BAB III

PELAKSANAAN PROYEK

3.1 Kedudukan dan Koordinasi

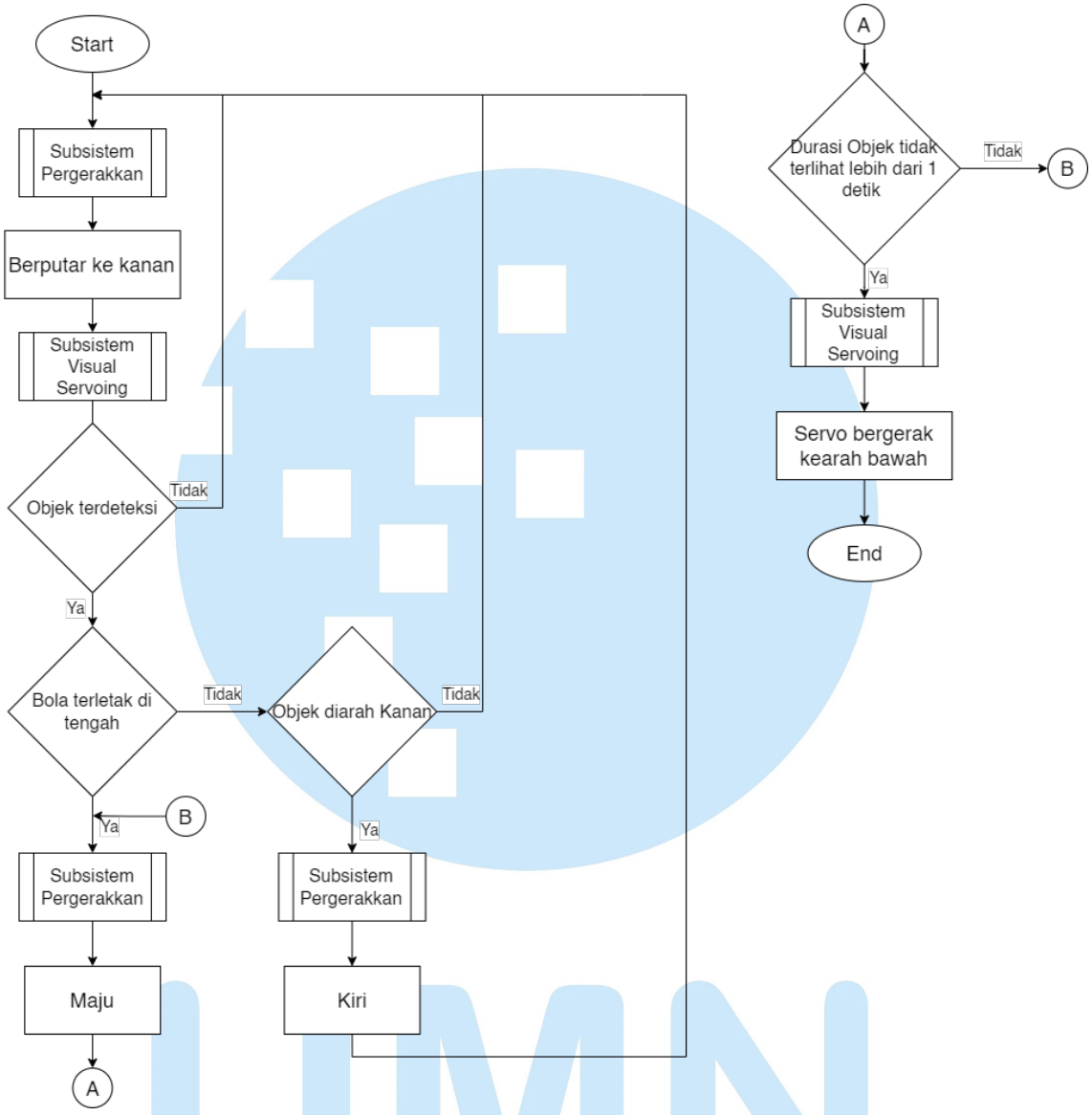
Pada proyek independen, penulis akan bekerja sama dengan divisi lainnya untuk membuat robot sepak bola bernama “Bison”. Penulis berperan sebagai divisi *Low Level Programming*. Divisi *Low Level Programming* bertanggung jawab dalam mengembangkan *framework* untuk mengatur pengambilan keputusan Bison melalui mengelola *input* dan mengontrol pergerakan. Tugas divisi meliputi implementasi antar-komponen *hardware* menggunakan *Robot Operational System* (ROS), serta mengintegrasikan subsistem yang telah dibuat oleh divisi *High Level Programming*.

Koordinasi antar divisi umumnya bersifat iteratif sehingga divisi *Low Level Programming* akan menunggu hasil pengembangan subsistem dari *High Level Programming*.

3.2 Sistem Kerja Robot “Bison”

Robot "Bison" beroperasi berdasarkan pengambilan keputusan (*decision making*) dari subsistem *Visual Servoing*. Pada awalnya, robot akan berputar ke arah kanan untuk melakukan pencarian objek (bola) menggunakan kamera. Ketika bola terdeteksi, sistem akan mengevaluasi posisinya. Jika bola berada di depan (tengah pandangan kamera), robot akan bergerak maju untuk mendekatinya.

Apabila bola terdeteksi namun tidak di tengah, robot akan menyesuaikan posisinya hingga bola berada di tengah pandangan sebelum bergerak maju. Proses gerak maju ini akan terus berlanjut. Jika robot kehilangan jejak visual bola selama lebih dari satu detik, maka diasumsikan bola berada di area bawah robot. Sebagai respons, *servo* kamera akan bergerak ke arah bawah untuk melakukan verifikasi akhir. Untuk *flowchart* sistem kerja dari robot “Bison” dapat dilihat pada Gambar 3.1.



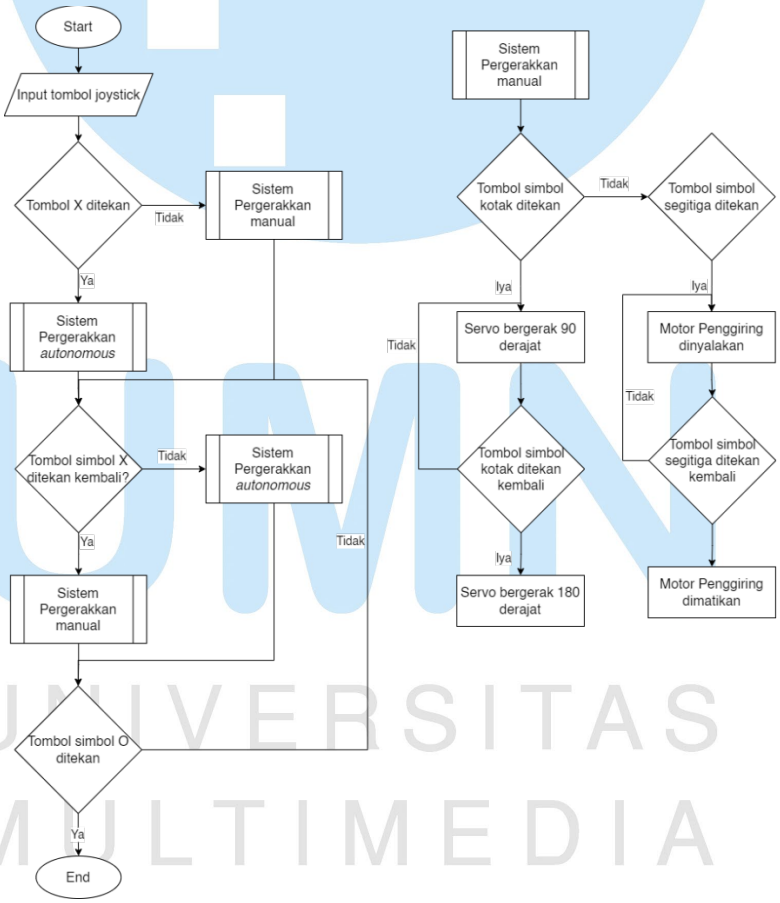
Gambar 3. 1 Flowchart Sistem Kerja Bison

3.3 Proses Pengembangan

Untuk melakukan implementasi sistem kerja robot yang telah dijelaskan pada bagian 3.2 dan *flowchart* (Gambar 1), proses pengembangan dilakukan melalui beberapa tahapan. Tahap pengembangan dimulai dengan pendekatan awal menggunakan metode *hardcoded* hingga implementasi akhir menggunakan *Robot Operating System* (ROS) sebagai *platform* dalam robot "Bison" mengambil keputusan.

3.3.1 Tahap Pengembangan Awal: Integrasi secara Hardcoded

"Pada tahap pengembangan awal, robot “Bison” memiliki dua mode operasi: manual dan *autonomous*. Mode manual memungkinkan pengendalian langsung oleh operator, di mana robot dapat bergerak maju, mundur, menyamping, dan berotasi. Pada mode ini, operator juga dapat mengaktifkan subsistem pengiring bola dan menggerakkan servo secara manual melalui *joystick*. Di sisi lain, mode *autonomous* dirancang agar robot dapat bergerak secara mandiri untuk mencari dan mengejar bola, yang mana alur logikanya diimplementasikan sesuai dengan flowchart pada Gambar 1. Untuk sistem kerja dari integrasi secara *hardcoded* dapat dilihat melalui *flowchart* pada Gambar 3.2.



Gambar 3. 2 Integrasi Subsistem secara *Hardcoded*

Meskipun integrasi dengan metode *hardcoded* fungsional, metode ini memiliki beberapa keterbatasan, seperti tingkat fleksibilitas yang rendah dan kesulitan dalam melakukan modifikasi atau penambahan fitur baru/perubahan subsistem. Keterbatasan inilah yang mendasari keputusan penulis untuk beralih ke *platform* ROS pada tahap pengembangan selanjutnya.

3.3.2 Tahap Pengembangan Akhir: Implementasi Robot Operating System (ROS) 2

Robot Operating System (ROS) merupakan sebuah *platform* atau kerangka kerja perangkat lunak bersifat *open source* yang dirancang untuk membangun sistem robot yang kompleks secara modular[5][6][7]. Memulai perkembangannya pada tahun 2009, ROS awalnya dirancang untuk menyederhanakan perkembangan robotika.

Namun, terdapat beberapa keterbatasan signifikan yang dimiliki oleh ROS1. Keterbatasan utamanya adalah penggunaan [8]. Selain itu, ROS1 juga memiliki kekurangan dalam dukungan untuk sistem *multi-robot support* dan belum dapat memenuhi kebutuhan pemrograman *real-time*[8][9]. Oleh karena itu, ROS2 hadir dengan arsitektur yang mendukung kinerja *real-time* dan kemampuan multi-robot yang lebih baik, sehingga memungkinkan komunikasi efektif antara robot penyerang dan kiper pada KSBRI-B.

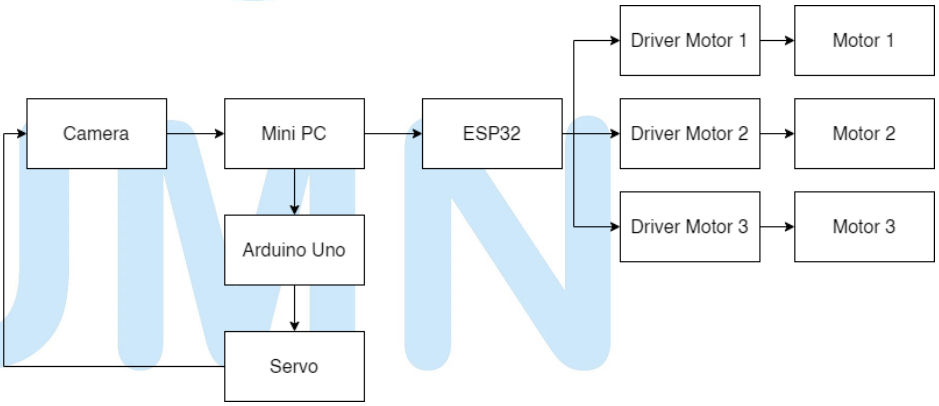
3.3.2.1.Persiapan Implementasi ROS2 pada Mini PC

Implementasi ROS 2 pada robot “Bison” dilakukan menggunakan sebuah mini PC dengan Sistem Operasi (SO) Ubuntu 24.04 dan ROS 2 distro Jazzy Jalisco. Pemilihan versi Ubuntu 24.04 dilakukan agar kompatibel dengan ROS 2 Jazzy. Distro Jazzy Jalisco dipilih karena memiliki tanggal *End of Life* (EOL) yang lebih panjang, yaitu hingga Mei 2029, dibandingkan dengan versi sebelumnya seperti

Humble (EOL Mei 2027)[10]. Pertimbangan ini juga dilakukan berdasarkan rencana persiapan robot untuk Kontes Robot Indonesia (KRI) di masa mendatang, sehingga dapat mempermudah angkatan berikutnya dalam melakukan pengembangan.

3.3.2.2.Desain Arsitektur Sistem

Untuk implementasikan sistem kerja robot "Bison" secara modular, fleksibel dan *real-time*, sistem dirancang menggunakan ROS2. ROS2 memiliki kemampuan dalam memecahkan fungsionalitas dari robot menjadi *nodes* independen yang dapat saling berkomunikasi melalui *topics*. *Nodes* yang dirancang akan mewakilkan subsistem yang terdapat pada robot "Bison". Visualisasi sistem secara keseluruhan dapat dilihat pada diagram blok sistem di Gambar 3.



Gambar 3. 3 Diagram Blok Sistem

Dengan *planning* ke depan akan ditambahkan subsistem dari pengiring, maka ROS2 menjadi *platform* yang cukup tepat untuk rencana pengembangan. Selain itu berdasarkan penelitian Wahyudi dan tim, ROS2 terbukti efektif dapat meningkatkan hasil performa YOLO[11].

Untuk visualisasi komunikasi dari subsistem yang digunakan saat ini dapat melihat Gambar 4.

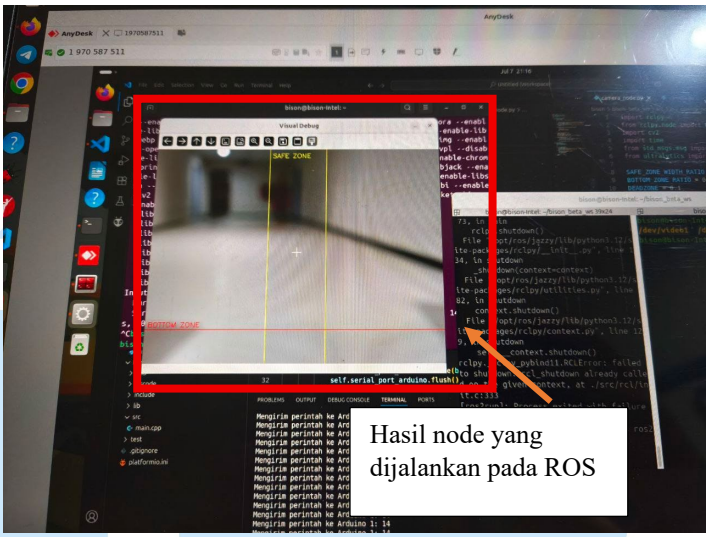
Gambar 3. 4 *rqt graph* sistem dari ROS2

Ketiga *node* yang ada akan merepresentasikan subsistem dan otak aktuator dari subsistem tersebut. Untuk /camera_node merepresentasikan subsistem *Visual Servoing*. /camera_node akan menjadi *master node* dari robot "Bison". Hal ini karena pergerakan akan dilakukan berdasarkan input yang dihasilkan oleh /camera_node.

3.3.2.3.Pengujian ROS2 pada Mini PC

Setelah arsitektur sistem dirancang, langkah selanjutnya adalah memverifikasi bahwa platform pengembangan (Mini PC) siap untuk implementasi. Oleh karena itu, sebuah pengujian fungsionalitas dasar ROS 2 dilakukan. Pengujian ini dilaksanakan dengan menjalankan *node* kamera untuk memastikan sistem mampu menangani

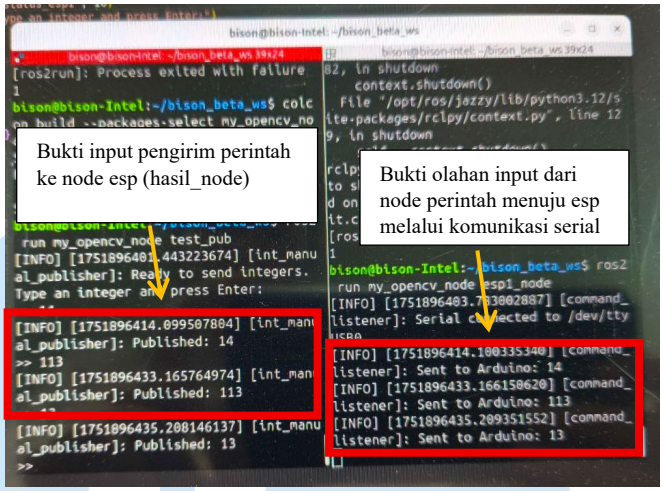
akuisisi dan visualisasi kamera secara *real-time*.



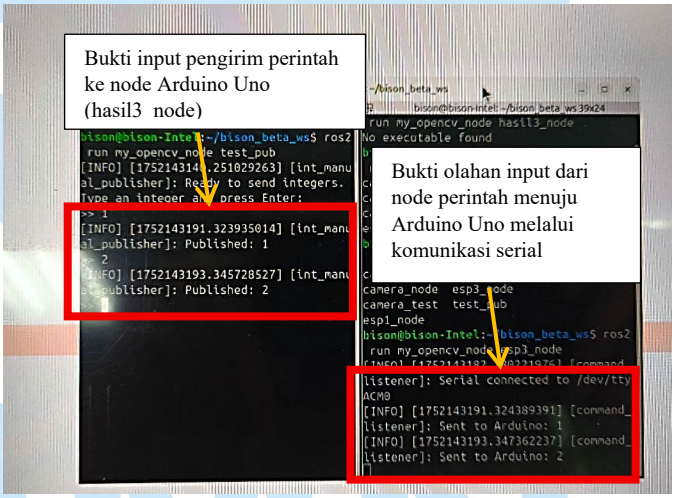
Gambar 3. 5 Hasil Pengujian Program Node Kamera

3.3.2.4.Pengujian Komunikasi ESP32 dan Arduino Uno dengan Mini PC

Pengujian ini bertujuan untuk memverifikasi komunikasi antar *node* pada Mini PC dengan mikrokontroler ESP32 dan Arduino Uno. Proses pengujian melibatkan sebuah *node* simulator (test_pub) yang mempublikasikan data perintah melalui *node* jembatan serial, yaitu hasil_node dan hasil3_node. Terdapat perbedaan konfigurasi *baud rate* untuk kedua mikrokontroler: ESP32 menggunakan 115200, sedangkan Arduino Uno menggunakan 9600. Perbedaan ini didasarkan pada kapabilitas perangkat keras dan stabilitas komunikasi yang dibutuhkan.



Gambar 3. 6 Hasil Pengujian Komunikasi ESP32 pada hasil_node



Gambar 3. 7 Hasil Pengujian Komunikasi Arduino pada hasil3_node

3.3.3 Analisis Komparatif Berbasis Skenario Uji

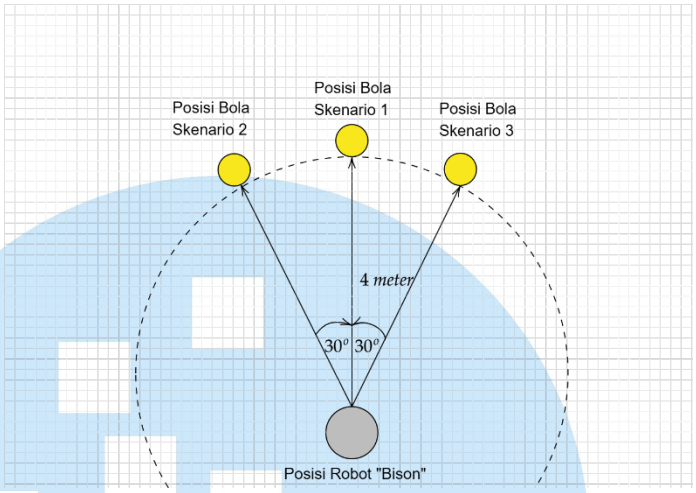
3.3.3.1.Skenario Pengambilan Data

Pengujian ini bertujuan untuk membandingkan performa kuantitatif antara implementasi sistem secara Hardcoded dan ROS2 pada robot "Bison". Fokus utamanya adalah membuktikan hipotesis bahwa arsitektur ROS2 menawarkan kinerja yang lebih efisien. Pengujian akan dilakukan dengan mengukur waktu pergerakan total yang

dibutuhkan robot untuk mengejar dan mencapai bola yang diletakkan pada jarak 4 meter di depannya.

Pada setiap percobaan, beberapa metrik waktu akan diukur secara spesifik, yaitu: waktu yang dihabiskan YOLO dalam melakukan deteksi objek dalam subsistem *Visual Servoing*, waktu transmisi data ke mikrokontroler ESP32, dan waktu pengiriman perintah ke Arduino Uno. Data yang terkumpul dari metrik-metrik ini akan dianalisis secara statistik untuk memperoleh nilai rata-rata sebagai indikator utama performa.

Eksperimen akan dijalankan dalam tiga skenario posisi yang berbeda untuk menguji kapabilitas robot secara komprehensif. Skenario pertama menempatkan bola tepat di depan robot (0°), skenario kedua pada posisi 30° ke arah kiri, dan skenario ketiga pada posisi 30° ke arah kanan. Hasil analisis komparatif dari ketiga skenario ini akan menjadi landasan objektif untuk menyimpulkan metode implementasi mana yang menunjukkan kinerja paling unggul untuk tugas robot pengejar bola. Untuk visualisasi disajikan pada Gambar 3.8. Pada gambar tersebut, lingkaran berwarna kuning menggambarkan posisi bola, sedangkan lingkaran berwarna abu-abu menggambarkan posisi robot.



Gambar 3. 8 Visualisasi Skenario Pengujian Pergerakkan Robot terhadap Bola

3.3.3.2.Prosedur Pengambilan Data Skenario

Prosedur pengambilan data dilaksanakan secara sistematis untuk memastikan objektivitas dan konsistensi. Untuk setiap skenario yang telah didefinisikan, sistem robot dioperasikan secara bergantian menggunakan implementasi *Hardcoded* dan ROS2. Setiap kombinasi skenario dan metode implementasi dieksekusi sebanyak tiga puluh kali untuk memperoleh data yang representatif. Selama setiap eksekusi berlangsung, sistem dirancang untuk mencatat (*me-logging*) secara otomatis dua metrik waktu krusial:

1) Waktu yang dibutuhkan oleh subsistem Visual Servoing untuk mendeteksi posisi bola.
2) Waktu latensi yang terjadi dalam pengiriman data perintah dari sistem pemrosesan utama ke mikrokontroler.

Proses akan diulang hingga seluruh data dari total 180 percobaan (3 skenario × 2 metode × 30 repetisi) berhasil dikumpulkan untuk dianalisis lebih lanjut.

Analisis perbandingan kinerja metode dilakukan berdasarkan rata-rata waktu tempuh robot. Waktu tempuh total diukur sejak robot mulai bergerak hingga berhasil menangkap bola. Namun, untuk mendapatkan data kinerja gerak yang murni, waktu tempuh total tersebut akan dikoreksi dengan mengurangi total durasi yang dibutuhkan oleh node subsistem *Visual Servoing* untuk berjalan. Pendekatan ini menghilangkan variabel waktu pemrosesan visual, sehingga perbandingan antar metode menjadi lebih objektif dan sebanding.

3.3.3.3. Hasil Pengujian dan Analisis Skenario 1

Pada Skenario 1, di mana target terletak pada posisi paling ideal (0°). Hasil pengujian menunjukkan bahwa implementasi ROS2 sebagai integrasi lebih unggul dibandingkan metode secara *hardcoded*. Hal ini terbukti dengan data pada Tabel 3.1 dan Tabel 3.2.

Tabel 3. 1 Hasil Pengukuran Waktu Skenario 1 Pergerakan Menuju Bola dengan Metode ROS2

No	Waktu keseluruhan (s)	Waktu Node (s)	Waktu *Subsistem Visual Servoing* (s)
1	13,59	3.46	10.13
2	15,65	3.27	12.38
3	15,52	3.12	12.4
4	16,3	3.25	13.05
5	16,4	3.5	12.9
6	15,22	3.14	12.08
7	15,69	3.26	12.43
8	15,41	3.19	12.22
9	15,3	3.14	12.16
10	15,14	3.14	12
11	15,83	3.49	12.34
12	15,93	3.31	12.62
13	15,83	3.22	12.61
14	15,64	3.26	12.38
15	15,33	3.49	11.84
16	15,43	3.56	11.87
17	15,65	3.16	12.49
18	16,15	3.71	12.44
19	15,17	3.36	11.81
20	16,2	3.16	13.04
21	15,2	3.23	11.97
22	15,53	3.34	12.19
23	15,65	3.2	12.45
24	16,15	3.71	12.44
25	15,83	3.42	12.41
26	15,93	3.31	12.62
27	15,83	3.22	12.61
28	15,64	3.26	12.38
29	15,33	3.49	11.84
30	15,44	3.32	12.12
Rata-rata			12,274

Tabel 3. 2 Hasil Pengukuran Waktu Skenario 1 Pergerakan Menuju Bola dengan Metode *Hardcoded*

No	Waktu keseluruhan (s)
1	14
2	14
3	14.18
4	12.61
5	12.95
6	13.35
7	13.73
8	13.64
9	12.33
10	14.06
11	13.52
12	14.1
13	13.65
14	12.99
15	13.4
16	13.27
17	12.66
18	13.47
19	12.5
20	13.88
21	12.97
22	12.81
23	12.53
24	13.36
25	13.42
26	12.36
27	13.07
28	14
29	13.49
30	12.77
Rata-rata	13,302

Berdasarkan Tabel 3.1 dan Tabel 3.2 terlihat bahwa rata-rata waktu pergerakan robot “Bison” dengan metode ROS2 lebih cepat 1,028 detik dibandingkan dengan metode

hardcoded. Keunggulan tersebut didukung melalui waktu deteksi bola yang dibutuhkan pada integrasi ROS2 pada seluruh log data pengujian adalah 45,77 ms dengan waktu pengiriman rata-rata pada ESP32 dan Arduino masing-masing sebesar 0,1 ms. Waktu tersebut juga lebih cepat bila dibandingkan dengan hasil integrasi secara *hardcoded*. Hal ini terlihat pada rata-rata waktu deteksi bola yang dibutuhkan adalah 55,52 ms dengan masing-masing waktu pengiriman pada ESP32 dan Arduino adalah 0,25 ms dan 0,46 ms. Waktu pengiriman yang lebih cepat yang dimiliki ROS2 secara tidak langsung dapat meningkatkan keakuratan sehingga perintah dapat tersampaikan secara real-time dengan *delay* yang nyaris tidak ada.

3.3.3.4. Hasil Pengujian dan Analisis Skenario 2

Pada Skenario 2, di mana target terletak pada posisi 30° ke arah kiri robot "Bison". Hasil pengujian menunjukkan bahwa implementasi ROS2 sebagai integrasi sedikit lebih unggul dibandingkan metode secara *hardcoded*. Hal ini terbukti dengan data pada Tabel 3.3 dan Tabel 3.4.

Tabel 3. 3 Hasil Pengukuran Waktu Skenario 2 Pergerakkan Menuju Bola dengan Metode ROS2

No	Waktu keseluruhan (s)	Waktu (s)	Waktu Subsistem *Visual Servoing* (s)
1	24,87	3,16	21,71
2	24,25	3,04	21,21
3	23,41	3,29	20,12
4	25,5	3,17	22,33
5	27,57	3,31	24,26
6	27,92	3,23	24,69
7	27,24	2,98	24,26
8	26,48	3,18	23,3
9	25,9	3,19	22,71
10	26,42	3,5	22,92
11	26,85	3,38	23,47
12	27,97	3,33	24,64
13	26,11	3,29	22,82
14	26,38	3,12	23,26
15	28,25	3,58	24,67
16	25,56	3,27	22,29
17	25,43	3,51	21,92
18	26,8	3,31	23,49
19	26,51	3,2	23,31
20	25,92	3,23	22,69
21	25,8	3,5	22,3
22	27,97	3,33	24,64
23	26,13	3,29	22,84
24	27,97	3,33	24,64
25	26,11	3,29	22,82
26	27,71	3,2	24,51
27	26,11	3,29	22,82
28	24,25	3,04	21,21
29	23,33	3,29	20,04
30	25,9	3,19	22,71
Rata-rata			22,95

Tabel 3. 4 Hasil Pengukuran Waktu Skenario 2 Pergerakkan Menuju Bola dengan Metode *Harcoded*

No	Waktu keseluruhan (s)
1	25,49
2	16,27
3	25,12
4	23
5	22,07
6	24,28
7	23,17
8	24,3
9	23,31
10	21,8
11	23,47
12	23,24
13	24,18
14	25,02
15	22,32
16	23,32
17	22,56
18	23,2
19	22,42
20	21,1
21	22,77
22	21,45
23	23,61
24	23,4
25	24,06
26	23,3
27	21,97
28	22,46
29	23,62
30	22,78
Rata-rata	22,969

Berdasarkan Tabel 3.3 dan Tabel 3.4 terlihat bahwa rata-rata waktu pergerakkan robot “Bison” dengan metode ROS2 lebih cepat 0,695 detik dibandingkan dengan metode

hardcoded. Sama seperti skenario sebelumnya, keunggulan tersebut didukung melalui waktu deteksi bola yang dibutuhkan pada integrasi ROS2 pada seluruh log data pengujian adalah 46,84 ms dengan waktu pengiriman rata-rata pada ESP32 dan Arduino masing-masing sebesar 0,1 ms. Waktu tersebut juga lebih cepat bila dibandingkan dengan hasil integrasi secara *hardcoded* yang mana rata-rata waktu deteksi bola yang dibutuhkan adalah 54,57 ms dengan masing-masing waktu pengiriman pada ESP32 dan Arduino adalah 0,19 ms dan 0,46 ms.

3.3.3.5. Hasil Pengujian dan Analisis Skenario 3

Pada Skenario 3, di mana target terletak pada posisi 30° ke arah kanan robot "Bison". Hasil pengujian menunjukkan bahwa implementasi ROS2 sebagai integrasi sedikit lebih unggul dibandingkan metode secara *hardcoded*. Hal ini terbukti dengan data pada Tabel 3.5 dan Tabel 3.6.

Tabel 3. 5 Hasil Pengukuran Waktu Skenario 3
Pergerakkan Menuju Bola dengan Metode ROS2

No	Waktu keseluruhan (s)	Waktu (s)	Waktu Subsistem Visual Servoing (s)
1	17	3,2	13,8
2	16,58	3,43	13,15
3	18,48	3,58	14,9
4	16,43	3,24	13,19
5	15,78	2,99	12,79
6	17,79	3,2	14,59
7	17,1	3,37	13,73
8	18,57	3,14	15,43
9	18,61	3,23	15,38
10	19,04	3,23	15,81
11	18,05	3,31	14,74
12	18,03	3,13	14,9
13	18,01	3,31	14,7
14	18,39	3,18	15,21
15	18,41	3,17	15,24
16	17,78	3,25	14,53
17	18,34	3,54	14,8
18	17,92	3,53	14,39
19	17,16	3,27	13,89
20	17,13	3,32	13,81
21	18,43	3,24	15,19
22	18,21	3,23	14,98
23	18,46	3,38	15,08
24	18,21	3,17	15,04
25	17,78	3,25	14,53
26	18,34	3,54	14,8
27	17,72	3,27	14,45
28	17,16	3,33	13,83
29	17,13	3,32	13,81
30	17	3,2	13,8
Rata-rata			14,589

Tabel 3. 6 Hasil Pengukuran Waktu Skenario 3 Pergerakan Menuju Bola dengan Metode *Hardcoded*

No	Waktu keseluruhan (s)
1	15,09
2	14,45
3	14,27
4	13,75
5	13,99
6	13,73
7	16,76
8	14,28
9	13,29
10	14,61
11	13,92
12	14,84
13	14,97
14	13,49
15	13,65
16	14,23
17	16,42
18	15,17
19	13,73
20	14,73
21	15,15
22	14,23
23	15,48
24	16,66
25	14,23
26	13,28
27	14,18
28	14,2
29	15,3
30	14,18
Rata-rata	14,542

Berdasarkan Tabel 3.5 dan Tabel 3.6 terlihat bahwa rata-rata waktu pergerakan robot "Bison" dengan metode *hardcoded* lebih cepat 0,047 detik dibandingkan dengan metode ROS2.

Hasil ini berbeda dengan skenario sebelumnya yang mana metode integrasi *hardcoded* lebih unggul dibandingkan ROS2. Walau begitu, ROS2 tetap unggul dalam waktu deteksi bola yang dibutuhkan. Hal ini terlihat pada integrasi ROS2 pada rata-rata nilai seluruh log data pengujian adalah 47,11 ms dengan waktu pengiriman rata-rata pada ESP32 dan Arduino masing-masing sebesar 0,1 ms. Waktu tersebut juga lebih cepat bila dibandingkan dengan hasil integrasi secara *hardcoded*. Hal ini dibuktikan pada rata-rata waktu deteksi bola yang dibutuhkan adalah 51,88 ms dengan masing-masing waktu pengiriman pada ESP32 dan Arduino adalah 0,29 ms dan 0,46 ms.