A New ROS-Based Operation Software for an Autonomous Underwater Vehicle HOBALIN and Its Test in Real Sea

Takahiro Seta
Offshore Energy and Underwater Technology Department
National Maritime Research Institute
National Institute of Maritime, Port and Aviation Technology
Tokyo, Japan
seta@nmri.go.jp

Akihiro Okamoto
Offshore Energy and Underwater Technology Department
National Maritime Research Institute
National Institute of Maritime, Port and Aviation Technology
Tokyo, Japan
okamoto@nmri.go.jp

Shogo Inaba
Offshore Energy and Underwater Technology Department
National Maritime Research Institute
National Institute of Maritime, Port and Aviation Technology
Tokyo, Japan
inaba-s@nmri.go.jp

Masahiko Sasano
Offshore Energy and Underwater Technology Department
National Maritime Research Institute
National Institute of Maritime, Port and Aviation Technology
Tokyo, Japan
sasano@nmri.go.jp

Abstract—**The software system for a hovering-type AUV named HOBALIN was replaced by newly developed system based on ROS, which is a widely-used robot software framework, and the AUV completed dives in the real sea almost successfully with the new software. An important objective of the software replacement was to make it easier to import new or experimental features, and some new features are also tested in the dives done in the real sea. Therefore, in this paper, the design of the new software and the results of the dives are reported.**

Keywords—ROS, Robot Operating System, AUV, software

I. INTRODUCTION

A hovering type AUV (Autonomous Underwater Vehicle) HOBALIN was developed in 2015 [1], as a part of a national project of Japan named "Next-Generation Technology for Ocean Resources Exploration" and nicknamed "Zipangu-in-the-Ocean" [2][3].

The AUV has been in work already [4]. However, since the development of the AUV was done in hurry, the operation software (in this paper, the word "operation software" does not mean the software like Windows or Linux, but the basic software system to be used for AUV operation or control) was based on an old design and was not fully matched to the AUV. It was enough for simple survey missions, however, the way in which the AUV acted sometimes does not match with what the authors expected and implementation of new features of functions were sometimes difficult or required much effort and large risk.

On the other hand, it is being popular for recent robots or AUVs to be developed on common frameworks like ROS (Robot Operating System, [5]), MOOS (Mission Oriented Operating System, [6]), etc. These common frameworks make development and maintenance of AUVs much easier. Software modules can be shared among multiple kind of AUVs relatively easily. Software engineers can easily apply the experience on one robot or AUV to another robot or AUV, or can easily take charge of multiple kinds of AUVs. Therefore, science on the ocean or development of the ocean will be boosted with these common frameworks.

In this situation, the authors decided to replace the operation software of the AUV HOBALIN to newly designed one based on the ROS [7], and the development has reached at the stage of real sea experiment on December 2017. Furthermore, the AUV completed real survey missions on February 2018.

In this paper, some characteristic examples of the detailed design of the new software and the results of the dives with the new software are reported.

II. THE TARGET AUV

In order to design the software suitable for the target AUV, HOBALIN, we need to see the characteristics of the AUV.

The appearance of the AUV as of the survey cruise on February 2018 is shown in Fig. 1.

Because one of the main purpose of the survey cruise was to test the newly developed acoustic modem with a positioning function, extra payloads were held both sides of the AUV (the transducer on the left side of the AUV and the electronic part on the right side). The normal modem was also left at the top middle of the front half. In addition, because the extra payloads are heavy, extra buoyancy material were added with belts. The AUV sometimes works as a testbed like this and needs flexibility.

The typical operation flow of the AUV is like Fig. 2.

The main observation instruments of the AUV are cameras. Two still cameras are equipped below the AUV and a video camera has been added from this cruise (however, lights for the video camera had not been equipped). Some stand-alone chemical sensors can also be loaded. In order to take high-resolution photos with an enough density, the typical request to the AUV is to cruise at 3 meter altitude and 0.2 m/s speed along the defined survey lines precisely and stably.

This research is supported by Council of Science, Technology and Innovation (CSTI), Cross-ministerial Strategic Innovation Promotion Program (SIP), "Next-Generation Technology for Ocean Resources Exploration" (lead agency: Japan Agency for Marine-Earth Science and Technology, JAMSTEC).

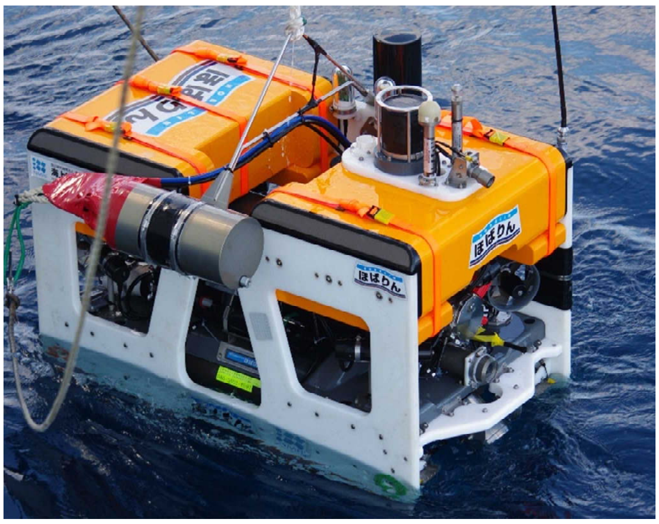
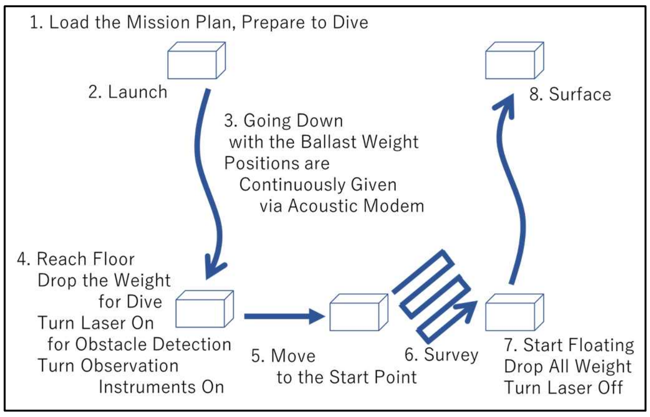


Fig. 1 AUV HOBALIN after the dive (heading to the right).


1. Load the Mission Plan, Prepare to Dive
2. Launch
3. Going Down with the Ballast Weight Positions are Continuously Given via Acoustic Modem
4. Reach Floor Drop the Weight for Dive Turn Laser On for Obstacle Detection Turn Observation Instruments On
5. Move to the Start Point
6. Survey
7. Start Floating Drop All Weight Turn Laser Off
8. Surface
Fig. 2 Typical AUV operation flow.

III. The ROS Framework

ROS (Robot Operating System) is probably the most popular open-source framework for the robots (on the land and in the air). Some AUVs seem to use this framework too.

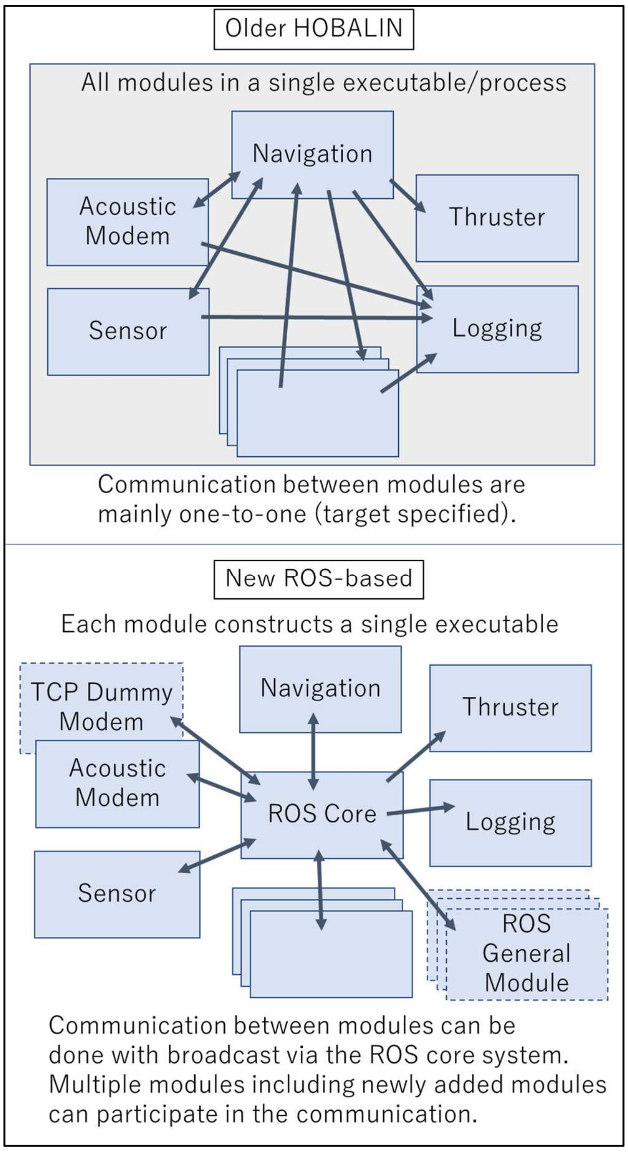
The most important idea on this framework is "publish-subscribe" or "message-passing" concept. Most data or messages are broadcasted (published) and any module can subscribe them (see Fig. 3 in the next section). This type of idea is not unique to ROS but popular among other frameworks or technologies, and has been proved to be successful. Some frameworks for AUVs like MOOS and LSTS toolchain [9] and some standards like CAN (Controller Area Network, [10]), which is a widely used standard by cars, and NMEA (National Marine Electronics Association) 0183 standard [11], which is the format GPS instruments use, also employ this type of idea.

One of the strong points of this type of idea is that it is easy to add new modules or to replace modules without changing other modules. This makes development and maintenance of the AUV easier and more efficient.

As one of other strong points of ROS, we can find out that ROS has a large community and many open-source libraries. This will also let the software maintenance easier and boost the development speed.

IV. New Software Features of the AUV

Largest part of the motivation of replacing operating software of the AUV using ROS is that it makes development and maintenance easier. Therefore, the authors implemented some new features together with this software replacement. (Note that what the authors did was "software replacement" and not just "porting to ROS". The software was almost newly designed and developed.) Most of these features became possible, or at least became easier and faster to be implemented by the ROS structure as shown in Fig. 3.


Older HOBALIN
All modules in a single executable/process
Navigation
Acoustic Modem
Thruster
Sensor
Logging
Communication between modules are mainly one-to-one (target specified).
New ROS-based
Each module constructs a single executable
TCP Dummy Modem
Navigation
Thruster
Acoustic Modem
ROS Core
Logging
Sensor
ROS General Module
Communication between modules can be done with broadcast via the ROS core system. Multiple modules including newly added modules can participate in the communication.
Fig. 3 Software structure of old and new HOBALIN.

A. *Video Camera Support*

One part of the most important new hardware added from the survey dive is a video camera. Though hardware trouble (noise problem probably caused by the dimmer module) made it unrealized, LED lights were also planned to be equipped. Therefore, software module to support them needed to be prepared.

The requirement for the video camera is just powering on at system wake up and the requirements for the LED lights were powering up at floor reached, powering down at leaving floor (or reaching surface) and manual powering on/off. Additionally, a function of temporal power off on shutter timings of the still cameras was also planned (but not implemented because the LED lights were removed before the effect to the photograph is checked by the noise trouble).

These requirements were easily achieved with the publish-subscribe structure of ROS. Because messages for events like “reach floor”, “start floating” and “reach surface” had already existed to be broadcasted by a module named navigation, and binary and analog output service module had already existed waiting service request messages, the modules for the video camera and LED lights can use those messages. Therefore, it was proved the easiness of preparing software to support these kind of new observation instruments. (Moreover, we may say it was also proved the easiness of removing modules or changing system configurations by having stopped the software module for the LED lights as an accident.)

Though it has not yet been done, the feature of temporal LED lights power off can be implemented easily, because an almost same feature is already working with the downward laser which is for altitude and floor shape observation. However, it needs change on already existing still camera support module, which is against the strength of the ROS structure. One possible design for avoiding this kind of effect is to prepare a message like “shutter is going to work, so turn lights down for 200 msec”, but it might be difficult design because message-passing cause delay. A little more time is necessary to decide the best way.

B. *Newly Developed Acoustic Modem & TCP Dummy Modem*

Another part of the most important new hardware at the survey dive is a newly developed acoustic modem with positioning function. Testing this hardware in the real environment is the necessary mission for the AUV in the survey cruise.

Generally it is necessary to change software to handle new hardware, however, this was not required this time by fully using the ROS structure. Especially, a module named “TCP dummy modem” had played a big role, which became possible to be realized with the ROS structure. Fig. 4 shows the data flow around the acoustic modems, which are newly developed one, the original one and also the TCP dummy modem. The red colored parts represent temporary changes applied in order to use the newly developed acoustic modem.

Because broadcasting is the key idea of the ROS structure, data to be sent via the (original) acoustic modem are also broadcasted in the AUV. Therefore, we can listen the acoustic data to be sent simultaneously via non-acoustic pass. In addition, we can send messages designed to be communicated via acoustic pass directly into the AUV via non-acoustic pass. This is what the TCP dummy modem does, it just works as a kind of bridge between the TCP network outside the AUV and the ROS world inside the AUV. This is useful for testing new acoustic protocols, and it is the original reason why this module was developed. The newly developed acoustic modem were connected to the ROS world via this module as a diversion. This might not have been the best way if there had been enough time, however, this should have been the fastest and reliable enough way this time.

One more interesting point is in the right side of Fig. 4. The output to the original acoustic modem was disabled by changing connection to be open. Because broadcasting also means target-unspecified, modules can listen unexisting message groups, or unexisting “topics” in the ROS terminology. (It is also possible for modules to send messages to a topic with no listeners, of course.) This can be done by just modifying the system launching script using an ROS function named “remap”. Note that only the output to the serial port was disabled, the whole part of the original acoustic modem was not disabled and input was still alive. Therefore, the AUV could receive acoustic messages via the original acoustic modem, in case the newly developed one (or software modules for it) did not work well. (Fortunately, the newly developed acoustic modem worked well, and this feature was not used.)

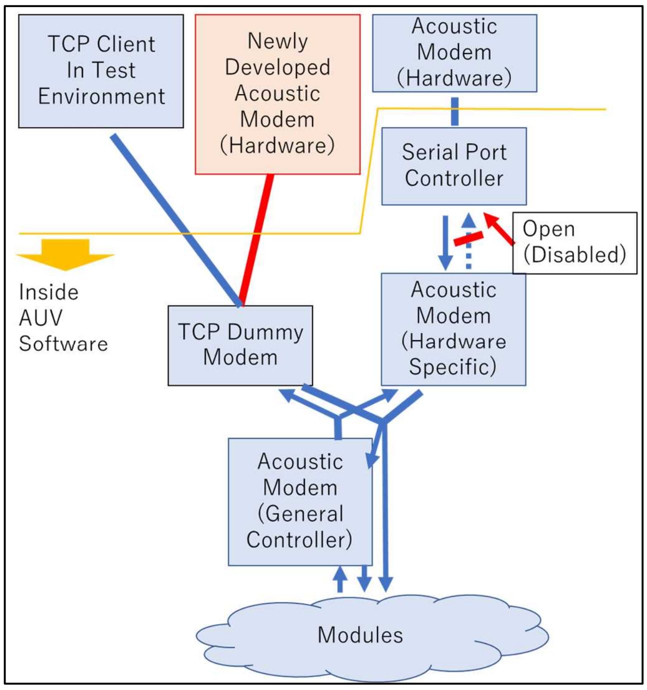

TCP Client In Test Environment
Newly Developed Acoustic Modem (Hardware)
Acoustic Modem (Hardware)
Serial Port Controller
Open (Disabled)
Inside AUV Software
TCP Dummy Modem
Acoustic Modem (Hardware Specific)
Acoustic Modem (General Controller)
Modules

Fig. 4 Data flow in the new software and changes at the survey cruise.

C. *Position Control & Related Futures*

The most important theme on AUV control is probably position (and attitude, velocity) control. In the new software, this part is also redesigned.

The position control algorithm was completely replaced from the older software. (The base technique is just simple PID control or PI, and it is not different.) In addition, as preparations for the future, the mission definition language is completely changed and thruster configuration and load allocation tuning became possible with configuration file (as reported in [7]).

These are not strongly related with the ROS structure, however, this type of try is easily done with the ROS structure because modules are easily replaced without affecting other modules. Modules are in separated executable and interactions between modules are limited in messages, the format of which is clearly defined.

Fig. 5 shows a characteristic part of the position control algorithm used in the last dives, Fig. 6 shows the trajectory of the dive at real sea experiment on December 2017, Fig. 7 shows the trajectory of a dive on at the survey cruise on February 2018 and Fig. 8 shows the altitude result on the same cruise. The distance between neighboring survey lines was 4 m, the cruising speed was 0.2 m/s and the cruising altitude was 3.0 m in both cases. The lengths of each survey line were 60 m for Fig. 6 and 100 m for Fig. 7. Green line is for the trajectory when the AUV were in initial waiting state with horizontal thrusters disabled, orange lines are for the trajectories when the AUV were moving to the start points and blue lines are the trajectories on the very survey lines. (The green line and most part of the orange line are omitted in Fig. 6 because they do not seem to include interesting problem to be considered.) It looks like the survey lines end suddenly because missions were stopped by the acoustic command manually. The programs used in these two dives were almost same but a little different, because it has been in the development and debugging stage, the dive on December 2017 is conducted to test the whole system and to find bugs before the real survey mission on February 2018 and some of important bugs and bad designs found were fixed indeed.

Seeing Fig. 6, we may find: (a) survey lines were almost followed, (b) overshoots occurred on corners, and (c) right turns and left turns were different. The problems are (b) and (c). First, for (c), it was found that there was a human error on defining mission. Because the dive was decided to be done a port area by weather problems at the morning of the day, the authors rewrite the mission definition file by hand, then by failing copy and paste, “wait” keywords were dropped on corners in the right. This explains the differences. A little shortcuts were done on rounding corners on the right side. The curves were what the authors intended as shown in Fig. 5. Second, for (b), the authors think this was caused by the PI control and some other complex reasons on the ordered mission, which are the intended waiting time in the corners, which was set to be 10 sec and the distance to be recognized as way point reached, which was set to be 3 m. Because the intended cruising speed was set to be 0.2 m/s, it was impossible to reach the corner precisely in the designated time. This cause wrong learning on the I parameter of the PI control. In addition, the authors think it might also effected that the PI control is directly applied from the error distance to the output power without using the speed information on some part. This problem was left unsolved at the survey cruise of Fig. 7 because of the time shortage of testing new methods.

Seeing Fig. 7, we may find: (a) water flow was a little strong as the AUV moves west to east with horizontal thrusters stopped (green line), (b) the AUV did zig-zag moves. For (a), it is found that the speed of the water current was 0.3 m/s from the log data and the green line shows the direction, which is crossing the survey lines. Because the AUV is not so strong with water current from the side direction (see the shape in Fig. 1 again), and the water current speed was not negligible because it was larger than the intended cruising speed, we may say that it was not best condition for the AUV. For (b), the authors think, this was a result of the bad condition described above in one aspect, and this was also a result of a gap of control in the position control algorithm shown in Fig. 5, which must be the more important reason. Because the authors made heading-keep range around the survey line and set heading target on the survey line there happened a gap on the target heading direction when the AUV passed the edge of the heading-keep range. The heading target should have been located on the edge of the heading-keep range. In addition, the authors are also considering about the possibility that the problem on PI control described above might have relation to this problem, and about the necessity of feed-forward type control as a future study.

As results of these dives in the real environment, it is appeared that the much more improvement in the stability is necessary, which is a big problem left, indeed, however, it can be also appeared that the AUV did understand the missions and did follow the ordered survey lines and the software almost worked well.

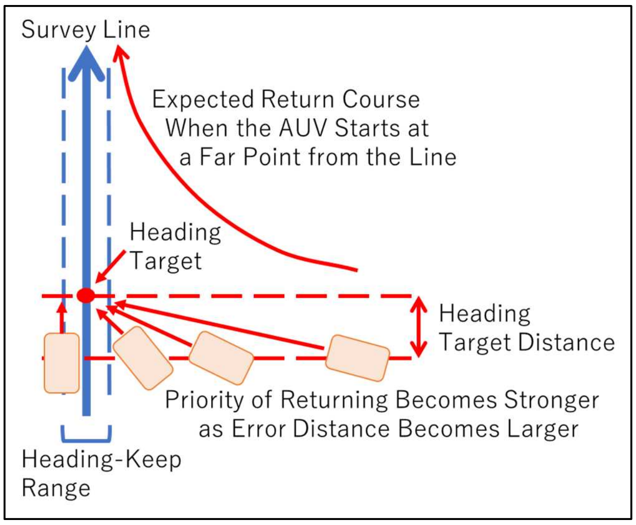

Survey Line
Expected Return Course When the AUV Starts at a Far Point from the Line
Heading Target
Heading Target Distance
Priority of Returning Becomes Stronger as Error Distance Becomes Larger
Heading-Keep Range

Fig. 5 The position control algorithm used at the survey cruise.

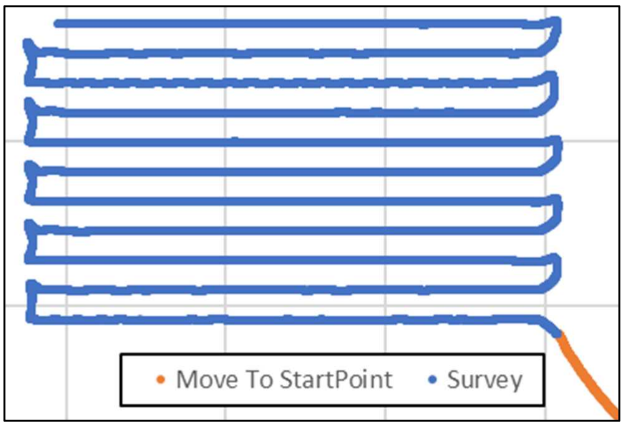


Fig. 6 Trajectory at the sea experiment on December 2017.

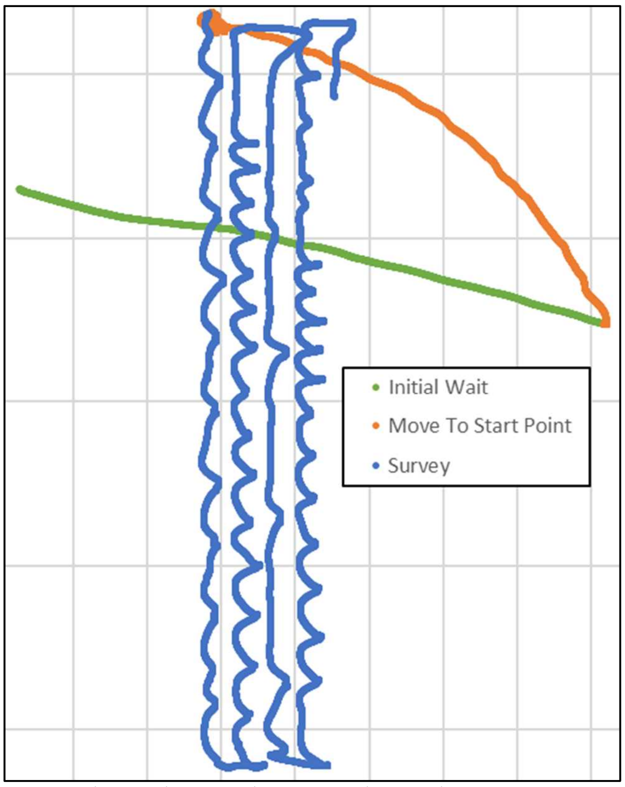


Fig. 7 Trajectory at the survey cruise on February 2018.

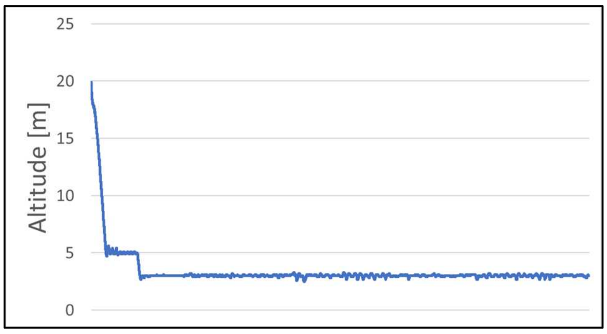


Fig. 8 Altitudes at the survey cruise on February 2018.

D. Purgeable Backward Compatibility

There are not only technical problems. Generally, software and other things with creativeness cannot be free with license problems. Older HOBALIN had such a license problem, too. The authors (and the institute they belong to) have been authorized to use the software with the AUV, however, it is in gray area to use it with another AUV in future. It is also difficult to lend HOBALIN to outside of the authors' institute without stuffs from the authors' institute. Therefore, the license problem was an important part of motivation of software replacement.

The license problem itself was cleared by just redeveloping the software based on ROS, which is an open-source framework. However, there was another requirement of backward compatibility that the AUV needs to communicate with existing software with old protocols, too.

This was also solved with the ROS structure. Because each module constitutes single executable, proprietary source codes are separated from new codes. The source code management repositories can be also separated. Because messages are broadcasted, proprietary modules can easily participate in the communication without affecting new codes. For example, all acoustic messages received by the hardware are broadcasted, if it is in the new protocol, it is processed by the new module, and if it is in the old protocol, it is just ignored by the new module and it is processed by the compatibility modules. Commands via TCP in the old protocol are received by a TCP server in the compatibility modules, and command messages are translated to the new protocol and broadcasted into the ROS communication structure. These compatibility modules are just purgeable option modules, the AUV works only with newly developed modules.

The ROS style launching scripts are also separated. This is done with the include mechanism of the launching script as shown in Fig. 9. An empty file is included without compatibility modules, and it is overwritten if the compatibility modules are used. It makes it easy to enable / disable the compatibility modules. (This mechanism was also used for the module created for the newly developed acoustic modem which is described in the section C, so that the main launching script are not affected by a temporarily added module.)

```
…
<!-- Compatibility -->
<include
file="$(find auv_hobalin)/launch/compatibility.launch" />
```

Fig. 9 Launching script separation (end of the main launching script).

V. Conclusion

In this paper, the replacement of the operating software of an hovering-type AUV named HOBALIN to ROS based one is reported with showing some examples of detailed design. The new software has some problems left yet, most important one of which is probably position control stability, and is on a development stage yet, however we can also say it already has ability to be used in the real environment. The first real survey mission has done as reported in this paper.

The software replacement is not the goal, but what is expected is it boosts science on the ocean and development of the ocean. Therefore, it is just the start point. Some effects are already appearing as also reported in this paper. The authors will continue improvement of the AUV using the new software, including both hardware improvements and software improvements.

Acknowledgment

The authors thank members of JAMSTEC (Japan Agency for Marine-Earth Science and Technology) and the crews of the research vessel Kaimei for giving a chance to dive the AUV HOBALIN in the survey cruise on February 2018 (KM18-02, chief researcher: Dr. T. Kasaya). This was the first time for the new software to be used in a real survey mission.

References

[1] A. Okamoto, K.Tamura, M. Sasano, K. Sawada, T. Seta, S. Inaba, T. Ura, Y. Nishida, J. Kojima and Y. Itoh, "Development of hovering-type AUV 'HOBALIN' for exploring seafloor hydrothermal deposits," OCEANS 2016 MTS/IEEE Monterey, 2016.

[2] T. Urabe, T. Ura, T.Tsujimoto and H. Hotta, "Next-generation technology for ocean resources exploration (Zipangu-in-the-Ocean) project in Japan," OCEANS 2015 MTS/IEEE Genova, 2015.

[3] Cabinet Office of Japan,"Pioneering the future: Japanese science, technology and innovation", http://www8.cao.go.jp/cstp/panhu/sip_english/sip_en.html, 2015.

[4] T. Seta, A. Okamoto, S. Inaba, M. Sasano, S. Takashima, M. Shimazu and T. Matsuda, "An observation of sea floor by a hovering type AUV HOBALIN with offshore multi-purpose work vessel Kaiyu at middle Okinawa trough," OCEANS 2017 MTS/IEEE Anchorage, 2017.

[5] Open Source Robotics Foundation, "ROS.org", http://www.ros.org/ (accessed March 5, 2018).

[6] MOOS-IvP.org, "MOOS-IvP home page", http://oceanai.mit.edu/moos-ivp/pmwiki/pmwiki.php (accessed March 5, 2018).

[7] T. Seta, A. Okamoto, S. Inaba, M. Sasano, "Development of a new operating system software for a hovering-type autonomous underwater vehicle HOBALIN", 11th Asian Control Conference (ASCC), 2017, pp. 37–42.

[8] A. Okamoto, M. Sasano, T. Seta, S. Inaba, K. Sato, K. Tamura, Y. Nishida and T. Ura, "Obstacle avoidance method appropriate for the steep terrain of the deep seafloor," Tecno-Ocean 2016, 1D.4, 2016, pp.195-198.

[9] LSTS (Underwater Systems and Technology Laboratory of the University of Porto), "Toolchain," http://lsts.fe.up.pt/toolchain (accessed March 5, 2018).

[10] BOSCH, "CAN specification version 2.0", 1991.

[11] National Marine Electronics Association, "Standards (NMEA 2000, 0183)", http://www.nmea.org/ (accessed March 5, 2018).