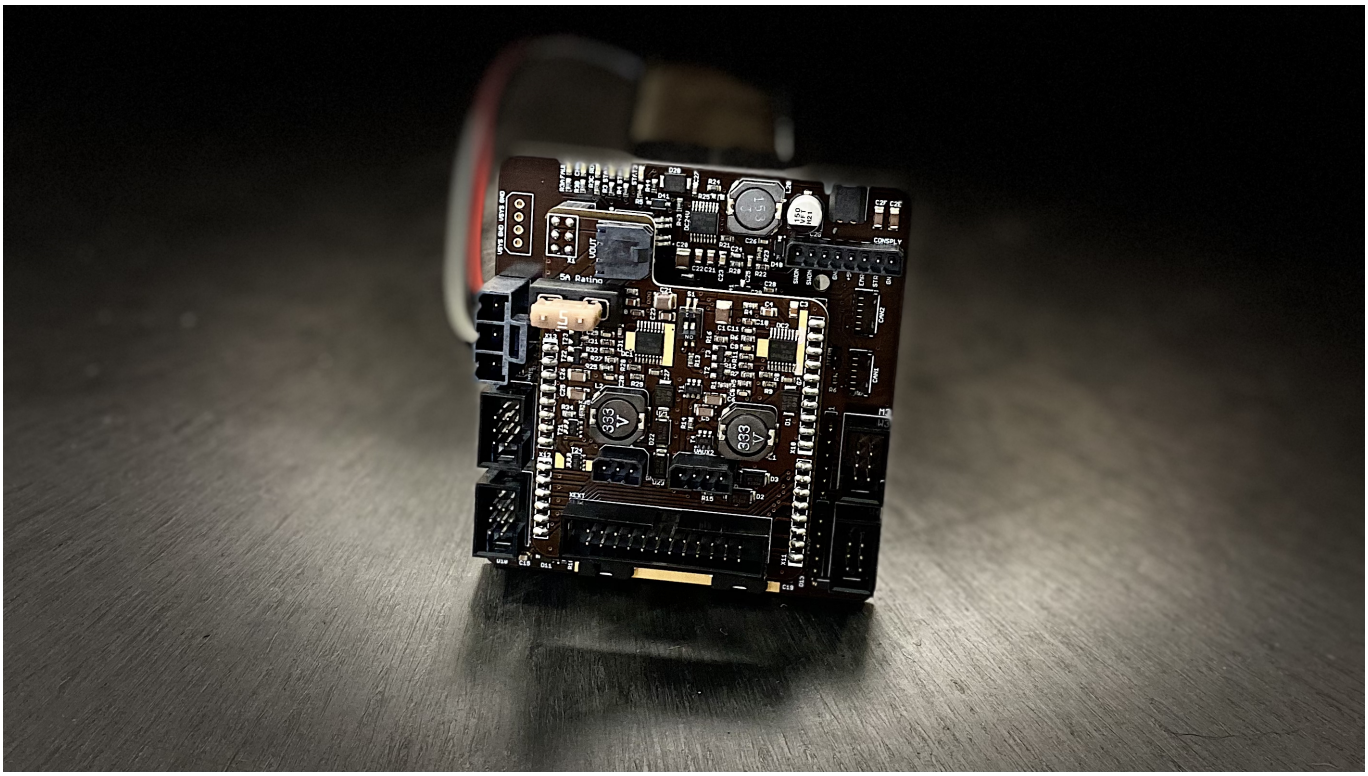
With this robot, the emergency stop is not primarily for the protection of the human operator, but rather for the protection of the machine in the event of unexpected movements due to programming errors or false signals from sensors. To activate it, press the button as vertically as possible from above. To unlock, turn it to the right until it disengages. Before unlocking, make sure that the source of danger has been eliminated. The robot does not start automatically after unlocking. Operation must first be enabled again.

Extension-Shield and onboard Sensors: iotbot_shield_node

The ROS2 node for communication between the microprocessor and the IOT2050 publishes on the following topics and receives these services:

Description	**Topic**	**Message type**
Controller Values	/joy	sensor_msgs/msg/joy
Taget Motorspeed values	/iotbot/rpm	iotbot_interface/msg/RotationSpeed
Current Motorspeed values	/iotbot/rpm/return	iotbot_interface/msg/RotationSpeed
Inertial Measurement Unit	/iotbot/imu	sensor_msgs/msg/Imu
Distance measurements	/iotbot/tof	std_msgs/msg/Float32MultiArray
Battery Voltage	/iotbot/battery	iotbot_interface/msg/Battery

Description	**Service**	**Message type**
Determination of light pattern	iotbot/srv/send_lighting	iotbot_interface/src/SendLighting
Enable signal for driving	iotbot/srv/send_enable	iotbot_interface/src/SendEnable



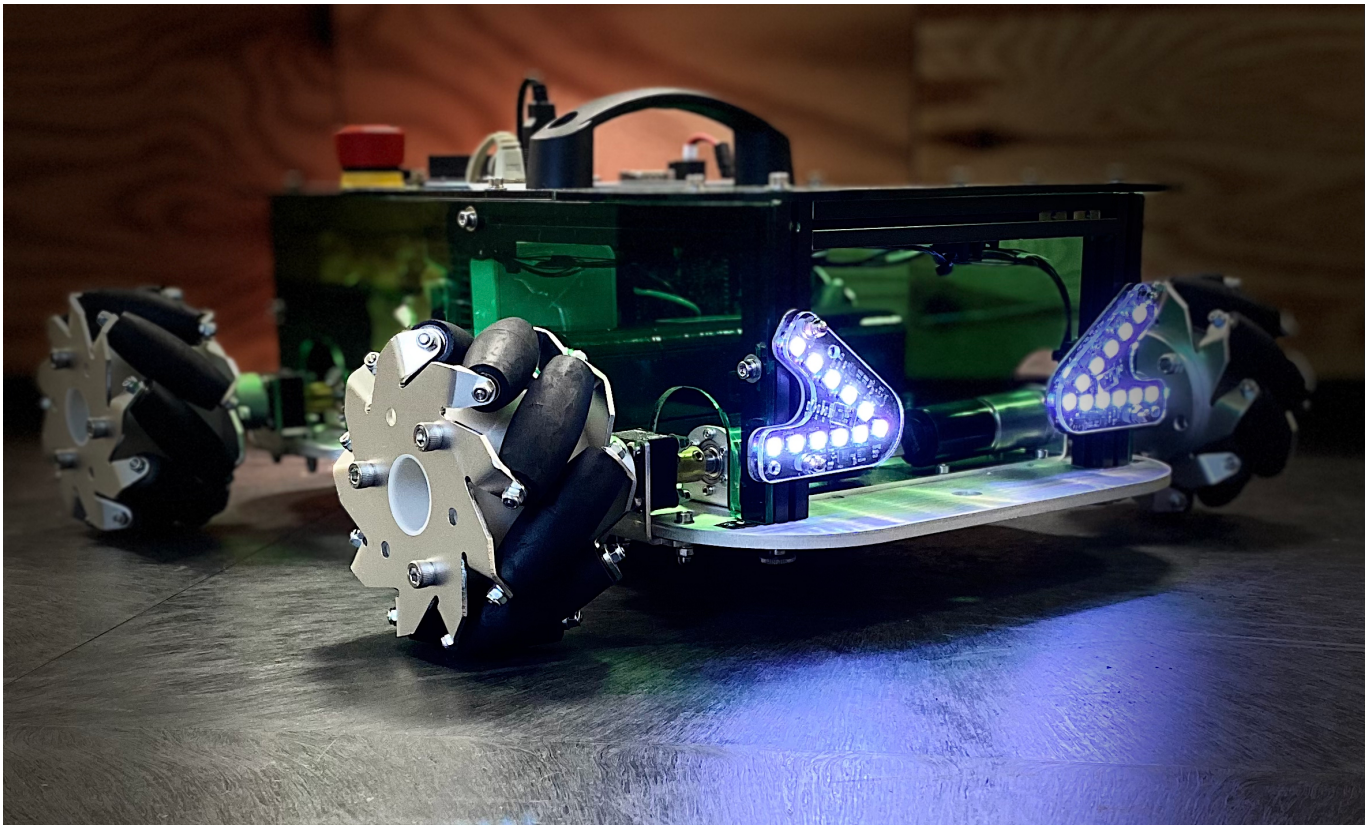
Kinematic variations: iotbot_motion_X_node

The ROS2 nodes for calculating the motion of the robot publish on the following topics:

Description	Topic	Message type
Taget Motorspeed values	/iotbot/rpm	iotbot_interface/msg/RotationSpeed
Taget Velocity vector	/cmd_vel	geometry_msgs/msg/Twist

Skid Drive: iotbot_motion_differential_node

Mecanum Drive: iotbot_motion_mecanum_node



Mechanical components and functions

Depending on the variant you chose at the time of building, your platform is equipped with either Faulhaber or Pololu motors. See the description of the respective drive systems in the further course of the chapter.

Wheel change

To change the wheel, do not loosen the screws on the axle couplings. The couplings are made of brass, which is why the threads cut in them are very sensitive. To change between the wheel concepts , use an Allen key or similar to be able to loosen the screws and/or to apply a leverage to the propeller screws.

Change Mecanum wheel to "off-road" wheel:

- Loosening the grub screw in the wheel hub
- Pulling the wheel off the drive shaft
- Mounting the "offroad" wheel
- Insert the Allen key into the hole in the propeller driver on the wheel.
- Lock the driver while holding the wheel.
- Check if it is firmly seated.

Change from "off-road" wheel to Mecanum wheel:

- Like the change described above, but backwards.
- Attention: Push the Mecanum wheel in as far as it will go but leave a small gap so that the wheel does not rub against the suspension.
- Arrangement: when looking from above, all the rollers of the individual wheels should point towards the centre of the robot so that a cross could be formed.

Differential drive:

- Remove front wheels
- Convert the rear wheels to "off-wheel" wheels.
- Insert the front castor through the hole in the middle and lock it with the wing nut.

Removing the upper frame

The top frame has to be removed if, for example, you want to make changes to the electronics in the interior or connect sensors, cameras or similar to the USB hub. Since unscrewing the top plate is too time-consuming and not recommended, knurled screws have been used on the underside of the robot to screw the bottom plate to the chassis. These screws can be loosened and tightened without any additional tools. Always use this option with caution and do not tighten the screws with pliers, as this can destroy the thread in the chassis.

How to open the frame

The frame must be opened, for example, if slot nuts or hammer head screws are to be inserted into the frame, as the ones already inserted are not enough. It cannot be opened at any point! Precondition: Upper frame removed as described in the previous section.

Proceed as follows:

- Remove the cables & carefully cut open the cable ties.
- Remove the Spotlight boards so they do not get damaged.
- Remove the side cover plates. Use an 2.5mm Allen key.
- Locate the profile with the hole in a side groove

- Use a flathead screwdriver to loosen the screw in the hole
- Turn the frame apart by slightly loosen one profile after the other and push them out of the grooves in the othe profiles. Attention: Slot stones falling out!
- Insert the sliding blocks or square-head screws and reassemble the frame.

Installation Instructions for Prerequirements 🗂️

The following preparations must be made for the operation of the robot.

Installing Debian Buster on the IOT2050 🖥️

The operating system for the IOT2050 is to be downloaded as SIMATIC IOT2050 SD-Card example image . After a successful download, a tool such as Win32DiskImager Win32DiskImager can be used to burn this image onto an SD card of at least 16GB.

The SD card can then be inserted into the device and initially switched on. For the first boot process, we recommend a direct connection via monitor and keyboard. The default username and password are `root`. Further installation instructions can be taken from the installation process.

> ***NOTE:*** Under no circumstances should the IOT2050 be switched on without a bootable SD card!

After successfully installing and setting up the operating system, the following commands should be used to update the system and install git. The last one enables the donwload of this repository:

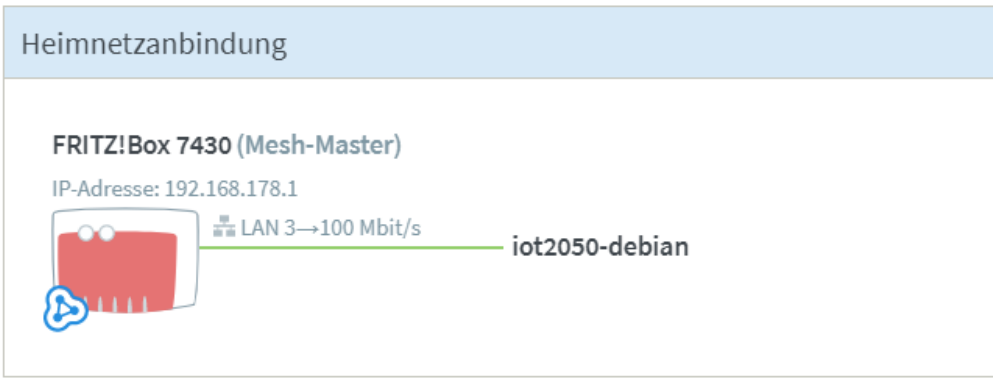
```
$ sudo apt update
$ sudo apt upgrade

$ sudo apt install git
$ git clone ...
```

Connecting to your robot via SSH

To access your robot remotely, the IOT2050 must be on a network with your machine. This can be done via a router or a direct connection between your device and the IOT2050.

When connecting via a router, both devices must be connected to it. Select the Ethernet port P2, which is configured with DHCP by default. This allows the IOT2050 to receive an IP from your router dynamically:



For the direct connection between your device and the IOT2050, select the Ethernet interface P1, which has the static IP address 192.168.200.1 by default. Here, a static IP address in the subnet 192.168.200 must also be assigned in the settings of the network adapter of your host computer.

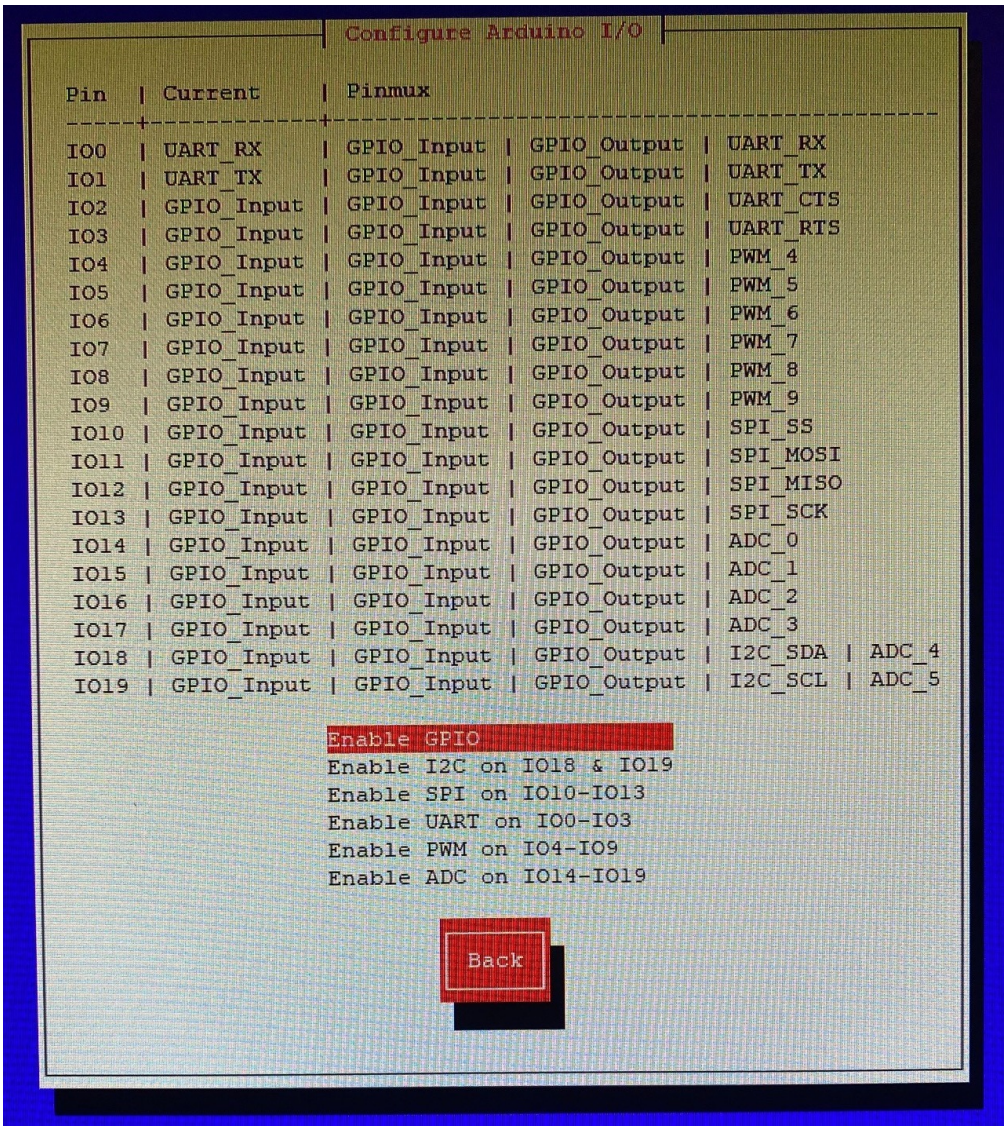
In both cases, a terminal of the IOT2050 can be reached via its IP address. To do this, open a command line. CMD in the search bar on Windows 10, Ctrl+Alt+T on Ubuntu. The SSH tools are already available in most operating systems, so you can access your robot via the user name and IP address of the IOT2050 as follows:

```
$ ssh root@[IP-ADDRESS]
```

Setting up the UART Interface 🖥️

To allow communication between the extension board and the computer, the corresponding pins of the serial interface must be configured accordingly.Use the terminal command `iot2050setup` to access the settings of the IOT2050. Navigate to the GPIO settings under the menu option `Configure Arduino I/O`.

Activate the communication via the menu option `Enable UART on IO0-IO3` and follow the instructions there:



Installing the Docker-Engine 🐋

The following installation instructions are taken from the official docker docs documentation. We recommend an installing through the use of their repository.

Before you install Docker Engine for the first time on a new host machine, you need to set up the Docker repository. Afterward, you can install and update Docker from the repository. Update the `apt` package index and install packages to allow `apt` to use a repository over HTTPS:

```
$ sudo apt-get update
$ sudo apt-get install \
    ca-certificates \
    curl \
    gnupg \
    lsb-release
```

Add Docker's official GPG key:

```
$ curl -fsSL https://download.docker.com/linux/debian/gpg | sudo gpg --
dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg
```

Use the following command to set up the stable repository:

```
$ echo \
  "deb [arch=$(dpkg --print-architecture) signed-
by=/usr/share/keyrings/docker-archive-keyring.gpg]
https://download.docker.com/linux/debian \
  $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list
> /dev/null
```

Update the `apt` package index, and install the latest version of Docker Engine and containerd:

```
$ sudo apt-get update
$ sudo apt-get install docker-ce docker-ce-cli containerd.io
```

Verify that Docker Engine is installed correctly by running the hello-world image:

```
$ sudo docker run hello-world
```

Setting up your Joystick 🎮

A joystick can be used to operate the IOTBot. For this purpose, the IOT2050 must be extended by a Debian-compatible Bluetooth stick. PlayStation® 4 and PlayStation® 5 controllers were used, which are interpreted identically in their operating interface. The initial start-up of a Bluetooth controller follows.

Start the Bluetooth controller in the operating system:

```
$ bluetoothctl
```

Set up the controller and prepare for scanning:

```
$ agent on
$ default-agent
$ power on
$ discoverable on
$ pairable on
```

Put the PlayStation® Controller into connection mode by pressing the Share and PS buttons simultaneously. Rapid flashing indicates the status.

Now start the scanning process:

```
$ scan on
```

The connection process so far should look like this. Your joystick is now recognised as a wireless controller. Copy the MAC address of the device for the rest of the procedure.

```
root@iot2050-debian:~# bluetoothctl
Agent registered
[bluetooth] # agent on
Agent is already registered
[bluetooth] # power on
Changing the power on succeeded
[bluetooth] # discoverable on
Changing discoverable on succeeded
[CHG] Controller XX:XX:XX:XX:XX:XX Discoverable: yes
[bluetooth] # pairable on
```

```
Discovery started
[CHG] Controller XX:XX:XX:XX:XX:XX Discovering:yes
[NEW] Device XX:XX:XX:XX:XX:XX Wireless Controller
[bluetooth] #
```

Connect the controller using the following commands and its MAC address. If needed, press the PlayStation button again when the light signals stop flashing.

```
$ pair XX:XX:XX:XX:XX:XX
$ trust XX:XX:XX:XX:XX:XX
$ connect XX:XX:XX:XX:XX:XX
$ exit
```

NOTE: These steps are only required once at the very beginning. From now on, when the PS button is pressed, the joystick should automatically connect to the IOT2050 once it has successfully booted up. These operations are only then necessary again if the controller has been connected to another device in the meantime.

Software Building Instructions 🛠️

Building the Docker-Image

```
$ docker build . -t --cpuset-cpus 0-2 iotbot_basis
```

Creating the Docker-Container

```
$ docker run --name iotbot_basis --privileged -v /dev:/dev --network host -
-group-add dialout iotbot_basis
```

Usage Instructions 🚙

Starting and entering your container

```
$ docker start iotbot_basis
$ docker exec -it iotbot_basis bash
```

Launching the nodes

```
$ ros2 launch /home/iotbot_ws/src/iotbot/iotbot_launch/skid_performance.py
```

Controlling the robot

A controller can be requested to connect by pressing a specific button once. For the recommended controllers, it is the symbol between the axes. To operate the Robot, the following buttons and axes of the controller are assigned as follows:

Axis	DS5	Idle position	Value range	function
[0]	Joystick L: left & right	0.0	1.0 to -1.0	Steering
[1]	Joystick L: up & down	0.0	1.0 to -1.0	not in use
[2]	L2	1.0	1.0 to -1.0	not in use
[3]	Joystick R: left & right	0.0	1.0 to -1.0	not in use
[4]	Joystick R: up & down	0.0	1.0 to -1.0	Throttle
[5]	R2	1.0	1.0 to -1.0	not in use
[6]	D-Pad: left & right	0.0	1.0 to -1.0	not in use
[7]	D-Pad: up & down	0.0	1.0 to -1.0	not in use

Button	DS5	Idle position	Value range	function
[0]	Cross	0	0 or 1	not in use
[1]	Cyrcle	0	0 or 1	not in use
[2]	Triangle	0	0 or 1	not in use
[3]	Square	0	0 or 1	Light pattern: Parking light
[4]	L1	0	0 or 1	Light pattern: Turning left
[5]	R1	0	0 or 1	Light pattern: Turning right
[6]	L2	0	0 or 1	not in use

Button	DS5	Idle position	Value range	function
[7]	R2	0	0 or 1	not in use
[8]	SHARE	0	0 or 1	Light pattern: Circular light
[9]	OPTIONS	0	0 or 1	Light pattern: Siren
[10]	PS	0	0 or 1	Enable driving
[11]	L3	0	0 or 1	not in use
[12]	R3	0	0 or 1	not in use
[13]	Map	0	0 or 1	Light pattern: Warning light

Let the docker container auto-start

```
$ docker start iotbot_basis
$ docker exec -it iotbot_basis bash
$ echo “ros2 launch
/home/iotbot_ws/src/iotbot/iotbot_launch/skid_performance.py" >> ~/.bashrc
$ exit
```

ROS1 Compatibility 🤖

ROS1 Noetic Node
ROS1 Noetic virtuall Joystick

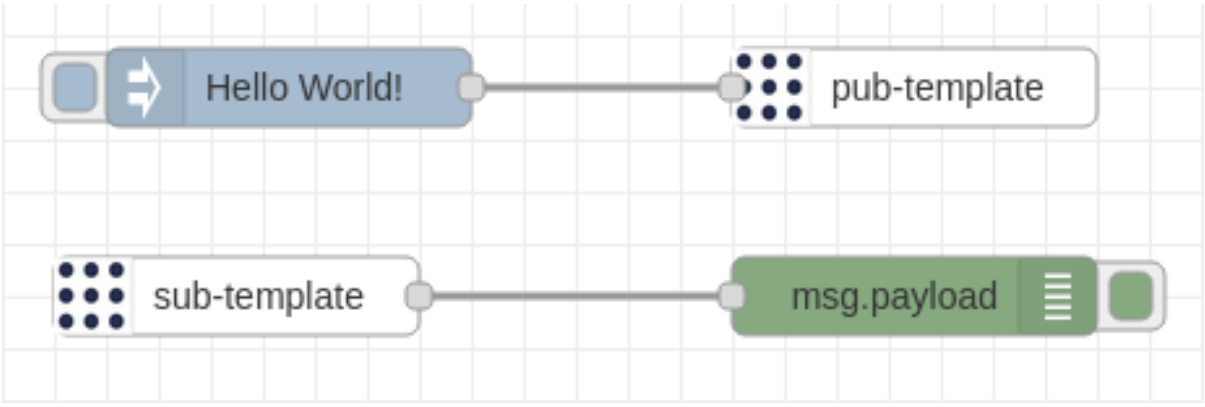
NodeRED extension 🎛️

Modules were programmed for communication between NodeRED and ROS2. The beta version is being tested as part of the Mission2Mars event and was provided as a pre-configured Docker container. The following subchapters thus describe a test status of these extensions and the handling of the modules with examples. NodeRED's browser-based IDE can be reached by the IOT2050's IP-Address on the Port 1880, e.g.:

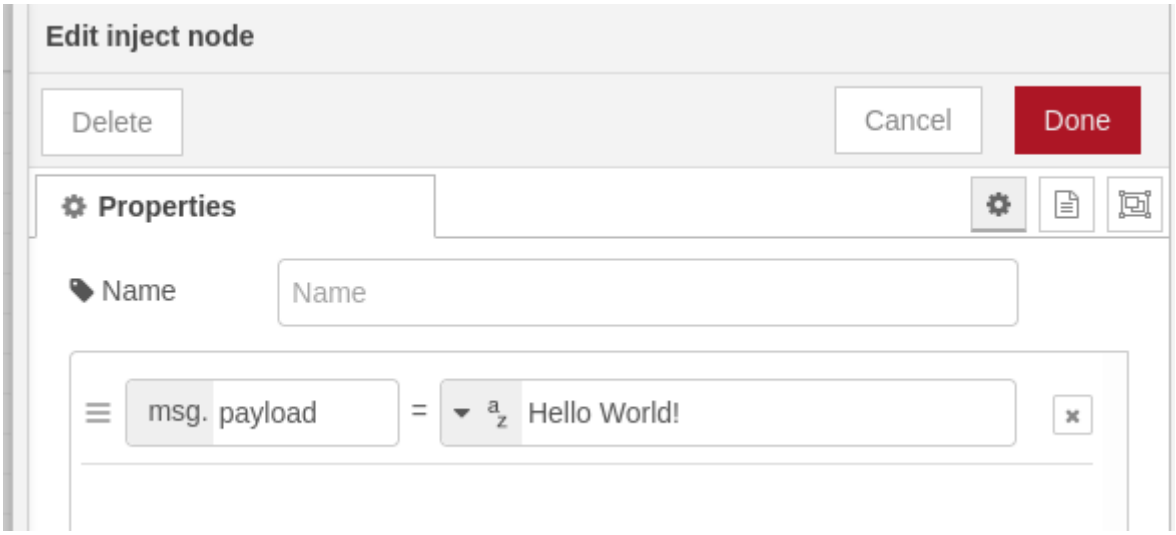
```
192.168.1.237:1880/
```

Example I: Sending a String and receiving it back

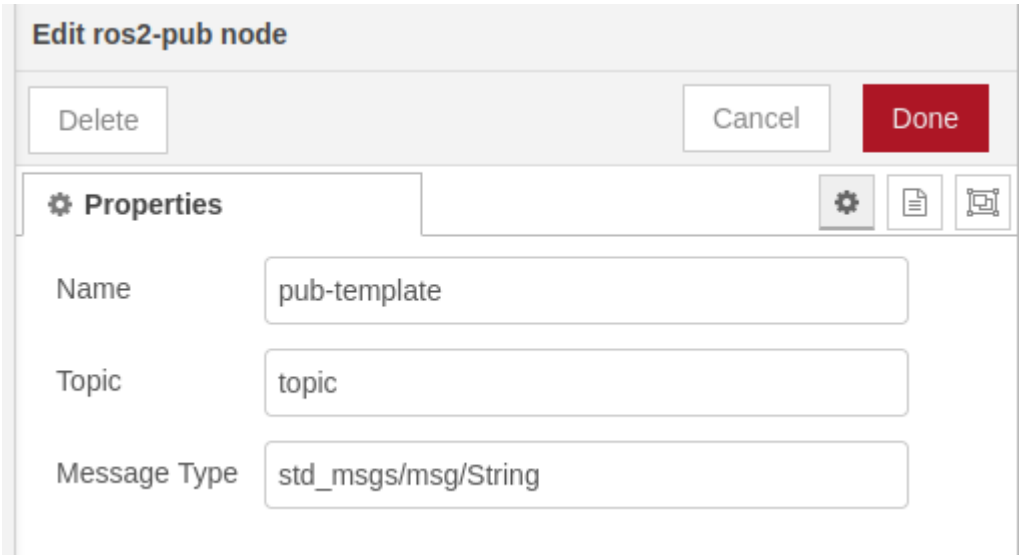
For a first introduction to the operation of the new ROS2 communication modes, we will first send a string into the ROS2 framework and then directly read it back in and display it:



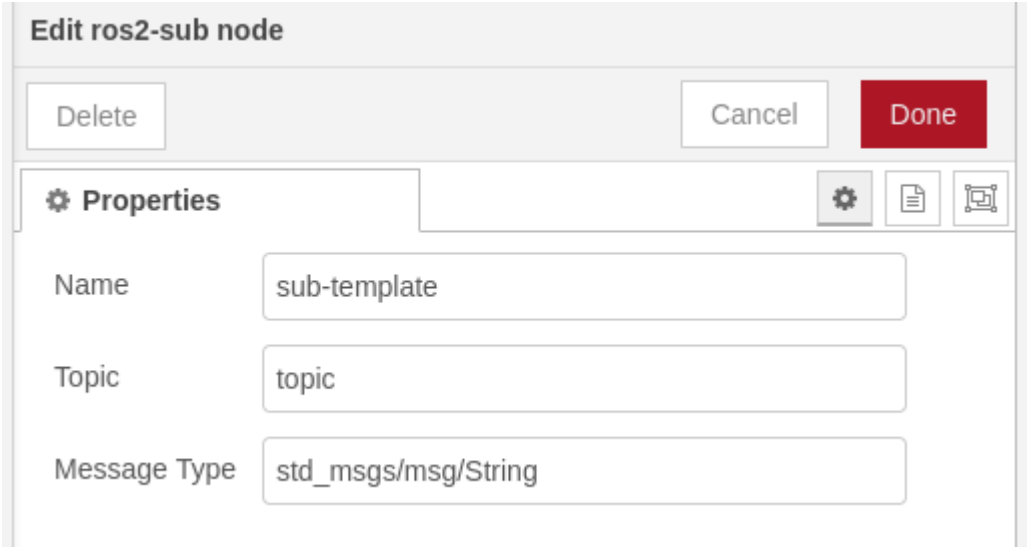
The Inject block is configured to output the string "Hello World!" as soon as its button is pressed:



To publish the data in the ROS2 framework, we create a ROS2-Pub block, which is configured as follows. Please note that the topic must be entered and the corresponding message type must be added:



Now that our string is available on the ROS2 network, we can read it back into NodeRED using a ROS2 Sub-Block:

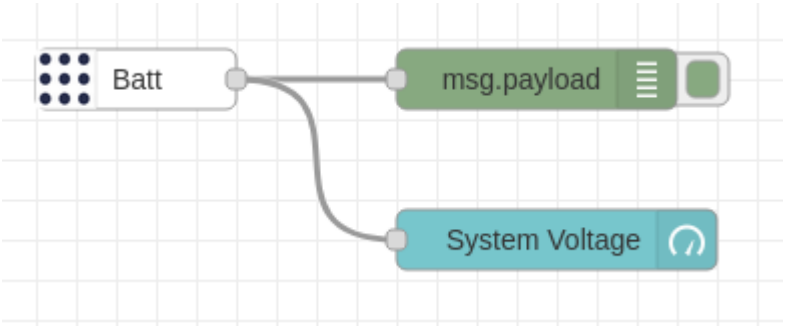


Finally, we use a debug block to be able to display the messages that have been read in:

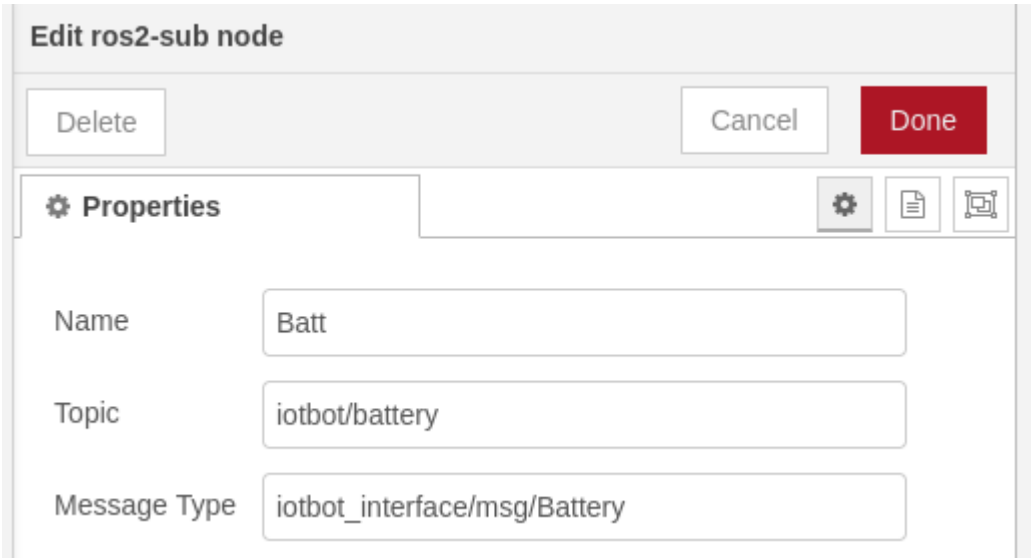
```
▾ object
  data: "Hello World!"
```

Example II: Displaying the robots battery voltage

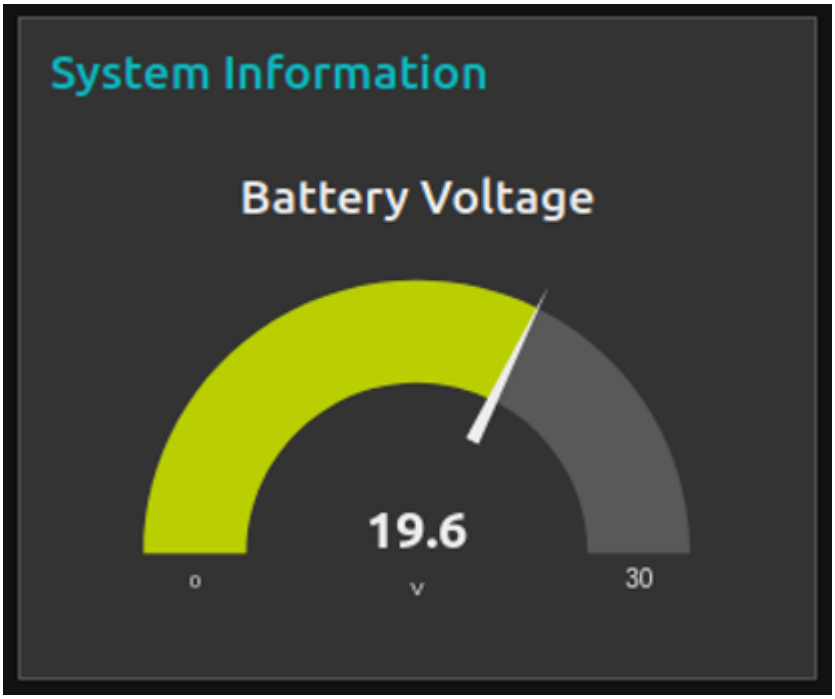
NodeRED provides us with a wide range of extensions. One of them is the creation of a graphical user interface. With the help of the module node-red-dashboard, which can be installed via the palette manager under settings, we can visualise operating data:



In this case, the battery voltage was read in with a ROS2 Sub-Block configured in this way:

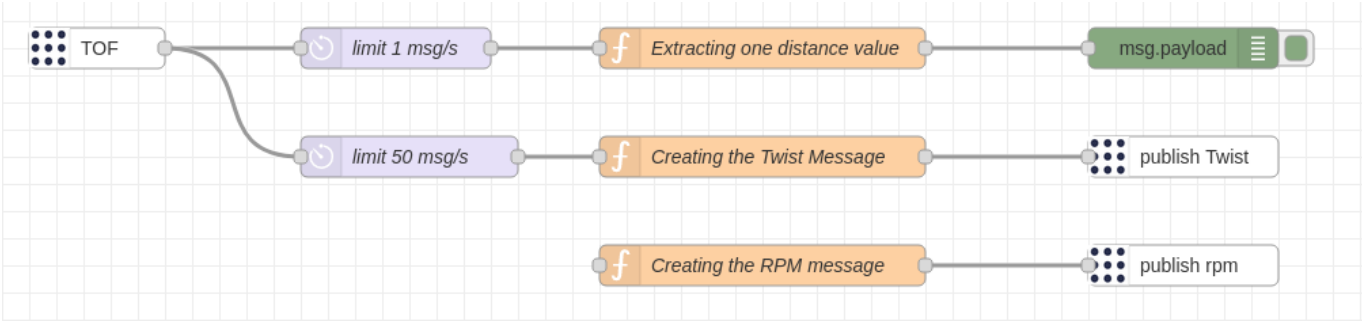


The GUI can be reached with an addition to the IDE address (e.g. `192.168.1.237:1880/ui`) and in our case displays the measurement data as a tachometer:



Example III: Dont hit the Wall! (please)

After the two previous examples, we are now ready to give our IOTBot a rather important feature - it should no longer be able to drive into walls (or us). To achieve this, we read the ToF distance measurements from ROS2, take the value for one of the front sensors and operate the robot with ROS2-messages from NodeRED:



Let's take a look at the code in the top function block and find out how we can seperate one measurement. We create a ne variable called `newMsg`, assign one element of the incoming payload to it and output it as the new message:

```
var newMsg = { payload: msg.payload.data["0"] };
return newMsg;
```

With this experience we can now create a so-called twist message, where we always check if our robot is not too close to an obstacle. Twist messages define the speed in direction and rotation, so we can reduce the speed if something is in front of our robot:

```
throttle = 0.0;

if (msg.payload.data["0"] < 0.2) {
    throttle = 0.0;
    }
    else {
    throttle = 0.2;
    }

var twist = {
    linear : {
        x : throttle,
        y : 0.0,
        z : 0.0,
    },
    angular : {
        x : 0.0,
        y : 0.0,
        z : 0.0
    }
    };
msg.payload = twist;
return msg;
```

Congratulations, with a few lines of code you have saved the life of a robot. Another way to control the robot is to give it the speeds for its four motors directly, as here. Try changing the code so that the distance is also checked here:

```
var rpm = { front_left_rpm: 20,
            front_right_rpm: 20,
            rear_left_rpm: 20,
            rear_right_rpm: 20 }
msg.payload = rpm;
return msg;
```

Next Steps 🚶

✅ Initial structure: separate nodes for the shield and the kinematic versions, the second inherits from a parent class.

✅ Creating a package for own message types: iotbot_interface.

✅ Implementation of the serial communication in iotbot_shield_node in a second thread.

✅ Calculation of the kinematic models in the iotbot_motion_X_nodes.

✅ Implementation of the IMU in the iotbot_shield_node.

✅ Implementation of the Time of Flight sensors in the iotbot_shield_node.

✅ Programming of calling all possible light patterns from the MCU in the iotbot_shield_node.

✅ Providing a service to call light signals from other nodes.

🔲 Integration of an operating concept for the PlayStation® 4 and PlayStation® 5 controller.

🔲 Integration and organization of all aplications as microservices within docker-compose for easy system-startup.

Related ongoing and future work 👷

🎛️ Completed NodeRED-Modules for general ROS2-Connectivity and IOTBot-related examples.

🕹️ GUI for operation via the keyboard respectively the ROS1 version.

🏏 Using IMU-data for controling a leveling-plattform including LIDAR scanner.

📷 Joystick-controlled tilt-plattform for POV cameras.

🕵️ ROS2-ready OpenMV H7 Extension with examples.

🛰️ ROS2-ready UWB-RTLS Extensionboard. Keep your eyes open for a swarm of IOTBots!

Safety instructions

Read this document carefully before using the product for the first time and make sure that no safety-related questions remain unanswered. Use this document only as an aid for expansions and handling of the robot. Pay attention to the warnings and symbols described below in order to understand potential dangers for the user and the device and to avoid accidents.

Limits of use

Risk of damage to the robot platform and/or objects in the surroundings due to operation in an unsuitable environment!

Do not operate the robotic platform

- not in areas with holes and/or stairs.
- not on uneven, wet and/or soft surfaces.
- if on raised platforms (e.g. table, pedestal, stage), then only on the rack provided for this purpose.
- not in outdoor areas.
- only at a suitable ambient temperature between 0°C and 40°C .

- not in wet or humid environments.
- not in potentially explosive atmospheres.
- if a minor, then only under the supervision of a parent or tutor.

Predictable misapplication

Dangers result from incorrect handling of the unit (electrical as well as mechanical)!

- Do not short-circuit the battery!
- Do not damage the battery with intent!
- Do not extend the device with extensions with sharp edges or tips!
- Risk of crushing or impact injuries if the robot falls down!
- Always switch off the unit completely before making mechanical or electrical changes and disconnect the accumulator from the entire system when making major modifications!

In case of inappropriate programming and use of the robotic platform, damage may occur to the platform itself or to surrounding objects.

Incorrect storage can also cause damage to the robot. Incorrect storage can also cause damage to the robot. Therefore, store the robot as described.

- not in direct sunlight.
- only on the storage rack provided.
- for long storage with the battery unplugged.
- only in dry rooms.
- not within reach of children when used unsupervised.

Remaining risks

Under certain circumstances, the platform can cause serious damage to health even when used professionally!

- Do not work with the platform if you suffer from epilepsy.
- Looking directly and continuously at the light emitting diodes from a short distance may cause irreversible eye damage.

Fire hazard due to overheating of the robot platform!

- Do not operate the robot unsupervised.
- Only charge the accumulator under supervision .

Danger of burns from touching heated parts!

The following parts of the robot platform heat up during operation and must not be touched until they have had enough time to cool down:

- IOT expansion board
- Motors after heavy use
- If applicable, components attached afterwards

Danger of squeezing due to rotating components.

- Avoid reaching into the drive system.
- Activate the emergency stop if you notice a malfunction during operation.
- Lift and carry the robot by the handle provided.

Injuries due to unexpected weight.

- Expect the weight of the platform to become heavier when lifting.

Risk of injury from falling!

- Do not place the platform in escape routes or walkways to avoid tripping over it.

References
