mbot_ros
User Manual

Edited by

Marco Barbosa
João Estilita
SELFTECH, Lda.

October 8, 2014

Changelog

2014-02-12 - 0.2.0

- First revision.

2014-02-14 - 0.2.1

- removed syslog messages that were being printed regularly with no purpose
- added serial communication warning messages to syslog (bad checksums and timeouts)
- regular TF publications are now done with the `static_transform_publisher` tool and not by mbot_ros
- increased speeds sent from the joystick
- increased mbotrosd sleep time to 10 seconds in order to allow the IMU to initialize the USB port
- corrected bug communicating with the IMU. Sample mode was not being activated
- added rviz config file mbot.rviz

2014-02-24 - 0.2.2

- changed respawn limit of mbotrosd to prevent it from continuously restarting
- implemented safety feature, when the software does not receive velocities for a set amount of time, it stops
- adding config parameters for joystick funtionality
- updated default value of l1 values due to update in robot documentation
- mbot_ros now expects manual controls to be in the cmd_vel_manual topic. These will override commands from the cmd_vel topic.
- updated default operation frequencies for motor board and imu after experimental tests
- implemented automatic device reconnect after connection has been lost (e.g. USB plug is disconnected and then reconnected)

- mbot board firmware version is now reported on startup
- hardware hardstop time is configurable
- hardstop status is now reported on a topic

2014-03-07 - 0.2.3

- corrected udev rules: now using a symlink and installing the file in `/lib/udev/rules.d/`
- messages now moved to `monarch_msgs` package.
- in mbot.launch: enabled time calibration for Hokuyo LRF

2014-06-20 - 0.2.4

- corrected bug which prevented automated control of the robot when joystick was disabled in configuration file.
- implemented variable speed control with joystick: the d-pad y axis controls a variable gain that is applied to both linear and angular speeds (suggested by Rodrigo Ventura)
- added startup of rosbridge to mbotros launch file
- implemented automatic system shutdown when PC voltage gets too low (suggested by Lorenzo Sarti)
- corrected wrong AHRS reference frame

2016-07-03 - 0.2.5

- Correcting LRF position as per information provided by IdMind.
- Changing LRF maximum and minimum angles.
- Adding scripts for correct LRF device naming.
- Changing topic names in order to be able to work within a namespace. The used namespace is specified with the variable robot_name in the mbot.launch file.
- Added possibility to configure ROS_MASTER_URI in config.xml: now the daemon waits for a roscore to be accessbile at the defined URI. Only after being accessible do all the robot's nodes start.

2016-10-03 - 0.2.6

- Changing LRF maximum and minimum angles.
- Adding scripts for correct LRF device naming.

2016-10-08 - 0.2.7

- Changed source of information for low battery voltage check: now using electronics battery as indicated by IDMind.
- Changed motor board interaction order to prevent message timeouts.

Preamble

This document describes mbot_ros and all of its components. It is supposed to be a usage manual for MOnarCH partners that manipulate the project's robots, the mbots . This document will focus on practical issues dealing with installing, configuring and daily usage of mbot_ros onboard the MOnarCH robots. For information on developing software for the mbots and the overall MOnarCH system, please consult the Software Integration Manual.

List of contributors

- Pedro Lima
- Rodrigo Ventura

Contents

List of Figures

List of Tables

Chapter 1

User tasks

1.1 Dependencies

mbot_ros uses custom-defined ROS messages that belong to the `monarch_msgs` package. In order to be able to use ROS tools to manipulate these custom message types, the `monarch_msgs` package should be available in your ROS system (for instance, you should be able to `roscd` into the `monarch_msgs` package).

1.2 Installation

This section describes the installation of software provided by SELFTECH that makes all of the robots' hardware available for usage within a ROS environment.

1.2.1 Obtaining installation package

The mbot-ros software is distributed as a ready-to-install debian package. You may download it from `https://selftech.com/monarch/`. Username is `monarch` and password is `hominibus`.

1.2.2 Installing software

Once you download the package you may install it with the command `dpkg -i mbot_ros-<version number>-Linux.deb`.

Uninstalling is just as simple with the command `dpkg -r mbot_ros`.

In order to have ROS working well with mbot_ros, add `source /opt/mbot_ros/setup.bash` to your `.bashrc` file. After doing this change either log out and log back in again or run in every terminal `source ~/.bashrc`.

Since mbot_ros installs `udev` configuration files, `udev` should be restarted prior to using the software. Either reboot the system or issue the command `sudo service udev restart`.

1.2.3 Checking hardware latency settings

The mbots use USB-to-serial converters manufactured by FTDI. These drivers use an internal timer that define when is data flushed from its internal buffers. The default value for these timers is normally quite high (¿= 16ms), which has an impact on devices that need to be highly responsive such as the robot's speed control hardware. These setting can be, however, changed by the user. In order to check what value is set the user can use the command `cat /sys/bus/usb-serial/devices/ttyUSB*/latency_timer`.

If there is any value reported greater than 1, the user should set the value for that device with the command `echo 1 > /sys/bus/usb-serial/devices/ttyUSB0/latency_timer`, substituting `ttyUSB0` with the appropriate device.

1.3 Configuration

The configuration file for mbot-ros can be found in `/opt/mbot_ros/config.xml`. The available configuration items are listed in table 1.3. An example configuration file is listed below:

```
<mbot_ros>
  <config_version>
    <number>2</number>
  </config_version>
  <autostart>
    <enabled>true</enabled>
  </autostart>
  <platform>
    <type>mbot_so</type>
  </platform>
  <four_wheel_mecanum><!-- Constants for 4 wheel mecanum kinematics -->
    <l1>0.14429</l1>
    <l2>0.175</l2>
    <r>0.05</r>
  </four_wheel_mecanum>
  <devices><!-- Hardware devices config -->
   <idmind_imu>
      <enabled>true</enabled>
```

```
      <device_path>/dev/mbot-imu</device_path>
      <frequency>45</frequency>
    </idmind_imu>
    <idmind_motor_board>
      <enabled>true</enabled>
      <device_path>/dev/mbot-motorboard</device_path>
      <clicks_per_turn>2000</clicks_per_turn>
      <gear_ratio>13.795918367</gear_ratio>
      <hardstop_time>2</hardstop_time>
      <frequency>45</frequency>
      <vel_timeout_ms>200</vel_timeout_ms>
    </idmind_motor_board>
    <idmind_sensor_board>
      <enabled>true</enabled>
      <device_path>/dev/mbot-sensorboard</device_path>
      <frequency>10</frequency>
      <num_sonars>12</num_sonars>
      <sonars_fov>0.785398163</sonars_fov>
      <sonars_max_range>6.4516</sonars_max_range>
      <sonars_min_range>0.15240</sonars_min_range>
    </idmind_sensor_board>
    <joystick>
      <enabled>true</enabled>
      <device_path>/dev/input/js0</device_path>
      <angular_gain>3.14</angular_gain>
      <linear_gain>1.0</linear_gain>
    </joystick>
  </devices>
</mbot_ros>
```

Item path	Description	Values	Default Value	Added in version
config_version	Indicates the version number of the configuration file structure	Integer greater than 0	0	1
autostart	Section that contains configurations for mbot-ros automated startup			1
autostart / enabled	Controls whether mbotrosd will run at startup	true or false	false	1
platform	Section that defines which physical platform is the software running on			2
platform / type	Type of the platform	mbot_po or mbot_so depending on whether the platform is a PO or SO robot.	mbot_so	2
four_wheel_mecanum	Four wheel mecanum kinematic configurations.			2
four_wheel_mecanum / l1	L1	double	0.14429	2
four_wheel_mecanum / l2	L2	double	0.175	2
four_wheel_mecanum / r	Wheel radius	double	0.05	2
devices	Section that contains configurations for the devices that mbotrosd interfaces directly			2
devices / idmind_imu	Configuration for IdMind's IMU board			2
devices / idmind_imu / enabled	Enables mbot_ros to connect to this board	true or false	true	2
devices / idmind_imu / device_path	Filesystem path to the device	string	/dev/mbot-imu	2
devices / idmind_imu / frequency	Frequency at which data is published, in Hz	double	45	2
devices / idmind_motor_board	Configuration for IdMind's motor board			2
devices / idmind_motor_board / enabled	Enables mbot_ros to connect to this board	true or false	true	2
devices / idmind_motor_board / device_path	Filesystem path to the device	string	/dev/mbot-motorboard	2
devices / idmind_motor_board / clicks_per_turn	Number of encoder clicks per motor rotation	integer	2000	2
devices / idmind_motor_board / gear_ratio	Gear ratio, from motor to wheels (number of rotation of the motor per wheel rotation)	double	13.795918367	2
devices / idmind_motor_board / hardstop_time	Time duration, in seconds, while a hardware hardstop will be active in case of the bumpers being activated	integer	2	2
devices / idmind_motor_board / frequency	Frequency at which data is published, in Hz	double	45	2
devices / idmind_motor_board / vel_timeout_ms	Maximum time to continue sending the same speed value to the electronics, in ms	unsigned int	200	2
devices / idmind_sensor_board	Configuration for IdMind's sensor board			2
devices / idmind_sensor_board / enabled	Enables mbot_ros to connect to this board	true or false	true	2
devices / idmind_sensor_board / device_path	Filesystem path to the device	string	/dev/mbot-sensorboard	2
devices / idmind_sensor_board / frequency	Frequency at which data is published, in Hz	double	10	2
devices / idmind_sensor_board / num_sonars	Number of sonars installed on the robot	integer	12	2
devices / idmind_sensor_board / shutdown_voltage	Voltage under which the PC is automatically shutdown by mbot_ros in order to prevent a hard shutdown	11.5	2	
devices / idmind_sensor_board / sonars_fov	Field-of-view of the sonars, in radians	integer	0.785398163	2
devices / idmind_sensor_board / sonars_max_range	Maximum range detected by sonars	double	6.4516	2
devices / idmind_sensor_board / sonars_min_range	Minimum range detected by sonars	double	0.1524	2
devices / joystick	Configuration for joystick capabilities	2		
devices / joystick / enabled	Enables mbot_ros to connect to the joystick			2
devices / joystick / enabled / device_path	Filesystem path to the device	string	/dev/input/js0	2
devices / joystick / enabled / angular_gain	Gain of the angular joystick commands. Value will equal the maximum angular speed in rad/s	double	3.14	2
devices /	Gain of the linear joystick commands.	double	1.0	2

1.4 Controlling startup and shutdown of mbot_ros

The software package mbot_ros contains a daemon named mbotrosd which is able to execute mbot_ros and `roscore` at system startup. This daemon is configured by default to start automatically at system startup. The daemon's status can be checked with the command `sudo service mbotrosd status`. Likewise it can be manually started using `sudo service mbotrosd start` and manually stopped using `sudo service mbotrosd stop`. As long as `mbotrosd` is running, all sensors and actuators of the robot should be available from ROS topics.

Since a ROS master is needed to have the software running, `mbotrosd` checks if a ROS master is available and, if not, will start a `roscore` instance itself.

If the user wants to prevent mbotrosd from starting automatically at startup it is just a matter of setting the `autostart / enabled` field in the configuration file to "`false`".

If mbotrosd autostart is disabled, it is still possible to run mbot_ros with the command `rosrun mbot_ros mbotros`. Please note that a `roscore` instance should be available prior to running mbot_ros.

1.5 Automatic system shutdown

In order to prevent the system from suffering a hard shutdown, an automatic shutdown feature was added. Once the PC voltage goes below a pre-defined limit (defined in the `config.xml` file in section `devices / idmind_sensor_board / shutdown_voltage`), mbot_ros issues a `shutdown` command. A 2 minute notice is given to all logged in users with the following message appearing on all open consoles:

```
The system is going down for power off in 2 minutes!
PC voltage too low!
```

1.6 View log files

The mbotrosd daemon logs to the standard system log file `/var/log/syslog`. An example of log entries produced by `mbotrosd` follows:

```
Nov 11 12:09:42 st-desk01 mbotrosd[4237]: starting
Nov 11 12:09:42 st-desk01 mbotrosd[4237]: daemonization complete
    without effort
Nov 11 12:09:42 st-desk01 mbotrosd[4238]: startup finished
Nov 11 12:09:42 st-desk01 mbotrosd[4238]: loading configuration
Nov 11 12:09:42 st-desk01 mbotrosd[4238]: configuration loaded
Nov 11 12:09:42 st-desk01 mbotrosd[4238]: time has passed
Nov 11 12:09:46  mbotrosd[4238]: last message repeated 4 times
Nov 11 12:09:46 st-desk01 mbotrosd[4238]: received signal 15 (
    Terminated)
Nov 11 12:09:46 st-desk01 mbotrosd[4238]: terminated
```

A user may view this information in real time using the command `tail -f /var/log/syslog | grep mbotros`. This will show only messages originating from the mbotrosd daemon.

1.7 Visualizing robot information on rviz

Be sure that your computer is in the same network as the robot and set the environment variable `ROS_MASTER_URI` to point to the ROS master instance running onboard the robot, for instance: `export ROS_MASTER_URI=http://mbot01:11311/`.

Then run `rviz` using `rosrun rviz rviz`. Once `rviz` is running, add visualizations for the available topics to see whatever you need.

A ready-to-use configuration for `rviz` is available in the mbot_ros directory with the name `mbot.rviz`.

1.8 Driving the robot manually

mbotrosd automatically uses USB joysticks that are connected to the robot. Once a joystick is connected, `mbotrosd` will publish target velocities to the `/cmd_vel_manual` topic and this will make the robot move according to the joystick input.

Joystick functionality was tested with a Logitech RumblePad 2 joystick that has keys as indicated in Figure 1.1

The key mappings of the joystick are defined in Table 1.8.

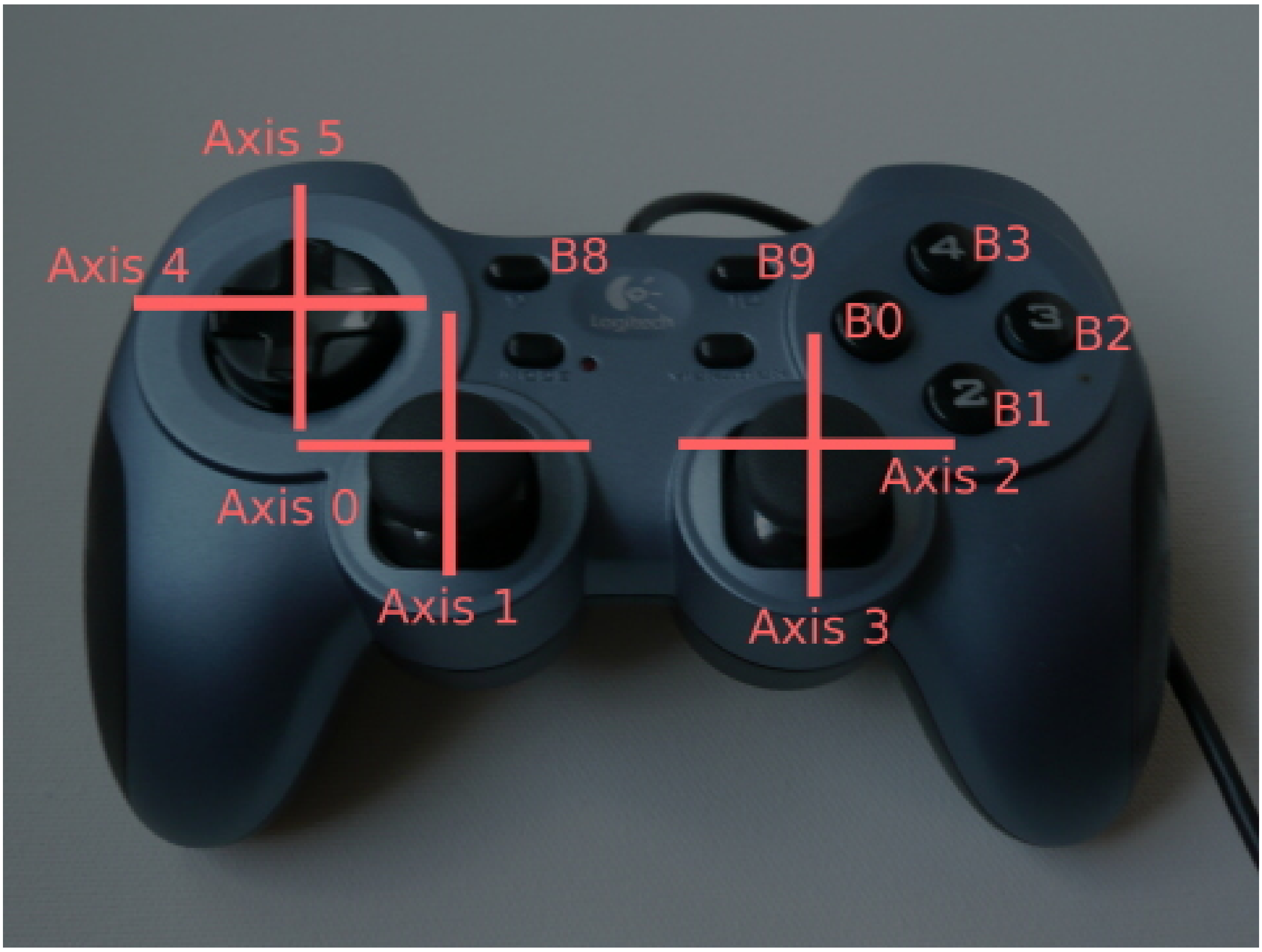

Figure 1.1: Joystick used for testing.

Key	**Mapped to**
Axis 0	Robot Linear Y-axis speed
Axis 1	Robot Linear X-axis speed
Axis 2	Robot Angular speed
Axis 4	Increase variable gain multiplier
Axis 5	Decrease variable gain multiplier

Please note that if other software is also publishing data to the `/cmd_vel` topic, the robot will only follow the `/cmd_vel_manual`, thus manual control overrides automatic control.

Chapter 2

Reference documentation

2.1 Robot reference frames

ROS provides a framework for registering relationships between coordinate frames over time named `tf`. Each of the robots' sensors measure features in its own coordinate frame. In order to successfully be able to relate information gathered from different sensors, the frame transformation data must be regularly published. This is done using a `static_transform_publisher`.

The base frame of all sensor frames is named "`base_link`". For the mbots this frame is defined as being in the 2D geometrical center of the robots' wheels, at the ground level. Therefore, if the robot is on a perfectly horizontal plane, the point (0,0,0) in the "`base_link`" is the point on the floor that is at the same distance from all of the wheels' centers.

All sensor frames follow the naming pattern "`<sensor_name><sensor_number>`" . Table 2.1 lists all the publicized frames and their posture relative to the "`base_link`" frame.

Besides the sensors positions, the robot's odometry position is published in `tf` as well, with the `base_link`'s posture being published within an `odom` frame.

Frame name	Description	Translation (x,y,z)	Rotation (pitch, roll, yaw)
`imu01`	First inertial measurement unit.	(0, 0, 0)	(0, 0, 0)
`lrf01`	First laser range findes.	(0.3, 0, 0.1365)	(0, 0, 0)
`rgbd01`	First RGB-D camera.	(0.3, 0, 1.0)	(0, 0, 0)
`rgbd02`	Second RGB-D camera.	(0, 0, 0)	(0, 0, 0)
`sonarN`, n=01...12	Nth sonar.	($0.15 \cos(angle)$, $0.15 \sin(angle)$, 0)	(0, 0, angle), angle = $\frac{2\pi(N-1)}{12}$

Table 2.1: Sensor frames.

2.2 Published topics

Data collected from onboard sensors is published in ROS topics. An effort was made to use as much as possible standard message types and topic names in order to make it easier to use existing ROS software with the mbots. Table 2.2 lists the publicized topics and which sensor data they contain. Some topics have custom data types that belong to the `monarch_msgs` package.

2.3 Subscribed topics

mbot_ros expects to receive information from other software in a set of specific topics. These are mainly related to control signals to be sent to the robot's actuators. Table 2.3 lists the expected topics and what is done with the information they contain.

2.4 mbot Kinematics Transformations

The mbot's kinematic model was adapted from [1]. Adjustments were made in order to account for different wheel rotation direction and wheel numbering.

Topic name	Data type	Description
auxiliary_batteries_voltage	AuxiliaryBatteriesVoltage	Readings from auxiliary batteries voltage sensors.
batteries_voltage	BatteriesVoltage	Readings from main batteries voltage sensors.
bumpers	BumpersReadings	Readings obtained from the robots bumpers.
charger_status	ChargerStatus	Readings from charger lines voltage sensors.
ground_sensors	GroundSensorsReadings	Readings from ground sensors.
hardstop_status	HardstopStatus	Status of the low level hardstop.
hokuyo_node/parameter_descriptions	TBD	TBD
hokuyo_node/parameter_updates	TBD	TBD
joy	sensor_msgs::Joy	Readings from USB Joystick.
motor_board_communication_status	MotorBoardCommunicationStatusReadings	Not yet implemented.
motor_board_cooling_fans	MotorsCoolingFans	Motor board cooling fans status.
motor_board_temperatures	MotorBoardTemperatures	Motor board temperature readings.
motor_board_voltages	MotorBoardVoltages	Motor board voltage readings.
odom	nav_msgs::Odometry	Odometry measurements calculated from measured wheel rotation. Note: the robot's posture (the `base_link` frame) is also published using `tf` within the `odom` frame.
relativeHumidity	sensor_msgs::RelativeHumidity	Range measurements collected from the onboard sonars.
sensor_board_communication_status	SensorBoardCommunicationStatusReadings	Not yet implemented.
sonars	sensor_msgs::Range	Range measurements collected from the onboard sonars.
temperature	sensor_msgs::Temperature	Ambient temperature measured by the robot.

Table 2.2: Published sensor topics.

Topic name	Data type	Description
`cmd_vel`	`geometry_msgs::Twist`	Target speeds of the robot in the robot frame (“`base_link`”), published by an automatic control source.
`cmd_vel_manual`	`geometry_msgs::Twist`	Target speeds of the robot in the robot frame (“`base_link`”), published by a manual control source.
`set_state_imu`	`SetStateImu`	Set the Inertial Measurement Unit state.
`set_motor_board_cooling_fans`	`SetStateMotorBoardCoolingFans`	Allow automatic or manual control of the fans to cool the motors and drivers.
`set_state_aux_batt1_power`	`SetStateAuxiliaryPowerBattery`	Enable/Disable/Charge auxiliary power battery 1.
`set_state_aux_batt2_power`	`SetStateAuxiliaryPowerBattery`	Enable/Disable/Charge auxiliary power battery 2.
`set_state_electronics_power`	`SetStateElectronicPower`	Enable/Charge electronic power battery.
`set_state_motors_power`	`SetStateMotorsPower`	Enable/Disable/Charge motors power battery.
`set_state_sonars`	`SetStateSonars`	Enable/Disable sonars.

Table 2.3: Subscribed topics.

Considering that:

- l_1 and l_2 are the x-axis and y-axis distances of the wheel center to the robot frame's origin.
- r is the wheel radius.
- the robot's wheels are number from 1 to 4 in counter-clockwise direction starting from the wheel that is in the robot's front left corner.
- ω_k represents the angular speed of wheel k
- vx, vy and $\dot{\theta}$ represent the robot's instantaneous speeds

$$L = l_1 + l_2 \tag{2.1}$$

2.4.1 Inverse Kinematics

$$\begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{bmatrix} = \frac{1}{r} \begin{bmatrix} -1 & -1 & L \\ -1 & +1 & L \\ +1 & +1 & L \\ +1 & -1 & L \end{bmatrix} \cdot \begin{bmatrix} vx \\ vy \\ \dot{\theta} \end{bmatrix} \tag{2.2}$$

2.4.2 Forward Kinematics

$$\begin{bmatrix} vx \\ vy \\ \dot{\theta} \end{bmatrix} = \frac{r}{4} \begin{bmatrix} -1 & -1 & +1 & +1 \\ -1 & +1 & +1 & -1 \\ \frac{1}{L} & \frac{1}{L} & \frac{1}{L} & \frac{1}{L} \end{bmatrix} \cdot \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{bmatrix} \tag{2.3}$$

Chapter 3

Experimental Results

3.1 Hardware Communication Latencies

In order to assess the appropriate update rates for data acquired from the mbots hardware, we conducted an experiment in which we measured the latency of each of three types of *interactions* with the robots' hardware. We consider an *interaction* to be the actions needed to acquire (or set) an individual piece of data. In this study, we focused on the three interactions that have higher update rates:

GetEncoders Get encoder information in order to calculate odometry.

SetVelocity Set wheel velocities.

GetImu Get IMU information.

We consider that the interaction starts in the instant the software running on the main computer is ready to send data to the hardware and it stops when the complete response has been received and correctly parsed by the software. The experiment duration was of roughly 1 hour.

The results obtained are showed in Figure 3.1 and statistics on this data is showed in Table 3.1. The `GetEncoders` and `SetVelocity` interactions belong to the same hardware board, so they must not occur simultaneously. By adding their 99.9% percentile time we obtain a total time of $0.0202150s$, which would result in a maximum frequency of about $49.46Hz$. As for the IMU, with a value of $0.0209570s$ this would result in a maximum frequency of about $47.7Hz$. In order to give a bit of slack to have a "round" number, it was decided that these devices should work at a frequency of $45Hz$.

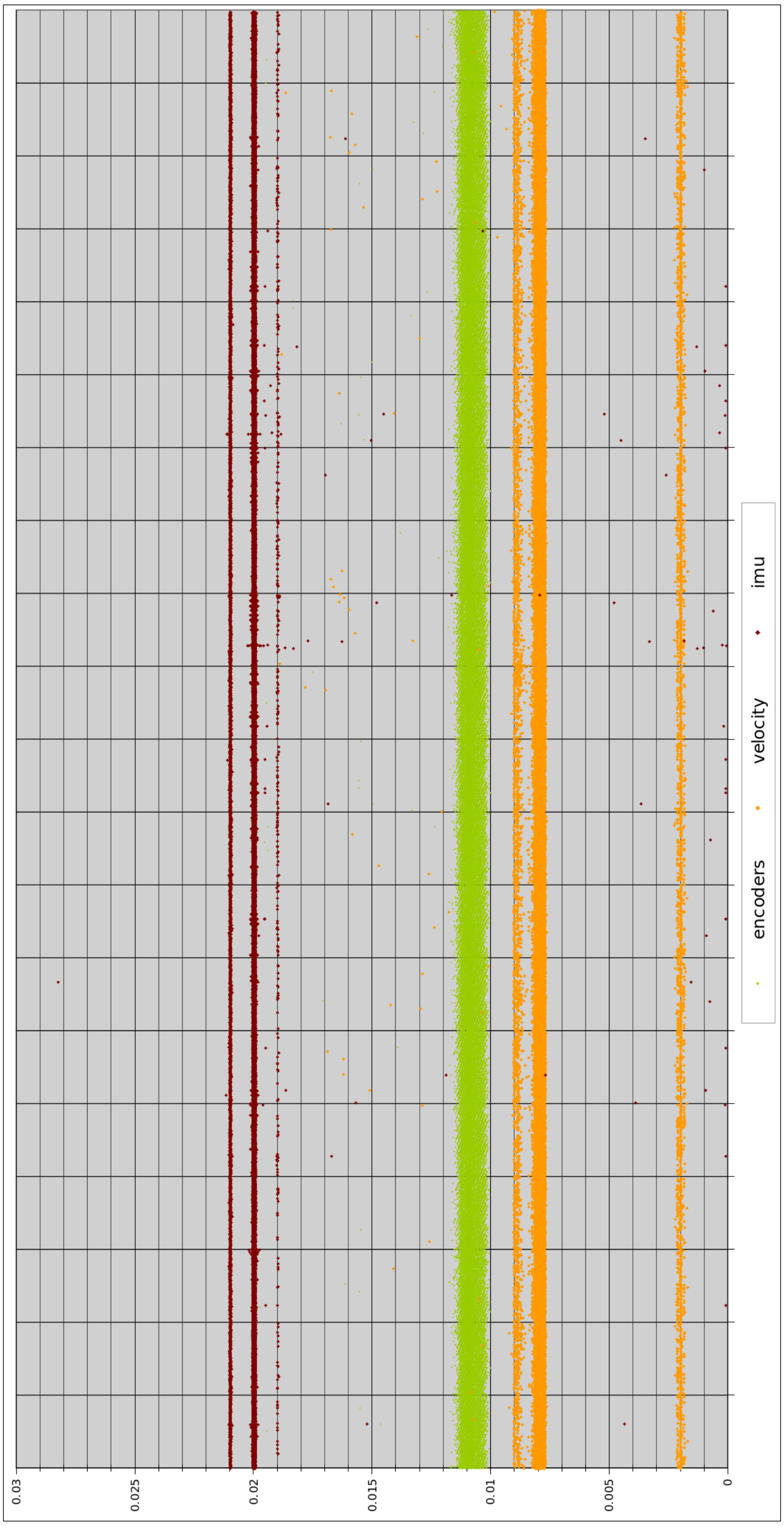

Figure 3.1: Interactions latencies per data type.

	`GetEncoders`	`SetVelocity`	`GetImu`
min	0.0099510	0.0016790	0.0000400
max	0.0196990	0.0188820	0.0282170
avg	0.0108120	0.0078534	0.0199712
stddev	0.0002853	0.0007979	0.0003140
median	0.0108410	0.0079650	0.0199600
99.9% percentile	0.0113580	0.0088570	0.0209570

Table 3.1: Interactions latency statistics per data type.

Bibliography

[1] Ioan Doroftei, Victor Grosu, and Veaceslav Spinu. Omnidirectional mobile robot–design and implementation. *Habib, Maki. Bioinspiration and Robotics, Walking and Climbing Robots. Viena, Austria: I-Tech Education and Publishing*, pages 511–528, 2007.

[2] João Estilita and Marco Barbosa. Monarch - study on partners' current development environments. Technical report, SELFTECH, March 2013.

[3] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, 2009.

[4] Junaed Sattar, Marc Grimson, and James Little. A software framework for multi-robot human interaction.