Jurnal Integrasi
Vol. 13 No. 1, April 2021, 68-77
e-ISSN: 2548-9828

Article History
Received January 2021
Accepted April 2021

The ROS: Kinetic Kame for Humanoid Robot BarelangFC

Susanto[*], Junito Suroto[*], and Riska Analia[*]

[*]Batam Polytechnics
Department of Electrical Engineering
Parkway Street, Batam Centre, Batam 29461, Indonesia
E-mail: riskaanalia@polibatam.ac.id, susanto@polibatam.ac.id

Abstrak

Sebuah robot kolaboratif seperti robot humanoid yang mampu bermain sepak bola terdiri dari banyak sekali *framework software* seperti servo *controller*, *vision system*, *receiver and transmitter* strategi, sensor, dan sistem koordinasi. Semua *framework software* ini perlu diintegrasikan untuk menyederhanakan perintah dalam menciptakan kompleksitas robot *behaviour*. Untuk mengatasi permasalahan tersebut, maka Robot Operating System (ROS) dapat diimplementasikan pada setiap robot. Makalah ini memaparkan implementasi ROS: Kinetic Kame untuk mengintegrasikan seluruh *framework* yang ada pada robot. Untuk memverifikasi kinerja sistem ini, beberapa percobaan telah dilakukan dalam aplikasi *real-time*. Dari hasil percobaan, ROS: Kinetic Kame mampu mengintegrasikan setiap *framework software* robot dengan respon yang sangat baik.

Kata kunci: Humanoid robot soccer, Software framework, ROS, Kinetic Kame.

Abstract

A collaborative robot such as humanoid robot which able to play soccer consist tons of software framework such as servo controller, vision system, strategy receiver and transmitter, sensors, and coordination system. All these frameworks needed to be integrated to simplify the command of creating the complexity of the robot behaviors. To overcome these problems, the Robot Operating System (ROS) can be implemented on each robot. This paper presented the implementation of the ROS: Kinetic Kame in order to integrated the whole framework which is existed in the robot. To verify the performance of this system, some experiments has been done in real-time application. From the experimental results, the ROS: Kinetic Kame able to integrate each software framework of the robot in very good response.

Keywords: Humanoid robot soccer, Software framework, ROS, Kinetic Kame.

1. Introduction

The ROS described as a flexible framework for writhing robot software. The ROS has some collection of tools, libraries, and conventions which aim to simplify the command for the complex movement of the robot. Since it has been released, the ROS has at least 13 distribution version with the latest one was ROS: Noetic Ninjemys. On progress of the ROS distribution version, many researchers already used the ROS system into their robot such as collaborative robot, navigation system, platform integration, object detection system, and also on the humanoid robot which has been developed by the author.

The usage of ROS on the robot interaction introduced by zhang, et.al [1] namely as Xinxin which able to interact the people in natural way. In this work, the

ROS is used as the core of information processing in order to integrated the hardware and software platform. While Belzunce, et.al [2] developed the control system design of an omni-directional mobile robot to determine the viability of such a system which has been tested along with Gazebo in simulation. The other autonomous robot which introduced by Köseoğlu, et.al [3] adapted the ROS to integrated the hardware platform and software which aim to make them interconnect each other. On the other hands, Park, et.al [4] developed a mapping and localization on the cooperative robot by ROS and SLAM in unknown working area. In this work, they add a new hybrid architecture which only one PC that able to communicate to the robot. The other localization and navigation system control that developed by the ROS presented by [5-9] that mostly they used the SLAM for the localization and used different sensors for the

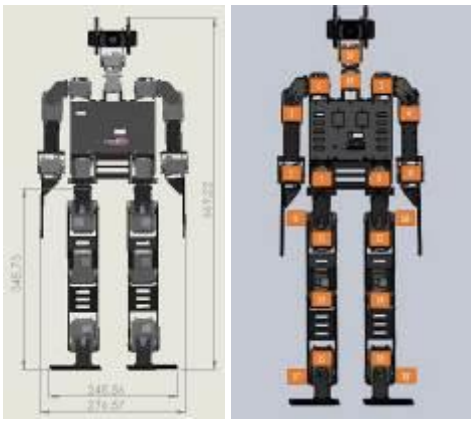

Figure 1: The architecture of the BarelangFC mechanical design.

environment detecting such as LiDAR, Laser Scanner and even the camera. The ROS also can be used for the industrial robot such presented by Chen, et.al [10]. In this work they developed the virtual teaching pendant system for the manipulator robot for establishing the robot model, direct and inverse kinematics manipulator and also the motion planning. In contrast with Chang, et.al [11], they used the ROS as a middleware framework to identify the object based on the deep learning method. In this work, they implement the system into Raspberrry Pi mobile robot. Mostly a collaborative robot consists more than three robots and need to be cooperated each other. One of the challenging works in developing the collaborative is establishing the software framework. Sometimes, one robot can be consisting of more than three software frameworks for make it works as the robot command. The integration of all these frameworks need to be emphasize regarding the performance of the robot in the field. To overcome this situation in [12-19], they implemented the ROS as a bridge to make each of framework on the robot able to communicate and integrate each other. Therefore, in this work we implement the tenth version of the ROS which is ROS: Kinetic Kame on the humanoid robot soccer called BarelangFC. The ROS will be used for send and receive command of all the software frameworks of the robot and make them communicate each other.

2. The Humanoid Robot Soccer

The humanoid robot soccer in this work so called BarelangFC build with 20 unit of servo motors. These servo motors are used as the actuator to move each joint of the robots. The hardware architecture of the BarelangFC can be seen on Error! Reference source not found.**. From Figure 1**, the robot construct in about 66 cm of the height and 27 cm of the width and has 20 DOF. It has 12 DOF for the legs joint, 6 DOF for the arms, and 2 DOF for the neck joint. In terms of servo position, each of servo has its own initial ID for easily troubleshooting problems and also the robot movement. On the neck joint mounted the webcam camera as the vision system. The robot

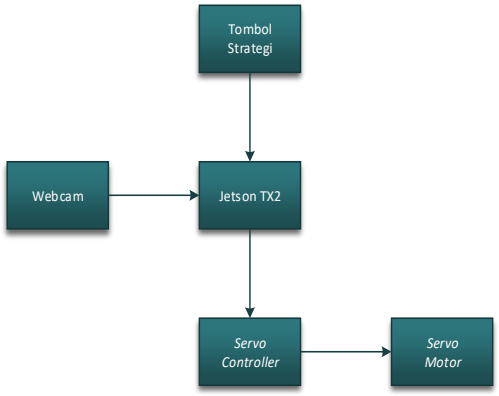

Figure 2: The block diagram of the system in general.

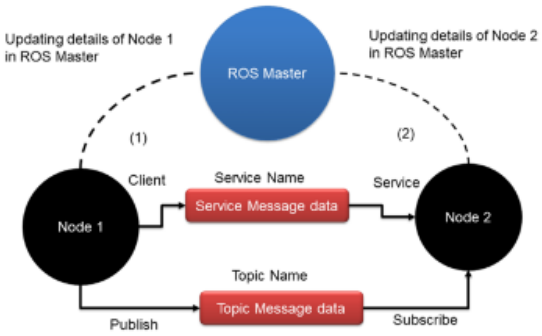

Figure 3: The ROS communication system.

body is made from the alluminium material with 2mm thickness.

This robot used a Jetson TX2 as a controller board to control the flow of all the software framework for each necessary. In order to send the signal to the servos, a servo controller also mounted in the robot. In general, the diagram block of this system can be seen on **Figure 2**. The system also has a "tombol strategi" for give the strategy command to the robot before robot playing soccer.

3. The ROS: Kinetic Kame System

The ROS: Kinetic Kame was the tenth version of the ROS distribution since it released. This ROS version will be implemented in the BarelangFC as a software flow control to command all the software framework that BarelangFC has to work properly based on the system command. The command such as moved the robot from one to another coordinate, detected the object in the field like the ball, goal, teammates, and opponents. A common ROS communication system can be seen on **Figure 3**, where it has a ROS master, Nodes, Service message data which consider as a service name, service, publish, topic message data so called topic name and subscribe. The ROS itself was concepted by three level such as system file, computational concepts, and community level.

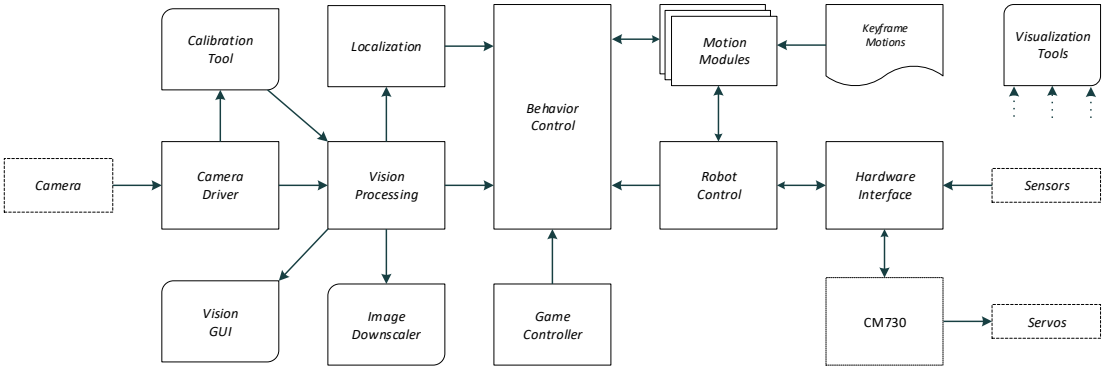

Figure 4: The software frameworks of BarelangFC ROS architecture.

As presented on Figure 4, the ROS was designed in order to move the robot, detect the ball and robot localization to understand the robot position by using the camera as an assistive hardware. Which consists of three main part nodes such as vision system, localization system, and the motion control. The software framework process described as follows: the dotted line on Figure 4 presented the hardware elements, while the curve square represented the extra software framework which not always necessary in framework implementation. The arrow described the information flow control, the square with a wavy bottom edge shown the data file. The visualization tools used to display all the information from the Figure 6.

Figure 6, ROS master will send the data to all three nodes for signing up to the ROS master. All the data which receive from the camera will be send to the calibration node as a receiver and transmitter. The data from camera will be proceed by OpenCV function and generated the camera calibration data such as x, y, z, 1. This calibration data will be published through calibration message with the topic name was usb_cam/image_rect_color to node darknet vision.

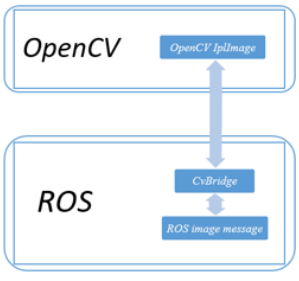

Figure 5: The ROS and OpenCV integration system.

architecture. At the beginning, the ROS system will run the camera to capture the image and process it then the image processing will be transmitted to CM730 for stir the servo motors.

The robot vision perception toward the environment surrounded was the most crucial parts in motion control, object detection while it play soccer. As for the vision system, the ROS was integrated to OpenCV in order to proceed the data visual. As seen on Figure 5, the integration between ROS and OpenCV was illustrated. The ROS used plugin namely as CVBridge in order to send and receive data from the OpenCV. These all components were arranged by one packet node so called image processing node. The image processing node schematic described on Figure 6

While the receiver data from the node calibration will be processed and generated the x, y ball and goal coordinate. Then the calibration data will be published via image message with the topic name was darknet_ROS/found_object toward to barelangmain node. The ROS master will always supervise the data flow through and towards the registered node.

The other node was localization robot node which used to determine the robot position in the field. This node has two methods such as odometry and particle filter that need two inputs such as goal stance coordinate and the steps of the robot. The localization node can be seen on Figure 7. As same as the other node, at first the ROS master will send to all four nodes for singing up to the ROS master. The darknet node on Figure 7 published the goal data and ball coordinate (x, y) throughout the image message with name was topic darknet_ROS/found_object. Then the

goal and the ball coordinate will be subscribed by node BarelangLocalization. The motion bride node on the other hands, published the coordinate robot data through the odometry message namely as topic imu_heading. Then the heading data subscribed by the BarelangLocalization node.

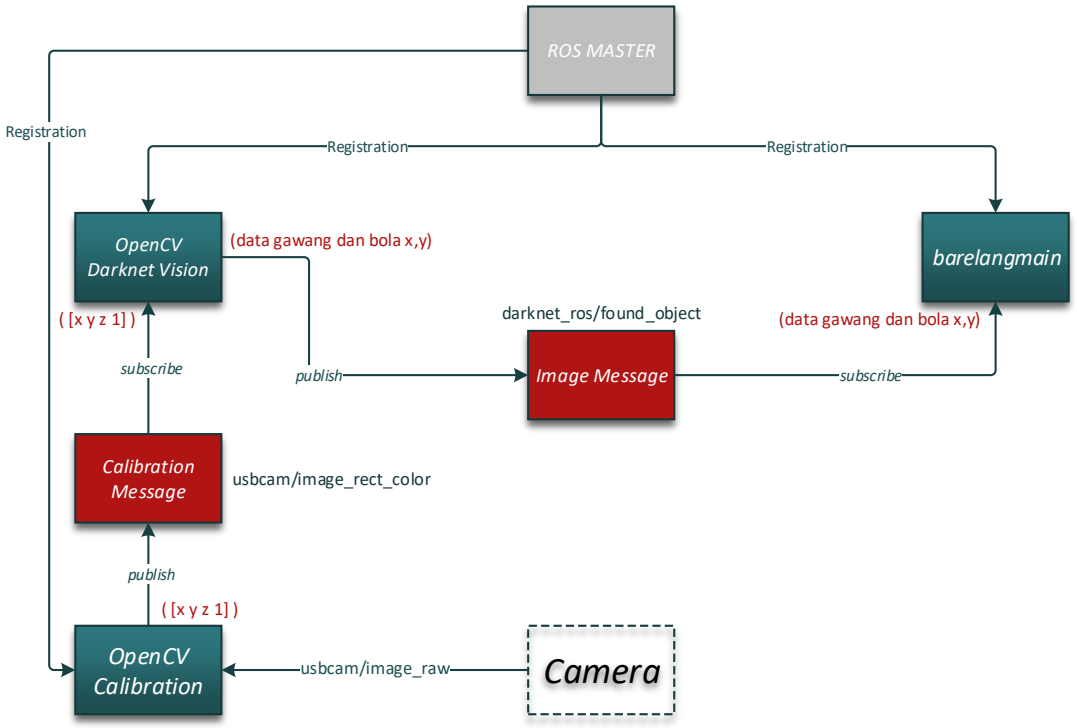

Figure 6: The vision system node in ROS.

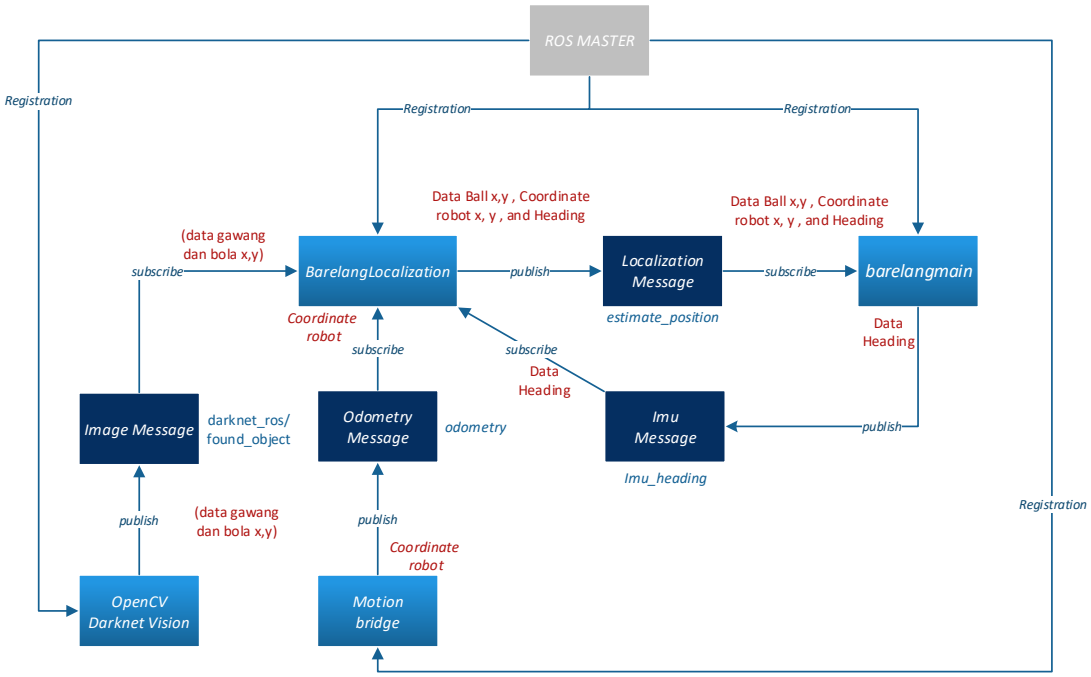

Figure 7: Localization node in ROS system.

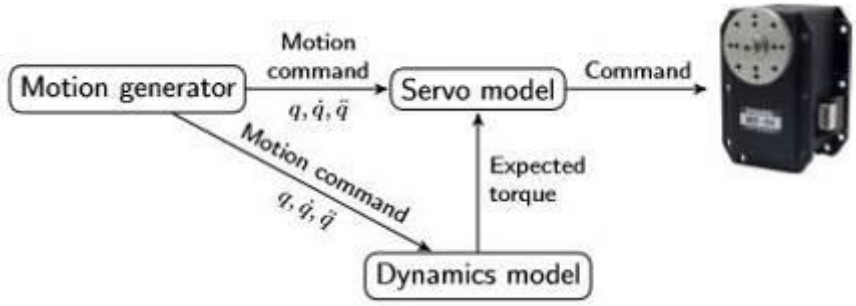

Figure 8: Servo controller system architecture.

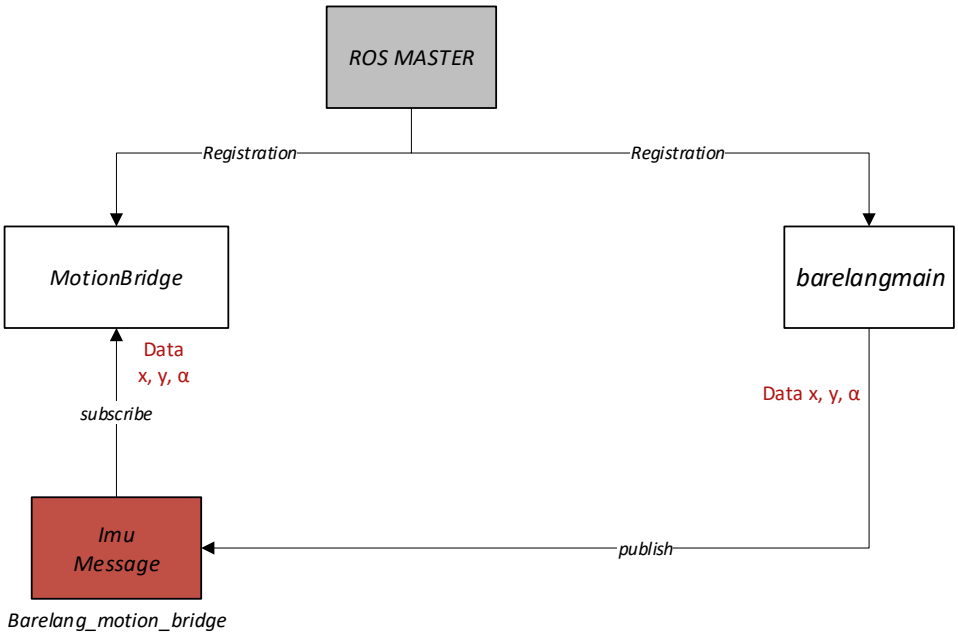

Figure 9: The motion control system in the ROS.

All of the heading, goal, and ball coordinate data will be proceeded in ROS master as the central controller to communicate to the BarelangLocalization node and generate each of (x, y) coordinate of the goal, ball, and heading. The localization message responsible to publish these all data through topic `estimate_position`.

For the motion control, the movement of the robot will be moved by the servo motors which consist of the robot model, kinematic, and dynamics. As seen on **Figure 8**, to command the servo motors it has some process such as motion generator which send the motion command to the servo model and dynamics **Figure 9**, the node consists of `MotionBridge` and `barealangmain`. The `barelangmain` node published the x, y, and α to the node `MotionBridge` to subscribe the data. The x and y were the coordinate while α represented the angle of movement so that the robot will move according to this data.

model. The dynamic model will send the expected torque to the servo model, then the servo model will send the command to each servo. Inside of the motion control, it has the controller node which consists of internal robot model, kinematics, and dynamics. The

main function of controller node on the motion control system was camera, actuators, and for the CM730 signal generator to the servos. The same as before, all the nodes in the ROS motion control system need to be registered to the ROS master. In this system which can be seen on

4. Experimental Results

In this section we do some experiments in order to we integrated the ROS system to each ROS topic of the robot. Then verify the ROS system performance on the BarelangFC such as distribution data process

between input and output which has been received and transmitted by each node. And also, we compared this system between the existing BarelangFC framework in data transferring system and the ROS which has been developed.

When all the ROS topic able to communicate each other in data distribution then each of topic only need to subscribe the node which already integrated. Which seen on **Figure 10**, the data published by the localization node consists of robot coordinate (x, y), heading, ball coordinate (x, y). All of these data can be generated the estimation of robot position. In this case, the localization node already integrated to the ROS topic. While for the motion control node, the list topic which already registered to the node which presented on

Figure 11 namely as `gripper/state`, `head/pos`, `motion/cmd_vel`, and the `motion/state`. The motion/state is used to move the robot motion by only send the string data to the `walk_server.lua` which can be seen on **Figure 12** to **Figure 15**. In this experiment, we send the string data such as stand, stop, start, and sit which represented on **Figure 12** to **Figure 15** respectively.

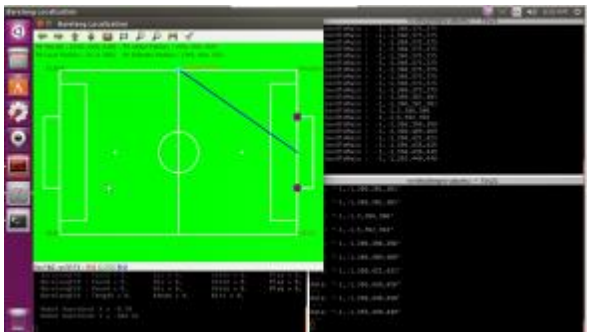

Figure 10: The data and GUI which published by localization node.

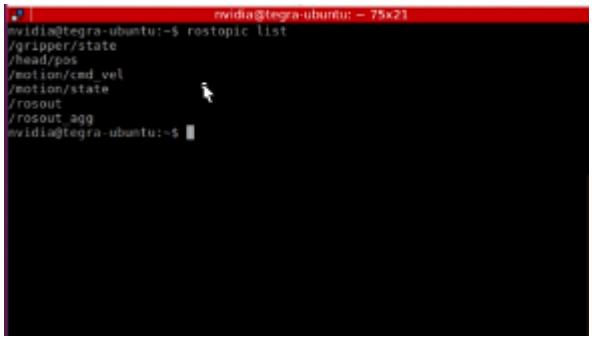

Figure 11: ROS topic list on motion control node.

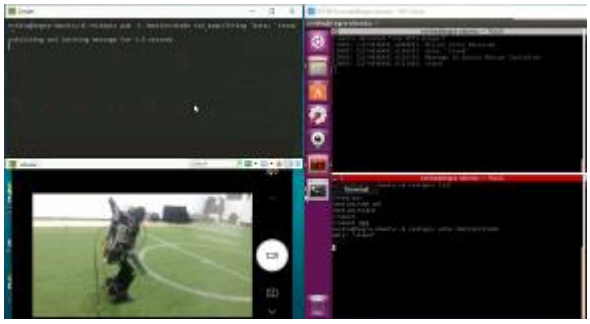

Figure 12: Node motion control subscribed the stand data.

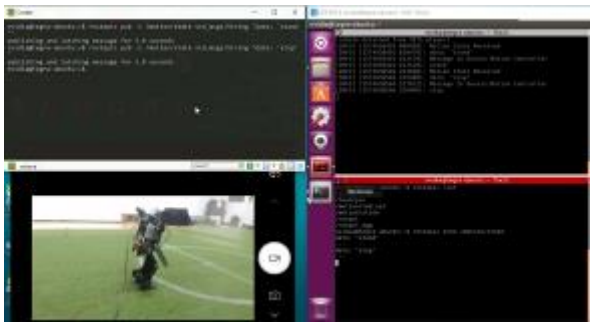

Figure 13: The motion control subscribed the stop data.

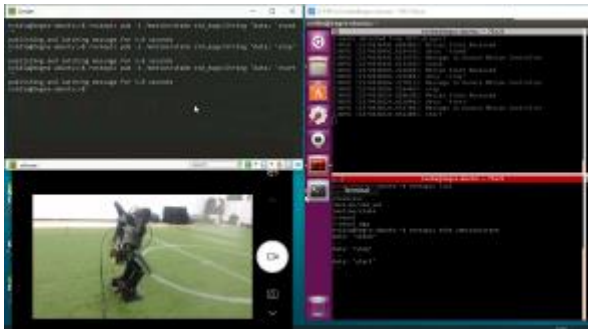

Figure 14: The motion control subscribed the start data.

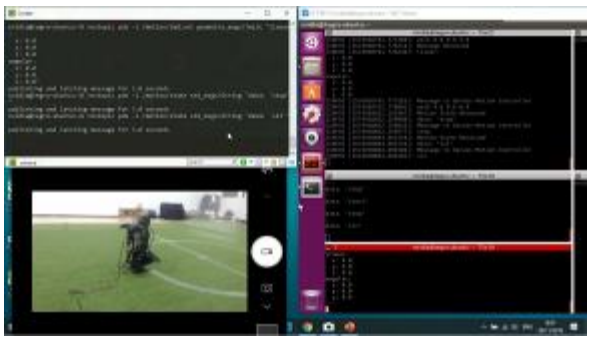

Figure 15: The motion control subscribed the sit data.

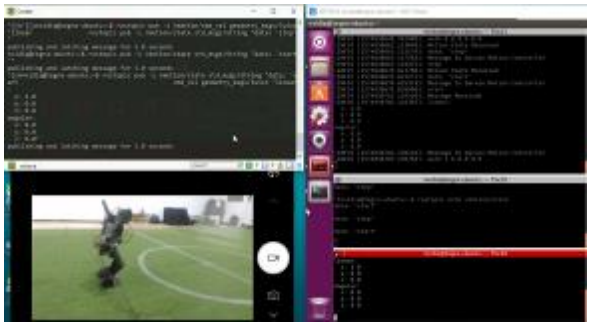

Figure 16: The motion control node subscribed the data "(*linier x=2, y=0, z=0)* and *(anguler x=0, y=0, z=0)".*

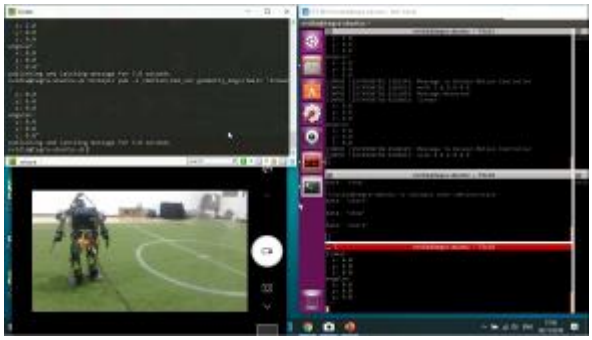

Figure 17: The motion control node subscribed the data *"(linier x=0, y=6, z=0)* and *(anguler x=0, y=0, z=0)".*

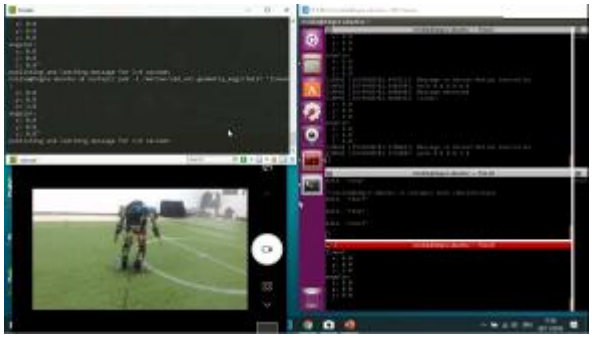

Figure 18: The motion control node subscribed the data "(linier x=0, y=0, z=3) and (anguler x=0, y=0, z=0)".

The ROS topic just need to send the name of the `topic/motion_state` and the published it to the motion control node. Then the motion control node will send the command to the `walk_sever.lua` to make the robot move according to the data which is subscribed before. The other experiments on motion control node was, we send the linear and angular data to make the robot moved into the linear and angular position. As presented on **Figure 16** to **Figure 18**.

On **Figure 16**, the ROS topic published the data "`(linier x=2, y=0, z=0)` and `(anguler x=0, y=0, z=0)`", then the node motion control subscribed this data and send it to `walk_server.lua`. After the `walk_server.lua` got the data it will command the robot to move forward with linear x=-, y=0, and z=0. The z here was the α namely the angle rotation of the robot. In this particularly experiment, the α set to 0 and make the robot only move forward. The next experiment presented on **Figure 17** command the robot to move in horizontal which the ROS topic data was "`(linier x=0, y=6, z=0)` and `(anguler x=0, y=0, z=0)`". Regarding to the data published by the ROS topic, the robot should be moved 6 cm in horizontal position. The last experiment in motion control was to request the robot to move around with the angle was 3 degree. The result presented on Figure **18** where the motion control node got the request from the ROS topic with the data "`(linier x=0, y=0, z=3)` and `(anguler x=0, y=0, z=0)`".

As for the vision system in this work, we integrated the vision orange node to the ROS topic. This topic command the robot to seek the orange ball in the field. As the example on **Figure 19** the vision orange node published the ROS topic called bola_oren which mean orange ball. To display the orange ball coordinate (x,y), the height of the goal stand from the left to the right, the distance of the goal stand from the right to the left, the distance of the ball, the goal distance (x,y) and the heigh of the ball presented on **Figure 20**.

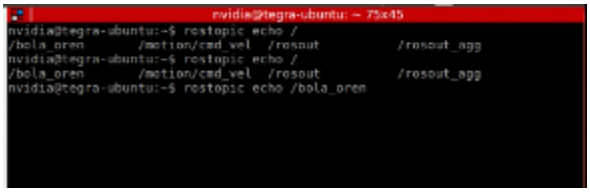

Figure 19: The ROS topic published by vision orange node.

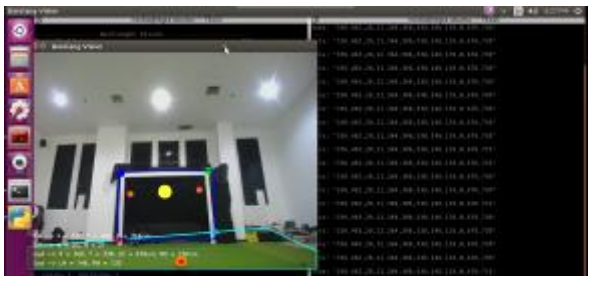

Figure 20: The data which is published by vision orange node.

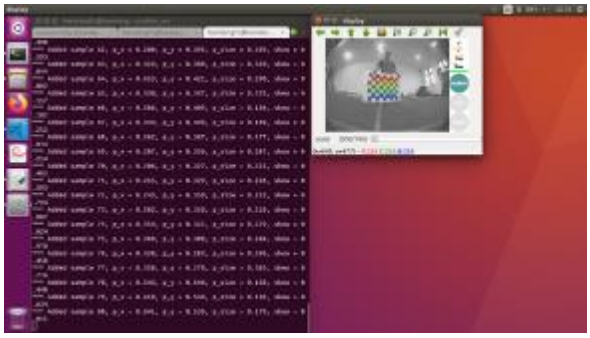

Figure 21: Calibration data collected process.

Before the vision seeking the ball, the vision needed to be calibrated to make sure that the camera was in the right position before capture the object and the robot able to collected some sample data of the object in real-time as much as 80 samples. In this experiment, we developed the calibration node in the ROS namely as vision calibrator node. The calibration collected data process can be seen on **Figure 21**. As the result of vision calibration process, presented on **Figure 22**. On **Figure 22**, the result of calibration process generated the matrix of the camera data, distortion, rectification and projection or x, y, z, 1 and saved the calibration process along with the sample data to "`/tmp/calibrationdata.tar.gz`" directory. The differences between with and without the vision calibration process illustrated on **Figure 23**. The vision results without the vision calibration process illustrated on red box where the camera captured the surrounded environment not centre and little bit tilt. While on the yellow box, the vision result generated by vision calibration process captured the image in the centre position without any tilting. The other differences lays on the green carpet, the red box presented the field look like curve and it was not happening on the yellow box even the camera used the fisheye camera.

Beside doing the calibration to the camera, the ROS topic also integrated to the darknet vision node. This node was the main vision system which able to detect the ball and the goal at the same time precisely. From the result presented on **Figure 24**, the darknet vision node generated the ball coordinate (x,y), left top goal pole (x,y), right top goal pole (x,y), left bottom goal pole (x,y), and right bottom top goal pole (x,y). The darknet vision node also able to generate both using the fisheye camera and a webcam camera. As presented on **Figure 25**. As seen on **Figure 25**, the darknet vision node already calibrated the camera that is why the result clearer to detect the object surrounded such as ball and goal position. In contrast with used the fisheye camera without implemented the calibration node which presented on **Figure 26**. The uncalibrated camera affected to not only the coordinate of each object but also the object detecting itself. As presented on **Figure 26**, because the we skip the calibration process then the vision failed to detect the ball and the right bottom position was unknown coordinate.

From the experimental results, it can be said that the implementation of the ROS in the BarelangFC system able to execute all the software framework inside the robot easily by only send the ROS topic command to each ROS node which has been developed.

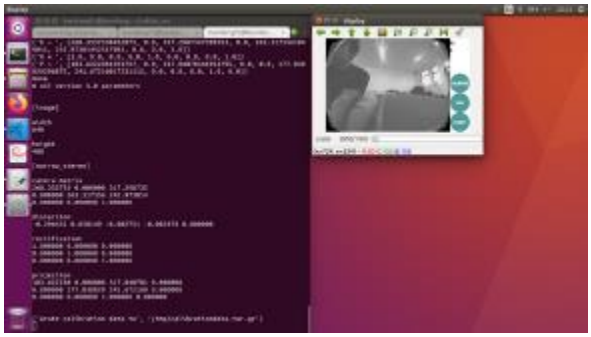

Figure 22: The result of vision calibration process.

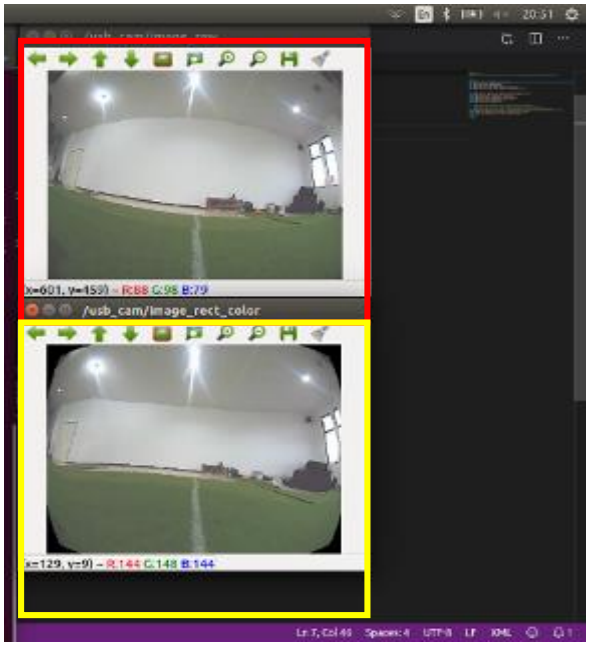

Figure 23: The output result of the vision calibrator node.

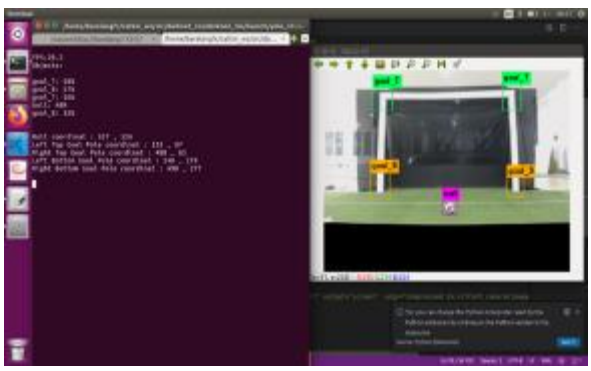

Figure 24: The output result of the darknet vision

node.

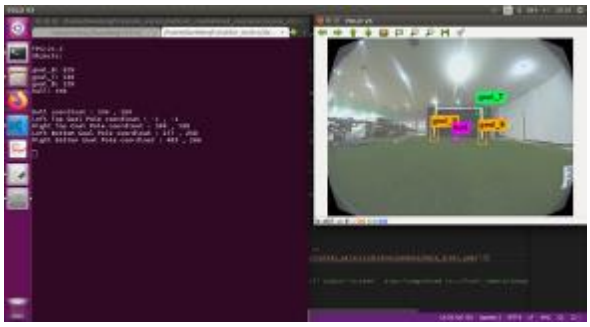

Figure 25: The darknet vision node used the fisheye camera.

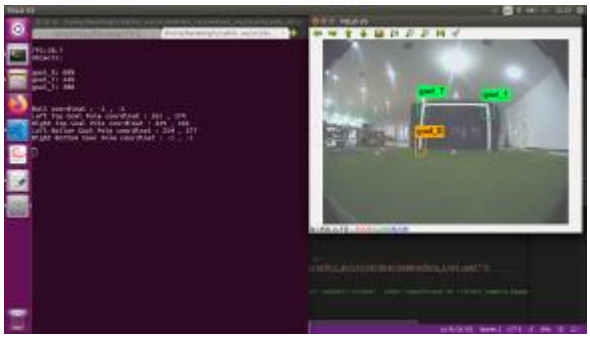

Figure 26: The darknet vision node using the uncalibrated fisheye camera.

5. Conclusions

This work presented the implementation of the ROS: kinetic kame on the humanoid robot soccer. The aim of this work was to integrate the last BarelangFC software framework into the ROS system. From some experimental results and the configuration of the ROS into the BarelangFC robot, this system able to implement in the robot well and also easier in order to send command and display the data from one robot to another. In the future, we will be developed the simulator system like presented by Ma, et.al [20] by using the open source simulator such as Gazebo or Ruiz to develop the simulation mode. So that, all the information of the robot able to monitor in anytime.

References

[1] Zhang Zhaohui, Mei Xuesong, Bian Xu, Cai Hanghang and Ti Jian, "Development of an intelligent interaction service robot using ROS," *2016 IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*, Xi'an, 2016, pp. 1738-1742. doi: 10.1109/IMCEC.2016.7867516

[2] A. Belzunce, M. Li and H. Handroos, "Control system design of a teleoperated omnidirectional mobile robot using ROS," *2016 IEEE 11th Conference on Industrial Electronics and Applications (ICIEA)*, Hefei, 2016, pp. 1283-1287. doi: 10.1109/ICIEA.2016.7603782

[3] M. Köseoğlu, O. M. Çelik and Ö. Pektaş, "Design of an autonomous mobile robot based on ROS," *2017 International Artificial Intelligence and Data Processing Symposium (IDAP)*, Malatya, 2017, pp. 1-5. doi: 10.1109/IDAP.2017.8090199

[4] S. Park and G. Lee, "Mapping and localization of cooperative robots by ROS and SLAM in unknown working area," *2017 56th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE)*, Kanazawa, 2017, pp. 858-861. doi: 10.23919/SICE.2017.8105741

[5] L. Zhi and M. Xuesong, "Navigation and Control System of Mobile Robot Based on ROS," *2018 IEEE 3rd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, Chongqing, 2018, pp. 368-372. doi: 10.1109/IAEAC.2018.8577901

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

[7] W. Wang, Y. Chien, H. Chiang, W. Wang and C. Hsu, "Autonomous Cross-Floor Navigation System for a ROS-Based Modular Service Robot," *2019 International Conference on Machine Learning and Cybernetics (ICMLC)*, Kobe, Japan, 2019, pp. 1-6. doi: 10.1109/ICMLC48188.2019.8949176

[8] S. Oajsalee, S. Tantrairatn and S. Khaengkarn, "Study of ROS Based Localization and Mapping for Closed Area Survey," *2019 IEEE 5th International Conference on Mechatronics System and Robots (ICMSR)*, Singapore, 2019, pp. 24-28. doi: 10.1109/ICMSR.2019.8835455

[9] 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)*, Bangkok, Thailand, 2019, pp. 151-154. doi: 10.1109/ICA-SYMP.2019.8645984

[10] L. Chen, Z. Wei, F. Zhao and T. Tao, "Development of a virtual teaching pendant system for serial robots based on ROS-I," *2017 IEEE International Conference on Cybernetics and Intelligent Systems (CIS) and IEEE Conference on Robotics, Automation and Mechatronics (RAM)*, Ningbo, 2017, pp. 720-724. doi: 10.1109/ICCIS.2017.8274867

[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)*, Chiba, 2018, pp. 66-69.

doi: 10.1109/ICASI.2018.8394348

[12] Jeong Seok Kang, Dong Uk Yu and Hong Seong Park, "A robot software bridge for interconnecting OPRoS with ROS," *2012 9th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, Daejeon, 2012, pp. 296-297. doi: 10.1109/URAI.2012.6462998

[13] S. Shin, D. Yoon, H. Song, B. Kim and J. Han, "Communication system of a segmented rescue robot utilizing socket programming and ROS," *2017 14th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, Jeju, 2017, pp. 565-569. doi: 10.1109/URAI.2017.7992670

[14] X. Ma, F. Fang, K. Qian and C. Liang, "Networked robot systems for indoor service enhanced via ROS middleware," *2018 13th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, Wuhan, 2018, pp. 852-857. doi: 10.1109/ICIEA.2018.8397832

[15] Z. GUO, W. YANG, M. LI, X. YI, Z. CAI and Y. WANG, "ALLIANCE-ROS: A Software Framework on ROS for Fault-Tolerant and Cooperative Mobile Robots," in *Chinese Journal of Electronics*, vol. 27, no. 3, pp. 467-475, 5 2018. doi: 10.1049/cje.2018.03.001

[16] S. Dong, W. Li, Z. An and Z. Xie, "A New Master Control Framework for the Next Generation Service Robots Based on ROS," *2018 37th Chinese Control Conference (CCC)*, Wuhan, 2018, pp. 5164-5168. doi: 10.23919/ChiCC.2018.8483374

[17] P. Anggraeni, M. Mrabet, M. Defoort and M. Djemai, "Development of a wireless communication platform for multiple-mobile robots using ROS," *2018 6th International Conference on Control Engineering & Information Technology (CEIT)*, Istanbul, Turkey, 2018, pp. 1-6. doi: 10.1109/CEIT.2018.8751845

[18] Y. Nitta, S. Tamura and H. Takase, "ZytleBot: FPGA Integrated Development Platform for ROS Based Autonomous Mobile Robot," *2019 29th International Conference on Field Programmable Logic and Applications (FPL)*, Barcelona, Spain, 2019, pp. 422-423. doi: 10.1109/FPL.2019.00077

[19] 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)*, Suzhou, China, 2019, pp. 626-630. doi: 10.1109/CYBER46603.2019.9066498

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