Implementation of Base Station as An Intermediary Referee Box in the delivery of Wheeled Football Robot Movement Commands

Marchiano Alfredo Fouk[1], Hudiono[2], Aad Hariyadi[3]

[1,2] Digital Telecommunication Network Program Study,
Electrical Engineering, State Polytechnic of Malang, Indonesia
[3] Telecommunication Engineering Program Study,
Electrical Engineering, State Polytechnic of Malang, Indonesia

[1]alvredoe1@gmail.com, [2]hudiono@polinema.ac.id, [3]aad.hariyadi@polinema.ac.id

***Abstract*— In the contest of Indonesian wheeled robot football or KRSBI wheeled robot which competes with wheeled football robots, there is a referee who controls the course of the match, the referee box. For referee box to provide commands that can set the course of the robot, a base station is required that serves to receive data from the referee box and translated and then sent it to the robot. The use of TCP socket programming is quite successful whereby 17 attempts to send commands from referee box can be accepted all by base station with a success rate of 100%. The use of robot operating system (ROS) to pass data in the form of coordinate points and commands for robots chasing balls or stops was also successfully carried out where from 17 attempts the delivery of orders, it can reach 100% success rate. After that, the measurement of distance analysis against quality of service (QoS) with 3 different distances, and obtained the following results, at a distance of 3 meters obtained an average delay of 44.41 ms, throughput of 767.29 kbps and no missing package, at a distance of 8 meters obtained an average delay of 7.22 ms, an average throughput of 1065.17 kbps and no missing packages, lastly at a distance of 15 meters obtained an average delay of 103.99 ms, an average throughput of 835.51 kbps and there was a missing package of 6.9% on 1 order so that the average lost package was obtained by 0.405%.**

***Keywords*— Referee Box, Base Station, socket, ROS, Wheeled Robot Football, QoS.**

I. INTRODUCTION

At this time robots have indeed been present in human life in various forms. There are simple robots to do easy or repetitive activities. There are also robots designed to "behave" very complex and to some extent can control themselves [1]. In general, the notion of robots has always been associated with "Living Things" in the form of people and animals made of metal and electrically powered [2]. While in the broadest sense a robot means a system consisting of a mechanism that has an electrical control [3] to carry out a certain task. This understanding is so closely related between robots and automation that it can be understood that almost every activity of modern life is increasingly dependent on robots and automation.

From the understanding of the robot, there are also various types of robotics contests or competitions held in Indonesia, one of which is the Indonesian Robot Contest (KRI). The Indonesian robot contest is a prestigious event attended by students from all universities in Indonesia who are engaged in the world of robotics [4]. The Indonesian robot contest consists of 5 foreign exchanges that will compete, one of which is the KRSBI (Indonesian Football Robot Contest) Wheeled. In the KRSBI event competing with wheeled football robots that will play football like humans using wheels. Meanwhile, there is a Referee Box used in the MSL (Middle Size League) RoboCup [5] or international football robot competition. Referee Box is a game controller that judges use to control matches, such as match time, will start when, calculate the score, the robot wants to be transported, fouls and penalties [1].

To receive data from the Referee Box each team is required to provide a laptop / notebook to run the team Base Station, because the Referee Box only sends commands to each team Base Station. Base Station is a programming application that is created by each team as a data receiver or client of the Referee Box and also acts as a server to forward and translate data from the Referee Box to the robot.

To build a Base Station, python programming is used where there is PyQt in python which can be used to build a Base Station graphic user interface (GUI). In addition to being able to receive commands from the Referee Box used socket programming [6] with python. Socket programming is a way of connecting two nodes/computers on a network to communicate with each other. One socket (node) listens on a particular port on an IP, while another socket reaches the other to form a connection. The server forms a listener socket while the client reaches the server [7].

Meanwhile, in robots, to be able to receive data (in the form of coordinate points) from the Base Station requires a laptop/ mini pc/ raspberry pi because the robot communication and Base Station using the Robot Operation System (ROS) [8][9] where access is needed to connect to the wifi network. But because the robot is not only a command sent by the Base Station, but there is image processing and also kinematic calculations for robot

positioning [10], the author suggests using a mini pc because of its size that is not too large like a laptop and also has better performance in terms of specifications. In addition, the Quality of Service (QoS) [11][12] is particularly measured in terms of delay [13], throughput [14] and packet loss [15].

Based on this background, a Base Station is needed that is able to act as a client against the Referee Box and act as a server for the robot simultaneously so that the robot can move according to the Referee Box command. So that "Implementation of Base Station as an intermediary Referee Box in the delivery of Wheeled Football Robot Movement Orders".

II. METHOD

A. *Research Design*

This design process goes through several stages as follows: Fig. 1 is a research design that will be carried out, the first stage is the study of literature on the research to be carried out. This stage is carried out by looking for various supporting theories in theoretical application. The literature study in this study is about python programming language, TCP socket programming using python, ROS (Robot Operating System), and QoS on data transmission.

The Second Stage is the design of the system, and the design of program creation is the stage of system planning that will be made, starting from the concept of the course of the program that the author will use to the creation of the program and the implementation of the hardware and software to be created.

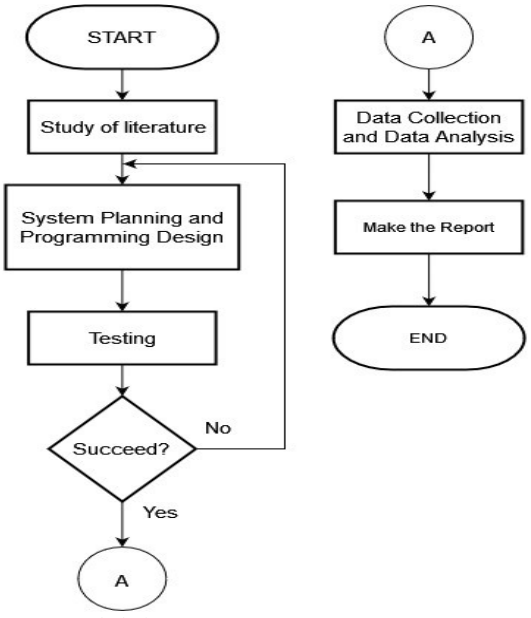


Figure 1. Research design

The third stage is the system testing stage that has been made in the second stage of this test is carried out to find out the results of the system test that has been planned. The fourth stage is the stage of data collection and analysis of data obtained from system testing. The fifth stage is the stage of making a report along with the withdrawal of conclusions and suggestions from the research results to be used as a report.

B. *System Diagram Block*

Based on Fig. 2, the suite of tools consists of a PC / laptop as a base station and referee box application, Access Point and also a wheeled football robot. Access Points are used for wireless communication between base stations and robots and also base station communication with referee boxes. The committee will use the Referee Box to send the match orders to the Base Station. The command from the Referee Box is only sent to the Base Station, the Base Station will forward and translate the command to the Robot so that the robot can move according to the Referee Box command through the Access Point.

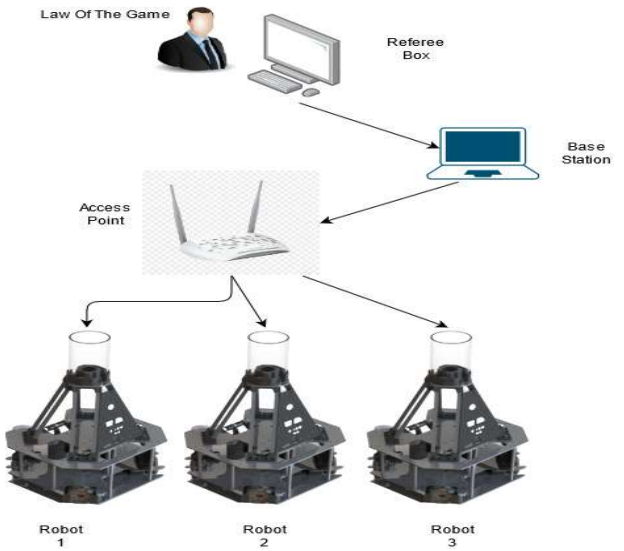


Figure 2 System diagram block [4]

C. *Base Station Layout Design*

Creating a graphic user interface (GUI) from Base Station using PyQt5 Qt Designer.

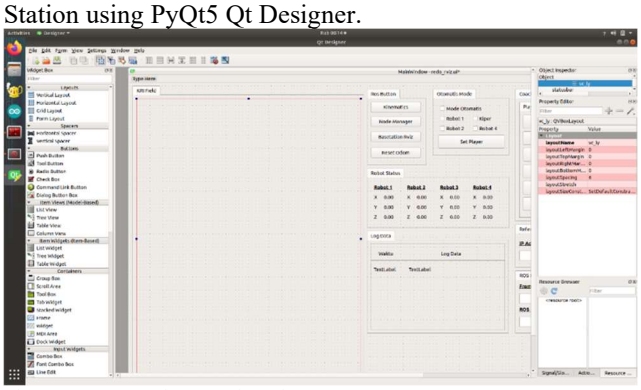


Figure 3. Base station creation view

Fig. 3 shows the creation of a Base Station design starting from the function of the button and the display of information to be displayed in the Base Station.

D. Design of data receipt from referee box

To receive data from the Referee Box used socket programming module python. Before using the socket module in Python, the rare steps are as follows:

1. The socket module must first be imported, as shown in Fig. 4.

```
import socket
```

Figure 4 Import socket

2. Next create a socket. Sockets are created through socket calling (family,type,[prosto]). Proto is optional and is usually worth 0. Here the author uses the IPv4 Protocol family and type Stream Socket (TCP), as ilustrated in Fig. 5.

```
952         self.s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

Figure 5. Creates socket

3. Connecting Socket (Connecting). When another process wants to connect with the server or use the server service, then the process must be connected to the address and specific port number specified by the server. This is done by calling the socket connect (address) method, where the address is a tuple (host, port) for the Internet domain and the pathname for the UNIX domain, as shown in Fig. 6.

```
953         self.s.connect((self.ip, self.port))
```

Figure 6 Connect socket

4. Receive data in the form of bytes and converted to char, as depicted in Fig. 7.

```
959             data = self.s.recv(1024)
960             data= data.decode("utf-8")
961             if not data:
962                 print("Close")
963                 break
```

Figure 7 Receives data and data conversion

E. Designing data transmission to Robots using ROS

To send data from the Referee Box to the robot is used by ROS by publishing the data received to ROS and will be subscribed by the robot. At Base Station, the steps are as follows:

1. Import rospy and message to be used, as ilustrated in Fig. 8.

```
import rospy
from std_msgs.msg import Float32MultiArray, Int32MultiArray
from geometry_msgs.msg import Vector3
```

Figure 8. Import rospy

2. Fig. 9 shows that init node is an identifier that the device will be one of the publishers or subscribers.

```
if __name__ == "__main__":
    rospy.init_node("redo_skripsi")
```

Figure 9. Init node

3. Initialize and create a ROS publisher topic to be used, as shown in Fig. 10.

```
self.pose_desired = rospy.Publisher('/robot1/path_planning_redo', Float32MultiArray, queue_size = 3)
self.mode_publisher = rospy.Publisher('/team/base_station_command', Int32MultiArray, queue_size = 2)
```

Figure 10. Initialization and create publisher topic

4. Fig. 11 and Fig. 12 highlight that translate the data provided by the Referee Box, an example of the data provided is kick off then prepare the nearest coordinate point or pose for the robot to kick off.

```
elif(data == 'K' or data == 'k'):
    self.command[0] = 2
    if(cond):
        self.msg = "Kick Off Cyan"
    else:
        self.msg = "Kick Off Magenta"
```

Figure 11. Translates the received data into command 2

```
#Kickoff
if (self.command[0] == 2 and self.command[1] == 0):
    self.pose[0] = 0.25
    self.pose[1] = 0.0
    self.pose[2] = np.radians(0)
    self.pose[3] = 2.5
    self.pose[4] = 3.5
    self.pose[5] = 1.0
```

Figure 12. Command 2 coordinate points

5. Publish data using the initialization of the publisher that has been created, as shown in Fig. 13.

```
poseSend = Float32MultiArray()
poseSend.data = self.pose
self.pose_desired.publish(poseSend)
cmd_pub = Int32MultiArray()
cmd_pub.data = [2,0,0]
self.mode_publisher.publish(cmd_pub)
```

Figure 13. Publish data

On The Robot, the steps are as follows:

a. Fig. 14 shows the import process.

```
import rospy
import numpy as np
import random
import tf
from numpy.linalg import inv
from geometry_msgs.msg import Vector3, Twist, Quaternion, PoseWithCovarianceStamped
from std_msgs.msg import Int32MultiArray, Float32MultiArray, Int32, Empty
```

Figure 14. Import rospy

b. Init node is as an identifier that the device will be one of the publishers or subscribers, as illustrated in Fig. 15.

```
if __name__ == "__main__":
    rospy.init_node("robot1_control_node")
```

Figure 15. Init node

c. Finally subscribe data from the Base Station and make sure the topic must be the same, as known in Fig. 16.

```
rospy.Subscriber("/team/base_station_command", Int32MultiArray, server.game_mode_callback)
rospy.Subscriber("/robot1/path_planning_redo", Float32MultiArray, server.path_planning_callbackRedo)
```

Figure 16. Subscribe data

F. Flowchart Determination Procedure

Fig. 17 depicts that the system can run when the Referee Box is run first to start the Socket protocol, and then we run the Base Station. Then we enter the IP input and port given by the committee on the Base

Station with the master being the IP and port of the Referee Box.

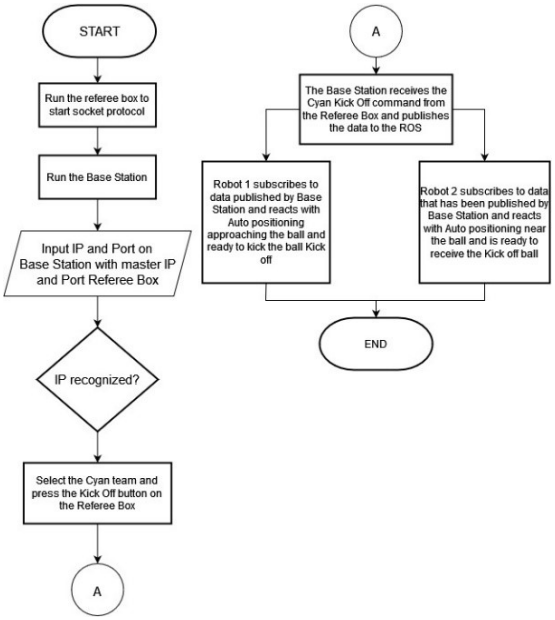


Figure 17. Flowchart Image determining procedures

If the IP has been recognized and appropriate then press the Kick Off button on the Referee Box to be sent to the Base Station, if it has been received by the Base Station then the Base Station will directly publish the data continuously into ROS, so when the command is no longer Kick Off but stop then the published data also changes. Robots 1 and 2 will also continuously subscribe to the data published by the Base Station and will React According to Orders from the Referee Box published by base station.

III. RESULTS AND DISCUSSION

A. *Base Station View*

Fig. 18 depicts base station view controlling numerous funtions.

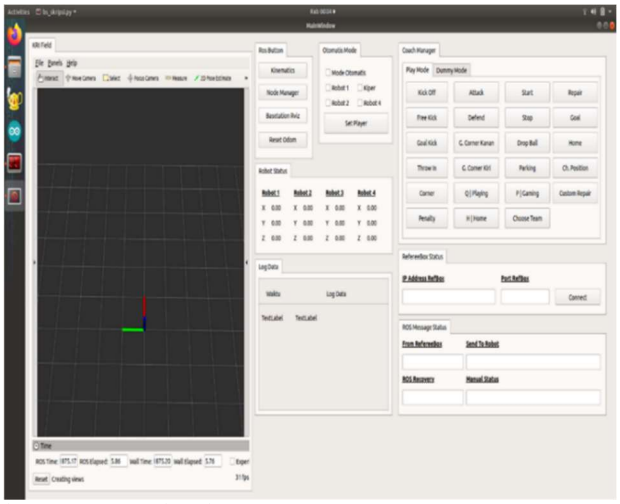


Figure 18. Base station view

B. *Base Station Parameter Testing*

The tests carried out include three parts, namely Testing the delivery of commands contained in the Referee Box to the Base Station, testing Subscriber and Publisher ROS, testing the influence of distance on QoS (Throughput, Packet Loss and Delay).

TABLE 1
DATA TYPE CHECKING TABLE SENT BY REFEREE BOX AND BASE STATION CONVERTED DATA

No.	Given command features	Data types received	Converted data
1.	Kick Off Cyan	Byte	'K'
2.	Stop	Byte	'S'
3.	Start	Byte	's'
4.	Drop Ball	Byte	'N'
5.	Free Kick Magenta	Byte	'f'

From the experiment that has been done, namely checking the data transmission sent by the Referee Box to the Base Station, where it is obtained that all commands sent by the Referee Box use byte data types and each command is successfully converted into a different character, as shown in Table 1.

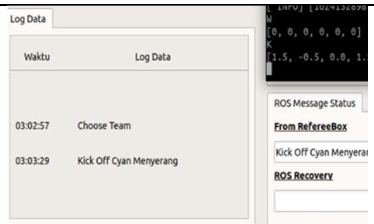
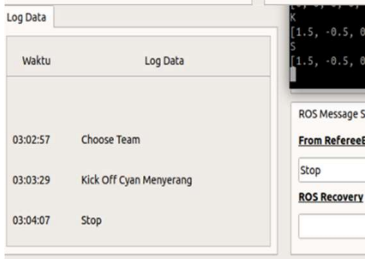
From Table 2 shows app feature checking of the experiments that have been done, it shows that all data transmission experiments from the Referee Box have been successfully received by the Base Station very well and the success rate is 100% of the 5 delivery attempts.

C. *Delay Testing*

This QoS delay measurement serves to find out the value of the delay during the lecturer's location data request process from Firebase to the android application.

The results of delay measurements for each building router based on delay values in accordance with the TIPHON version as standardization, namely the average delay index in Table 3 for each building with 7 Users in AH Building, 7 Users in AI Building, and 7 Users in AL Building.[8]

TABLE 2
TABLE OF TEST RESULTS ON BASE STATION

No.	Features	Test Results	Information
1.	Kick Off Cyan		Succeed
2.	Stop		Succeed

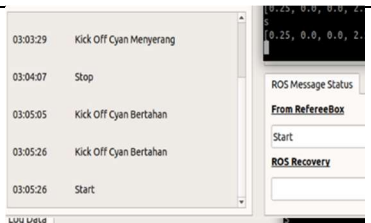
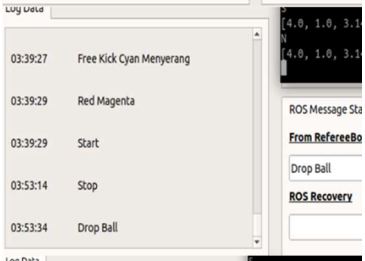
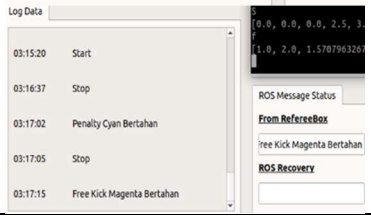
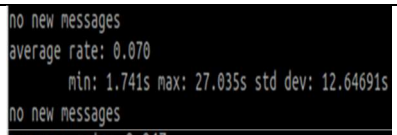
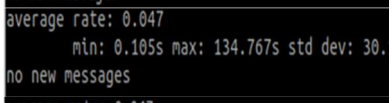
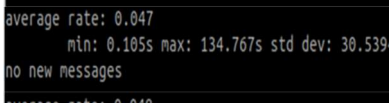
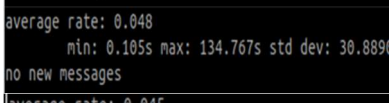
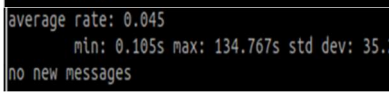
No.	Features	Test Results	Information
3.	Start		Succeed
4.	Drop Ball		Succeed
5.	Free Kick Magenta		Succeed

TABLE 3
TESTING PUBLISH DATA TO ROS

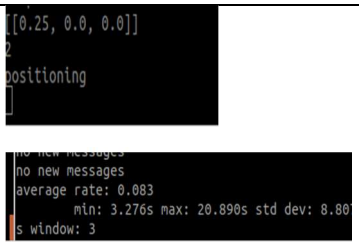
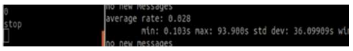
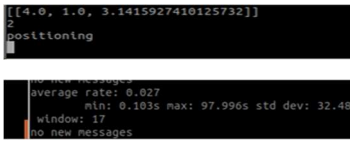
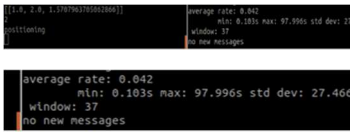
No.	Features	Test Results	Information
1.	Kick Off Cyan		Succeed
2.	Stop		Succeed
3.	Start		Succeed
4.	Drop Ball		Succeed
5.	Free Kick Magenta		Succeed

From table 3 it obtained very good results, namely from the 5 commands received by the Base Station from the Referee Box successfully published all by the Base Station to ROS.

D. *Test results subscribe data from base station to robot and robot reaction*

From the Table 4. that the test have been done, very good result, where the robot managed to move according to the commands subscribed from ROS (sent by the Base Station) with a 100% accuracy rate.

TABLE 4
TEST SUBSCRIBE ROS DATA AND ROBOT REACTIONS

No	Features	Test Results	Information
1.	Kick Off Cyan		Successfully to the coordinate point
2.	Stop		Successfully Stop
3.	Start		Successfully Chasing the Ball
4.	Drop Ball		Successfully Reaching the Coordinate Point
5.	Free Kick Magenta		Successfully Reaching the Coordinate Point

E. *Qos Testing at a Distance of 3 Meters*

TABLE 5
QOS TESTING AT A DISTANCE OF 3 METERS

No	Throughput (Bytes /second	Total Packet Loss (%)	Average delay(millisecond)
1	140166.66	0	6,347157
2	4161.76	0	204.279946
3	141500	0	5.831139
4	106125	0	7.533571
5	141500	0	6.0113
6	120142.85	0	6.982781
7	121285.71	0	7.428607
8	120142.85	0	7.406059
9	94333.33	0	9.211206
10	141500	0	6.369827
11	141500	0	5.950113
12	93181.81	0	10.679091
13	120142.85	0	6.576263
14	121285.71	0	7.446034
15	50875	0	47.907047
16	6691.17	0	204.496399
17	5097.56	0	204.629705
Total Average	98213.66	0	44.416

From table 5 obtained measurement results at a distance of 3 meters, namely the average throughput value of 98213.66 Bytes / second or 767.29 kbps, with no packets lost during delivery and an average delay value of 44,416 ms.

F. QoS Testing Distance of 8 Meters

TABLE 6
QOS TESTING AT A DISTANCE OF 8 METERS

No	Throughput (Bytes /second)	Total Packet Loss (%)	Average delay(millisecond)
1	140166.66	0	6.300789
2	121285.71	0	7.162129
3	106125	0	7.600477
4	141500	0	6.03399
5	150166.66	0	6.22268
6	120142.85	0	6.541954
7	141500	0	6.199733
8	140166.66	0	5.945424
9	141500	0	5.526767
10	102500	0	9.724251
11	141500	0	5.938019
12	144166.66	0	6.00421
13	145285.71	0	7.459621
14	147500	0	6.141892
15	146800	0	15.151326
16	117700	0	9.748791
17	169800	0	5.19793
Total Average	136341.52	0	7.229

From the tests at Table 6 that have been carried out, the measurement results were obtained at a distance of 8 meters, namely the average throughput value of 136341.52 Bytes / second or 1065.17 kbps, with no packets lost during delivery and an average delay value of 7,229 ms.

G. QoS testing at a Distance of 15 Meters

TABLE 7
QOS TESTING AT A DISTANCE OF 15 METERS

No	Throughput Bytes /second	Total Packet Loss (%)	Average delay(millisecond)
1	595.83	6.9	1632.63757
2	94333.33	0	9.32627
3	121285.71	0	6.996346
4	106125	0	8.213324
5	101700	0	10.388606
6	146166.66	0	6.372932
7	110625	0	7.878188
8	64692.30	0	13.438615
9	84900	0	9.692888
10	106125	0	8.047618
11	147500	0	6.304454
12	106125	0	7.944511
13	96142.85	0	7.122304
14	141500	0	6.215347
15	123727.27	0	10.835347
16	117700	0	10.154077
17	148833.33	0	6.364304
Total Average	106945.72	0.405	103.996

Based on the Table 7 that have been carried out, throughput results at a distance of 15 meters with an average value of 106945.72 and an average value of lost packages of 0.405% and also an average delay of 103,996 ms.

IV. CONCLUSIONS

Referee Box need an intermediary to send the command movement for the robot, so we must make a software named Base Station which can received the command and send it to the robot according to the rules. And now we had made the Base station used python programming with socket programming method and it had successfully done. The command data from Referee Box can be received very well by Base station and also can be forwarded to robots with 100% successful data accuracy rate of all tried commands. The results of the quality of service from the Base station were also not too bad with a very good category with the highest average value at a distance of 15 meters at 103.9960412 ms and the lowest average delay value at a distance of 8 meters of 7.229410765 ms. For the highest average throughput result at a distance of 8 meters with a value of 136341.52 bps1065.17 kbps and the lowest at a distance of 3 meters with a value of 98213.66 Bps or 767.29 kbps. And for Packet Loss obtained the result of no packets lost at a distance of 3 and 8 meters while at a distance of 15 meters obtained an average of 0.405% lost packets.

REFERENCE

[1] N. Solehah, A. Siahaan, W. Astuti, and R. R. Larengi, "Read TCP/IP Using the Matlab platform on Wheeled Football Robots," no. 3, pp. 19–24.

[2] K. Cahyono, I. K. Wibowo and M. M. Bachtiar, "A New Kicker System of Wheeled Soccer Robot ERSOW Using Fuzzy Logic Method," 2020 International Electronics Symposium (IES), 2020, pp. 219-225, doi: 10.1109/IES50839.2020.9231882.

[3] and W. D. Fitri*, Kiki Reski R, Ady Rahmansyah, "Use of Python Programming Language as Control Center on 10-D Robots," 2017.

[4] Ristekdikti, 2019 Wheeled KRSBI Guide. 2019.

[5] Y.B. P. Fahriza Azwar Muhammad, Rizky Arif Windiarto, "Socket Programming For Raspberry Pi Abtara Connection With Referee Box," 2016.

[6] K.B. Kusumo, P. H. Trisnawan, and K. Amron, "Implementation of Socket Programming in Packet Size Adjustment on IPv6 Networks," vol. 4, no. 2, pp. 572–580, 2020.

[7] J. F. Kurose et al., Computer Networking A Top-Down Approach Seventh Edition. 2017.

[8] I. Siradjuddin, S. Wibowo, R. P. Wicaksono, and G. Al Azhar, "Distribution of Wheeled Robot Motor Control using ROS," 2019.

[9] J.M. O'Kane, A gentle introduction to ROS, no. 2.1.3. 2016.

[10] MAATA, Rolou Lyn R., et al. Design and implementation of client-server based application using socket programming in a distributed computing environment. In: 2017 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC). IEEE, 2017. p. 1-4.

[11] R. Wulandari, "QoS (QUALITY OF SERVICE) ANALYSIS ON THE INTERNET NETWORK (CASE STUDY: UPT SITE TEST JAMPANG KULON – LIPI MINING TECHNIQUE)," J. Tek.

Inform. and Sist. Inf., vol. 2, no. 2, pp. 162–172, 2016, doi: 10.28932/jutisi.v2i2.454.

[12] D. Irawan, "Analysis and Tapping of Computer Network Data Packet Transmission Using Wireshark," Anal. and Transm Wiretapping. Data Jar Plan. Comput. Using Wireshark, vol. 7, no. 1, pp. 1–5, 2017.

[13] NDATINYA, Vivens, et al. Network forensics analysis using Wireshark. International Journal of Security and Networks, 2015, 10.2: 91-106.

[14] SANDERS, Chris. Practical Packet Analysis, 3E: Using Wireshark to Solve Real-World Network Problems. No Starch Press, 2017.

[15] SHIN, Seona, et al. Communication system of a segmented rescue robot utilizing socket programming and ROS. In: 2017 14th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI). IEEE, 2017. p. 565-569.