Jurnal Integrasi
Vol. 13 No. 2, October 2021, 113-121
e-ISSN: 2548-9828

Article History
Received September 2021
Accepted October 2021

Establishing ROS on Humanoid Soccer Robot-BarelangFC Software System

Susanto[1*], Eko Priono[1], Riska Analia[1]

* Batam Polytechnics
Electrical Engineering Study Program
Parkway Street, Batam Centre, Batam 29461, Indonesia
E-mail: susanto@polibatam.ac.id

Abstrak

Robot humanoid dikembangkan dari beberapa sub program atau sistem yang terintegrasi pada setiap program utama untuk memerintahkan robot bergerak selayaknya seperti pemain sepak bola. Masing-masing program utama terdiri atas sistem gerak, sistem indra visual (*vision system*), sistem sub-kontroler, dan strategi permainan. Saat ini, masing-masing sistem utama didesain menggunakan bahasa pemrograman yang berbeda, misalnya: sistem visi menggunakan python sedangkan yang lain menggunakan C dan LUA untuk kinematika gerakan. Penggunaan bahasa pemrograman yang berbeda akan mempengaruhi respon sistem karena masing-masing sistem utama perlu diintegrasikan menggunakan socket pada proses awal. Respon robot akan lambat dan menghabiskan banyak penggunaan memori. Oleh karena itu, dalam makalah ini akan disajikan proses migrasi ke dalam sistem operasi robot (ROS) dan mengalihkan semua sistem utama robot ke dalam bahasa python. Program terintegrasi akan diperiksa secara real-time aplikasi saat robot bergerak di lapangan. Serta menggunakan python ROS untuk membuat robot dapat bermain secara mandiri di lapangan.

Kata kunci: Integrasi, marjin

Abstract

Humanoid robot is built on several sub-programs or systems which is integrated to each main programs in order to command the robot to move as a soccer player. Each main programs namely as a movement system, a visual sense system (vision), a sub-controller system, and a game strategy. Currently, each of main system constructed using different programming language, for instance: the vision system used python while the others used C and LUA for the movement kinematics. Employing different programming language will affect to response system because each of main system need to be integrated using socket in the beginning process. Robot response will be slow and cost a lot of memory usage. Therefore, in this paper will present a migrating process into robot operating system (ROS) and switch all the robot main system into python language. The integrated program will be examined in real-time application while the robot moved on the field. We used a python ROS in order to make the robot play autonomously on the field.

Keywords: *humanoid robot soccer, system migration, ROS, real-time application*

1. Introduction

Developing and implementing the humanoid robot soccer need a lot of effort and really challenging work. It starts from design the mechanical, electrical, programming, and also the strategy of playing football. Moreover, in programming side mostly it consists of several framework path which is need to be connected one another so that it can fully automatically moved according to the strategy that has been choosing. As

introduced in decade, the robot operating system (ROS) was one of the powerful operating system for the robot which consists a lot of software part that needed to be integrated. As presented by Kim, et. al [1], they utilized the manipulation robot which is communicated to the Gazebo-ROS in order to examine the automation program like PLC controller through TCP/IP protocol. And also, Megalingam, et.al [2], used

ROS navigation stack to analyze the differences of path planning and path travel drive robot on Gazebo-ROS simulator.

Another work done by Rivera, et.al [3], aimed to

analyse the ROS monitoring tool effectiveness by investigated the effect between extended Berkely Packet Filters (eBPF) and eXpress Data Path (XDP) in developing a high-performance inline network monitoring for ROS framework. Meanwhile, Ma, et.al [4] They developed the ROS and Qt for the Graphical User Interface (GUI) of the multi-robot system (MRS) simulator in order to understand their coordination behavior and control the real robot with minor correction. Amontamavut, et.al [5] on the other hand, they reconstructed the Blue-Sky web-based environment for ROS publish and subscribe messaging distribution also monitored and traced all the data access on the ROS environment. And Wei, et.al [6] Presented a hybrid real-time ROS architecture on multi-core processor which can be run in both real-time and non-real-time subsystem in order to control a 6-DOF modular manipulator. While, Gatesichapakorn, et.al[7] utilized a 2D LiDAR and RGB-D camera which is equipped with ROS 2D navigation stack to implement an autonomous mobile robot with low power consumption. And did the ROS-based mobile robot has been proposed in this work for achieving a remote 3D reconstruction for a mobile autonomous robot using an onboard RGB-D camera in order to sharpen robot pose planning view in their next research [8]. Again, Nitta, et.al [9], in this work they integrated the FPGA platform to the ROS system on the ZytleBot efficiency in order to detect the traffic signal for autonomous and Mishra, et.al [10], they utilized the ROS platform for developing a service robot which is able to do self-mapping, localization, and do navigation in indoor environment with static obstacle. The ROS itself has been used for robot ability to mapping in real-time application using a low cost RGB-D camera. Also, Chang, et.al [11], they involved middleware ROS to recognize object in a Raspberry Pi for mobile robot

The implementation of the ROS is done by Hasegawa, et.al [12], where they implemented a ROS-based autonomous vehicle in FPGA board in order to recognized a lane, traffic signal, and also obstacle detection. In this work, they concluded that deployed the essential components of the vehicle on an FGPA board with the ROS-based system can be successfully run together. While, Staschulat, et.al [13], they introduced an advanced Executor for the ROS 2 C API on a 32-bit microcontroller included deterministic scheduling and aid domain-specific requirements. And Rhoades, et.al [14], they connected a Controller Area Network (CAN), Radio Frequency (RF), All-Terrain Vehicle (ATV) to the ROS based platform system to extract all the sensor information easily. To verify the robustness of the ROS system, Mirkhanzadeh, et.al [15], in this work they demonstrated a robustness and trusthworthiness of an orchestrated two-laye network tes-bed (PROnet) on a ROS industrial for the end-to-end distributed flow services. And Cai, et.al [16], the ROS is used in this work to simulate a depth control of hybrid-driven underwater vehicle-manipulator system (HD-UVMS). Stojanović, et.al [17], utilized one of the ROS features to let the Adaptive AUTOSAR application is able to run on developing platform.

While developing ROS for the robot system, a mechanism protection of the ROS should be taken note in order to avoid the network hacker. Goerke, et.al [18], in this work they discussed about the ROS protection mechanism which is proposed in this paper. Even though the effectiveness and usable of it did not clearly state, this mechanism protection of ROS should be emphasized for the developer before build the ROS system for avoiding a network-based attacker. Garcia, et.al [19], in this work, they developed a software ecosystem by developing the model-driven engineering (MDE) based on ROS to identify a possible way to understand the accessibility of ROS merit towards MDE. Hong, et. al [20], they presented a stable and bidirectional connection of RoverOS to integrated ROS system and Web client via WebSocket layer which is allowed the users to control and navigate the Turtlebot robot.

Regarding to all the beneficial which is given by the ROS for developing robot software, therefore in this work we proposed the ROS to rejuvenate our software system and migrate all the source code in python code. The same work also has been done in our previous work [21], however in our previous work just moved a part of whole system while in this work we moved whole system in ROS.

2. Mechanical Design

The humanoid robot BarelangFC was designed as a human being which has two arms, two legs, body, and the vision which is constructed with several parts such as 20 servo motors for the robot movement, camera for vision system, sub-controller cm-730 as the servo motors controller and a NVIDIA Jetson TX2 as the central controller for controlling and connecting all devices which are mounted on the robot body. The robot mechanical design presented on **Figure. 1** which has the same construction with our previous work [21], on this figure each of servo has its own number related to the servo ID for utilizing the servo movement from the sub controller. On the other hand, the robot also supported by some electrical devices which integrated each other to help the robot moved according to the command. The electrical block diagram system can be seen on **Figure. 2**. On **Figure.2**, the robot equipped with

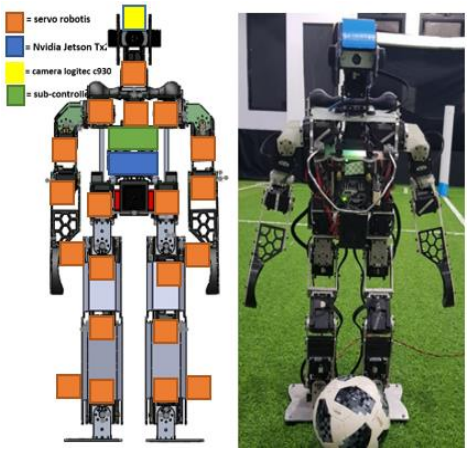


Figure. 1. The mechanical design of BarelangFc.

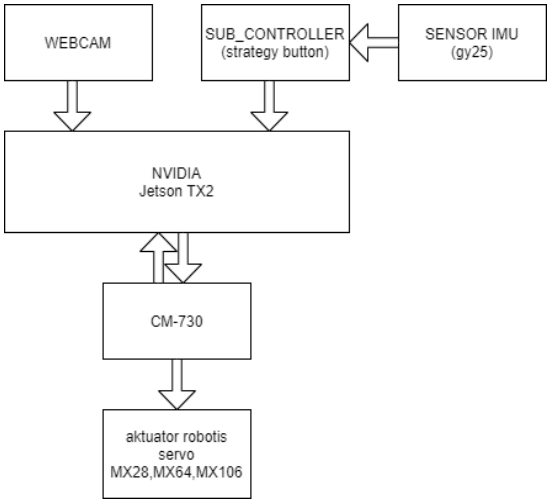


Figure. 2. The full electrical block diagram system.

webcam camera as the vision to detect ball, pole of goal, landmark, and also the robot rival. The vision detection will be proceeded on the NVIDIA Jetson TX2 therefore the robot will distinguish any object toward them. Moreover, robot also mounted the IMU sensor as a robot navigation so that the robot is able to decide which way should go on the field. The IMU sensor was selected as the input for the SUB_CONTROLLER of strategy button. This button is used to make the robot choose the strategi action whether moved to attack the opponents, remained still to protect the goal, or even to decide when it should dribble the ball or kick the ball towards opponent's goal. The decision of robot movement is depended on the vision system and strategy button which is computed on the NVIDIA Jetson TX2. Then the movement command will be transferred into CM-370 afterword to make sure each of servo motors moved according to the CM370 command.

3. The ROS Construction Plan

As seen on **Figure. 3**, the BarelangFC ROS structured by several nodes, subscribers, and publishers, the former node or package consisted of the Vision, the second one was the Sub_Controller, then the Main_Strategy, and the latter was the Bridge_Kinematic. While each of ROS publishers will transmit the data into the subscribers, for instance the Vision and Sub_Controller published their data to the Main_strategy node and then it will be also distributed the decision information to the Bridge_kinematic node through the ROS publisher. Another, the ROS subscriber was responsible on receiving the information from the publishers and delivering it to the package or node. At the last of process, the Bridge_kinematic node filled with the robot movement command and need to be sent to the kinematic procedures which was in this work used the LUA system via socket. The illustration of each package block diagram can be seen from **Figure. 4** to **Figure. 6** respectively. The vision package on Fig.4 attempted to process the vision data when the camera detected the goal and poles on the field. The output data from the vision was ball coordinate (x, y) also the poles as well. This coordinate will be published to the subscriber which is Main_Strategy node.

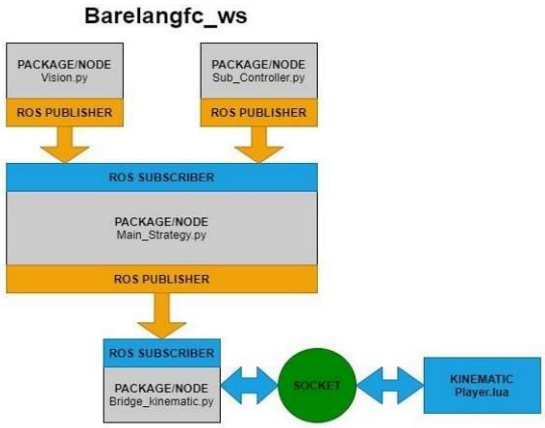


Figure. 3 The BarelangFC ROS architecture.

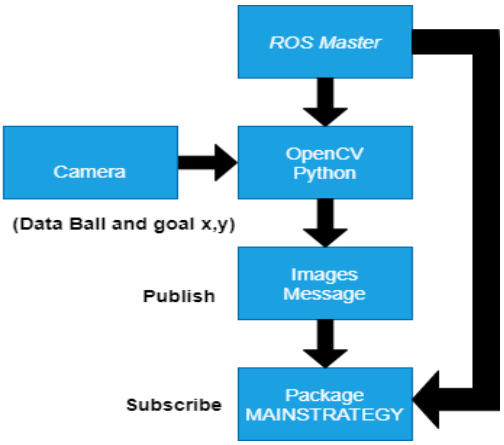


Figure. 4 The vision package block diagram.

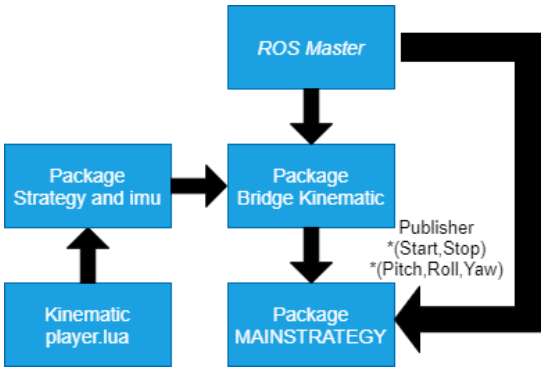


Figure. 5 The Sub_Controller package block diagram.

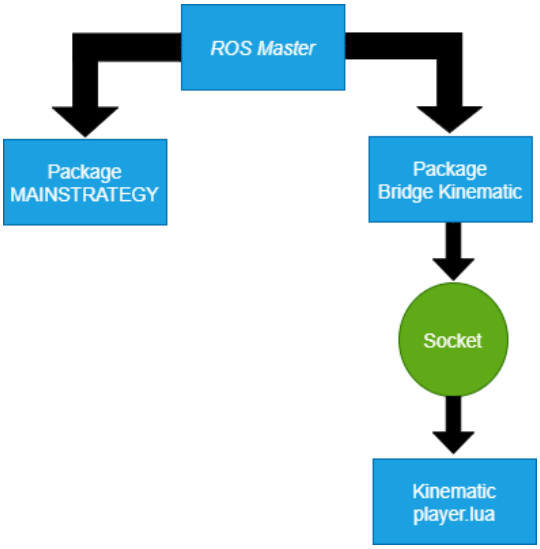


Figure. 6 The bridge kinematic package block diagram.

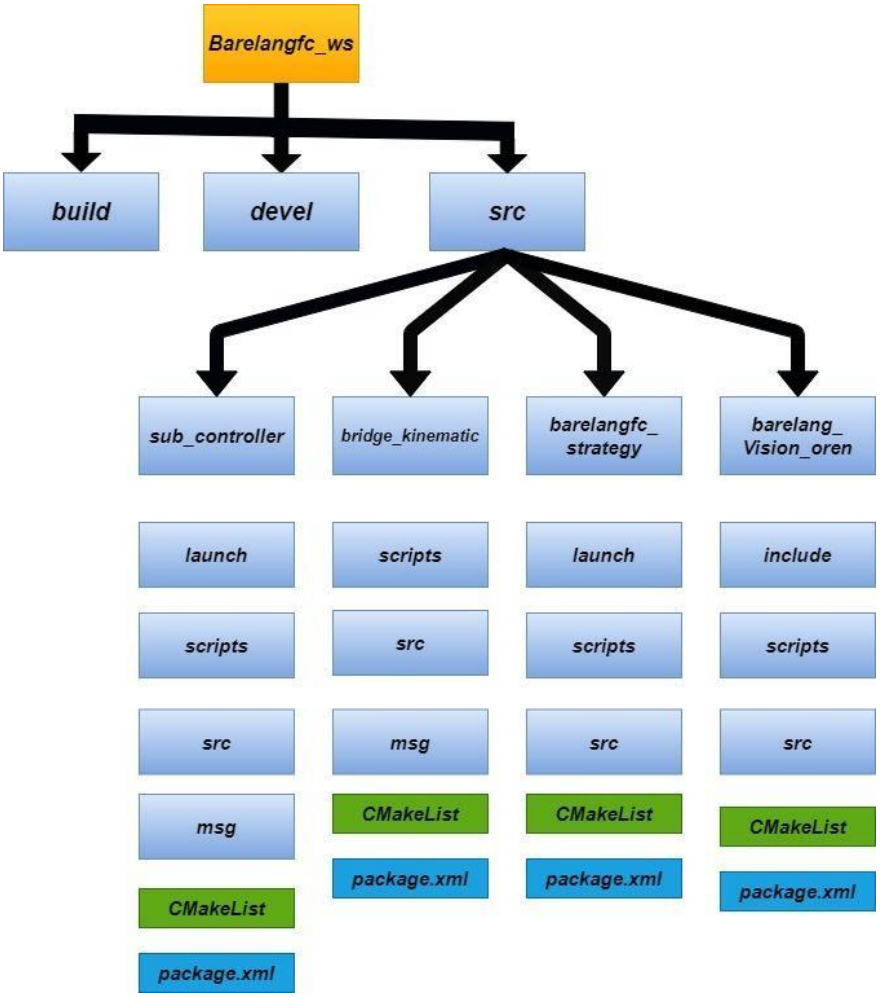


Figure. 7. The ROS workspace of BarelangFC.

Subsequently, the Sub_Controller depicted on **Figure. 5** including the IMU sensor for the robot heading and the start and stop strategy button. All of these data also transmitted to the subscriber Main_Strategy node. As for the Main_Strategy node, it became the server which control was controlled all of the function inside the robot in such visual system, localization, and strategy. This node has a special treatment due to it task to publish the robot movement data as well as subscribed data from the publisher. The latter package was Bridge_kinematik to transfer all the movement command from the Main_strategy node to the kinematic player. The block diagram system described on **Figure. 6** sketchily. Considering that the kinematic player has different platform, then all the command movement data from the package should be send to kinematic LUA through socket to translate the movement data to LUA. On the other hand, **Figure. 7** represented the completed BarelangFC ROS workspace which was consisted with several block that connected to each other. The BarelangFC workspace on **Figure. 7** was consisted the build and devel space where these two spaces donoted to the CMaked that requested to build the catkin packages in the source space and debug the targets that placed prior to being installed. Also, the scr or source space has varieties of packages or nodes for instance Sub_controller, Bridge_controller, Barelangfc_strategy, and Barelang_Vision_oren. All of these nodes will be generated the xml file before transmitting to the ROS subscribers.

4. Experimental Results

This section will discuss about all the experiments regarding the ROS transmitted and received data from the publisher to the subscriber. The first experiment was carried out to publish the data from publisher to the subscriber by utilizing the ROS topic. As seen on Figure. 8, the list of ROS topic on the workspace consists of kinematics or motion, walking, button, IMU for the robot heading, and also the vision indicated from "oren" which was denote to the orange color (ball) and "pos" represented the robot heading. For instance, as seen on **Figure. 9** (a), one of the ROS topics was running which is IMU topic, then the data not only will be shown up on the terminal screen but also transmitted to the subscriber. Meanwhile, **Figure. 9** (b) shown the kinematics ROS topic form the robot motion.

The first experiment was carried out in order to integrate the Sub_controller package to the ROS topic. The Sub_controller package consists of strategy button and the IMU sensor for the robot heading. This part was communicated to the core system through serial communication.

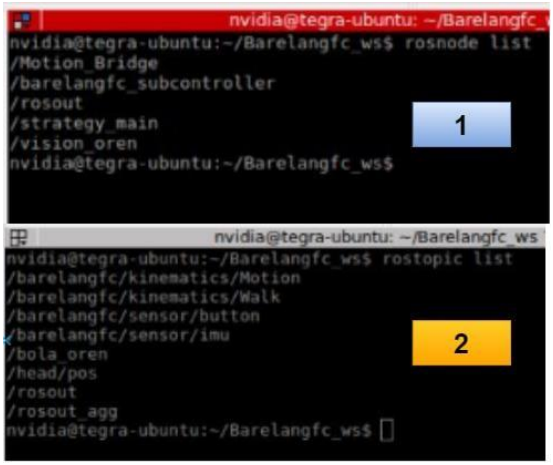
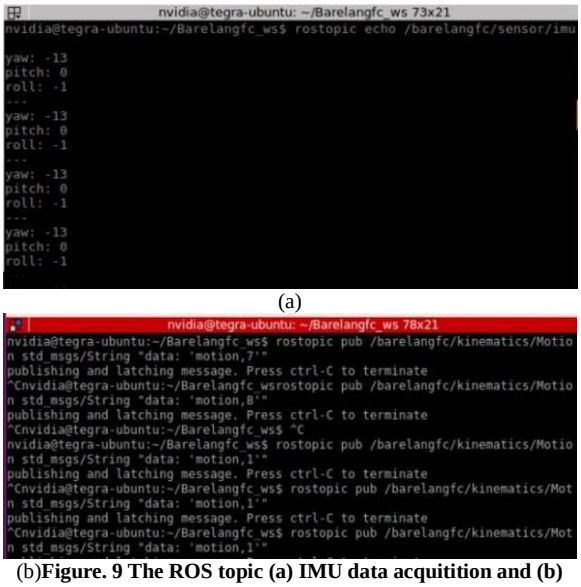


Figure. 8. The ROS topic list on the workspace.



(a)

(b)**Figure. 9 The ROS topic (a) IMU data acquitition and (b) kinematics motion data.**

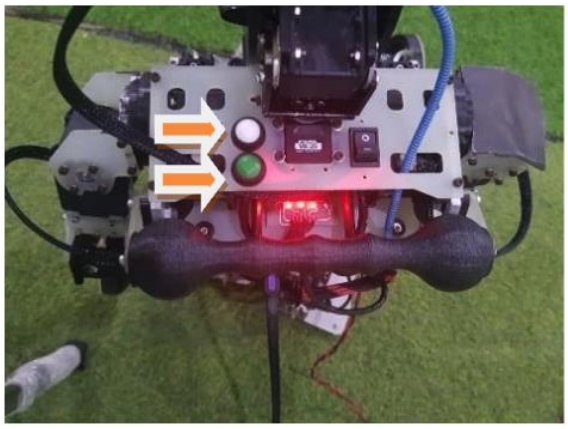


Figure. 10. The strategy button position of the robot.

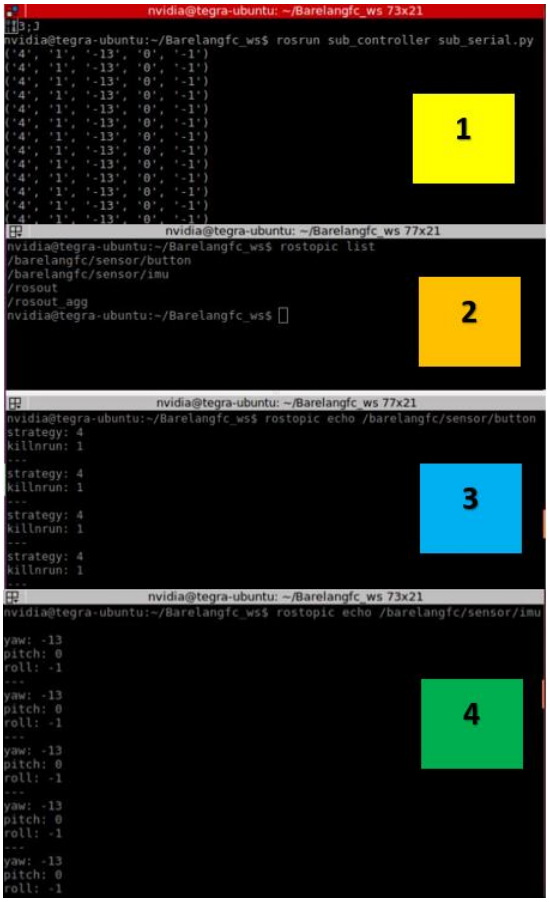


Figure. 11 The ROS topic for the Sub_controller published data.

The two strategy buttons are shown on **Figure. 10** denote with the orange arrows. When the Sub_controller node was connected to the ROS master, then this package will show the data which was published including the button and IMU data by using `"tombol1", "tombol2", "roll", "pitch", "yaw"`, data format, where the `"tombol1", "tombol2"` represented the strategy button data while `"roll", "pitch", "yaw"` acquired from the IMU sensor . For instance, when the Sub_controller was ready to send the data to the subscriber, the ROS topic will send `("4", "1", "-13", "0", "-1")` data format towards it which related to the button strategy and the IMU data sensor. The sending and receiving data processes are illustrated on **Figure. 11** labelled with number. The number one on **Figure. 11** denote to the whole package data from the Sub_controller node which is consists of strategy button and the IMU sensor data. When the strategy button and the IMU sensor have been separated from the Sub_controller node, then the ROS topic list will be added automatically on the workspace which can be seen on **Figure. 11** number 2. Therefore, in order to send the ROS topic for the strategy button and the IMU sensor, it just called the ROS topic name as presented on **Figure. 11** number 3 dan number 4.

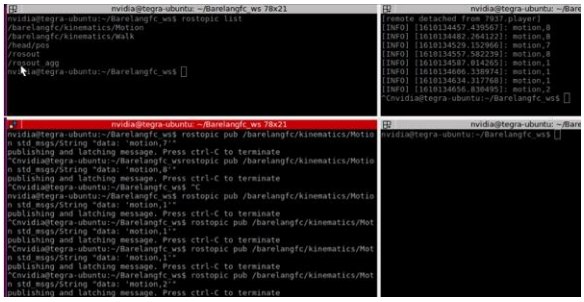


Figure. 12 The Bridge_kinematic ROS topic list.

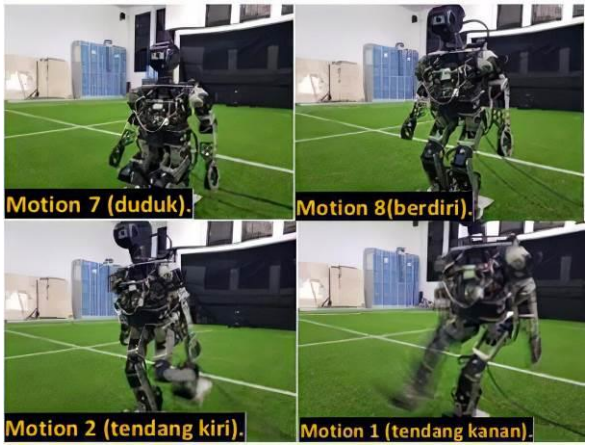


Figure. 13 The robot movement according to the ROS topic command prompt.

The second experiment was integrating the Bridge_kinematic node to the ROS topic which list can be seen on **Figure. 12** including the motion and walk. On **Figure. 12**, the integrating Bridge_kinematic package to the ROS topic consists of four topics were motion 7, motion 8, motion 1, and motion 2. On **Figure. 12** described the command prompt from ROS topic to move the robot according to the ROS topic which was send by the publisher. The result of the command from **Figure. 12** can be seen on **Figure. 13** where motion 7 for sit-down where in **Figure. 13** denote to the "duduk", motion 8 as stand-up command labeled with "berdiri" on **Figure. 13**, motion 2 for kicking to the left described with "tendang kiri" on **Figure.13**, and motion 1 as kick to the right which represented on **Figure. 13** as "tendang kanan". Another, when command the robot for moving forward and backward (x), left and right (y), and even rotating clockwise and counter clockwise (z) for the linear movement and the angular data for rotating movement of the robot. the ROS topic from Bridge_kinematic node can publish not only the linear data (x,y,z) but also the angular data of (x,y,z) including the walking movement of the robot. As the result of sending the Bridge_kinematic node can be seen on **Figure. 14** where the command prompt described on **Figure. 15**. On **Figure.14**, the robot when all the x,y,z send by the ROS topic equals to zero, then robot will stand still on its start point, and when the z has given to 0.6 it will turn to the right and turn to the left when the z filled with -0.9.

Figure. 14 The robot movement for each robot mobility according to the ROS topic prompt.

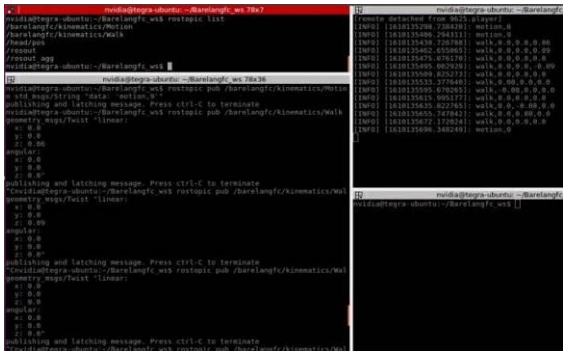


Figure. 15 The ROS topic format sending data from Bridge_kinematic node to the lua kinematics.

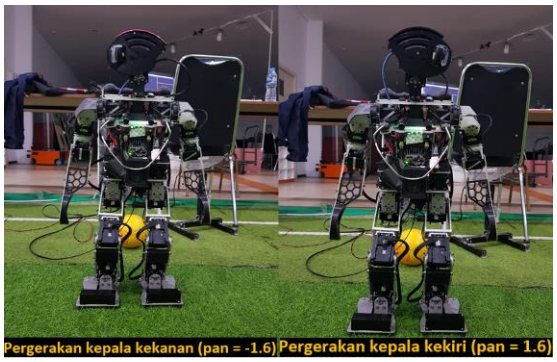


Fig. 16 The head position result from the ROS topic command.

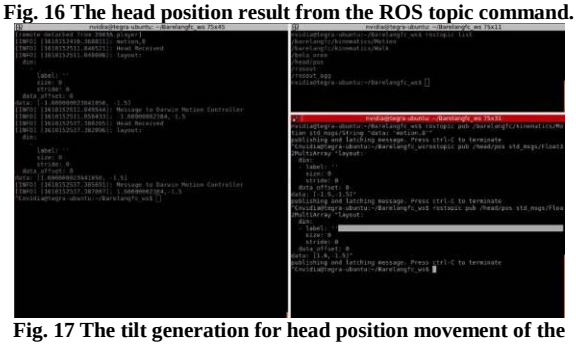


Fig. 17 The tilt generation for head position movement of the ROS topic.

These command on **Figure. 15** used geometry_msgs/Twist command from ROS topic to the subscriber. In this node, the robot also can be commanded to tilt its head to the right and right position by using the head/pos ROS topic. To make the pan movement resembles as human being, then the constraint position needs to be given for each action for instance we set for right pan to -2.0, left 2.0, up -20, and down -0.6. The results of given the pan data to the robot head can be seen on Fig. 16 with the ROS topic prompt described on **Figure. 17**. The prompt on **Figure. 17** generated the head movement result as seen on **Figure. 16**, made the robot experience to pan its head to the right on **Figure. 16** denote to "pergerakan kepala ke kanan" for about -1.6 and pan to the opposite direction for 1.6.

The next experiment integrated the vision system to detect the orange ball through barelang_vision node in order to produce the ball coordinate (x,y), height and width of the ball, ball distance, pole coordinate (x,y), the heigh of each right and left pole and the distance of right and left pole. These whole vision data will be published to the BarelangFC _strategy node as a detection system for BarelangFC robot. As the result of ROS topic vision gave the whole information of the detection represented on **Figure. 18**. On Figure. 18, the robot vision is asked to detect only the orange ball and then gave all the information of the orange ball. As seen on **Figure. 18**, the system only performs the information of the ball considering that the vision did not detect the goal and pole then the goal coordinate, distance, and height data shown only -1 and 0, meanwhile, robot was able to detect the coordinate of the orange ball which x 339, y 420 and the distance (D) from robot to the ball was 113cm, also the height (H) 60 and width (W) 64.

The other work was to integrated the Main_strategy package to the ROS topic which was published from the other packages. In order to verify the inter connection between ROS topic, the vison and Main_strategy packages were run at the same time which can be seen on **Figure. 19**. On **Figure.19**, each of nodes succeed to communicate one another. Moreover, on **Figure. 20**, described whole information about the ROS topic which were connected to the barelangFC_strategy node.

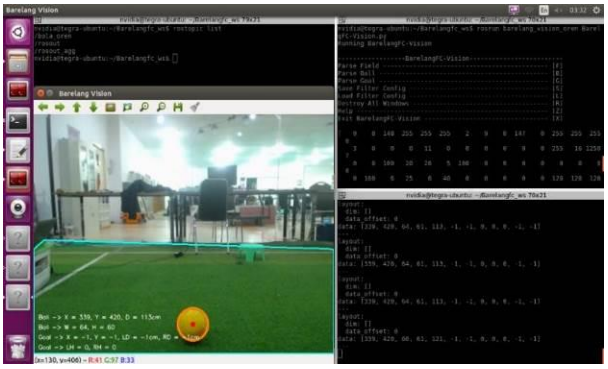


Fig. 18 The detection information from orange ball of the ROS topic vision.

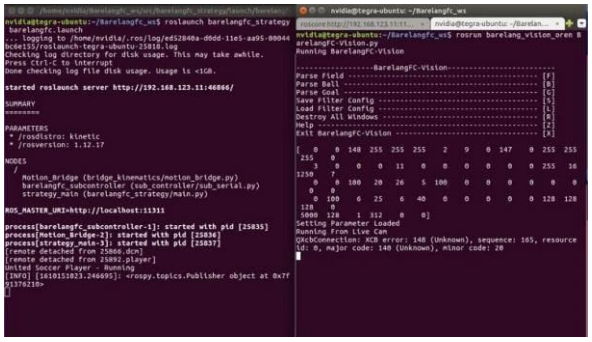


Fig. 19 Running the Main_strategy package and the vision node at the same time.

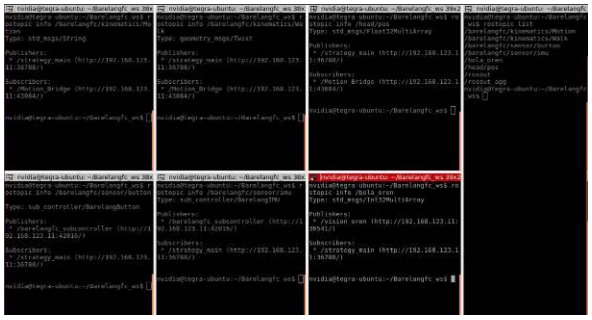


Fig. 20 The ROS topic package information which was published and subscribed on BarelangFC_strategy node.

5. Conclusions

This paper presented a new look of software system on BarelangFC humanoid robot including implementation and construction of the ROS in it. From the experimental results, it can be concluded that all the packages or nodes of the ROS are able to publish their data to the subscriber. The python platform has been utilized in this work as a main programming language for all the packages available in the system. The transmitted and received data flow process also can be seen on the RQT in real-time application from the packages or nodes. Due to the results of ROS construction in this work succeed, therefore, we are going to be reconstructed the other robot software system with the same architecture as we done in this work as the future work.

References

[1] Y. Kim, S. -y. Lee and S. Lim, "Implementation of PLC controller connected Gazebo-ROS to support IEC 61131-3," 2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), 2020, pp. 1195-1198, doi: 10.1109/ETFA46521.2020.9212096.

[2] R. K. Megalingam, A. Rajendraprasad and S. K. Manoharan, "Comparison of Planned Path and Travelled Path Using ROS Navigation Stack," 2020 International Conference for Emerging Technology (INCET), 2020, pp. 1-6, doi: 10.1109/INCET49848.2020.9154132.

[3] S. Rivera, A. K. Iannillo, S. Lagraa, C. Joly and R. State, "ROS-FM: Fast Monitoring for the Robotic Operating System(ROS)," 2020 25th International Conference on Engineering of Complex Computer Systems (ICECCS), 2020, pp. 187-196, doi: 10.1109/ICECCS51672.2020.00029.

[4] Z. Ma, L. Zhu, P. Wang and Y. Zhao, "ROS-Based Multi-Robot System Simulator," 2019 Chinese Automation Congress (CAC), 2019, pp. 4228-4232, doi: 10.1109/CAC48633.2019.8996843.

[5] P. Amontamavut and E. Hayakawa, "ROS Extension of Blue-Sky web based development environment for IoT," 2016 Fifth ICT International Student Project Conference (ICT-ISPC), 2016, pp. 45-48, doi: 10.1109/ICT-ISPC.2016.7519232.

[6] H. Wei, Z. Huang, Q. Yu, M. Liu, Y. Guan and J. Tan, "RGMP-ROS: A real-time ROS architecture of hybrid RTOS and GPOS on multi-core processor," 2014 IEEE International Conference on Robotics and Automation (ICRA), 2014, pp. 2482-2487, doi: 10.1109/ICRA.2014.6907205.

[7] S. Gatesichapakorn, J. Takamatsu and M. Ruchanurucks, "ROS based Autonomous Mobile Robot Navigation using 2D LiDAR and RGB-D Camera," 2019 First International Symposium on Instrumentation, Control, Artificial Intelligence, and Robotics (ICA-SYMP), 2019, pp. 151-154, doi: 10.1109/ICA-SYMP.2019.8645984.

[8] S. Gatesichapakorn, M. Ruchanurucks, P. Bunnun and T. Isshiki, "ROS-Based Mobile Robot Pose Planning for a Good View of an Onboard Camera using Costmap," 2019 10th International Conference of Information and Communication Technology for Embedded Systems (IC-ICTES), 2019, pp. 1-6, doi: 10.1109/ICTEmSys.2019.8695969.

[9] Y. Nitta, S. Tamura, H. Yugen and H. Takase, "ZytleBot: FPGA Integrated Development Platform for ROS Based Autonomous Mobile

Robot," 2019 International Conference on Field-Programmable Technology (ICFPT), 2019, pp. 445-448, doi: 10.1109/ICFPT47387.2019.00089.

[10] R. Mishra and A. Javed, "ROS based service robot platform," 2018 4th International Conference on Control, Automation and Robotics (ICCAR), 2018, pp. 55-59, doi: 10.1109/ICCAR.2018.8384644.

[11] Y. Chang, P. Chung and H. Lin, "Deep learning for object identification in ROS-based mobile robots," 2018 IEEE International Conference on Applied System Invention (ICASI), 2018, pp. 66-69, doi: 10.1109/ICASI.2018.8394348

[12] K. Hasegawa, K. Takasaki, M. Nishizawa, R. Ishikawa, K. Kawamura and N. Togawa, "Implementation of a ROS-Based Autonomous Vehicle on an FPGA Board," 2019 International Conference on Field-Programmable Technology (ICFPT), 2019, pp. 457-460, doi: 10.1109/ICFPT47387.2019.00092.

[13] J. Staschulat, I. Lütkebohle and R. Lange, "The rclc Executor: Domain-specific deterministic scheduling mechanisms for ROS applications on microcontrollers: work-in-progress," 2020 International Conference on Embedded Software (EMSOFT), 2020, pp. 18-19, doi: 10.1109/EMSOFT51651.2020.9244014.

[14] B. B. Rhoades, D. Srivastava and J. M. Conrad, "Design and Development of a ROS Enabled CAN Based All-Terrain Vehicle Platform," SoutheastCon 2018, 2018, pp. 1-6, doi: 10.1109/SECON.2018.8479285.

[15] B. Mirkhanzadeh et al., "A two-layer network Orchestrator offering trustworthy connectivity to a ROS-industrial application," 2017 19th International Conference on Transparent Optical Networks (ICTON), 2017, pp. 1-4, doi: 10.1109/ICTON.2017.8025148.

[16] M. Cai, Y. Wang, S. Wang, R. Wang and M. Tan, "ROS-Based Depth Control for Hybrid-Driven Underwater Vehicle-Manipulator System," 2019 Chinese Control Conference (CCC), 2019, pp. 4576-4580, doi: 10.23919/ChiCC.2019.8865762.

[17] D. Stojanović, M. Krunić, N. Četić and N. Lukić, "Source code generators for ADAS feature deployment in context of ROS and adaptive AUTOSAR applications," 2019 27th Telecommunications Forum (TELFOR), 2019, pp. 1-4, doi: 10.1109/TELFOR48224.2019.8971074.

[18] N. Goerke, D. Timmermann and I. Baumgart, "Who Controls Your Robot? An Evaluation of ROS Security Mechanisms," 2021 7th International Conference on Automation, Robotics and Applications (ICARA), 2021, pp. 60-66, doi: 10.1109/ICARA51699.2021.9376468.

[19] N. Hammoudeh Garcia, M. Lüdtke, S. Kortik, B. Kahl and M. Bordignon, "Bootstrapping MDE Development from ROS Manual Code - Part 1: Metamodeling," 2019 Third IEEE International Conference on Robotic Computing (IRC), 2019, pp. 329-336, doi: 10.1109/IRC.2019.00060.

[20] H. Hong, Z. Wen, S. Bi, Y. Zhang and W. Yang, "RoverOS: Linking ROS with WebSocket for moblie robot," 2019 IEEE 9th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER), 2019, pp. 626-630, doi: 10.1109/CYBER46603.2019.9066498.

[21] Susanto, Junito Suroto, Riska Analia, " The ROS: Kinetic Kame for Humanoid Robot BarelangFC" , Jurnal Integrasi - April 2021, Vol 13 No 1 (2021), pp. 68-77, DOI: https://doi.org/10.30871/ji.v13i1.2686.