Design of a Wireless Communication System for Barelang V Mobile Robot Control

Muhammad Zainul Daffa[1] and Rifqi Amalya Fatekha[2]

[1]Department of Electrical Engineering, Robotics Engineering Study Program, Batam State Polytechnic, Batam, Indonesia

[*]Email: muhammadzainuldaffa526@gmail.com

***Abstrak*— Penelitian ini mengevaluasi performa komunikasi nirkabel berbasis WiFi pada sistem kendali robot bergerak yang terdiri dari ESP32 sebagai pengirim data, mini-PC sebagai penerima dan pencatat data, serta Teensy 4.1 sebagai pengendali motor. Analisis dilakukan melalui pengujian interval pengiriman data, mode hemat daya WiFi, dan daya pancar transmisi untuk memperoleh konfigurasi komunikasi terbaik. Parameter yang dievaluasi meliputi latensi, jitter, standar deviasi latensi, dan tingkat kehilangan paket. Selanjutnya, konfigurasi terpilih diuji pada jarak 5 meter, 10 meter, dan 15 meter. Hasil pengujian menunjukkan bahwa sistem mampu mempertahankan komunikasi yang stabil dengan latensi rata-rata rendah dan tanpa kehilangan paket hingga jarak 15 meter, meskipun nilai latensi maksimum tidak selalu meningkat secara linier terhadap jarak akibat sifat stokastik jaringan nirkabel.**

Kata Kunci- ESP32, WiFi, Komunikasi Nirkabel

Abstract— This study evaluates the performance of WiFi-based wireless communication in a mobile robot control system consisting of an ESP32 as the data transmitter, a mini-PC as the data receiver and recorder, and a Teensy 4.1 as the motor controller. The analysis was conducted by testing the data transmission interval, WiFi power saving mode, and transmission power to obtain the best communication configuration. The parameters evaluated included latency, jitter, latency standard deviation, and packet loss rate. Furthermore, the selected configuration was tested at distances of 5 meters, 10 meters, and 15 meters. The test results showed that the system was able to maintain stable communication with low average latency and no packet loss up to a distance of 15 meters, even though the maximum latency value did not always increase linearly with distance due to the stochastic nature of wireless networks.

Keyword- ESP32, WiFi, Wireless Communication

I. INTRODUCTION

Communication systems play an important role in mobile robot control because they directly affect system latency, reliability, and responsiveness [1], [2], [3]. Low latency is necessary for robots to respond to commands in real time, while reliable communication ensures operational stability and safety [4].

Various wireless protocols such as Bluetooth and Wi-Fi have different characteristics in terms of range, bandwidth, and power consumption. Bluetooth is generally used for short-range communication, while Wi-Fi offers greater bandwidth and range. The combination of these two technologies enables a flexible and adaptive robot control system that can be tailored to the needs of the application [5].

A Multi-layer architecture is widely used in modern robotics to separate input, data processing, and control execution functions [6]. This approach improves system modularity and allows for optimization at each layer. The use of devices such as ESP32, mini-PC, and Teensy 4.1 supports this division of tasks, where ESP32 handles wireless communication, mini-PC acts as a data manager, and Teensy 4.1 executes motor control.

Time synchronization and transport protocol selection also affect communication system performance [7], [8]. Network Time Protocol (NTP) is used to maintain time consistency between devices [9], while UDP is chosen for its low latency and suitability for continuous control data transmission [10]. Based on this, this study designs and evaluates an integrated communication system for mobile robot control with a focus on analyzing latency, jitter, and communication reliability performance.

II. METHOD

A. System Design

The communication system for robot control designed in this study uses a multi-layer architecture that integrates various communication protocols. Figure 1 shows the overall system design consisting of five main components: PS3 Controller, ESP32, Router (Access Point), Mini-PC, and Teensy4.1.

Fig 1. System Design

The PS3 controller functions as an input device for entering robot control commands [11]. This controller provides two analog sticks for translation and rotation control, four digital buttons for special functions, as well as additional triggers and accelerometers. Communication uses the Bluetooth Classic protocol with MAC address-based pairing.

The ESP32 acts as a protocol bridge between Bluetooth and Wi-Fi [12]. The ESP32 receives data from the PS3 controller via Bluetooth, adds a timestamp and sequence number for packet tracking, and then transmits the data in the form of UDP packets to the mini-PC via Wi-Fi. The ESP32 also synchronizes time with an NTP server for accurate timestamping.

The wireless router functions as an access point and DHCP server [13]. The router manages the local network connecting the ESP32 to the mini-PC, providing an isolated network with encryption for data security and automatic IP address allocation.

The Mini-PC acts as middleware that receives UDP packets from ESP32 [14], parses controller data, calculates communication latency, detects packet loss based on sequence numbers, logs data to CSV files, and forwards data to Teensy via serial communication. The Mini-PC also implements an NTP server for system time synchronization.

Teensy 4.1 as the main microcontroller for executing control of the three-wheeled omnidirectional robot [15]. Teensy receives data from the mini-PC, parses commands, and executes control algorithms that include kinematic transformations to calculate target wheel speeds, PID control for motor regulation based on encoder feedback, odometry calculations for position estimation, and sensor fusion with the BNO055 IMU for heading correction. Teensy implements a safety mechanism in the form of timeout detection that automatically stops the motor if there is no new data.

The communication flow begins with the operator providing input via the PS3 controller. The data is transmitted via Bluetooth to the ESP32, then the ESP32 sends a UDP packet to the mini-PC. The mini-PC performs processing, logging, and forwards the data to Teensy via serial. Teensy executes inverse kinematics, PID control, and odometry updates based on the commands received.

B. Communication Protocol

The communication protocol is designed with a text-based format using comma-separated values for easy parsing and debugging. The protocol consists of a sequence number for tracking and packet loss detection, a timestamp with microsecond precision for time synchronization, and controller data that includes analog stick values and button status. Each field is identified by a key followed by a colon and a value, with commas as delimiters.

Wireless communication between ESP32 and mini-PC uses UDP as the transport layer protocol. UDP was chosen because of its connectionless characteristics, which eliminate connection establishment overhead and the absence of an acknowledgment mechanism that can cause variable latency. The mini-PC performs packet loss detection by comparing the sequence numbers of consecutive packets.

Communication between the mini-PC and Teensy uses a USB serial connection with a newline character as the message delimiter. USB serial was chosen because of its deterministic timing and error-free transmission, which are critical for control execution.

1) ESP32 Algorithm

The algorithm on the ESP32 is divided into two main stages, namely the initialization phase and the operational phase. The ESP32 algorithm flowchart is shown in Figure 2.

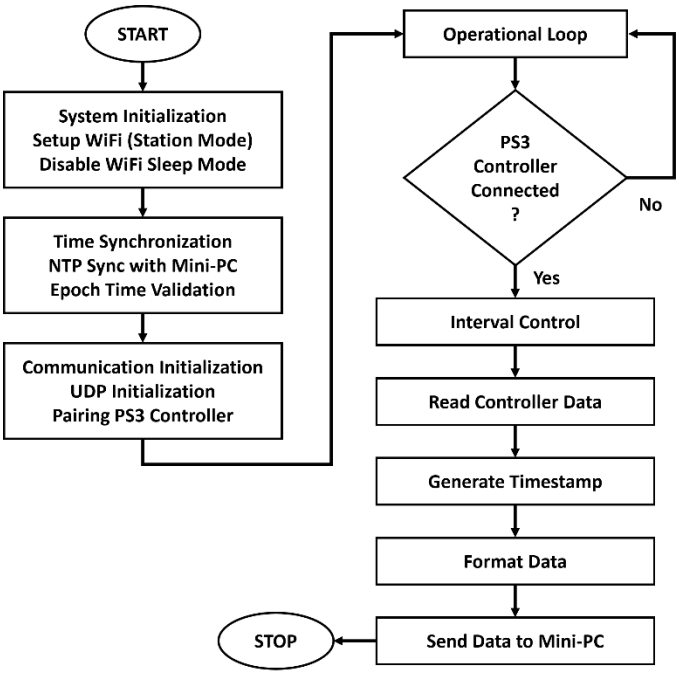

Fig 2. ESP32 Algorithm Flowchart

During the initialization phase, the ESP32 prepares the communication system and time. The device first connects to the WiFi network in station mode, then synchronizes the time with the NTP server so that the entire system has the same time reference. After that, UDP-based network communication is activated, and the ESP32 performs the pairing process with the previously configured PS3 controller.

The operational phase runs continuously. At this stage, the ESP32 checks the connection status of the PS3 controller. If the controller is connected, the system maintains a consistent data transmission interval according to the specified frequency. When the transmission time is reached, the ESP32 reads the input data from the controller, adds time and data sequence information, and then sends the data to the mini-PC via the network. This process ensures that control data is sent periodically and in a structured manner.

2) Mini-PC Algorithm

The mini-PC algorithm is implemented as a data receiver and management system. The flowchart of the mini-PC algorithm is shown in Figure 3.

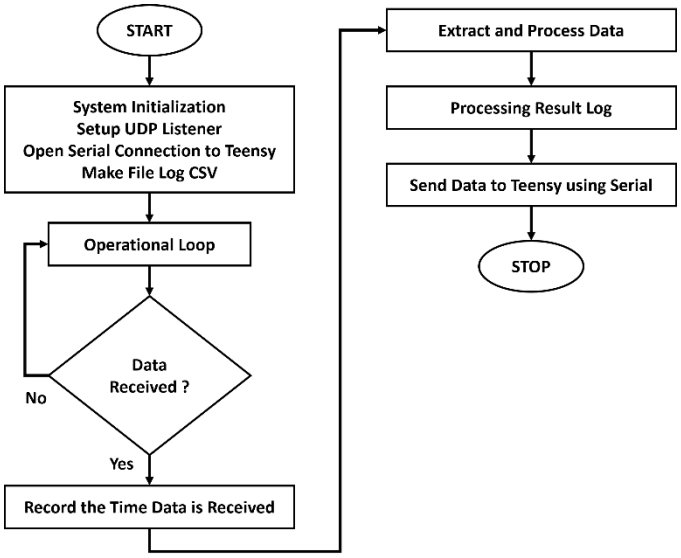

Fig 3. Mini-PC Algorithm Flowchart

During the initialization phase, the mini-PC sets up network communication to receive data from the ESP32, opens a serial connection to the Teensy device, and prepares log files for data recording. The recording structure is set up from the outset so that all data received can be recorded systematically.

During the operational phase, the mini-PC waits for data sent by the ESP32. When data is received, the reception time is immediately recorded as a reference for analysis. The received data is then processed to extract important information such as sequence numbers and timestamps. Based on this information, the system can calculate communication latency and detect possible data packet loss.

All processing results are recorded in a log file, while control data is forwarded to Teensy via serial communication. Thus, the mini-PC acts as both a connector and a recorder of communication system performance.

3) Teensy 4.1 Algorithm

The algorithm on Teensy 4.1 focuses on motor control execution. The Teensy algorithm flowchart is shown in Figure 4.

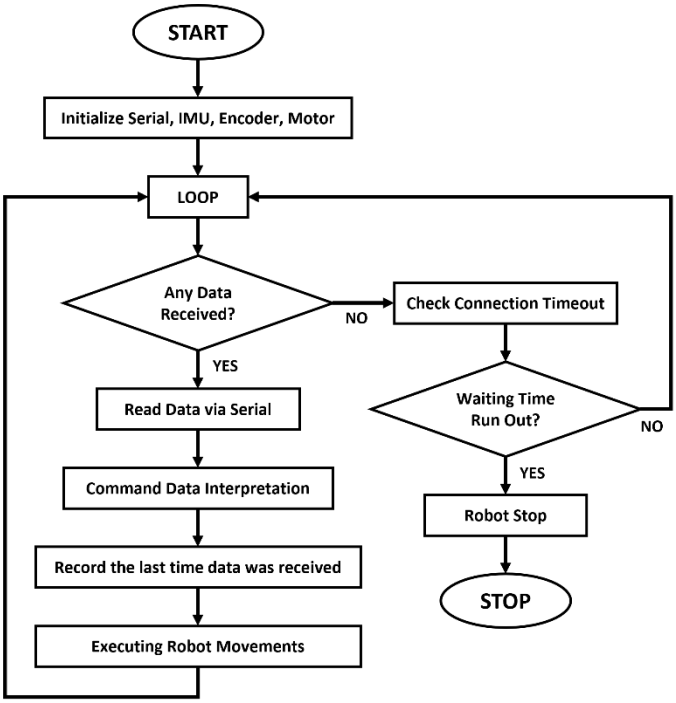

Fig 4. Teensy 4.1 Algorithm Flowchart

During the initialization phase, Teensy sets up serial communication, activates the IMU sensor for orientation readings, configures the motor control system, and initializes the encoder and PID controller. This stage ensures that all components are ready before the system is run.

During the operational phase, Teensy continuously monitors whether control data is received from the mini-PC. If data is available, it is processed and stored as control input reference, while updating the last data reception time. Based on this input, the system calculates the robot's motion commands through the stages of sensor reading, kinematic calculation, and motor speed control. In addition, the system also estimates the robot's position based on speed and time data.

As a safety mechanism, Teensy continuously monitors the continuity of the data received. If no data is received within a certain period of time, the system automatically stops all motors to prevent uncontrolled movement of the robot.

4) Time Synchronization

Time synchronization is used to ensure that all devices in the system have the same time reference, so that latency measurements can be performed accurately. The Mini-PC acts as a time server, while the ESP32 functions as a client.

During initialization, the ESP32 synchronizes the time with the mini-PC via the NTP protocol until it obtains a valid time. After the initial synchronization is complete, the ESP32 maintains time accuracy using a calibrated internal clock. All data sent by the ESP32 is accompanied by a timestamp.

With this time synchronization, the difference between the data transmission time from the ESP32 and the reception time on the mini-PC can be calculated accurately. This enables the evaluation of network communication performance, particularly in the analysis of latency and data transmission reliability.

III. TEST AND RESULT

A. Sent Interval Analysis

Initial testing focused on analyzing the effect of data transmission intervals and transmission frequency on system communication performance. These parameters are very important in robot control systems because they determine the balance between system responsiveness and network load. The testing was conducted with three different transmission interval configurations, namely 5 ms (200 Hz), 10 ms (100 Hz), and 20 ms (50 Hz).

Each test was conducted for approximately one minute under the same network conditions. During the test, the mini-PC recorded the sent timestamp from the ESP32 and the reception timestamp to calculate latency, jitter, and packet loss.

TABLE I

RESULT OF SENT INTERVAL ANALYSIS

Parameters	200 Hz	100 Hz	50 Hz
Average Latency	12.173 ms	11.787 ms	11.314 ms
Minimum Latency	2.137 ms	2.367 ms	2.217 ms
Maximum Latency	55.413 ms	56.143 ms	134.862 ms
Standard Deviation	11.909 ms	11.58 ms	12.004 ms
Average Jitter	4.271 ms	7.273 ms	11.647 ms
Packets Loss Rate	0.00%	0.00%	0.00%
Total Packets Sent	12449	6311	3241
Total Packets Received	12449	6311	3241
Test Duration	1 minute	1 minute	1 minute

Based on these three tests, it can be concluded that changes in transmission intervals do not have a significant effect on average latency, which ranges from 11 to 12 ms for all configurations. However, transmission frequency does affect jitter and maximum latency.

The 200 Hz configuration produces the lowest jitter, but has a higher transmission load. Conversely, the 50 Hz configuration shows greater maximum jitter and latency due to variations in delay between packets. The 100 Hz configuration provides the best balance between timing stability, network load, and system responsiveness.

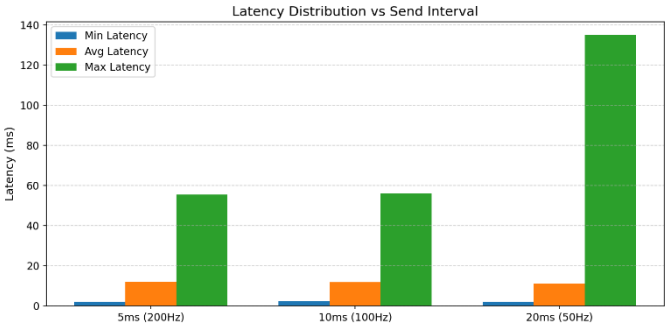

Fig 5. Comparison Chart of Latency Against Sent Interval

Based on these results, a transmission frequency of 100 Hz was selected as the primary operational configuration, as it provides low latency, controllable jitter, and high communication reliability for robot control applications.

B. WiFi Sleep Mode Analysis

This test aims to analyze the effect of the WiFi power management mode (WiFi sleep mode) on the ESP32 on system communication performance, particularly in terms of latency, jitter, and transmission reliability. In robot control systems, WiFi sleep mode settings are an important factor because they have the potential to reduce power consumption, but can affect communication stability.

The test was conducted with a fixed transmission interval of 5 ms (200 Hz), while the WiFi sleep mode configuration was varied into three conditions, namely WiFi sleep disabled (WiFi.setSleep(false)), WiFi sleep enabled (WiFi.setSleep(true)), and WiFi sleep with minimum modem mode (WIFI_PS_MIN_MODEM).

Each configuration was tested for approximately one minute under the same network conditions.

TABLE II

WiFi SLEEP MODE ANALYSIS RESULTS

Parameters	False	True	WIFI_PS_MIN_MODEM
Average Latency	12.173 ms	12.406 ms	12.424 ms
Minimum Latency	2.137 ms	2.203 ms	2.211 ms
Maximum Latency	55.413 ms	62.798 ms	86.236 ms
Standard Deviation	11.909 ms	12.089 ms	12.409 ms
Average Jitter	4.271 ms	4.288 ms	4.246 ms
Packets Loss Rate	0.00%	0.00%	0.00%
Total Packets Sent	12449	12492	12577
Total Packets Received	12449	12492	12577
Test Duration	1 minute	1 minute	1 minute

Based on the test results, it can be concluded that activating WiFi sleep mode does not have a significant effect on average latency and jitter at a transmission interval of 5 ms. All

configurations showed an average latency in the range of 12 ms and jitter of around 4 ms, with a packet loss rate of 0.000%.

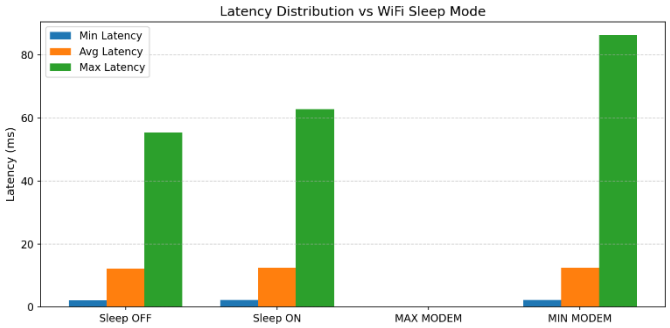

Fig 6. Comparison Chart of Latency Against WiFi Sleep Mode

However, a difference can be seen in the maximum latency, where configurations with WiFi sleep enabled—especially in minimum modem sleep mode—show a greater delay spike compared to configurations without sleep mode. This phenomenon is caused by the additional time required for the WiFi module to return to an active state before transmitting data.

For robot control applications that prioritize timing consistency and minimize extreme delays, disabling the WiFi sleep configuration is recommended. Although power consumption is higher, this configuration provides better communication stability and minimizes the risk of latency spikes that can affect system response.

C. WiFi Transmission Power Analysis

This test aims to analyze the effect of WiFi transmission power on the ESP32 on the communication performance of the system. Transmission power settings affect signal strength, communication range, and power consumption, which can indirectly impact latency and data transmission stability.

The tests were conducted with a fixed transmission interval of 5 ms (200 Hz) and WiFi sleep mode disabled. The WiFi transmission power was varied to several levels, namely 19.5 dBm, 17 dBm, 15 dBm, 13 dBm, and 7 dBm. Each configuration was tested for approximately one minute under the same network conditions.

TABLE III

WiFi TRANSMISSION POWER ANALYSIS RESULTS

Parameters	19.5 dBm	17 dBm	15 dBm	13 dBm	7 dBm
Average Latency	13.51 ms	14.118 ms	13.703 ms	13.635 ms	14.028 ms
Minimum Latency	2.065 ms	2.131 ms	2.253 ms	2.297 ms	2.205 ms
Maximum Latency	159.60 ms	165.84 ms	151.55 ms	152.36 ms	197.72 ms
Standard Deviation	16.14 ms	16.195 ms	15.382 ms	15.766 ms	16.786 ms
Average Jitter	4.398 ms	4.48 ms	4.327 ms	4.22 ms	4.37 ms
Packets Loss Rate	0.00%	0.00%	0.00%	0.00%	0.00%
Total Packets Sent	12801	12743	12482	12555	12617
Total Packets Received	12801	12743	12482	12555	12617
Test Duration	1 minute	1 minute	1 minute	1 minute	1 minute

Based on the test results, it can be concluded that the reduction in WiFi transmission power did not have a significant effect on packet loss, which remained at 0.000% across all configurations. This shows that the communication system has high reliability at the test distance used.

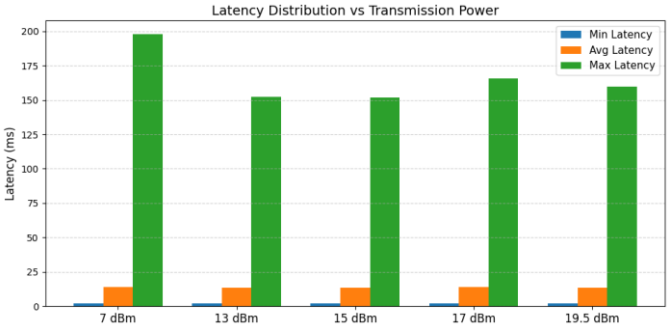

Fig 7. Comparison Chart of Latency Againts Transmission Power

However, transmit power variation affects maximum latency and latency standard deviation. Transmit power that is too high or too low tends to result in greater delay spikes, which are likely caused by WiFi channel dynamics and transmission adaptation mechanisms.

Medium transmission power configurations, particularly in the range of 13–15 dBm, show the best balance between average latency, jitter, and communication stability. Therefore, this configuration is more recommended for robot control applications because it is able to maintain stable communication performance while reducing power consumption compared to maximum transmission power.

D. Distance Testing

After optimizing the system parameters through sent interval analysis, WiFi sleep mode, and WiFi transmission power, the next step is to test the communication distance. This test aims to evaluate the performance and reliability of the wireless communication system when the distance between the ESP32 and the mini-PC is increased using the best parameter configuration obtained previously.

Distance testing was conducted by varying the distance between the ESP32 and the mini-PC at several test points, namely 5 meters, 10 meters, and 15 meters. At each distance, the system was run for a certain duration at a constant data transmission rate, in accordance with the optimal parameters from previous tests.

TABLE IV

DISTANCE TEST RESULTS

Parameters	5 Meters	10 Meters	15 Meters
Average Latency	9.72 ms	11.635 ms	12.867 ms
Minimum Latency	0.8 ms	1.029 ms	2.271 ms
Maximum Latency	136.831 ms	137.008 ms	87.418 ms
Standard Deviation	11.816 ms	12.663 ms	12.364 ms
Average Jitter	7.177 ms	7.559 ms	7.322 ms
Packets Loss Rate	0.00%	0.00%	0.00%
Total Packets Sent	21236	24073	21667
Total Packets Received	21236	24073	21667
Test Duration	3 minute	3 minute	3 minute

Test results show that the system is capable of maintaining stable communication without packet loss up to a distance of 15 meters. The average latency value tends to increase with distance, due to a decrease in signal quality and an increase in wireless channel variation. However, this latency value is still within an acceptable range for robot control applications.

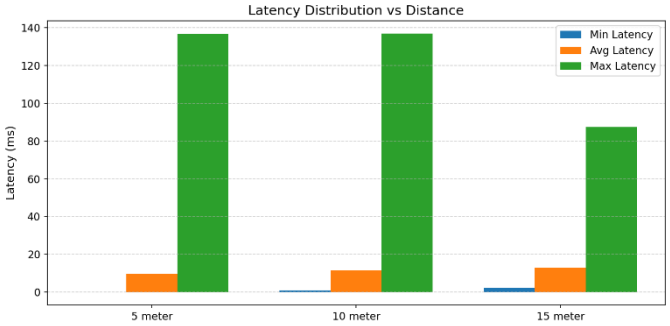

Fig 8. Graph Comparing Latency Againts Distance

Although there are variations in maximum latency values, these values do not show a monotonous increase with distance. This is because maximum latency represents momentary extreme events that are influenced by network interference and operating system scheduling processes, not solely by communication distance. Therefore, average latency and packet loss rate are used as the main indicators in assessing system performance.

Based on the test results, it can be concluded that the optimal parameters obtained in the previous stage remain effective for use in tests at distances of up to 15 meters. The communication system shows consistent performance with stable latency, controlled jitter, and no packet loss, making it suitable for use in mobile robot control applications.

IV. CONCLUSION

This study evaluates the performance of a wireless communication system based on ESP32, mini-PC, and Teensy 4.1 for robot control applications. Testing was conducted through analysis of data transmission intervals, WiFi power saving mode, and transmission power to identify the most suitable communication parameters.

TABLE V

OBTAINED OPTIMAL PARAMETER CONFIGURATION

Parameters	Configuration
Sent Interval	100 Hz
WiFi Sleep Mode	False
WiFi Transmission Power	13-15 dBm

The results of the transmission interval testing show that the 10 ms interval provides the best balance between average latency, jitter, and transmission stability compared to faster or slower intervals. In WiFi power saving mode testing, disabling the WiFi sleep mode configuration resulted in more consistent latency and jitter without an increase in packet loss. Meanwhile, transmission power testing showed that a power range of 13–15 dBm was able to maintain stable communication without packet loss and without a significant increase in latency variation.

Based on these parameter configurations, distance testing at 5 meters, 10 meters, and 15 meters showed that the system was able to maintain a relatively low average latency and zero packet loss at all test distances. The variation in maximum latency values across distances indicates natural fluctuations in the wireless network during testing.

REFERENCES

[1] D. P. Mtowe, L. Long, and D. M. Kim, “Low-Latency Edge-Enabled Digital Twin System for Multi-Robot Collision Avoidance and Remote Control,” *Sensors*, vol. 25, no. 15, Aug. 2025, doi: 10.3390/s25154666.

[2] K. Sankhe, D. Jaisinghani, and K. Chowdhury, “ReLy: Machine Learning for Ultra-Reliable, Low-Latency Messaging in Industrial Robots,” *IEEE Communications Magazine*, vol. 59, no. 4, pp. 75–81, Apr. 2021, doi: 10.1109/MCOM.001.2000598.

[3] S. Rakhimboeva, “APPLICATION OF 5G MOBILE COMMUNICATION TECHNOLOGY INTEGRATING ROBOT CONTROLLER COMMUNICATION METHOD IN COMMUNICATION ENGINEERING,” *International Multidisciplinary Research in Academic Science (IMRAS)*, vol. 8, 2025, doi: 10.5281/zenodo.15121754.

[4] A. Iqbal, T. Khurshaid, A. Nauman, and S. B. Rhee, “Energy-Aware Ultra-Reliable Low-Latency Communication for Healthcare IoT in Beyond 5G and 6G Networks,” *Sensors*, vol. 25, no. 11, Jun. 2025, doi: 10.3390/s25113474.

[5] H. P. Firtiani, A. Maulana, M. Hasbi, T. A. Septiani, and R. F. Azzahr, “Analisis Perbandingan Performa Jaringan Wireless 2.4GHz dan 5GHz dalam Penggunaan Sehari-hari,” *Jurnal Komputer, Informasi*

dan Teknologi, vol. 5, no. 2, 2025, doi: 10.53697/jkomitek.v6i1.33.

[6] G. Pise, S. D. Babar, and P. N. Mahalle, "Design and Analysis of a Multi-Protocol Gateway for Enhanced Machine-to-Machine Communication in IoT Environments," *International Journal of Scientific Research in Science, Engineering and Technology | www.ijsrset.com Published in*, vol. 10, no. 7, [Online]. Available: www.ijsrset.com

[7] Y. Liu *et al.*, "Enhancing Time Synchronization in Smart Grid With White Rabbit: Theory, Architecture, and Challenges," 2025, *Institute of Electrical and Electronics Engineers Inc.* doi: 10.1109/ACCESS.2025.3599486.

[8] Q. Li, J. Guo, W. Liu, W. Gao, Y. Zhang, and Y. Hu, "An enhanced time synchronization method for a network based on Kalman filtering," *Sci Rep*, vol. 14, no. 1, Dec. 2024, doi: 10.1038/s41598-024-71929-8.

[9] W. Zhang, I. Rodriguez, and P. Mogensen, "Implementation of Wireless Synchronous Robotic Control in 5G and Edge-Cloud Environments," in *Proceedings - IEEE Consumer Communications and Networking Conference, CCNC*, Institute of Electrical and Electronics Engineers Inc., 2025. doi: 10.1109/CCNC54725.2025.10976110.

[10] T. Hafwenia, O. Himmatana Prasetyo, M. Pandu Pamungkas, R. Bayu Saputra, P. Wisnu Agung Sucipto, and U. Negeri Malang, "IMPLEMENTATION OF A REAL-TIME COMMUNICATION SYSTEM FOR HUMANOID ROBOTS USING THE GROUPSYNC FEATURE IN DYNAMIXEL PROTOCOL 2.0."

[11] H. Sandip Telengre, P. Dike, S. T. Bandu, V. R. Jadhav, and S. Trupti Bandu, "International Journal of Sciences and Innovation Engineering A Study on Robotics Arm Vehicle using ESp32 and PS3 controller," vol. 2, no. 6, 2025, doi: 10.70849/ijsci.

[12] Mr. A. Kuppuswamy, Mr. S. L. Hariharan, Mr. R. B. Kalishwaran, Mr. S. Hanuram, and Mr. S. A. kumar, "Wireless Joystick Control Robotic Arm Using ESP32 Microcontroller," *International Journal of Research Publication and Reviews*, vol. 6, no. 3, pp. 5589–5592, Mar. 2025, doi: 10.55248/gengpi.6.0325.1268.

[13] H. Dhokane, V. Kasar, H. Patil, and S. I. Bakhtar, "HOME AUTOMATION USING ESP32 AND ACCESS POINT," 2025. [Online]. Available: www.ijprems.com

[14] L. Arifandi, E. Setyaningsih, and Y. Calvinus, "AUTOMATED DELIVERY ROBOT SYSTEM DESIGN FOR OFFICE USING ESP32 TECHNOLOGY," *International Journal of Application on Sciences, Technology and Engineering*, vol. 1, no. 3, pp. 985–996, Aug. 2023, doi: 10.24912/ijaste.v1.i3.985-996.

[15] S. Candidate prof Marcello Chiaberge Nunzio Villa, "POLITECNICO DI TORINO Motion Control Architecture for Service Robotics".