Grant Agreement No. 611373

FP7-ICT-2013-10

D6.3 Fully customizable system for automatic report generation with auto-alerting capabilities

Due date of deliverable: 30/09/2016

Actual submission date: 15/10/2016

Start date of project: 01 January 2014 Duration: 36 months

Organization name of lead contractor for this deliverable: UNIZG-FER

Revision: version 1

Dissemination level		
PU	Public	x
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Contents

1 Outline of the deliverable

This document describes the fully customizable software package for report generation with auto-alerting capabilities developed for in the scope of the CADDY project. This software package is composed of several parts covering online interaction and data collection duting a mission, misson analysis, mission replay and simulation.

Deliverable is composed of three sections. Section *Control center monitoring – Situational* awareness describes the software subset designated for vehicle monitoring and control, diver monitoring, interaction and alarming in case of safety/health hazard. *Diver data transmitted through acoustics* describes which measured data is used in online diver monitoring and displayed using the *Diver State Monitor*, while the *Chat GUI* is used for diver interaction. *Vehicle monitoring and control* section covers the graphic user interface (GUI) used in monitoring the vehicle behavior and control of various available capabilities. *Data logging and processing* section describes which systems and software is used to log all available data during the mission. Diver related data is stored in the diver tablet, while vehicle related data is stored in the vehicle ROS bag. After the misson logged data can be merged and exported to MATLAB compatible formats allowing easy analysis of data of interest.

For a better picture in mission analysis, ROS data replay and Uwsim are used. *Data replay* and *UWsim* sections describe how these two software packets can be used for a customizable mission analysis giving higher level of situational awareness and a global picture of mission results.

2 Control center monitoring – Situational awareness

2.1 Diver monitoring

2.1.1 Diver data transmitted through acoustics

Acoustic channel low bandwidth defines constraints on how much data can be transferred between the surface, the buddy vehicle and the diver. Apart from buddy vehicle telemetry data, remaining acoustic message payload had to be compressed in as little bits as possible while keeping useful data resolution. Diver monitoring data transferred through the acoustic channel are as shown in table below.

TRANSFERRED DATA	SIZE	DESCRIPTION
Diver depth	7 bits	Depth obtained from the pressure sensor integrated in the acoustic modem
Average flipper rate	4 bits	Rate calculated on the diver tablet using DiverNet inertial sensors mounted on diver flippers
Heart rate	8 bits	Obtained from commercially available Polar heart rate sensor integrated with DiverNet; measured in heart beats per minute
Breathing rate	7 bits	Obtained from Nerites system integrated with DiverNet; measured in breaths per minute
Motion rate	2 bits	Calculated on the diver tablet using the combination of inertial sensor measurements from DiverNet; values displayed are “low”, “medium”, “high”
PAD space	5 bits	Diver state in the pleasure-arousal-dominance space, represented with three values
Diver state alarms	3 bits	Displays alarms if any of the measured or calculated values (breathing rate, heart rate and flipping difference) cross the threshold
Predefined chat	5 bits	Codes for predefined chat messages

2.1.2 Diver State Monitor

Small factor GUI that shows the supervisor real time diver status using several critical values like: heart rate, breathing rate, paddling rate, motion rate, calculated PAD space. These values are calculated by the diver underwater tablet and transmitted periodically through acoustic messaging. There are three alarms sent through acoustic messaging: Breathing rate, Heart rate, Flipping difference; which are generated when the corresponding value surpasses a certain threshold. Each alarm received on the top side is shown as a pop-up dialog to divert the supervisor's attention and reduce response time. Developed using *WxWidgets* and as an RQT plugin, it can be run on any machine with ROS installed. When run as standalone GUI it enables two graphic modes:

I. **Compact view** – small factor window, indented to be kept on top (or on the side) of a mission control GUI. Shows heart rate, breathing rate, paddling rate, motion rate, calculated PAD space current values. Compact view enables the operator to work in the mission control GUI while still being able to keep watch of the diver status. Compact view is shown in Figure 1.

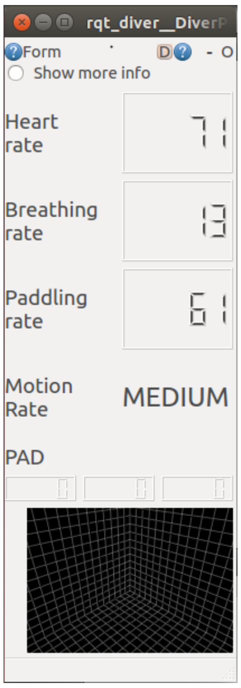
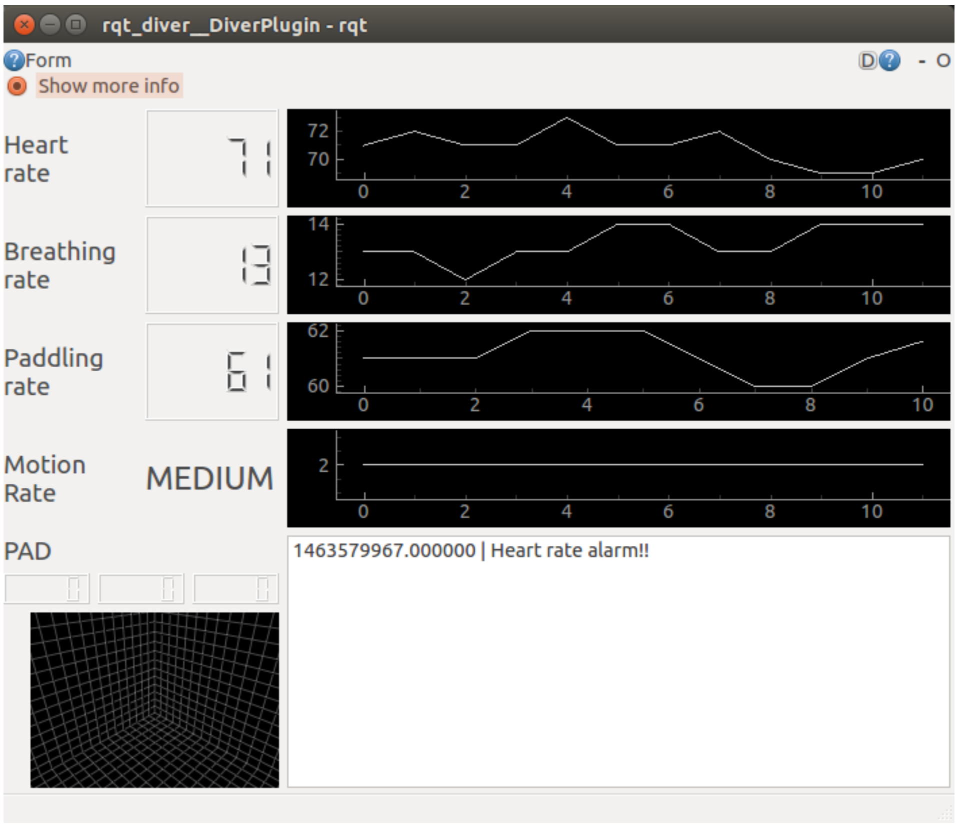


Figure 1: Compact view of the diver state monitor.

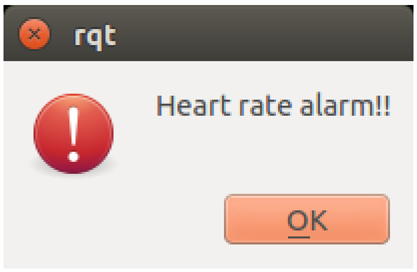
II. **Extended view** – extends the compact view with graphs showing heart rate, breathing rate, paddling rate, motion rate trends and alarms history list containing alarm type and corresponding timestamps. Extended view is shown in Figure 2.



1463579967.000000 | Heart rate alarm!!

Figure 2: Extended view of the diver state monitor.

Pop-up dialog windows are generated for each alarm to divert supervisor's attention to diver status and potential health problem. An example of the pop-up dialog window is shown in Figure 3.



Heart rate alarm!!

Figure 3: Alarm pop-up dialog.

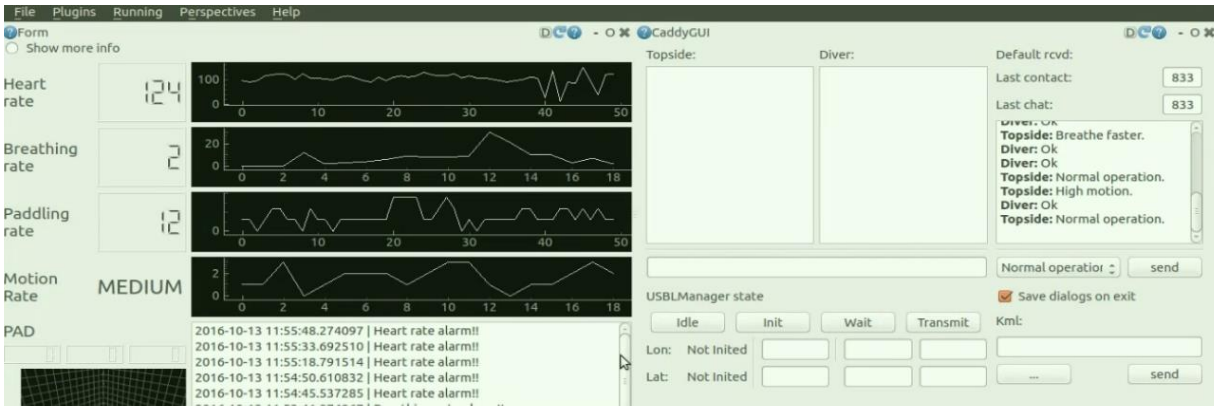
<u>Testing and validation summary:</u>

<u>Diver state values:</u>
Heart rate (bpm) - validated
Breathing rate (bpm) - validated
Paddling rate (ppm) - validated
Motion rate - validated
PAD space – tested with simulation results

<u>Diver state alarms:</u>
Breathing rate - validated
Heart rate - validated
Flipping difference - validated

2.1.3 Chat GUI

Graphic interface used for communication between the surface operator and the diver provides an intuitive way for the operator to issue commands to the diver, request tasks and check the diver status. Chat GUI Developed as an RQT plugin and it can be run on any machine with ROS installed. In Figure 4, Chat GUI is shown inside the RQT scope adjacent to the Diver State Monitor together making a more powerful diver monitoring and interaction tool.



```
2016-10-13 11:55:48.274097 | Heart rate alarm!!
2016-10-13 11:55:33.692510 | Heart rate alarm!!
2016-10-13 11:55:18.791514 | Heart rate alarm!!
2016-10-13 11:54:50.610832 | Heart rate alarm!!
2016-10-13 11:54:45.537285 | Heart rate alarm!!
```

Topside: Breathe faster.
Diver: Ok
Diver: Ok
Topside: Normal operation.
Topside: High motion.
Diver: Ok
Topside: Normal operation.

Figure 4: Chat and Diver State Monitor combined

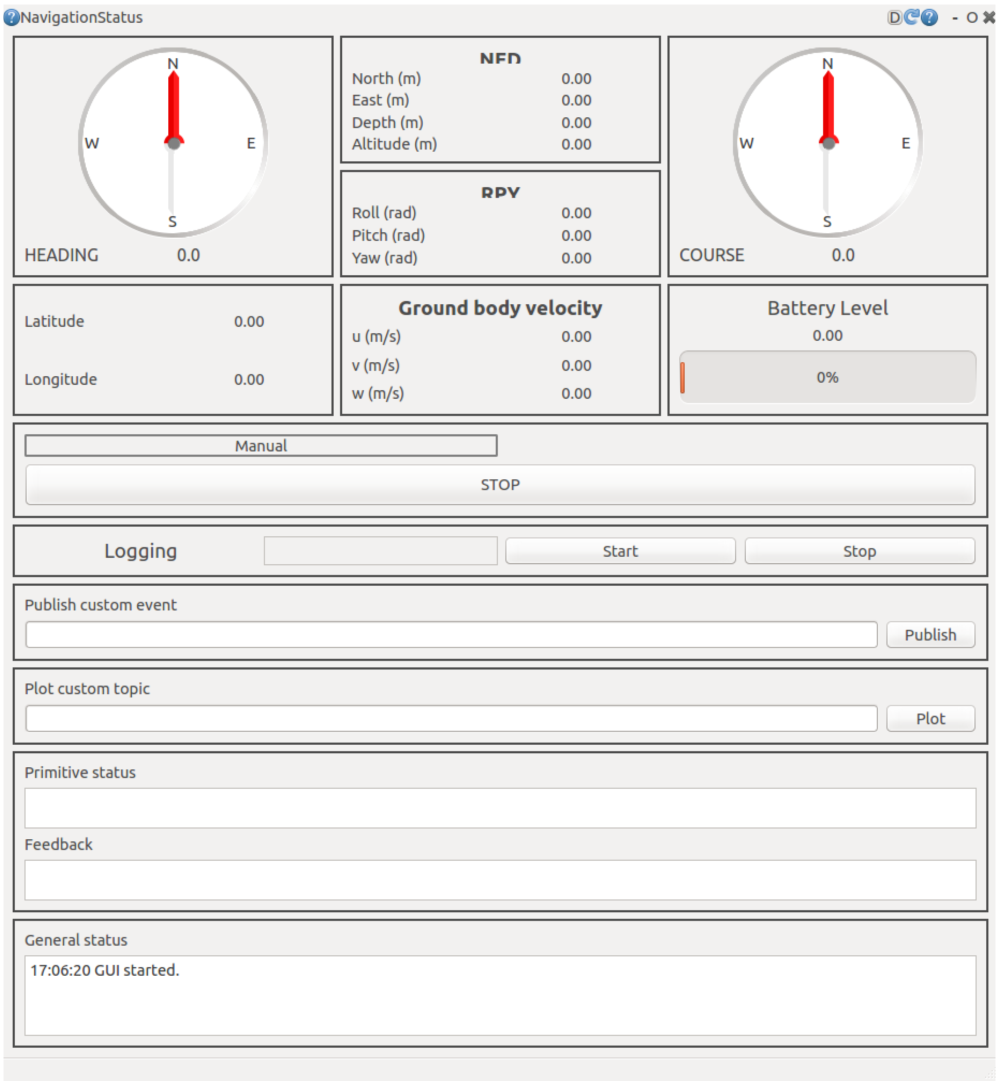
Communication is enabled using a predefined set of messages as follows (indexed as in the predefined messages list):

SURFACE TO DIVER MESSAGES:	**DIVER TO SURFACE MESSAGES:**
1 - Are you OK?	8 - "Alert"
2 - Flip with only one flipper.	9 - "Ok"
3 - Normal operation.	10 - "Yes"
4 - Breathe faster.	11 - "No"
5 - High motion.	12 - "Repeat instruction"
6 - Issue alarm.	13 - "Diving out"/"Found Object"
7 - Acknowledged.	
14 - Dive out.	
15 - Follow BUDDY.	

2.2 Vehicle monitoring and control

One of the project tasks was to develop a system that allows consumers to build a modular application which displays data of interest during mission execution and displays alerts generated by active vehicles. Every entity in the system is accessible to the consumer through different widgets that allows acquisition, processing or manipulation of data. All widgets are implemented as Robot Operating System (ROS) rqt plugins which can be connected together into different perspectives which are deployed depending on application and consumers preferences.

The collection of gadgets will include data acquisition gadgets which acquire data from diver's tablet, AUV, USV, generation of table reports, data reports (plots, charts), and logic widgets which enable specification, detection and reaction to specific user defined conditions. Such widget based graphical user interface (GUI) is shown in Figure 5. It shows navigational information about deployed vehicles to the ground operator. It enables plotting of custom data, and alerts ground station about important events. In the case of the emergency switching to manual control can be achieved through the interface. The user interface also lists active primitives and gives all primitive feedback information to the ground operator.



17:06:20 GUI started.

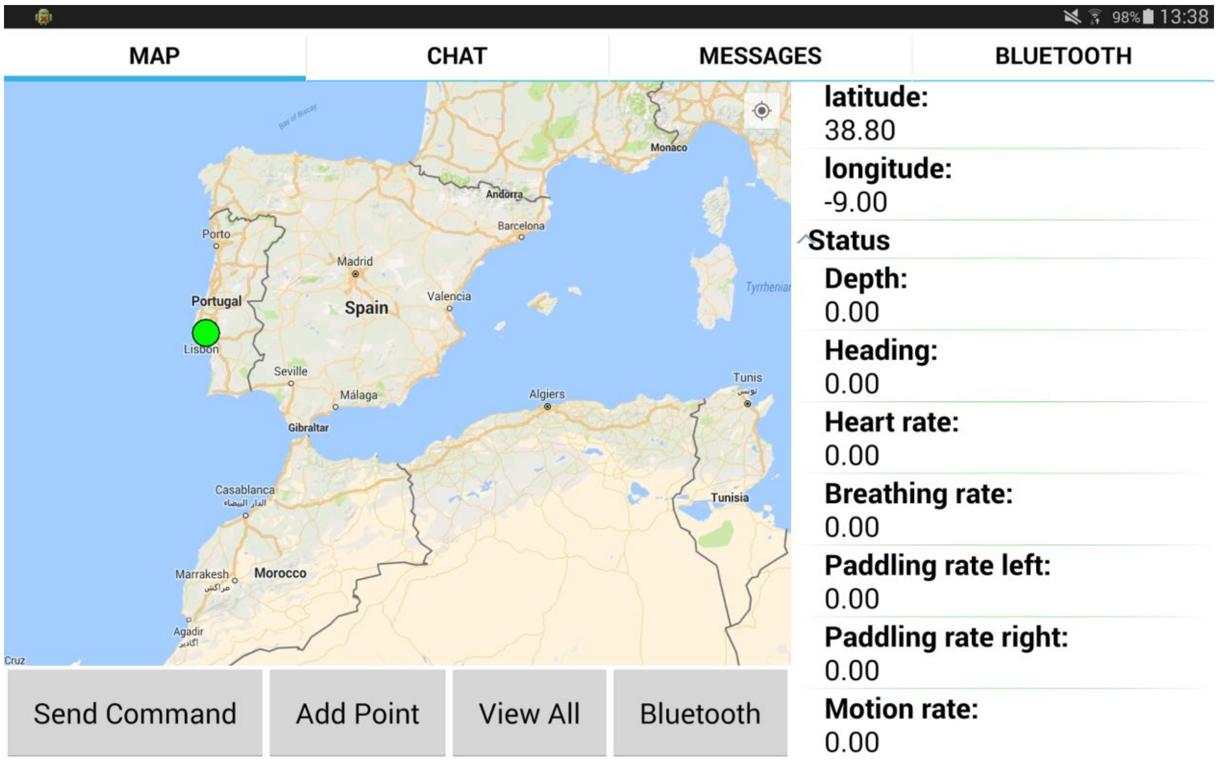
Figure 5: Vehicle monitoring and control

2.3 Data logging and processing

2.3.1 Diver tablet logging

The diver tablet logs both raw data received from the DiverNet sensors and the processed data it uses to provide real-time information to the diver and later acoustically transmits to the other agents in the system and finally the surface station. The logs are in a widely-used CSV format, suitable for post-processing in MATLAB and similar tools. The tablet also creates separate logs for the magnetometer readings of the DiverNet heading sensor (the sensor placed on the diver's chest) for heading calibration purposes.

The diver tablet hosts an FTP server, which allows all logs to be fetched from the surface station once the diver has surfaced and the tablet has automatically reconnected to a Wi-Fi network. The logs are then processed and converted to ROS-compatible and replayable bag files using a simple Python script, converting measurements into timestamped ROS topics and making them ready for use with the mission replay software.



latitude:
38.80
longitude:
-9.00
Status
Depth:
0.00
Heading:
0.00
Heart rate:
0.00
Breathing rate:
0.00
Paddling rate left:
0.00
Paddling rate right:
0.00
Motion rate:
0.00

Figure 6: Example view of the diver tablet display

The tablet itself carries out the gathering and pre-processing of the DiverNet sensor data. It calculates the diver's heart rate by extrapolating from the recorded periods between positive pulses from the digital sensor on the diver's torso. In order to account for false positives, there is a minimum threshold placed on pulse readings which disregards positive pulses that are immediately one after the other, as this is anatomically impossible.

Breathing rate is calculated using a simple falling edge detection algorithm applied to the readings from the pressure sensor included in the DiverNet. As the diver draws breath, the pressure sensor displays a sharp drop. The breathing rate is then calculated using measured time periods between each detected breath.

Paddling rates for both flippers are calculated by applying a FFT to the accelerometer measurements of the DiverNet nodes placed on the diver’s feet. If there is a difference between the left and right flipper higher than a preset threshold, the tablet reports an alarm to the surface - the assumption being that one of the sensors on the flippers might have fallen off, or the diver is experiencing difficulties. Only the average flipper rate is reported to the surface, in the interest of minimizing the size of the acoustic payload.

Diver motion rate is calculated by applying a similar FFT algorithm to readings from the DiverNet nodes placed on the hands, feet, and head of the diver and calculating an average, then thresholding this measurement in order to report motion rate as a simple measure of 0 - No Motion, 1 - Low Motion, 2 - Medium Motion, and 3 - High Motion. This allows for simple detection of sudden sharp or twitchy movements on the diver’s part that might indicate distress.

2.3.2 ROS based logging and processing – MATLAB and ROS bag

Main difficulty which arises when logging information from multiple robots during a mission is timestamp synchronization. In ROS it is possible to log all data on some centralized location when Ethernet or Wi-Fi communication is available, however that puts a lot of strain on communication channel. Furthermore, when we are dealing with underwater robotics usually the only mean of communication is over acoustic channel which is not suitable for sending such information. Due to that every vehicle logs its own data aboard its own CPU, and these logs are then merged together in post processing phase using custom rosbag command line tool called “rosbag_filter”.

It enables merging multiple bags into one bag file. There are multiple additional options, like selecting which topics to merge, adding namespaces, compensating for possible errors in clock synchronization between multiple vehicles or selecting time windows to merge. The tool is called from command line interface using simple commands like: “rosbag_filter -o ‘output.bag’ -b ‘input1.bag’ ‘input2.bag’”. All available parameters are described in Table 1.

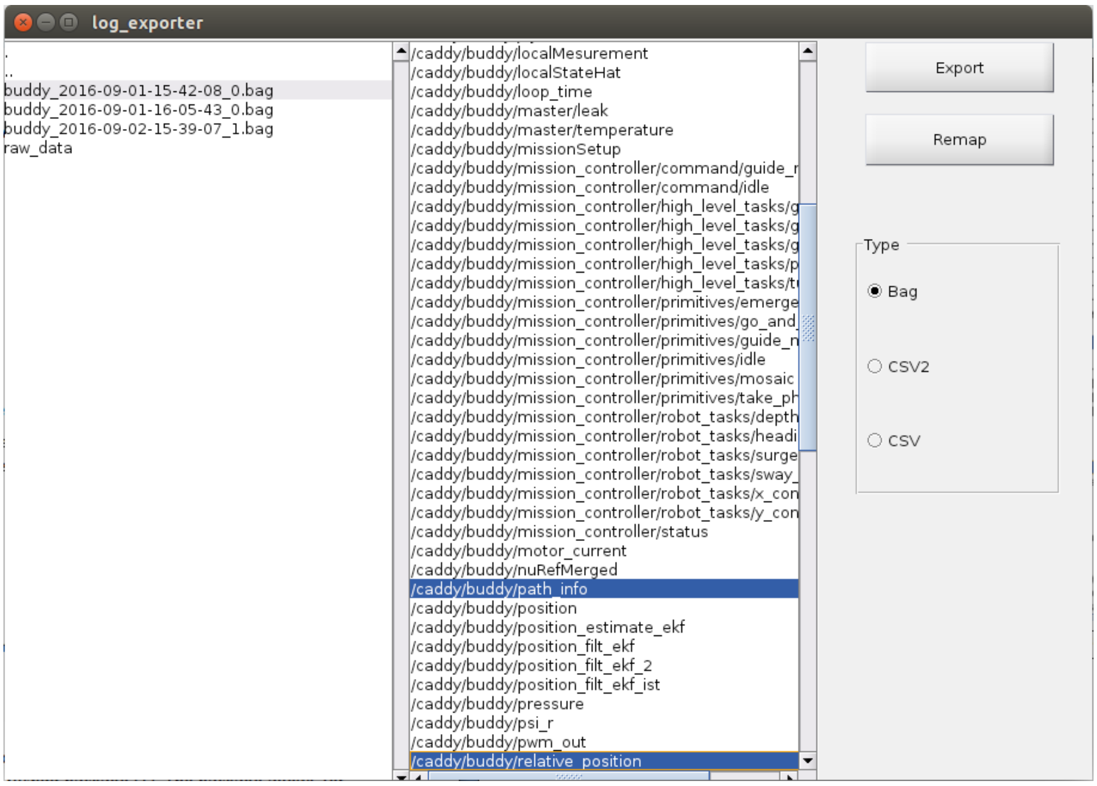
Table 1: rosbag_filter command line parameters

Parameter	**Description**
-b	List of input bag names to merge.
-t	List of topics to use when merging.
-ns	List of namespaces to append when merging
-delay	List of input bags delay values.
-o	Output bag name.
-start_time	Start merging from timestamp.
-end_time	End merging from timestamp.

All this features enable constructing different bags with different information depending on the intended usage. Locally aboard the vehicle we record all the data and if we want to check only navigation filter data, we can easily filter other data that is not important at the moment, e.g. sonar or camera images, which speeds up data processing in other software like MATLAB.

MATLAB is the main tool used for log post-processing and analysis and it supports reading of ROS bags through the *matlab_rosbag* package (1). The package allows for basic access to bag files and the main log analysis framework for MATLAB is built on top of this package. Alternate way of importing ROS data into MATLAB includes bag reply with combined conversion of individual topics into **.csv* files. This can be achieved by using options of the *rostopic* executable provided with ROS for topic introspection. However, this results with many data preparation step which can be inconvenient when multiple topics are required. The developed MATLAB scripts consist of two parts: the exploratory part providing a GUI for easy topic selection and exporting and the post-processing part providing scripts for automatic topic extraction, resampling and preparation for deeper analysis.

Initial analysis can be performed by means of the log_exporter application, see Figure 7, which allows selecting topics of interest. Once selected, the topics can be exported into MATLAB as raw cell data or be remapped as standard vector data. As shown on the figure the user interface consists of two columns. The leftmost column is used for selecting desired files, a choice between two CSV versions and ROS bag file exists. The second column displays topics available in this bag file allowing the user to choose which topics are interesting. Once selected, the topics can be "Exported" as is provided from the matlab_rosbag package. Alternatively, a remapping procedure can be carried out to provide the numerical data in a standard matrix format. The data is structured under a common element with each topic expanding as a set of fieldnames in the log super structure. In this way the class nature of ROS topics is maintained even in ROS.



```
.
..
buddy_2016-09-01-15-42-08_0.bag
buddy_2016-09-01-16-05-43_0.bag
buddy_2016-09-02-15-39-07_1.bag
raw_data
```

```
/caddy/buddy/localMesurement
/caddy/buddy/localStateHat
/caddy/buddy/loop_time
/caddy/buddy/master/leak
/caddy/buddy/master/temperature
/caddy/buddy/missionSetup
/caddy/buddy/mission_controller/command/guide_r
/caddy/buddy/mission_controller/command/idle
/caddy/buddy/mission_controller/high_level_tasks/g
/caddy/buddy/mission_controller/high_level_tasks/g
/caddy/buddy/mission_controller/high_level_tasks/g
/caddy/buddy/mission_controller/high_level_tasks/p
/caddy/buddy/mission_controller/high_level_tasks/t
/caddy/buddy/mission_controller/primitives/emerge
/caddy/buddy/mission_controller/primitives/go_and
/caddy/buddy/mission_controller/primitives/guide_n
/caddy/buddy/mission_controller/primitives/idle
/caddy/buddy/mission_controller/primitives/mosaic
/caddy/buddy/mission_controller/primitives/take_ph
/caddy/buddy/mission_controller/robot_tasks/depth
/caddy/buddy/mission_controller/robot_tasks/headi
/caddy/buddy/mission_controller/robot_tasks/surge
/caddy/buddy/mission_controller/robot_tasks/sway_
/caddy/buddy/mission_controller/robot_tasks/x_con
/caddy/buddy/mission_controller/robot_tasks/y_con
/caddy/buddy/mission_controller/status
/caddy/buddy/motor_current
/caddy/buddy/nuRefMerged
/caddy/buddy/path_info
/caddy/buddy/position
/caddy/buddy/position_estimate_ekf
/caddy/buddy/position_filt_ekf
/caddy/buddy/position_filt_ekf_2
/caddy/buddy/position_filt_ekf_ist
/caddy/buddy/pressure
/caddy/buddy/psi_r
/caddy/buddy/pwm_out
/caddy/buddy/relative_position
```

[✓ Bag] CSV2 CSV

Figure 7: ROS bag exporter (log_exporter) user interface

Once exported in a matrix format the data can be plotted with MATLAB tools. Scripts for plotting position, orientation, velocities and forces are already predefined while more specific analysis plots should be provided manually for each domain.

This type of semi-automatic bag analysis can become tedious when multiple or large bag files are involved. Therefore, a more automatic method for extraction can be used. It requires specification of bag files, time frames and topics of interest. These are specified in a XML format providing documentation about the data preparation procedure for experiment repeatability.

Listing 1 shows an example of the XML format. The bag XML tag specifies basic information about each bag file that has to be processed. The attributes include the filename, output filename and start and end dates of interest. For each bag file topics or a topic group (tgroup) can be specified. For topics that are missing ROS header timestamps the meta_time attribute can be specified in order to create the header timestamps based on the time the data itself was logged.

Such XML files, in addition to document the bag file data, are used by the created bag2mat.m MATLAB function. The function loads the XML data and extracts the required data from each bag file. The function utilizes export_bag.m and bagstruct_resample.m MATLAB functions. The first function reads the bag file data and creates the superstructure containing individual topics. Unsupported characters, e.g. '/', are replaced to fit the MATLAB fieldname naming scheme. Once the bag structure is created, the second function performs data resampling on the timebase of the desired main variable. All ROS topics in the bag structure have their individual timestamps in the topic header. Additionally, topics have different sampling intervals, meaning that vector lengths are not the same. Most plotting and data manipulation functions expect same vector lengths and a common time base for all data. Therefore, by selecting a common time base, e.g. the main state estimate, all topics can be resampled to have same length and a common timestamp. Interpolation is done using a zero-order hold to avoid introducing false data.

```xml
<filter_info>
  <tgroup name="common">
    <topic name="/caddy/buddy/position_filt_ekf" />
    <topic name="/caddy/buddy/nuRefMerged" />
    <topic name="/caddy/diver/position_filt_ekf" meta_time="true" />
    <topic name="/caddy/buddy/path_info" />
    <topic name="/caddy/diver/camera_heading" />
  </tgroup>

  <!-- Rotation 1 -->
  <bag file="buddy_2016-09-01-15-42-08_0.bag"
      out="buddy_2016-09-01-15-42-08_0_rotation1.bag"
      dstart="01-Sept-2016 15:58:03"
        dend="01-Sept-2016 16:00:34">
    <tgroup name="common" />
  </bag>

  <!-- Rotation 2 -->
  <bag file="buddy_2016-09-02-15-39-07_1.bag"
      out="buddy_2016-09-02-15-39-07_1_rotation2.bag"
      dstart="02-Sept-2016 16:19:42"
        dend="02-Sept-2016 16:24:00">
    <tgroup name="common" />
  </bag>

  <!-- Movement -->
  <bag file="buddy_2016-09-01-16-05-43_0.bag"
      out="buddy_2016-09-01-16-05-43_0_movements.bag"
      dstart="01-Sept-2016 16:11:10"
        dend="01-Sept-2016 16:14:00">
    <tgroup name="common" />
  </bag>
</filter_info>
```

Listing 1: Example of the XML specification for bag file parsing

Finally, both the resampled and raw data is saved in the corresponding mat file to allow easier post-processing without requiring users to install or utilize ROS tools.

3 Data replay

Previous section described how data logging is performed and the tools developed to analyze and extract the data of interest. This data can be used to generate plots and statistics from within MATLAB. However, a more visual insight is preferred for data replay as part of the generated report. By visualization more information is presented to the user analyzing the report than is the case with plots and statistics. Additionally, the dynamic relation between the data is better preserved. For visualization two approaches where analyzed. The rqt_bag approach combines plots and imaging data with a classical timeline capability. The UnderWater SIMulator (UWSIM) provides a complete 3D replay of all agent movements. Note that combining both approaches provides 1D, 2D and 3D representations of the available data.

3.1.1 Rqt_bag

The rqt_bag utility can be used to load one or more bag files and merge them in the timeline. One example of the setup is shown in Figure 8. The available topics are listed on the left and each can be individually presented in raw, plot or image version. Additionally, topics can be published into the ROS system that is running in the background. This allows connecting other utilities and provides them with emulated real-time data from recorded bag files.

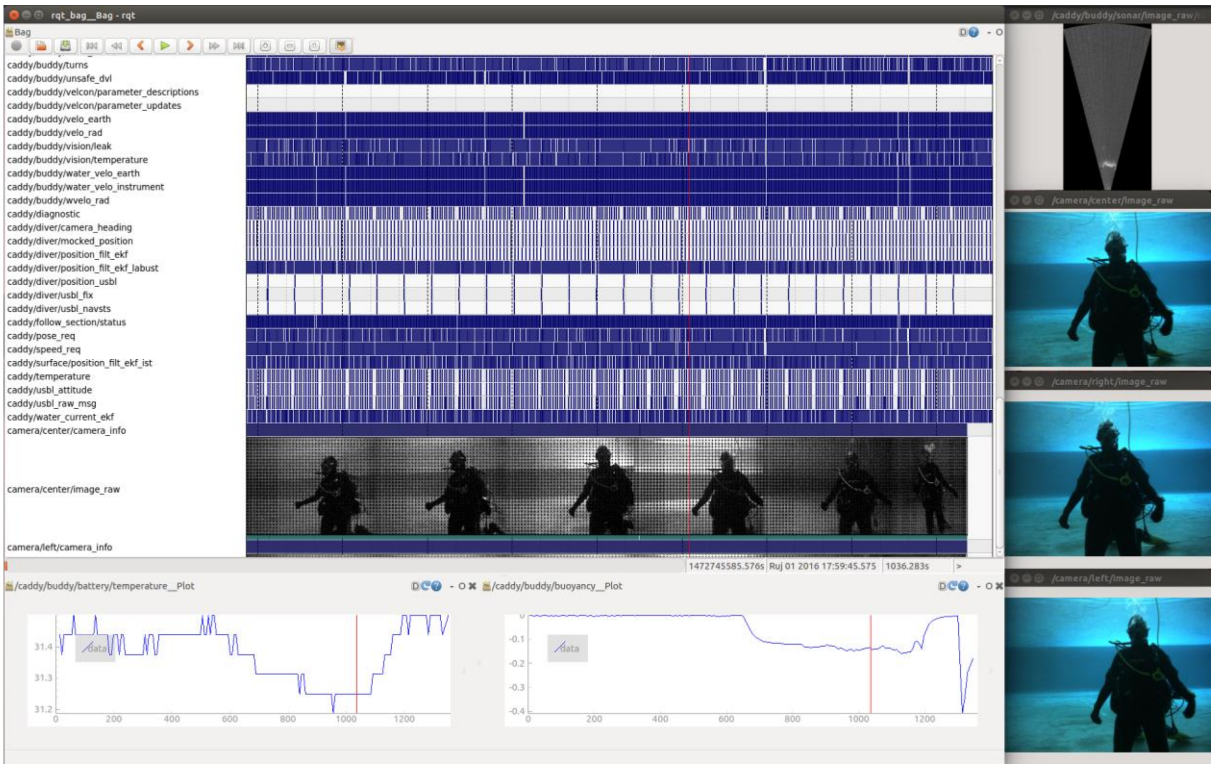

rqt_bag__Bag - rqt
Bag
caddy/buddy/turns
caddy/buddy/unsafe_dvl
caddy/buddy/velcon/parameter_descriptions
caddy/buddy/velcon/parameter_updates
caddy/buddy/velo_earth
caddy/buddy/velo_rad
caddy/buddy/vision/leak
caddy/buddy/vision/temperature
caddy/buddy/water_velo_earth
caddy/buddy/water_velo_instrument
caddy/buddy/wvelo_rad
caddy/diagnostic
caddy/diver/camera_heading
caddy/diver/mocked_position
caddy/diver/position_filt_ekf
caddy/diver/position_filt_ekf_labust
caddy/diver/position_usbl
caddy/diver/usbl_fix
caddy/diver/usbl_navsts
caddy/follow_section/status
caddy/pose_req
caddy/speed_req
caddy/surface/position_filt_ekf_ist
caddy/temperature
caddy/usbl_attitude
caddy/usbl_raw_msg
caddy/water_current_ekf
camera/center/camera_info
camera/center/image_raw
camera/left/camera_info
1472745585.576s Ruj 01 2016 17:59:45.575 1036.283s
/caddy/buddy/battery/temperature__Plot
/caddy/buddy/buoyancy__Plot
data
/caddy/buddy/sonar/image_raw
/camera/center/image_raw
/camera/right/image_raw
/camera/left/image_raw

Figure 8: Bag file analysis example

On the right the acoustic and stereo camera images can be observed during the playback, while on the bottom some plots of interesting variables are shown. Note the red vertical guideline which indicates the corresponding time between plots and the main timeline. From the timeline different topic rates can be easily observed. Visual data can be represented in thumbnails for easier navigation along the timeline.

4 UWSim

Just after the end of a mission, it is really useful to replay the mission data in the console so that one can have a global picture of the success of the mission. A diagram with the architecture can be seen in Figure 9.

In order to achieve so, the first step is to download the data from the diver tablet and both robotic vehicle, surface and underwater. This is done by a simple script on the console computer that uses a file synchronization protocol such that it downloads the latest data.

While in the vehicles ROS is running, thus ROS bags are already ready to be retrieved, on the diver tablet this is not true. The logging on the tablet is done to a txt file, which is downloaded and converted to a bag on the shore computer. When the 3 bag files are ready the same script that grabs the data sends launches UWSim.

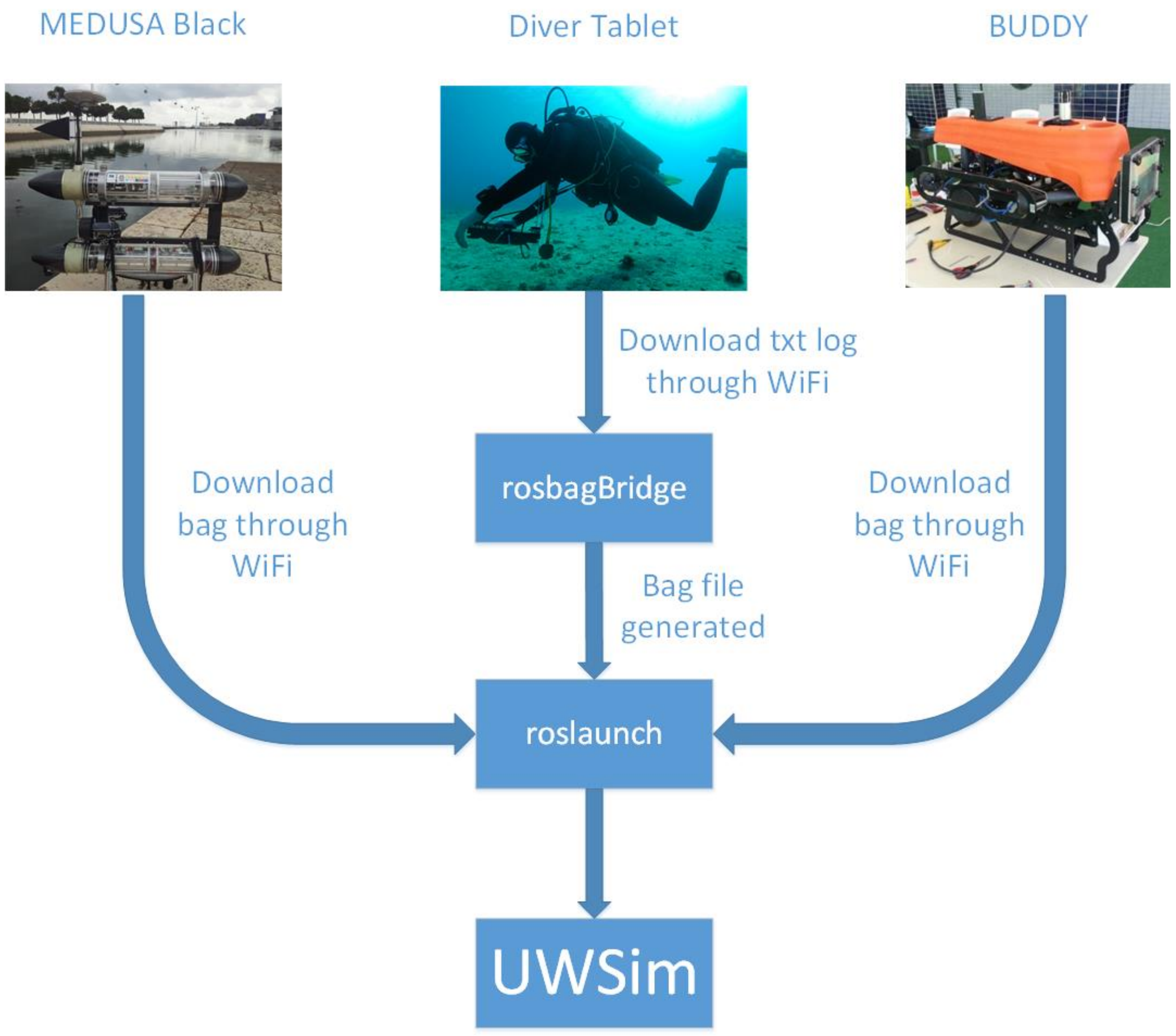

MEDUSA Black
Diver Tablet
BUDDY
Download txt log through WiFi
Download bag through WiFi
rosbagBridge
Download bag through WiFi
Bag file generated
roslaunch
UWSim

Figure 9: The implemented architecture for merging the obtained data.

5 References

1. MATLAB support for ROS bag files. [Online] https://github.com/bcharrow/matlab_rosbag.