CONTINUOUS INTEGRATION AND DEPLOYMENT SOFTWARE FOR MOBILE SERVICE ROBOTS

KOLESOV, Y., PRYANICHNIKOV, V. & TARASOV, R.

Abstract: *Development of the intelligent control software for mobile robots usually requires a lot of changes; the problem is to find evolutionary way of the process. This article describes the study of the problems of testing, integration of parts of hardware and software for mobile service robots (Amur-307 type). The complex of software tools for simplification, integration, testing and speed up debugging of compilation and reliability errors are described. The approach is using emulation of hardware, sensors and electric drivers, additional software modules, which allows creating a complete virtual analogue of robotic complex and speeding up software development, using unit tests.*

Key words: *Integration software Service mobile robots, Emulation and testing of hardware and software, based on microprocessors and on-board computers.*

Authors' data: PhD students: Kolesov, Y[uri], Tarasov, R[adomir] & Leading researcher, Dr.techn.sc. Pryanitchnikov V[alentin]*,**, *International Institute for New Educational Technologies RSUH, **KIAM Russian Academy of Sciences, Miusskaya sq. 6 & 8 Moscow 125047 Russia, v.e.pr@yandex.ru, radtarasov@gmail.com

This Publication has to be referred as: Kolesov, Y[uri]; Tarasov, R[adomir] & Pryanichnikov V[alentin] (2021). Continuous Integration and Deployment Software for Mobile Service Robots, Chapter 20 in DAAAM International Scientific Book 2021, pp.241-250, B. Katalinic (Ed.), Published by DAAAM International, ISBN 978-3-902734-31-0, ISSN 1726-9687, Vienna, Austria
DOI: 10.2507/daaam.scibook.2021.20

1. Introduction

In KIAM Russian Academy of Sciences, together with universities and organizations, is carrying out a complex of works on the creation of intelligent mobile robots for various purposes within the framework of the Intelligent Robotics project. One of the key problems of such developments is finding a reliable mechanism for upgrading software and hardware in the conditions of changing versions and meeting the conflicting requirements of various customers. Therefore, when developing such complex software, it is often necessary to use continuous integration and deployment systems to automate the development process. This approach to software development is the most flexible and convenient at the moment. The continuous integration system automatically prepares the software for launch and execution. The main task of this system is to ensure that execution errors do not occur at any arbitrary time, due to the preliminary passage of all testing stages (Davydov et al., 2017). This approach also justifies itself when developing client software for the Amur-307 service robot (Aryskin et al., 2020). The developed software can be used for the entire spectrum of robotic systems, including underwater ones (Aryskin et al., 2020) - (Ksenzenko et al., 2017), providing better results compared, for example, with work (Chernyshev et al., 2019).

This system provides all the necessary and sufficient tools for software testing with coverage of all levels of abstraction, such as: unit testing, integration testing and user interface testing. When testing the final program code, it is also necessary to test on the target hardware to exclude hardware-dependent errors (Pryanichnikov, 2018). For such testing, it is advisable to use virtualization of the entire hardware environment (Pryanichnikov et al., 2018). As a result, a system based on this method was developed.

2. The software used

The proposed architecture for testing and debugging client software allows, after making edits to the source codes, to automatically assemble and test software for the Raspberry Pi platform without resorting to manual testing on a real microcomputer.

Fig. 1. Technologies of the client software development workflow on the part of the programmer.

Due to the need to speed up work with the program code, from the sent commit to the launch of the finished application, a system of continuous integration and continuous testing of Github Actions working inside the repository with the code was used. The developed system is conditionally divided into 3 parts: part on the programmer's computer, part on Github servers and part on the target platform (Fig. 1.).

Technologies of the client software development workflow from the software side, various specialized libraries were used as components of the software being developed. These libraries allow you to speed up development, automate testing of system modules, as well as reduce the load on the processor and network (Tarasov et al., 2020).

After the research, the following libraries of the software codamist were selected:

- gRPC + Protobuf. This framework provides mechanisms for convenient and fast serialization and deserialization of data into binary format, as well as for reducing the transmitted data and increasing the exchange rate in comparison with the REST API. Also, this framework allows you to maintain backward compatibility with microservices of previous iterations of the developed software, which makes it possible to flexibly change the data structures of working with the network..
- piGPIO. This library provides mechanisms for interacting with peripheral equipment of the Raspberry Pi microcomputer, such as GPIO I/O ports. For this microcomputer, there are several programs for working with GPIO ports, the most popular are: WiringPi and piGPIO. Currently, the WiringPi library is not supported, and it also does not have the necessary functionality to work with a DMA controller, which would increase the number of PWM signals. pi GPIO is often used, due to its functionality, multilingual support and speed of operation.
- libconfig. This library is used to work with software configuration data. Its main advantage is cross-platform compatibility, as well as compatibility with the POSIX standard. The configuration file format has a simple syntax, and the markup language itself has flexible functionality.

3. Continuous integration and project deployment tools

For representativeness, versioning and unification of commits, a separate commitizen program is used. This program allows you to significantly save time for accepting a request to add code to the repository when viewing the commit history, and also allows you to do some work with the project in semi-automatic mode, such as assigning the current software version and generating a change file.

Fig. 2. The structure of the client software development workflow on the Github server.

The clock software package is used to estimate the number of lines of code, comments and obtain statistics on the volume of program code. The Git system has a multifunctional built-in git hooks mechanism that allows you to perform certain actions on a selected event. This mechanism is used to check the commit before it is accepted. Continuous Integration and Continuous Deployment technologies based on the Github Actions service are used to accelerate development and testing. On the Github server side, all the basic operations on comments are performed, such as: checking the code with a static analyzer, generating documentation, compiling and testing (Fig. 2.).

These technologies allow you to find out the test result, generate documentation, and assemble an application package on a third-party server, which in turn greatly reduced the time spent to obtain results. When a new commit is received, two scripts describing development pipelines are run on the server, one of which describes the assembly of documentation, and the second describes the assembly and testing of the project itself. The documentation is assembled by Doxygen (***, 2011). Documentation and UML diagrams of the client software structure are generated based on the program code. After the documentation is assembled, a status notification is sent to the Telegram work chat and if the assembly was successful, the documentation is updated on the project website.

To generate binary data for a large project, a configuration of a multifunctional assembly system is required. The build system allows you to assemble all the files in the correct order, and also allows you to run additional precompilation steps. The following build systems are used for the C++ language: qmake, make, cmake, autotools. The Raspberry Pi distribution does not have a pre-installed Qt library, so the qmake build system is not suitable due to the large size of the library, and the make system, in turn, is not a cross-platform system as well as autotools. As a result, for the above reasons, the CMake build system was chosen to build the program, which provides a flexible and user-friendly interface, and is also cross-platform.

Due to the fact that the final application must be run on a mobile platform with an ARM processor, there is a need to use cross-compilation, which allows you to compile the project on the server and not load the platform processor. Next, it is necessary to conduct unit testing to assess the performance of individual modules using the Gtest library, this testing is performed on the computing power of Github.

After successfully passing the unit tests, hardware interfaces are tested to confirm the operability of the system as a whole, a QEMU virtual machine is used for this task, which allows emulating Raspberry Pi hardware, including I/O ports. With the help of this set of tools, an image is created with compiled and tested AMUR-307 software (Aryskin et al., 2020).

4. Creation of a virtual prototype of software testing equipment

The prototype of the equipment uses a number of auxiliary programs that interact with virtual equipment, namely: tun/tap, iproute2, iptables, qemu, socat. A similar implementation is discussed in the article (Simple, 2011). In the virtual prototype of the hardware, the TCP/IP stack is completely emulated, since this is necessary for testing software for working with the server, which is beyond the scope of this article.

The network topology is described in the configuration of the virtual machine, which completely sets the context of its operation. This is necessary because there is no automatic access to the external network by default. Therefore, the network stack is supported by automatically creating the assignment of network interfaces to various virtual network clients. The described functionality is implemented using the TAP/TAP mechanism (QEMU, 2011), which is present in all modern versions of the Linux OS kernel.

The task of managing network interfaces, routes, addresses and virtual switches (bridges) is solved using the iproute2 package (Shahin et al., 2017). Bridges are also available in all modern versions of the Linux kernel. The iproute2 package was chosen to achieve code portability between different kernel versions since similar tools (ifconfig and route) are no longer supported. The connection of a virtual network to an external network interface can be carried out again either using a virtual switch (bridge) or using the virtual address translator (NAT) mechanism, which is activated using the iptables utility. This choice is also due to compatibility and support criteria, since the analogues (ipfwadm and ipchains) are outdated.

Inside the virtual machine, using the qemu translator, the target software is launched using the serial port provided by the socat utility (QEMU, 2011). As a result, the final result of integration testing is obtained, which shows the absence or presence of execution errors when running the AMUR-307 software on a virtual machine (Aryskin et al., 2019). The result of this launch gets into the pipeline of the sequential integration environment, in this case it is Github Actions, and after the execution status is sent to the Telegram channel.

5. Source code correctness analysis and executable code generation

On the git server, execution errors and new functionality are debugged using the integration software described above, if any. Also, a qemu virtual machine is used on the connected external server, which allows the use of gpio virtual ports, which allows debugging the functionality of the components of the AMUR-307 client software on a virtual machine, such as basic tests of the functionality of various sensors and other hardware modules. It is also possible to emulate the operation of a robotic complex in a virtual environment as a whole, namely, movement along the route, interaction with manipulators and much more, which significantly reduces the debugging time directly on the robot's equipment and allows you to identify machine-independent errors.

After successful testing, the installation apk package for Raspbian OS is automatically assembled, which is published in the repository releases and also published on the project website.

6. The mechanism of sequential software deployment on the operated work

The other-the-air programming (OTA) update mechanism is known. For sequential deployment of the AMUR-307 software, a specialized linux distribution based on buildroot, busybox implementation of the boot and diagnostic management interface (based on kexecboot) and a miniature linux kernel, which is placed in a 128 Mbit SPI flash on the robot's cross-board, are used.

This specialized distribution solves several problems:

- Implementation of remote debugging by supporting a full-fledged network stack by the operating system kernel and ssh service. This mode provides a remote serial port interface through which the boot process of the target OS can be tested..
- The task of automatic updating when a new version of the AMUR-307 software is released. In the case of software debugging, the update is launched after the successful completion of the gitlab pipeline. If the system is updated, the latest release from the repository is installed on the operated robot. Also, before performing the update, the correctness of the already installed software and the version of the new release is checked.
- Storing parameters specific to this instance of the AMUR-307 controller. The SPI flash memory section stores identifying and configuration data for a specific instance of the robot, such as: serial number of the board, configuration of peripheral equipment, MAC address and repository addresses.
- Implementation of debugging mode (with network support) with full access to the administration of the target operating system, in case of an emergency completion of automatic updates or downloads.
- Transfer of control to the target AMUR-307 operating system using a kexec system call.

Fig. 3. The structure of the workflow for deploying client software on a target job.

When the OS is launched on the AMUR-307 robot (Fig. 4.), the cron service is launched, which receives information about the latest github release at a predetermined interval. If the current version of the software has been updated on the server, cron plans to restart the Amur client board at a certain time, having previously recorded information about the need for an update in the board parameters.

Fig. 4. Example of the electronic box (6 channels PWM 30A, 6 – 10A, 12 sensor input) for mobile robot control, operating with the developed software.

In turn, the microkernel reads information at startup that an update is required and connects via Wi-Fi channel via http to the repository with the compiled binary image of the target OS and performs the update.

7. Conclusion

As a result, the conducted research made it possible to identify, create and configure a tool environment for continuous integration and deployment with the possibility of testing software on virtual hardware.

The created system allows you to save the time of software developers for the AMUR-307 robot by refusing manual testing on real robot hardware, as well as counting the automatic launch of code analysis and the necessary tests with each new commit.

As a result of the application of this software package, the programmer does not carry out the stages of assembly and publication of the release, and the installation of the package, or its update from the project repository, on the robot occurs automatically. In this way, the process of continuous development, continuous integration and continuous deployment is carried out.

8. Acknowledgments

The authors express their gratitude to the members of the Keldysh Institute of Applied Mathematics RAS, MINOT RSUH and other organizations involved in the development of mobile robots, including the Amur-307 series within the framework of the joint project "Intelligent Robotics", especially O.I.Davydov, A.V.Plotnikov, A.Ya.Ksenzenko, S.R.Eprikov and others.

9. References

***Simple, Painless Application Testing on Virtualized Hardware – Star Lab Software. Available at: <https://www.starlab.io/blog/simple-painless-application-testing-on-virtualized-hardware> (Accessed 20 October 2011).

A.Y. Ksenzenko, Yu.S. Marzanov, E.A. Prysev, V.E. Pryanichnikov, V.V. Chernyshev, Prototyping of contactless data exchange and power supply of underwater satellite robots with a base station walking on the bottom, Extreme robotics. Collection of abstracts of the international scientific and technical conference, 2017, St. Petersburg, p. 268.

Aryskin, A. Grigorev, A., Khelemendik, R., Petrakov, M., Pryanichnikov, V. & Tarasov, R. (2020). System for Monitoring and Control in Industrial Technological Processes, Proceedings of the 31st DAAAM International Symposium, pp.0644-0649, B. Katalinic (Ed.), Published by DAAAM International, ISBN 978-3-902734-29-7, ISSN 1726-9679, Vienna, Austria, DOI: 10.2507/31st.daaam.proceedings.089

Aryskin, A., Bogdanovich, A., Davydov, O., Grigoriev, A., Khelemendik, R., Kharin, K., Kuvshinov, S., Petrakov, M., Plotnikov A., Pryanichnikov, V. & Tarasov R. (2019). Control algorithms for service and industrial transport robots, Proceedings of the 30th

DAAAM International Symposium, pp.1166-1173, B. Katalinic (Ed.), Published by DAAAM International, ISBN 978-3-902734-22-8, ISSN 1726-9679, Vienna, Austria. DOI: 10.2507/30th.daaam.proceedings.164

Aryskin, A., Bogdanovich, A., Davydov, O., Khelemendik, R., Petrakov, M. & Pryanichnikov, V. (2019). Control and Interaction Algorithms for Industrial and Service Robots, Chapter 33 in DAAAM International Scientific Book 2019, pp.377-384, B. Katalinic (Ed.), Published by DAAAM International, ISBN 978-3-902734-24-2, ISSN 1726-9687, Vienna, Austria DOI: 10.2507/daaam.scibook.2019.33

Aryskin, A., Davydov, O., Eprikov S., Ksenzenko A., Pryanichnikov, V., Punenkov O., Shipovalov E., Solovyova M., Stepanova D., Tarasov R. & Tikhomirov A. (2020). Robotariums implementations with the Elements of Artificial Intelligence Control, Proceedings of the 31st DAAAM International Symposium, pp.0973-0985, B. Katalinic (Ed.), Published by DAAAM International, ISBN 978-3-902734-29-7, ISSN 1726-9679, Vienna, Austria DOI: 10.2507/31st.daaam.proceedings.1351 (P. 3)

Bielich T., Ksenzenko A. Ya., Kirsanov K. B., Pryanichnikov V. E. Increasing the time of continuous operation of the underwater inspection robotic complex. Information-measuring and control systems. 2015, No. 7, pp. 51-56, ISSN 2070-0814.

Davydov D.V., Eprikov S.R., Kirsanov K.B., Pryanichnikov V.E. (2017). Service Robots Integrating Software and Remote Reprogramming, Proceedings of the 28th DAAAM International Symposium, pp.1234-1240, B. Katalinic (Ed.), Published by DAAAM International, ISBN 978-3-902734-11-2, ISSN 1726-9679, Vienna, Austria, DOI: 10.2507/28th.daaam.proceedings.172.

Ksenzenko A.Y, Prysev E.A., Pryanichnikov V.E., Chernyshev V.V., (2017). Intelligent Robotronics: Design the Contactless Charger and Contactless Data transfer between underwater Robot-Satellites and Underwater 6-legged Vehicle. Proceedings of the 28 th DAAAM International Symposium on Intelligent Manufacturing and Automation, pp. 1197-1201, B. Katalinic (Ed), Published by DAAAD International, ISBN 978-3-902734-08-2, ISSN 1726-9679, Vienna, Austria, DOI: 10.2507/28th.daaam.proceedings.166.

M. Shahin, M. Ali Babar, and L. Zhu, —Continuous Integration, Delivery and Deployment: A Systematic Review on Approaches, Tools, Challenges and Practices”, IEEE Access, 2017. <https://arxiv.org/pdf/1703.07019.pdf>

Pryanichnikov V. E. Artificial intelligence and software and hardware robotic systems. Information-measuring and control systems. - 2018. — Vol. 16, No. 12. - Pp. 3-11, ISSN 2070-0814.

R.B. Tarasov, O.I. Davydov, V.E. Pryanichnikov, M.D. Solovyova, A.N. Tikhomirov, E.A. Shapovalov, S.R. Epikov // Software-Control Shell for Service Robots With Parallel Computing. Extreme robotics. // Collection of abstracts of the 31st International Scientific and Technical Conference "Extreme Robotics". - St. Petersburg: CPI LLC "Polytechnic-print", 2020. - 86 p., pp.41-42. ISBN 978-5-85875-602-6.

S.Y. Yoo, B.H. Jun, H. Shim, "Design of Static Gait Algorithm for Hexapod Subsea Walking Robot: Crabster", Transactions of the Korean Society of Mechanical Engineers A. Vol.38, 2014, pp. 989-997, 10.3795/KSME-A.2014.38.9.989.

Ushakov Yu.A., Polezhaev P.N., Shukhman A.E. Creation of a multiservice multipoint VPN network with dynamic auto-tuning. // Bulletin of Orenburg State University. - 2015. - No. 9. - pp. 170-178.

V. Arykantsev, A. A. Ariskin, O.O. Belyaev, A.Ya. Ksenzenko, E.A.Prysev, V.E. Pryanichnikov, V.V. Chernyshev, S.R.Eprikov. Supervisory control of underwater walking apparatus, Extreme robotics and conversion trends. Collection of abstracts of the international scientific and technical conference, 2018, St.Petersburg, p. 203.

V.E. Pryanichnikov, E.A. Shapovalov, R.B. Tarasov // Intellectual Support of Logistics Missions in Medical Institutions with Mobile Service Robots AMUR-307. Extreme robotics. // Collection of abstracts of the 31st International Scientific and Technical Conference "Extreme Robotics". - Saint Petersburg: CPI LLC "Polytechnic-print", 2020. - 86 p., p.73. ISBN 978-5-85875-602-6.

V.V. Chernyshev, V.E. Pryanichnikov, V.V. Arykantsev, I.P. Vershinina, Ya.V. Kalinin, "Research of the walking type of movement in underwater conditions", OCEANS 2019 – Marseille, IEEE, Marseille, France, 2019, pp. 1–6, <https://doi.org/10.1109/OCEANSE.2019.8867233>.