Teletransport V1.0 beta
User Manual

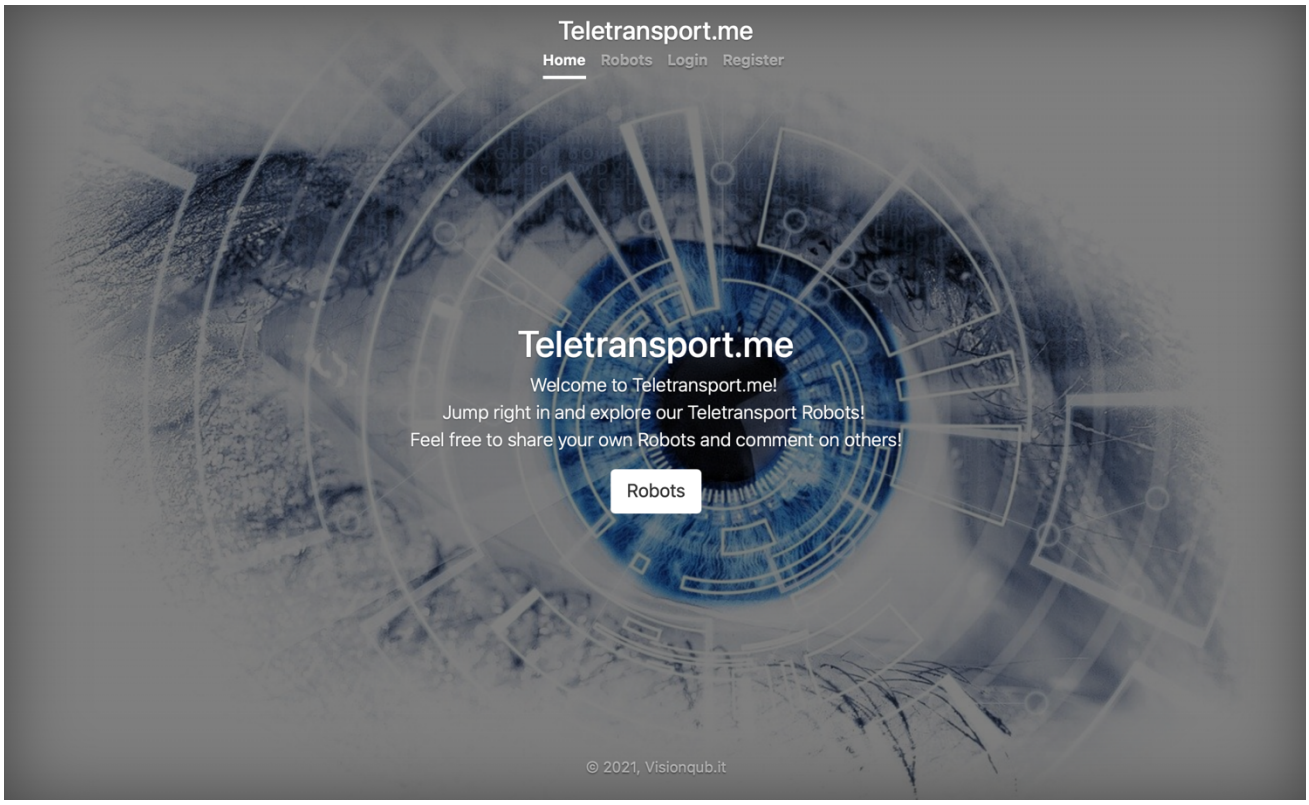


Table of Contents

Introduction

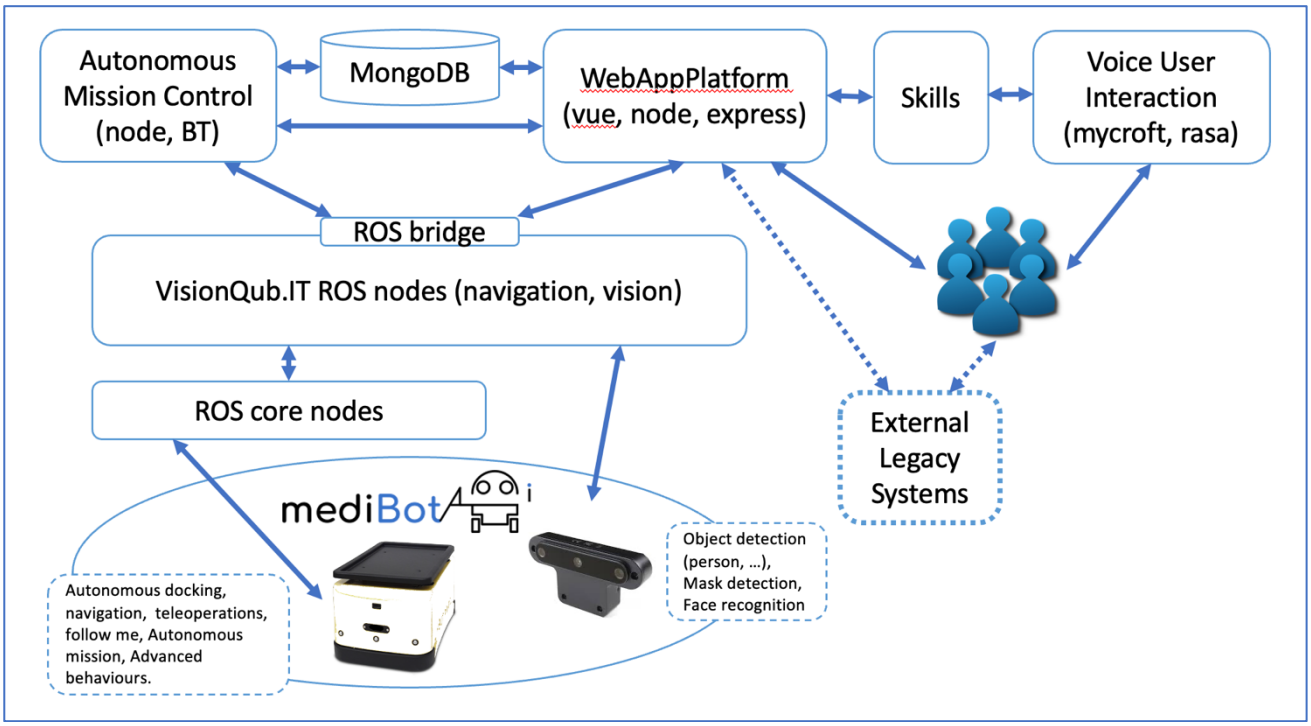
The purpose of this document is to describe the current functionalities of Teletransport. Teletransport is a platform to control fleet of heterogeneous mobile robots. The name comes from the concept of telepresence … telepresence can be defined as a sort of videoconference with wheels but can be also imagined as a way to be tele-transported in a different location where you can teleoperate a robot or simply launch a mission that the robot will execute while you are monitoring from a remote location.

Teletransport can provide generally limited access to robots built by third party provided that there is a plug-in developed and available and full access to robots with a ROS control system developed by VisionQub.IT. We call the former type of robots “Hosted Robots” and the latter “Native Robots”.

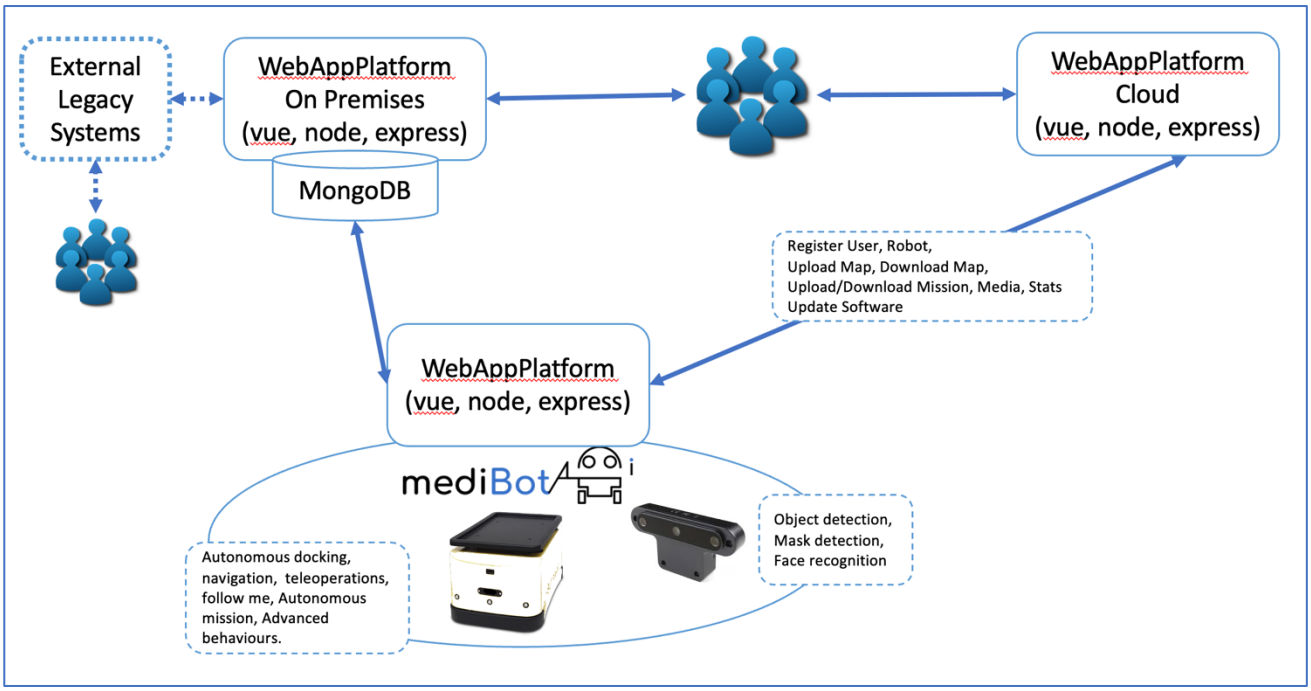
In case of “Native Robots” Teletransport can run on the same robot hardware, on the customer premises and on the cloud. Communication is available between the local teletransport and the remote one. The purpose of the local installation is the remote control of the single robot. The purpose of the remote version running on the customer premises or on the cloud is the remote control of fleet of robots.

Architecture

Teletransport is a scalable, responsive web application based on NodeJS, MongoDB, Express and VueJS. The picture below describes the overall architecture in the case of a “Native Robot” connection. In the case of “Hosted Robots” connection the web application simply connects to the specific robot provided API through the available plug-in.



The following picture describes the communication between the local and remote instances of the Teletransport application.



Hosted Robots

At the moment the currently supported hosted robots are the following:

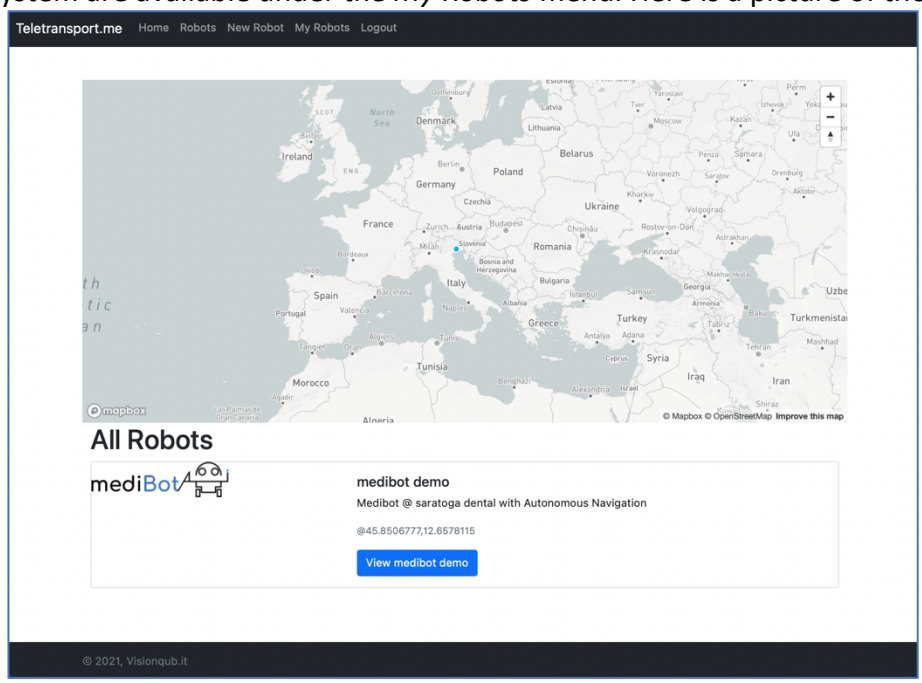
- Double by (https://www.doublerobotics.com)
- Ohmi by (https://ohmnilabs.com/)
- Ubbo Expert by (http://www.axyn.fr/)

Native Robots

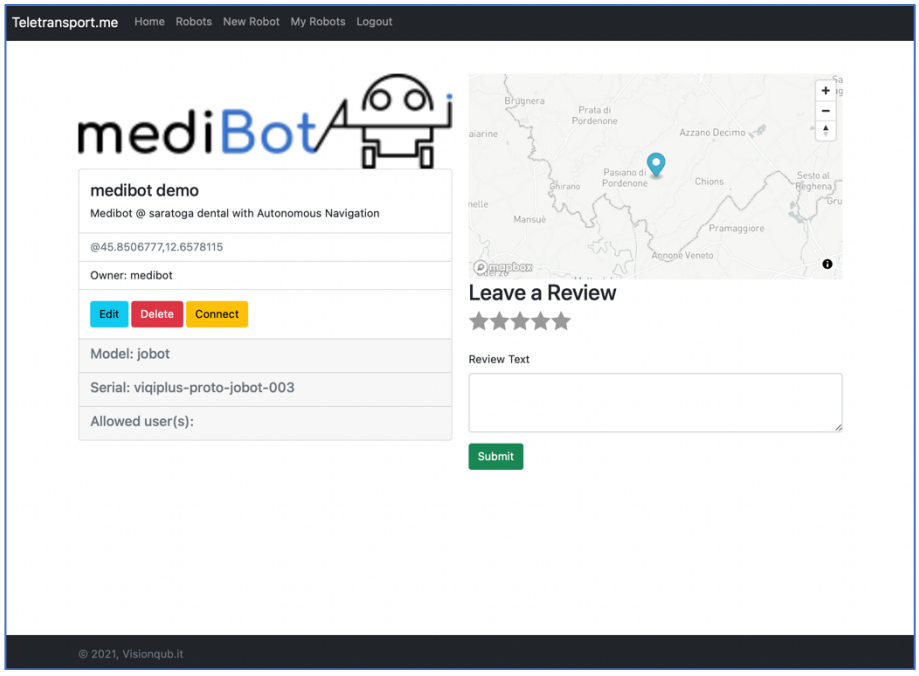
Native robots are robots controlled by a ROS control system developed by VisionQub.it with the services responding fully to the architecture presented in the previous chapters. At the moment are the supported Native robots are described in the following figure. ViQi+ hardware is Jobot and is developed by Eutronica S.r.l. while ViQi* hardware is RoverZero4WD developed by Rover Robotics. An integration with ViQi based on Temi is under development for a specific application called ViQi Smart Health for safe drug delivery in hospitals and much more. ViQi is based on the Temi platform (https://www.robotemi.com).

Dashboard

In order to access a robot you need to sign in and then log in. In order to register use the *Register* menu on the top bar. In order to login use the *Login* menu. When you select the *Robots* menu you are presented with the list of all available robots. You can select any robot and leave a comment. You can edit the robots you have added to the system (owner). You can control the robots you own or the robots the owner has given you the permission to access (allowed user). The robots you have added to the system are available under the *My Robots* menu. Here is a picture of the interface.

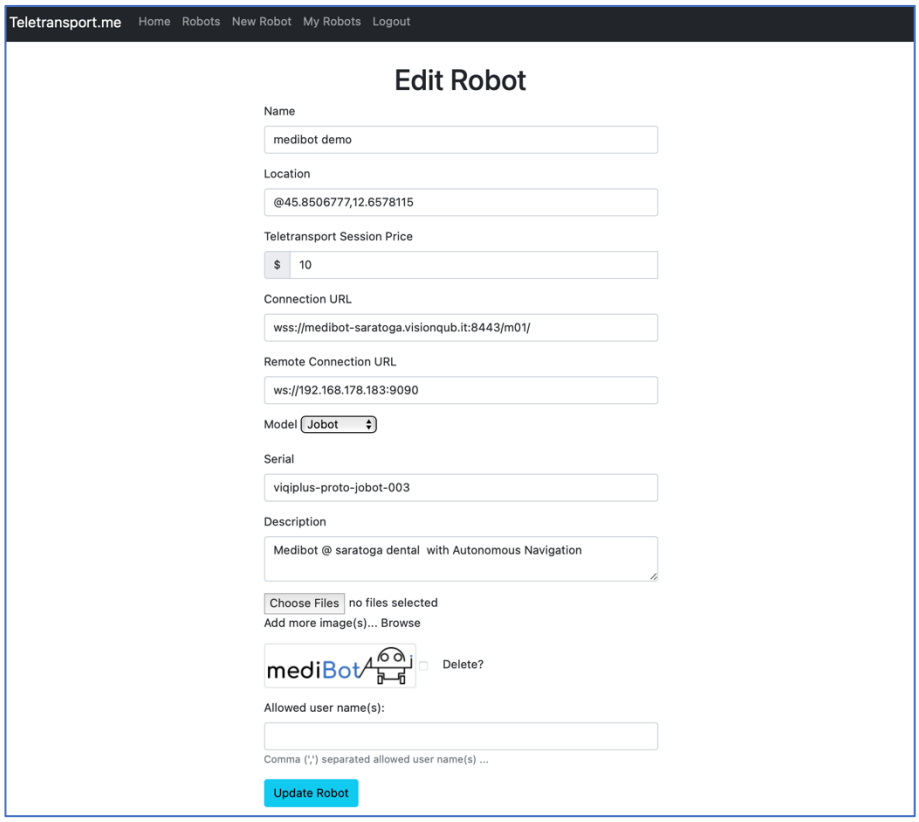


The picture below shows the interface presented when you select a specific robot via *View <robot name>* button. You can use the *New Robot* menu to add a new robot to the platform.

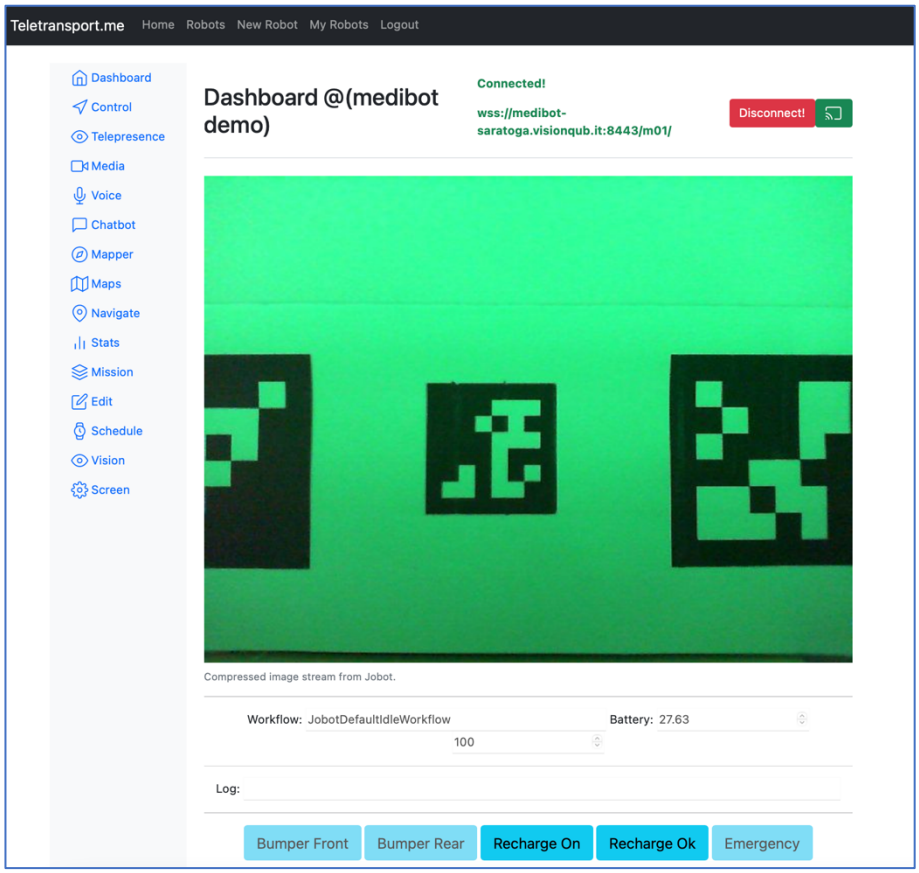


Use *Edit* to edit the robot, *Delete* to remove the robot, *Connect* to connect to the robot Dashboard.

The *Edit* interface is similar to the interface presented when you create a robot. The picture below presents the *Create/Edit* interface. You can also submit a comment / evaluation of the robot if you like.



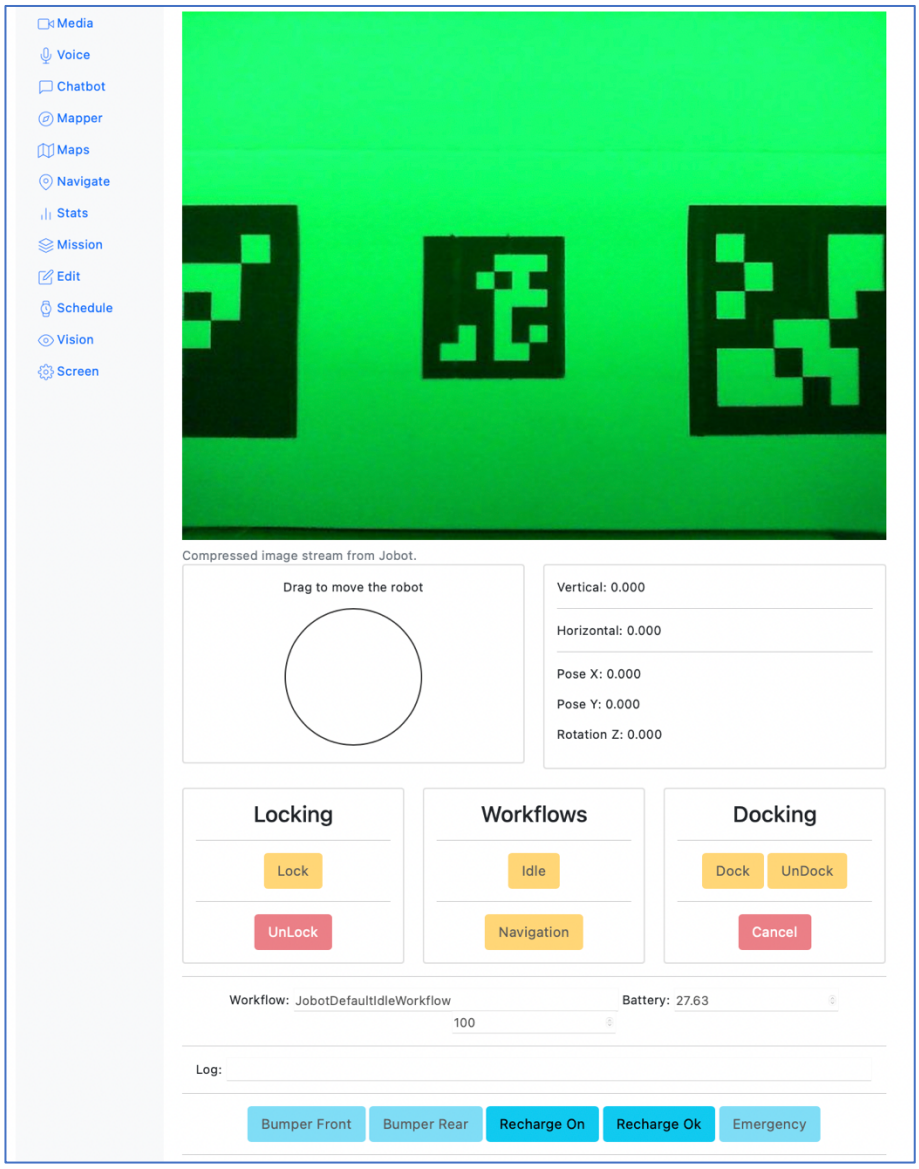
Most of the fields are self explanatory and the interface checks that all the mandatory fields are filled correctly. The location field lets you add the specific coordinates (latitude,longitude) with the following syntax *@latitude,longitude* while the connection URL syntas depend on the specific robot API and for native robots corresponds to the robot web socket. Select the correct robot model and add at least one image to describe the robot. Select *Add Robot* (if you are adding the robot to the system) or *Update Robot* (if you are updating the robot) in order to save the robot configuration. If you select *Connect* the specific robot *Dashboard* is presented. The picture below is an example Dashboard.



The dashboard presents on the left the menu with all the available functions, on top right the connection status, immediately below the video stream of the main camera normally used also to support docking. On the bottom right the current workflow (status) is reported together with the battery voltage and charge percentage and a field to report the robot control system log. A list of animated buttons presents the status of the robot bumbers (*Bumper Front* and *Bumper Rear*), of the recharge status (*Recharge On, Recharge Ok*) and of the emergency button (*Emergency*). The buttons are highlighted when the corresponding status is active. For example Recharge On active means that the robot is at the recharging (docking) station and Recharge Ok is active when the recharge process is complete. The *Emergency* is active when someone has pressed the emergency button if any.

Control

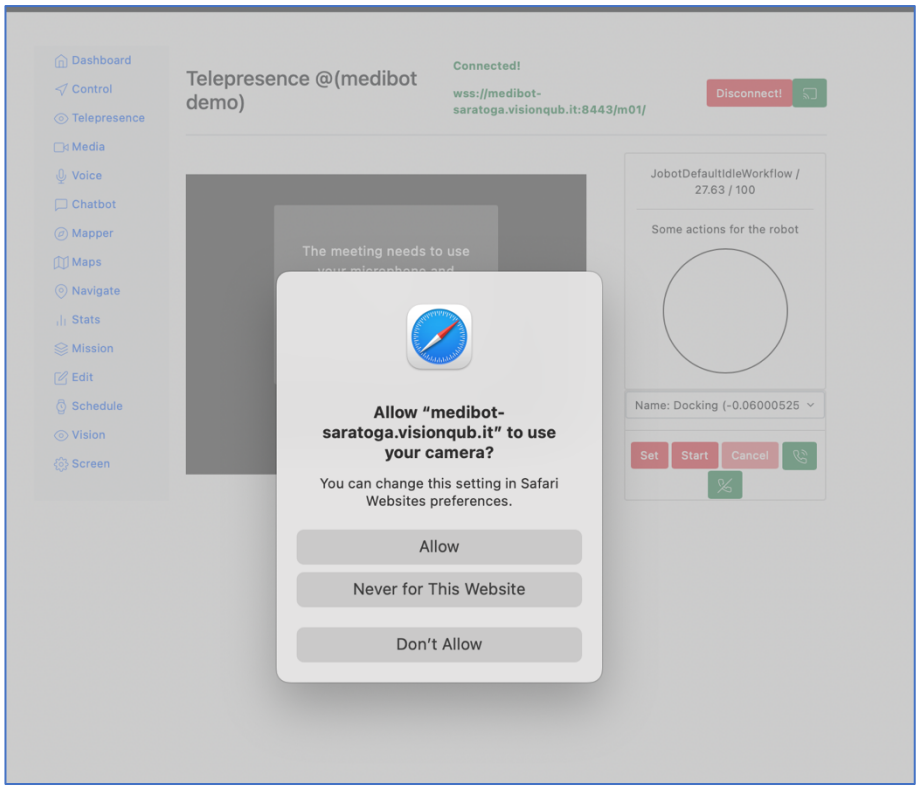
The figure below shows the Control interface that is activated when you select the *Control* link from the left link bar.



On the top you have the stream coming from the service camera. Just below, on the left a virtual joystick where you have to drag to move the robot. On the right you can see the vertical and horizontal position of the mouse in the joystick and the X, Y positions and the Z rotation from the robot navigation system. Below from left to right a set of buttons to control robot locking (when locked only you can control the robot), the workflows (robot modes) and automated docking. Note that in order to undock the *Recharge Ok* button has to be activated on some robots (e.g. Jobot). At the bottom of the page, you can find the usual status information described in the previous chapters.

Telepresence

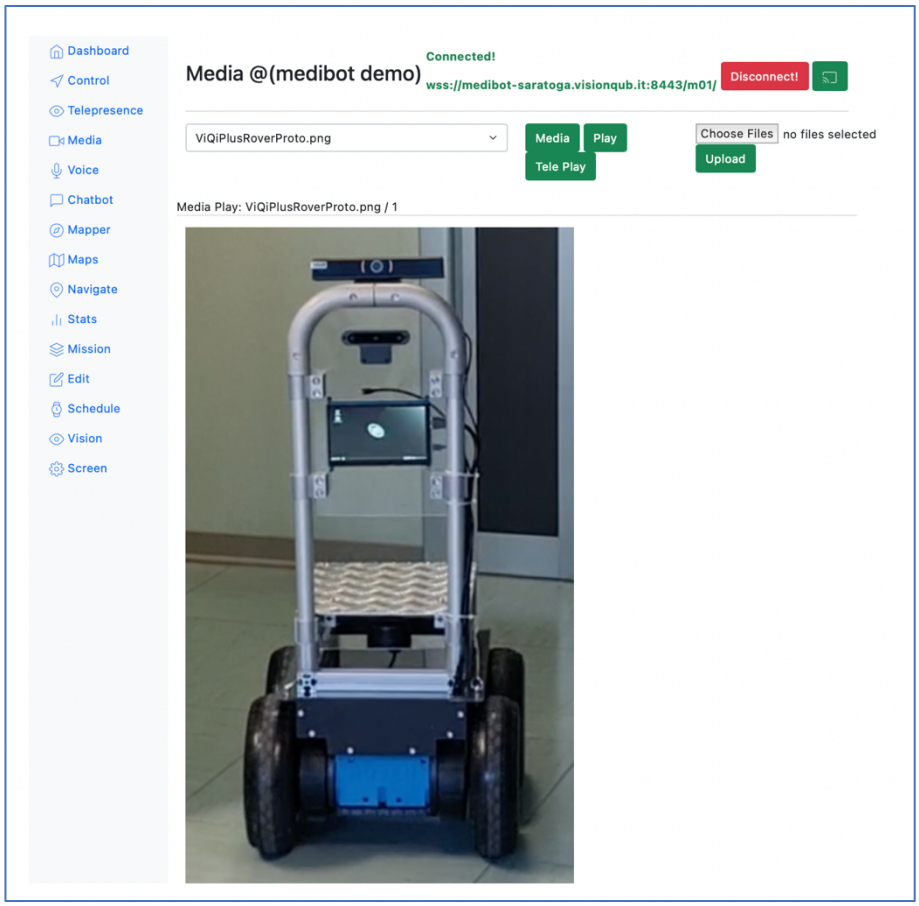
The figure below shows the Telepresence interface that is activated when you select the *Telepresence* link from the left link bar.



The Telepresence interface permits to activate a videoconference session with the robot giving you the idea to be tele-transported in the location where the robot is positioned. Besides the audio video stream coming from the robot, the people at the robot location see in the robot screen (if present) and hear you. On the right there is a control panel where you have a joystick to move the robot manually and a list of location that you can select. When you have selected the desired location, you can press set and then start and the robot will move autonomously to the desired location while the videoconference session is active. The cancel button can be used to cancel the navigation goal. There are also a couple of icons to cancel and restart the videoconference call.

Media

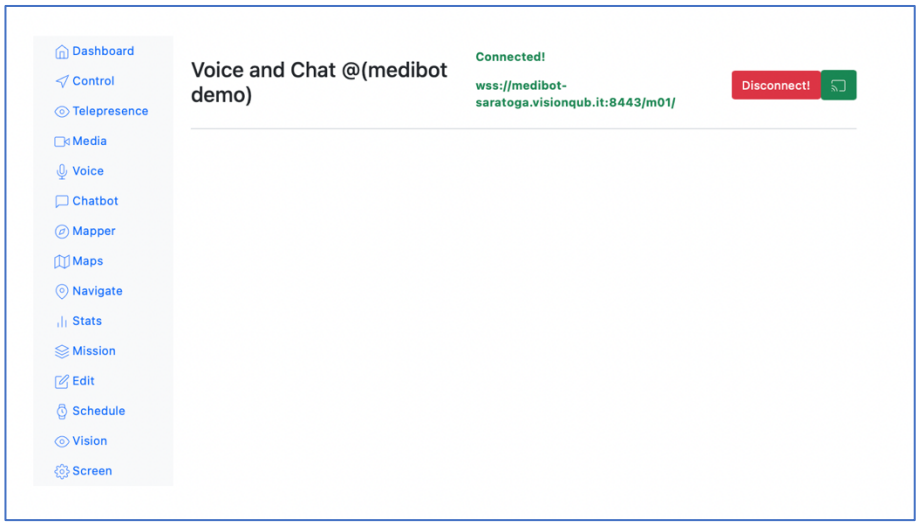
The figure below shows the Media interface that is activated when you select the *Media* link from the left link bar.



The Media interface has the purpose to permit the upload to the robot media files (images, video, audio). The *Media* button when pressed update the list with all media loaded on the robot. You can select a media and play it using the *Play* button or play the media remotely on the robot (using the robot audio and screen) using the *Tele Play* button.

Voice

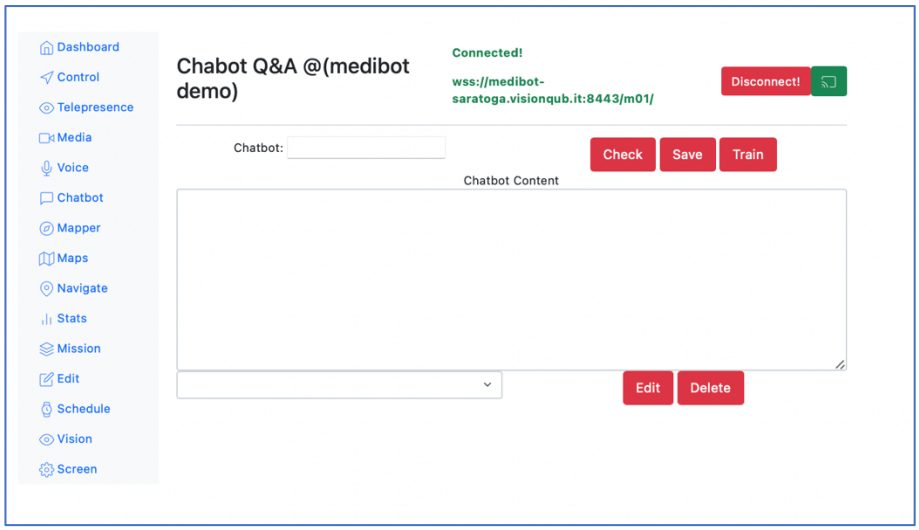
The figure below shows the Voice interface that is activated when you select the *Voice* link from the left link bar.



The Voice interface that is meant to control the robot via voice and is currently under implementation and testing. When released it will permit you to give the robot simple movement commands, like turn right, turn left, go forth, go back and navigation commands like go to <location> and stop. It will also permit to interact with the robot using the chatbot which will permit more complex conversations. With this feature the robot will work as a personal assistant.

Chatbot

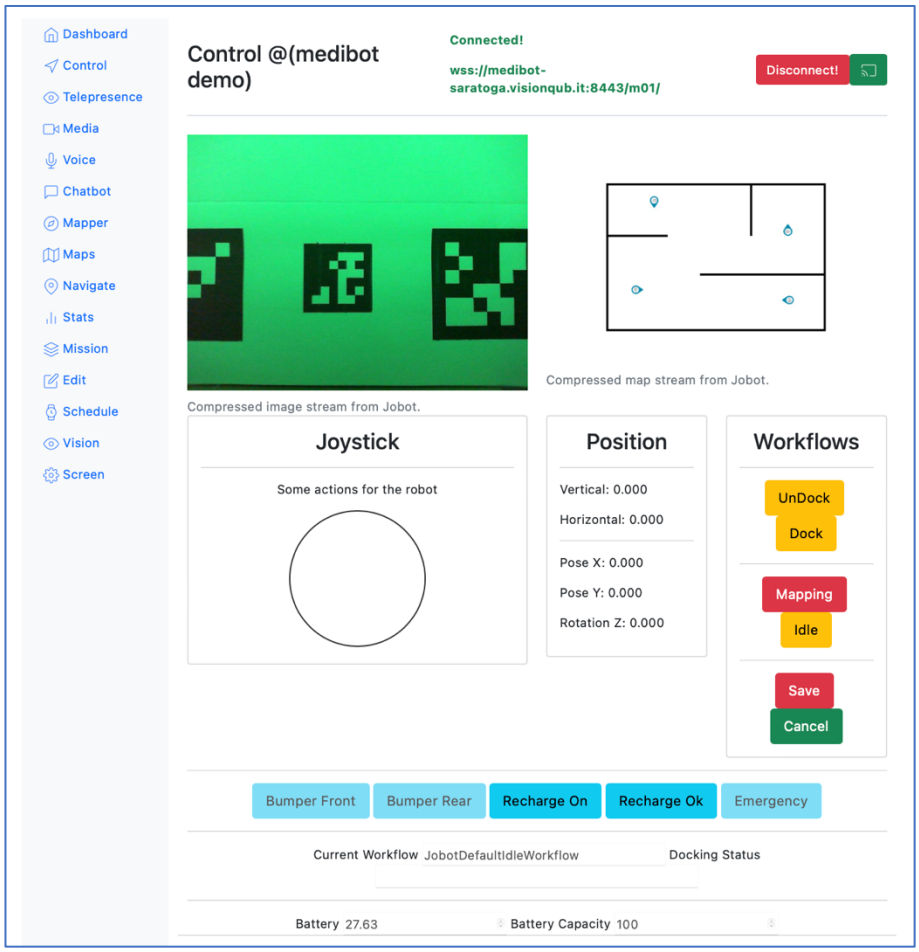
The figure below shows the Chatbot interface that is activated when you select the *Chatbot* link from the left link bar.



The Chatbot interface allow the programming of the robot chatbots. You can select the specific file, *Edit* (loads it to the text editor), *Delete* (removes the currently selected file), *Save* (saves on the robot the currently edited file), *Check* (checks the syntax of the currently loaded file). In order to create a new file you use the *Chatbot:* field to enter the name, the *Chatbot content* field to enter the file content and then *Save*. In order to make the chatbot active you need to select the *Train* button. This will activate the machine learning process that trains the Deep neural network model corresponding to the loaded Chatbot files. The Chatbot interface is currently under implementation and testing. When available together with the *Voice* interface will make the robot work as a personal assistant.

Mapper

The figure below shows the Mapper interface that is activated when you select the *Mapepr* link from the left link bar.

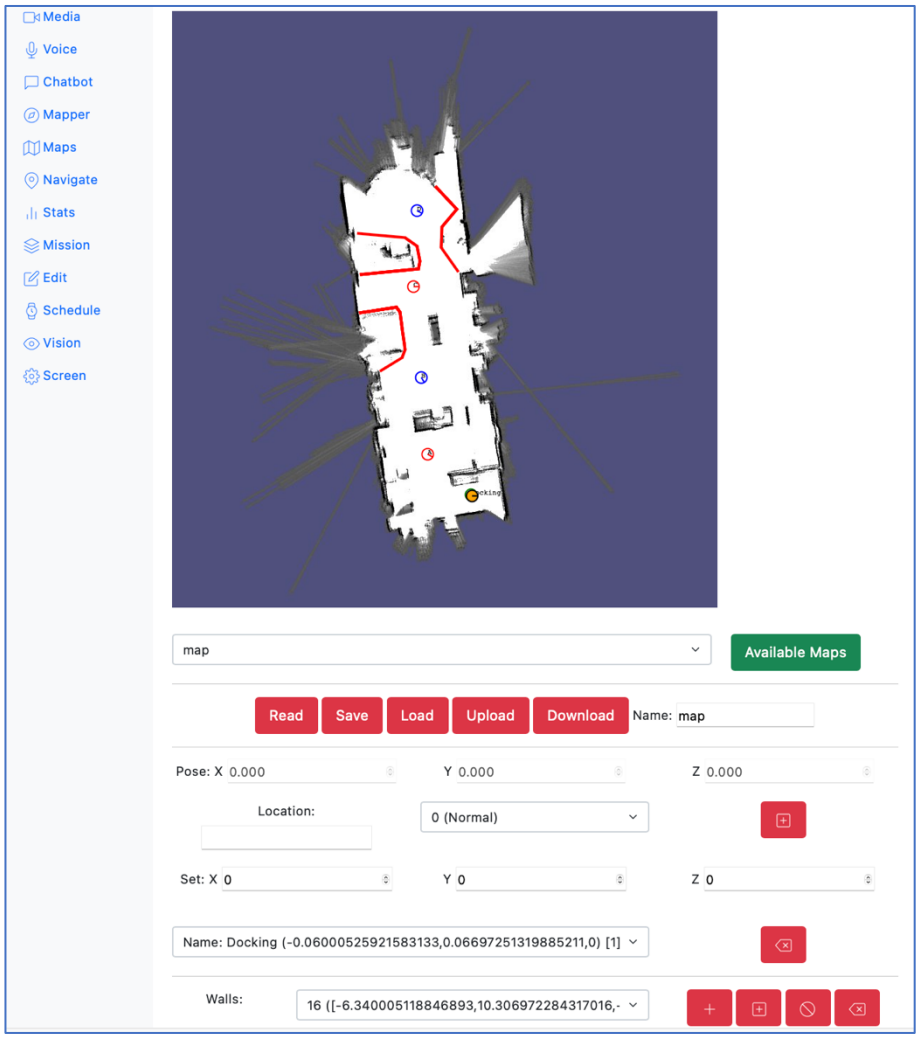


The purpose of the *Mapper* is the acquisition of a map even remotely. At the bottom of the screen, you can find the usual status information. At the top you have two image streams. On the left there is a video stream coming from the service camera that you can use to see the environment in front of the robot and on the right, you can see the stream coming from the mapper that presents the map while being acquired. You can use this stream to see if the map builds correctly or with some leaks. Just below the stream there is the virtual joystick to control the robot, the position information and some buttons to dock and undock and to activate the mapping workflow. Note that the mapping workflow when activated resets the current map (this is why the button is red). The correct sequence is:

1. Activate the Mapping workflow by selecting the *Mapping* button.
2. Acquire the map using the joystick and controlling the result by looking at the map stream
3. When satisfied activate the Idle workflow by selecting the *Idle* button (at this point the map is saved in the working memory of the mapper).
4. Save the map in the map repository on the robot (this is where the map can be edited) by selecting the *Save* button (the Save button is in red because this operation overrides the old map).
5. In case you are not satisfied with the map you can select the *Cancel* button and the old map can be recovered.

Maps

The figure below shows the Maps interface that is activated when you select the *Maps* link from the left link bar.



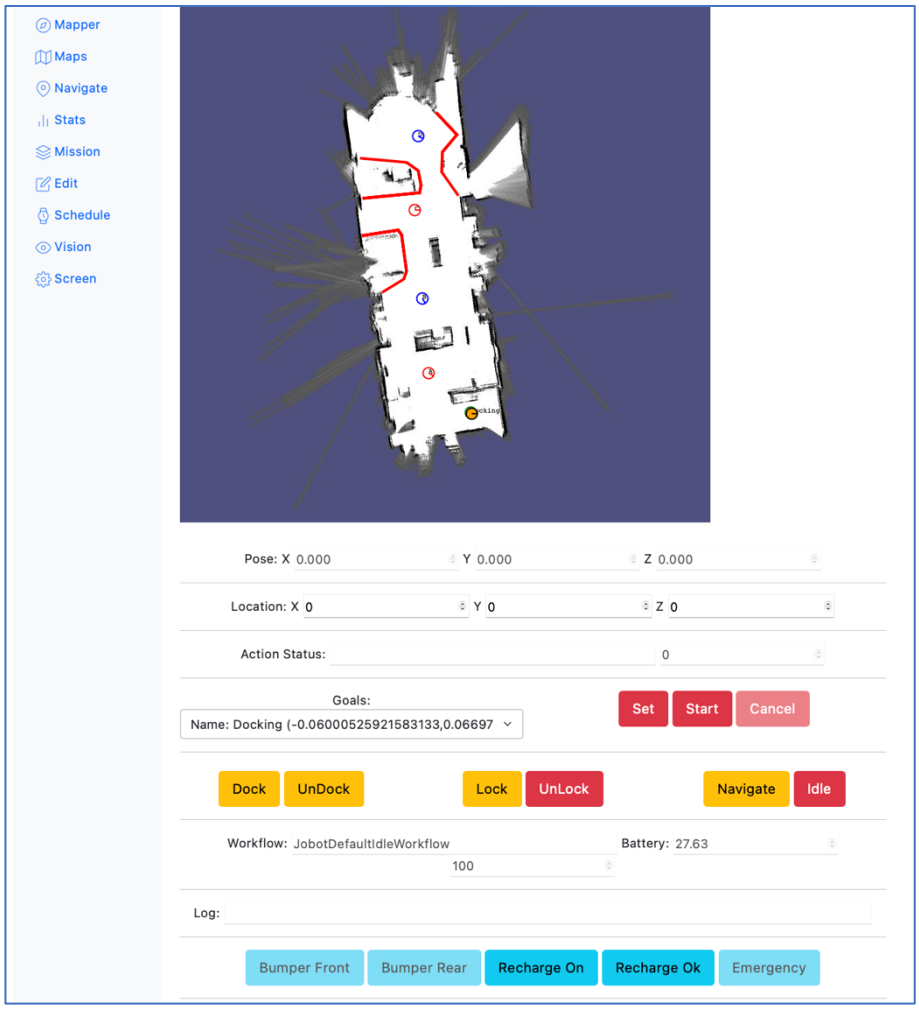
The Maps interface has the purpose to let you edit the map and add locations and virtual walls. By selecting the *Available Maps* button, you get the list of maps saved in the robot. The *Read* button loads the selected map and visualize it with all the meta information associated (locations, virtual walls, size, …). You can define new locations by pointing the mouse to a specific point in the map. The point coordinates are loaded in the corresponding fields. You can set the name of the location and the type (normal, special, dock) and the rotation in radiants. The by pressing the button with the ikon *[+]* the new point is saved in the map and visualized. You can list and select a specific location: the button with the *<+]* ikon can be used to remove the point from the map. In order to define a virtual wall, you have to define a path. Use the *+* ikon close to the *Walls* list to start the path and then click on the map to define the starting point. Every new click on the map with add a new segment to the path. Use the *[+]* ikon to save the virtual wall in the map. Use the ~~O~~ ikon to cancel the path without saving. The *<+]* ikon can be used to remove the virtual wall from the map.

When finished editing you can define if the case the map name by editing the *Name:* field and save the map with the *Save* button. In order to load the map in the working memory of the navigator you have to push the *Load* button. The Upload and Download buttons will be used to upload and download the map on the cloud peer if present in case of a multi-tier configuration. The map named

map is the default map. When you acquire a new map and with the Mapper interface, if you are not satisfied with the saved map, you have anyhow to load back the old default map in the working memory of the navigator by selecting immediately the map named *map* and by pressing the *Load* button. Be careful because if you press the *Save* button and the old map has not been back up by saving it with another name, the old map will be overwritten and hence lost. In order to back up a map with a different name you have to select the map, *Read* it, select the new name using the *Name:* field and then press the *Save* button.

Navigate

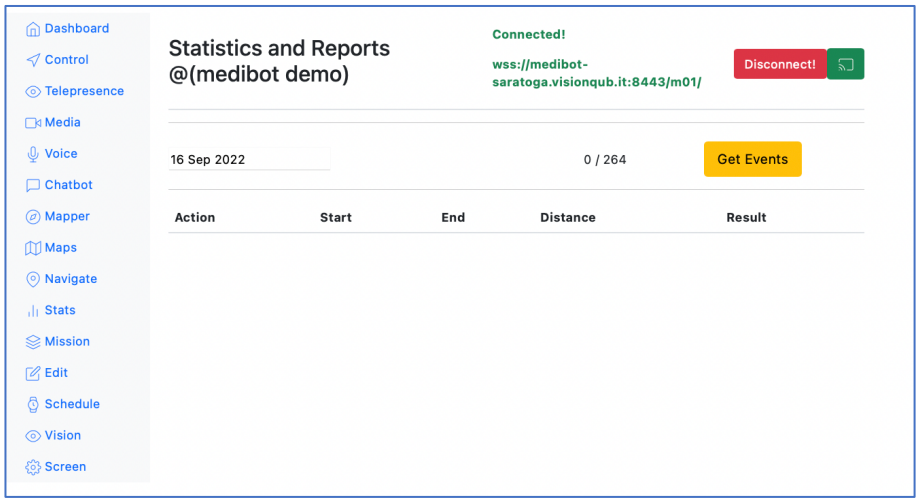
The figure below shows the Navigate interface that is activated when you select the *Navigate* link from the left link bar.



The Navigate interface that is used to control and test the navigator. The bottom part is as usual dedicated to the status information. Just above the buttons to *Dock, UnDock, Lock, UnLock, Navigate, Idle* that control respectively the automatic docking behaviour, the locking functions, and the robot workflows. In order to navigate you need to be in the navigation workflow that can be activated by pressing the *Navigate* button. The robot position is shown just below the interactive map. The map shows the robot position in real time, locations and virtual walls. You can select any location from the *Goals:* list and then by pressing *Set*. But you can select any point in the map as well. In order to navigate to the desired you have to press the *Start* button. To cancel the robot navigation goal and stop the robot from moving, press the *Cancel* button.

Stats

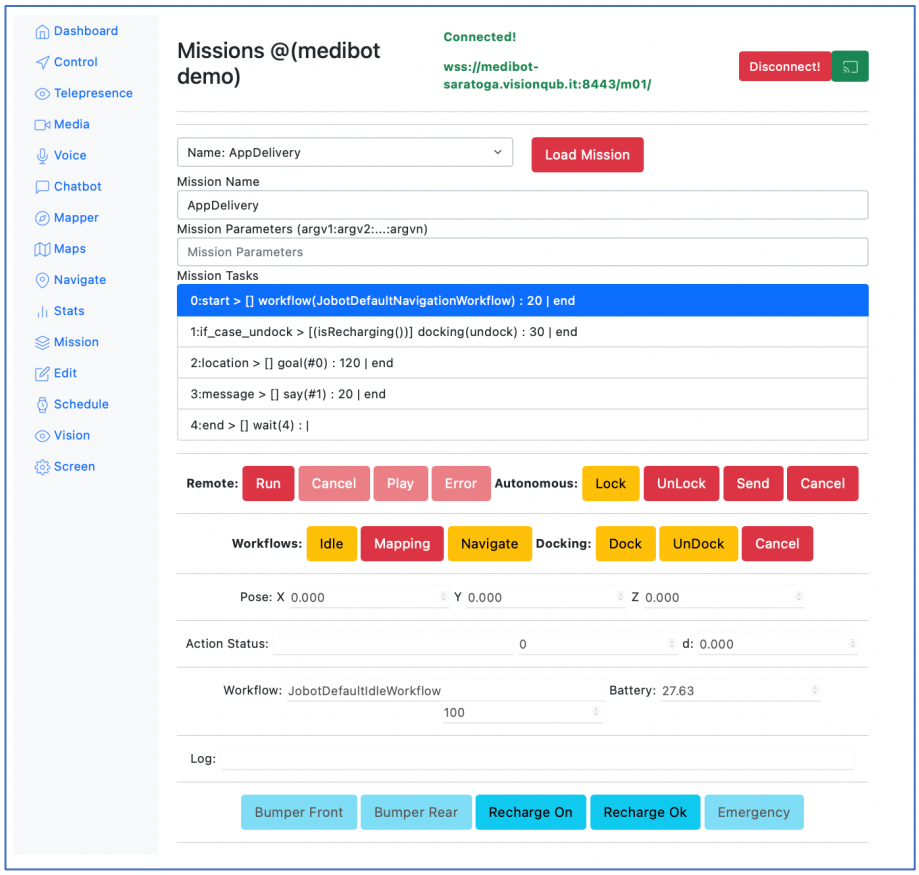
The figure below shows the Stats interface that is activated when you select the *Stats* link from the left link bar.



The Stats interface can be used to visualize very simple statistics about the robot mission execution and navigation. You can get an idea of the success and failures. Select the date from the date selector and then press the *Get Events* button.

Mission

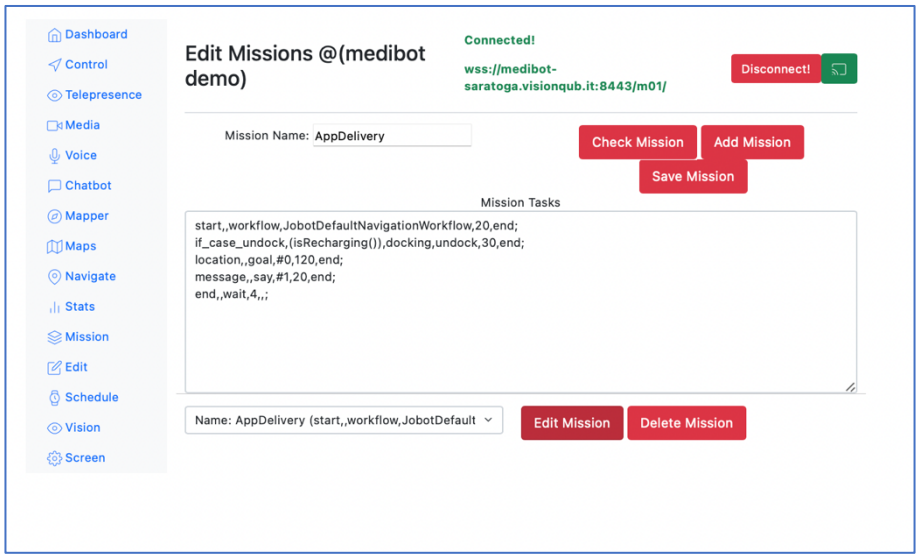
The figure below shows the Mission interface that is activated when you select the *Mission* link from the left link bar.



The Mission interface has the purpose to run and test missions. Missions are a sequence of useful task that implement a useful behaviour. In order to run a mission, you have to select a mission from the list of available missions and then press the *Load Mission* button. The mission parameters can be set using the *Mission Parameters* field. The mission is visualized in the mission debugger. The loaded mission is run by pressing the *Run* button and cancelled by pressing the *Cancel* button. Using these buttons let the browser control the robot by sending the actions one after the other. In case of network problems, the interface may be not able to send the correct command at the right time. When the mission has been tested, you can instead use the Send and Cancel buttons to do the same thing by letting the robot work autonomously. In this case the mission is loaded in the robot memory and executed by the *Selfcontrol* process. The *Lock, UnLock, Idle, Mapping, Navigate, Dock, UnDock, Cancel* buttons positioned below let you accomplish the usual tasks. The status information is reported in the bottom of the page.

Edit

The figure below shows the mission Edit interface that is activated when you select the *Edit* link from the left link bar.



The mission Edit interface has the purpose to let you create and modify the mission using the mission language described in the next section. You can load a mission by selecting it from the list and by pressing the *Edit Mission* button. The selected mission is loaded in the editor. You can delete the mission using the Delete Mission button. A new mission can be created by editing and selecting the mission's name using the *Mission name:* field and by pressing the *Add Mission* button. The mission syntax can be checked using the *Check Mission* button. An edited mission can be saved by pressing the *Save Mission* button.

Mission Language

The syntax of the Medibot mission language is quite simple.
A Mission is a list of actions separated by ';'. We have been inspired by behavioral trees with the whole mission implements the sequence construct (success if all the actions complete in order with success) where each actions implements a sort of selector construct.
Each action is described in a line with 6 components some of which can be null. The components are separated by ','; Each line is completed by ';'.
The structure of an action is the following:

```
[<label>],[<condition>],<type>,<parameters>,[<timeout>],[<fallback>];
```

Where:
<label> is a string and if not present you will not be able to goto this action.
<condition> is a javascript expression. In order to refer to any variable in the robot app you need to use helper functions (distanceFromLocation(<location),batteryLevel(), isRecharging(), isBumped(), isEmergency(). The default is true or no condition. The condition should be evaluated true in order to execute the action otherwise the action is skipped.
<type> is the type of action.
<parameters> are the action parameters usually separated by **':'**. If **<parameters>** starts with the **'@'** character the system treats **<parameters>** as a variable whose value is loaded as the value of parameter. If **<parameters>** starts with the **'#'** character the system sets <parameters> value taking it as the **<parameter>-th** value of the current mission parameters array (whose syntax is **<argv1>:<argv2>:…:<argvn>**).

<timeout> is the maximum allowed duration of the action in seconds: when the timeout is greater than 0 (default) and expires and the action has not yet completed it is considered failed.
<fallback> represents the instruction label to go in case of failure: the default is that the whole mission exits with fail.

Here is the list of possible action types:

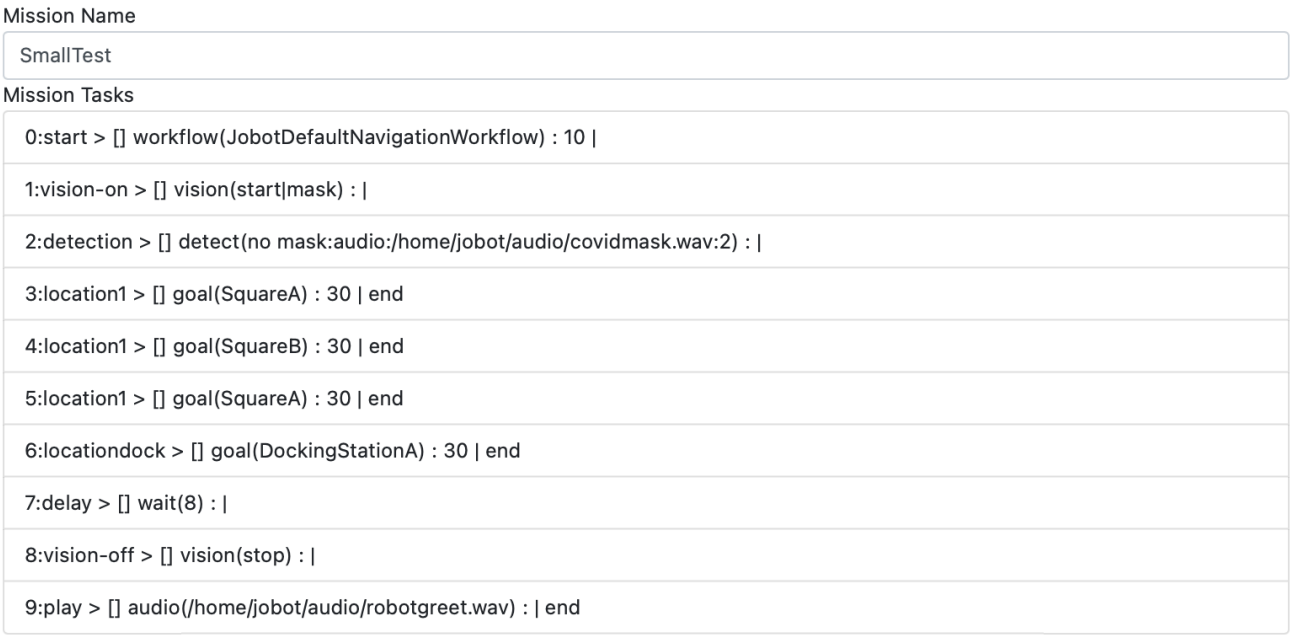
- **wait:** waits for <parameters> seconds;
- **goto:** jumps to mission action with <parameters> label;
- **move:** the robot base moves according to <parameters> specification where <parameters> is **<horizontal joystick>: <vertical joystick>** that have both to be between -0.5 and 0.5;
- **velocity:** the robot base moves according to <parameters> specification where <parameters> is **<linear_x>: <angular_z>** that are the linear and angular meaningful speeds that have both to be between -0.5 and 0.5;
- **goal:** navigates to goal named <parameters>;
- **eval:** evaluates the <parameters> expression;
- **vision:** executes the vision <parameters> command where <parameters> is one of start,start|yolo, start|mask,stop;
- **detect:** sets the detection <parameters> where <parameters> is **<target>: <action>: <parameter>: <delay in seconds>** and <action> is say or audio (see below) or eval;
- **say:** the robot converts the <parameters> text to speech;

- **workflow:** executes the workflow named <parameters> where <parameters> is one of JobotDefaultIdleWorkflow, JobotDefaultNavigationWorkflow, JobotDefaultMappingWorkflow (be careful … it destroys the current map);
- **docking:** executes docking <parameters> command where <parameters> is one of dock,undock,cancel;
- **audio:** plays <parameters> audio file;
- **tele:** launches a telepresence command where <parameters> is **<type>:<name>** (type is 0 text, 1 image, 2 audio, 3 video, 4 vconf call, 5 vconf hang, -1 unknown) and <name> is the media file name or the vconf parameter. The content is displayed on the robot screen;
- **begin:** begins a subroutine declaration and <parameters> is the name of the subroutine.
- **end:** ends a subroutine declaration; during execution returns to the caller.
- **call:** saves the register and executes the <parameters> routine.
- **map:** changes the current map to the map named <parameters>; this action to be safe should be combined by a previous workflow action to JobotDefaultIdleWorkflow and followed by workflow action to JobotDefaultNavigationWorkflow.

Note that subroutines cannot be nested: a runtime error will occur.
Here is a simple mission example:
JobotDefaultMappingWorkflow
Start,(batteryLevel() < 30),goal,DockingStationA,30,end;
,,docking,dock,60,end;
,,wait,4,,;
end,,audio,public/media/robotgreet.wav,,;

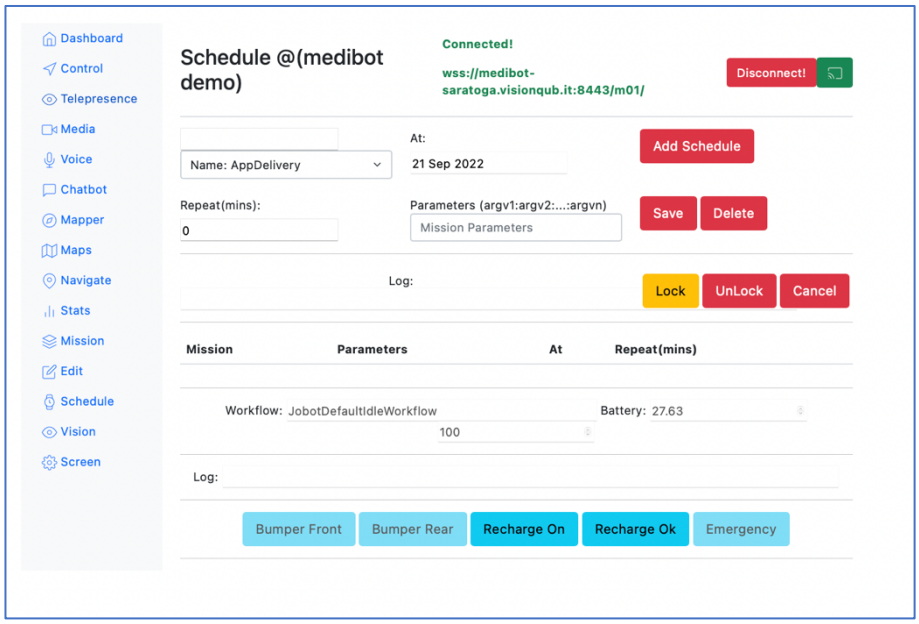
The meaning is that if the battery voltage is less than 24.0 Volt the medibot navigates to the point named in the map ‘DockingStationA’, then excutes the docking command dock and waits for 4 secs, thens plays the audio file with path ‘public/media/robotgreet.wav’.The first action is labelled ‘start’, the last is labelled ‘end’.

Here is how a mission is visualized in the mission manager:



Schedule

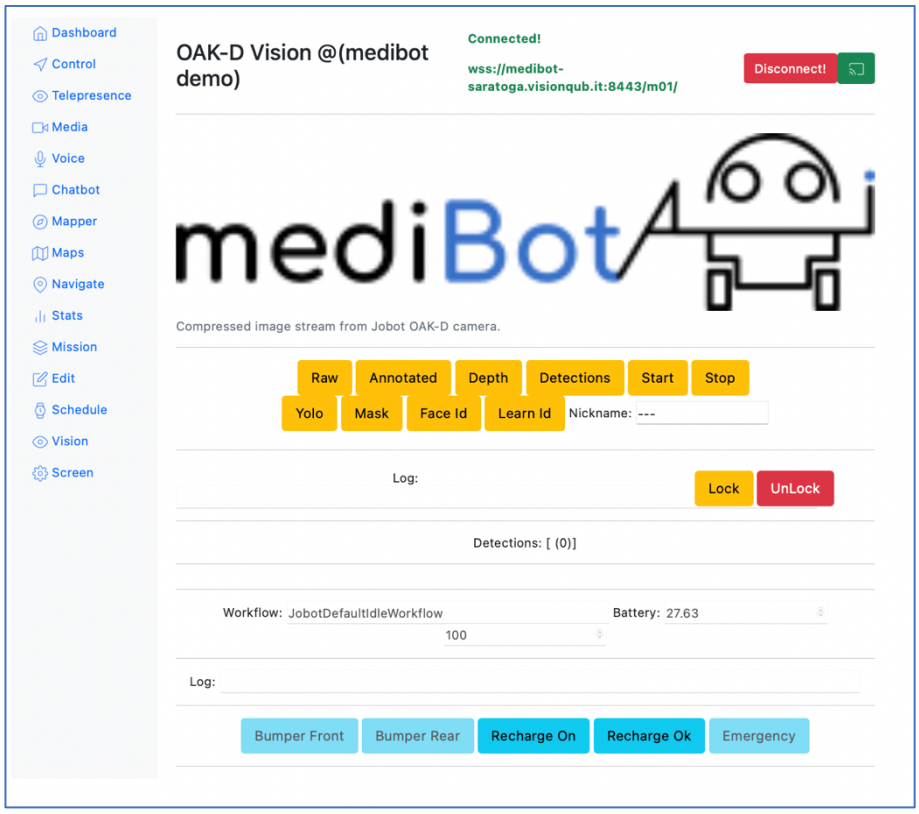
The figure below shows the Schedule interface that is activated when you select the *Shedule* link from the left link bar.



The Schedule interface can be used to schedule the execution of a mission. You have to select the mission from the mission list. Specify the mission execution date using the *At:* date selector. You can specify the mission parameters by using the *Parameters* field. The *Repeat (mins):* field can be used to specify when the mission should be repeated (in case of recurrent executions). At this point you can schedule the mission by pressing the *Add Schedule* button. The mission schedule is reported in the table below. Any schedule can be selected by clicking on the corresponding table row. The selected mission parameters are reported in the mission form. You can change the parameters and press *Save* to upload the modified schedule or delete the schedule by pressing the *Delete* button. The Lock, Unlock, Cancel, buttons and the status information at the bottom of the page have the usual functions.

Vision

The figure below shows the Vision interface that is activated when you select the *Vision* link from the left link bar.

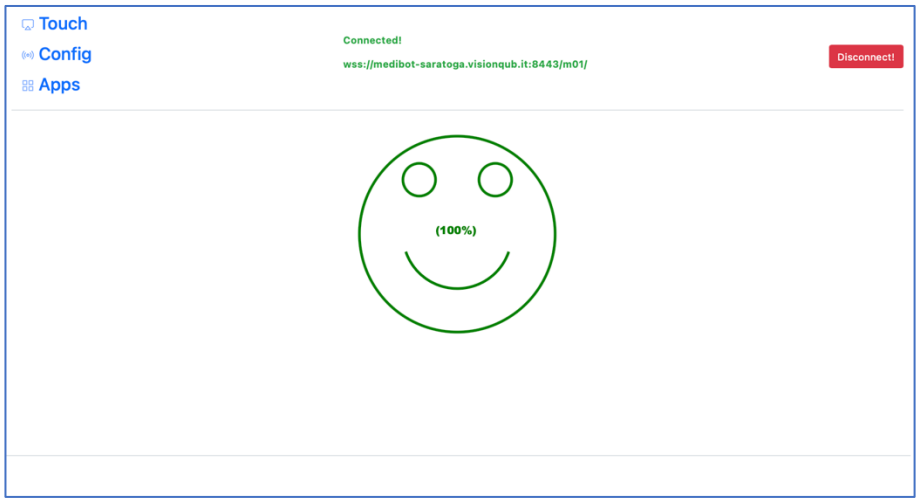


The Vision interface can be used if the robot is equipped with the OAK-D vision camera. The camera stream is displayed at the top of the page. You can visualize the *Raw, Annotated, Depth* stream by pressing the corresponding buttons. The Detections button activates the Detections stream that is visualized textually in the *Detections:* field. The Start and Stop buttons activate the OAD-K that loads the corresponding Deep Neural network pipeline. The *Yolo* (default) button loads the Yolo Deep Neural network. The *Mask* button activates the mask detection Neural network. The *Face Id* button activates the Face recognition Deep Neural network. In order to learn a new face you have to set the name in the *Nickname:* field and press the *Learn Id* button.

The Lock, Unlock, buttons and the status information at the bottom of the page have the usual functions.

Screen

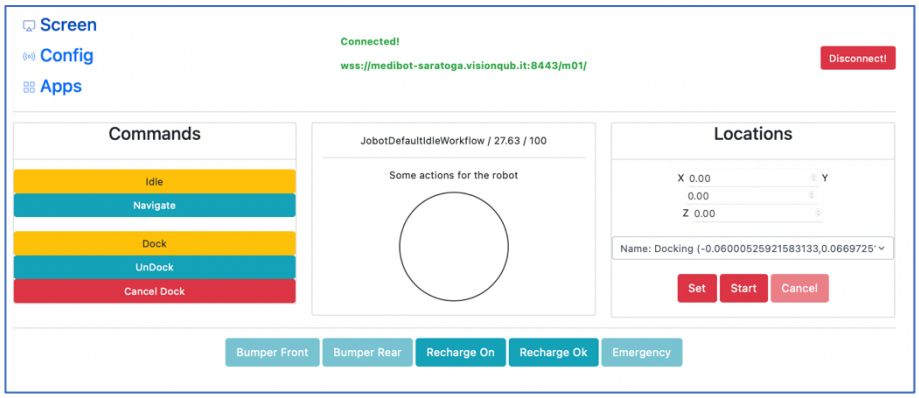
The figure below shows the Screen interface that is activated when you select the *Screen* link from the left link bar.



The Screen interface presents a smiley with the battery percentage in real time. The following pages reflect what is actually displayed on the robot touch screen (if the robot has it).

Touch

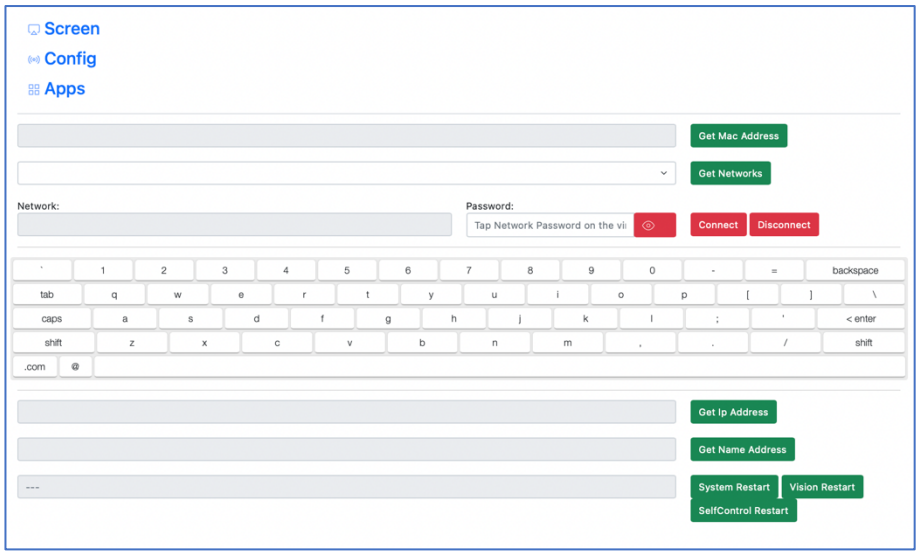
The figure below shows the Touch interface that is activated when you select the *Touch* link from the top left link bar.



The Touch interface allows the robot control from the touch screen of the robot. You can see the status from the status bar. Use the virtual Joystick, use the command buttons (*Idle, Navigate, Dock, UnDock, Cancel Dock*). You can select a location from the list of available locations and then press *Set* and *Start* to launch the navigation goal or press *Cancel* to cancel it.

Config

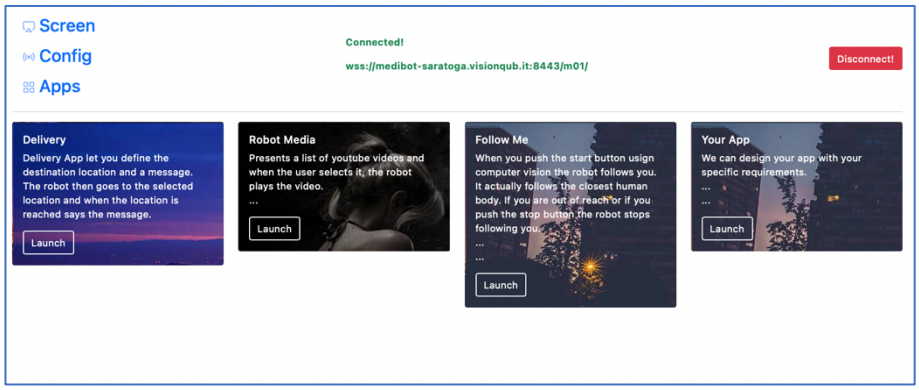
The figure below shows the Config interface that is activated when you select the *Config* link from the left link bar.



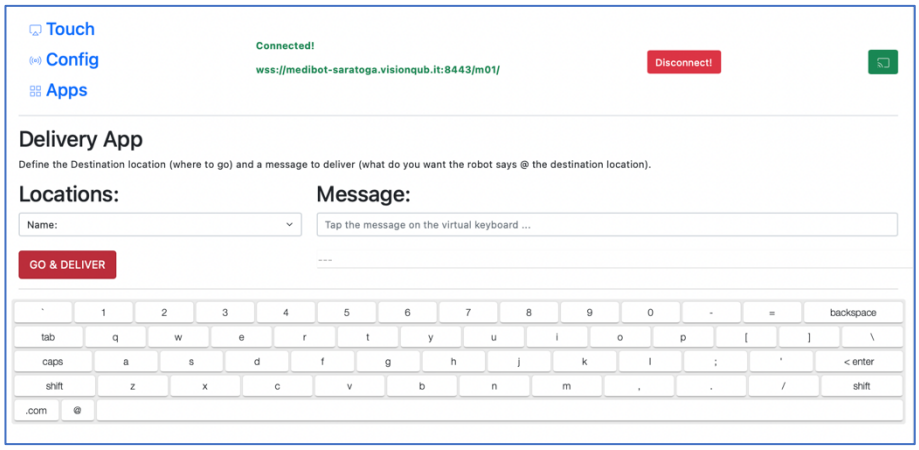
The Config interface allows you to configure the robot. You can get the Mac Address by pressing the *Get Mac Address* button. Get the available WIFI networks by pressing the *Get Networks* button. You can select the preferred network from the list and set the network password using the *Password:* field. *Connect* and *Disconnect* buttons allows you to connect and disconnect from the selected network. The *Get Ip Address* and *Get Name Address* allows you to read the robot ip address and the network name address. The buttons *System Restart, Vision Restart, SelfControl Restart,* activate the corresponding maintenance functions.

Apps

The figure below shows the Apps interface that is activated when you select the *Apps* link from the left link bar.



When you select a specific app by pressing the corresponding Launch button the specific app interface is loaded on the screen. The picture below presents for example the Delivery App interface. You have to select the specific location and insert a message and press the GO & DELIVER button. The robots goes to the specified location and when at destination says the specified message.



References

[1] ROS, Robot Operating System, https://www.ros.org