BAB IV
PENGUJIAN DAN ANALISA

4.1 Pengujian Kinerja Algoritma *Q-Learning*

Pengujian kinerja algoritma *Q-Learning* dilakukan untuk mengevaluasi kemampuan algoritma dalam mempelajari kebijakan (*policy*) navigasi yang optimal pada lingkungan simulasi sebelum diimplementasikan pada robot AGV. Pengujian dilakukan dengan menganalisis pengaruh parameter algoritma *Q-Learning*, yaitu *learning rate* (α), *discount factor* (γ), dan *epsilon decay*, serta pengaruh jumlah episode pelatihan terhadap proses pembelajaran. Evaluasi dilakukan berdasarkan beberapa parameter, meliputi *Reward Training*, *Step Training*, *Testing Success Rate*, *Step Testing*, episode awal terbentuknya *policy* optimal, serta hasil visualisasi *policy map* dan struktur *Q-table*. Hasil pengujian ini digunakan untuk menentukan kombinasi parameter algoritma dan jumlah episode pelatihan yang memberikan performa terbaik sebagai dasar implementasi sistem navigasi pada robot AGV.

4.1.1 Pengaruh Parameter Algoritma Q-*Learning*

Pengujian ini bertujuan untuk mengetahui pengaruh parameter algoritma *Q-Learning* terhadap proses pembelajaran dalam menentukan jalur navigasi yang optimal. Parameter yang diuji meliputi *learning rate* (α), *discount factor* (γ), dan *epsilon decay*. Pada setiap pengujian hanya satu parameter yang diubah, sedangkan parameter lainnya dipertahankan tetap sehingga pengaruh masing-masing parameter terhadap performa algoritma dapat diamati secara objektif. Evaluasi dilakukan menggunakan empat parameter, yaitu *Reward Training*, *Step Training*, *Testing Success Rate* (*Testing SR*), dan *Step Testing*. *Reward Training* merupakan rata-rata *reward* yang diperoleh selama proses pelatihan, sedangkan *Step Training* merupakan rata-rata jumlah langkah selama pelatihan. Sementara itu, *Testing SR* menunjukkan persentase keberhasilan agen mencapai tujuan pada tahap pengujian menggunakan *greedy policy* ($\varepsilon = 0$), sedangkan *Step Testing* menunjukkan rata-rata jumlah langkah yang diperlukan agen untuk mencapai tujuan pada tahap pengujian. Pada penelitian ini, nilai *Reward Training* dan *Step Training* dihitung berdasarkan rata-rata 200 episode terakhir untuk memperoleh hasil evaluasi yang lebih stabil.

Tabel 4.1. Parameter tetap

Pengujian	*Learning* Rate (α)	Discount Factor (γ)	Epsilon Decay	Episode
Learning Rate	Variabel	0,90	0,9995	500
Discount Factor	0,10	Variabel	0,9995	500
Epsilon Decay	0,10	0,90	Variabel	500

a) Pengaruh *Learning* Rate (α)

Tabel 4.2. Pengaruh *Learning* Rate (α) terhadap Kinerja Algoritma Q-*Learning*

α	*Reward Training*	*Step Training*	*Testing* SR	*Step Testing*
0,01	-11.962	151,18	100%	64,2
0,10	-10.582	138,17	100%	19,2
0,30	-10.509	139,36	100%	19,2
0,50	-10.470	140,93	100%	19,3

Berdasarkan Tabel 4.2, seluruh variasi *learning rate* menghasilkan *Testing Success Rate* sebesar 100%, sehingga seluruh agen mampu mencapai tujuan navigasi pada tahap pengujian. Meskipun demikian, setiap variasi menunjukkan karakteristik pembelajaran yang berbeda. Nilai α = 0,01 menghasilkan *Training Reward* sebesar −11,962, *Training Steps* sebesar 151,18 langkah, dan *Testing Steps* sebesar 64,2 langkah, yang menunjukkan proses pembelajaran berlangsung lebih lambat karena pembaruan *Q-value* dilakukan secara bertahap. Sementara itu, nilai α = 0,10, 0,30, dan 0,50 menghasilkan rata-rata *Testing Steps* sekitar 19 langkah dengan perbedaan *Training Reward* dan *Training Steps* yang relatif kecil. Menurut [41], evaluasi *Q-Learning* mempertimbangkan *Training Reward*, *Training Steps*, kecepatan konvergensi, dan *Testing Success Rate*. Berdasarkan indikator tersebut, nilai α = 0,10 memberikan *Training Steps* terendah (138,17 langkah) dengan *Testing Success Rate* sebesar 100% dan rata-rata *Testing Steps* sebesar 19,2 langkah, sehingga dipilih sebagai *learning rate* yang digunakan pada penelitian ini karena mampu memberikan proses pembelajaran yang efisien tanpa mengurangi performa navigasi.

b) Pengaruh Discount Factor (γ)

Tabel 4.3. Pengaruh Discount Factor (γ) terhadap Kinerja Algoritma Q-*Learning*

γ	*Reward Training*	*Step Training*	*Testing* SR	*Step Testing*

0,50	-10.579	139,94	100%	19,3
0,70	-10.635	140,12	100%	19,2
0,90	-10.582	138,17	100%	19,2
0,99	-10.914	141,29	100%	20,1

Berdasarkan Tabel 4.3, seluruh variasi *discount factor* menghasilkan *Testing Success Rate* sebesar 100%, sehingga perubahan nilai *discount factor* tidak memengaruhi tingkat keberhasilan agen dalam mencapai tujuan navigasi. Perbedaan terlihat pada karakteristik pembelajaran, di mana nilai $\gamma = 0,99$ menghasilkan *Training Reward* terendah (−10,914), *Training Steps* tertinggi (141,29 langkah), dan rata-rata *Testing Steps* sebesar 20,1 langkah, sedangkan nilai $\gamma = 0,50$, 0,70, dan 0,90 menghasilkan rata-rata *Testing Steps* sekitar 19 langkah. Menurut [41], *discount factor* menentukan keseimbangan antara *immediate reward* dan *future reward*, sehingga nilai yang terlalu tinggi menyebabkan agen lebih berorientasi pada *reward* jangka panjang dan proses pembelajaran menjadi kurang efisien. Berdasarkan hasil pengujian, nilai $\gamma = 0,90$ memberikan *Training Steps* terendah (138,17 langkah) dengan *Testing Success Rate* sebesar 100% dan rata-rata *Testing Steps* sebesar 19,2 langkah, sehingga dipilih sebagai nilai *discount factor* yang digunakan pada penelitian ini.

c) Pengaruh Epsilon Decay

Tabel 4.4. Pengaruh Epsilon Decay terhadap Kinerja Algoritma Q-*Learning*

Epsilon Decay	*Reward Training*	*Step Training*	*Testing* SR	*Step Testing*
0,9900	895	20,84	100%	19,3
0,9970	61	32,20	100%	19,3
0,9995	-10.582	138,17	100%	19,2
0,9999	-21.931	218,52	100%	19,2

Berdasarkan Tabel 4.4, variasi nilai *epsilon decay* memberikan pengaruh yang signifikan terhadap proses pembelajaran agen selama tahap *training*, meskipun seluruh variasi menghasilkan *Testing Success Rate* sebesar 100%. Nilai 0,9900 dan 0,9970 menghasilkan *Training Reward* yang lebih tinggi serta *Training Steps* yang lebih rendah, menunjukkan bahwa agen lebih cepat beralih dari proses *exploration* menuju *exploitation*. Sebaliknya, nilai 0,9999 menghasilkan *Training*

Reward terendah (−21,931) dan *Training Steps* tertinggi (218,52 langkah), yang menunjukkan bahwa agen mempertahankan proses eksplorasi lebih lama sehingga pembelajaran menjadi kurang efisien. Meskipun perbedaan rata-rata *Testing Steps* antara nilai 0,9900 (19,3 langkah) dan 0,9995 (19,2 langkah) hanya sebesar 0,1 langkah, analisis terhadap hasil pelatihan menunjukkan bahwa kebijakan yang dihasilkan tidak identik. Perbandingan *Q-table* menunjukkan sebanyak 1.242 dari 1.680 nilai (*73,93%*) memiliki nilai *Q-value* yang berbeda, dengan 103 dari 560 state-action pairs (*18,39%*) menghasilkan aksi terbaik yang berbeda serta selisih *Q-value* maksimum mencapai 458,85. Perbedaan tersebut juga terlihat pada hasil pengujian, di mana lintasan menuju *Goal* A menggunakan *epsilon decay* 0,9995 membutuhkan 9 langkah, sedangkan nilai 0,9900 memerlukan 10 langkah, sementara lintasan menuju *Goal* B dan *Goal* C sama-sama membutuhkan 9 langkah. Menurut [41], pemilihan *epsilon decay* harus memberikan keseimbangan antara proses *exploration* dan *exploitation*. Nilai *epsilon decay* 0,9900 menyebabkan nilai epsilon mencapai batas minimum lebih cepat sehingga kesempatan agen untuk mengeksplorasi seluruh ruang keadaan menjadi lebih terbatas, sedangkan nilai 0,9995 mempertahankan proses eksplorasi lebih lama sehingga menghasilkan kebijakan yang lebih konsisten pada seluruh skenario navigasi. Oleh karena itu, nilai 0,9995 dipilih sebagai parameter yang digunakan pada penelitian ini karena memberikan keseimbangan terbaik antara efisiensi proses pembelajaran, kualitas kebijakan yang dihasilkan, dan performa pengujian.

4.1.2 Pengaruh Jumlah Episode *Training*

Pengujian ini dilakukan untuk mengetahui pengaruh jumlah episode pelatihan terhadap proses pembelajaran algoritma Q-Learning. Pada pengujian ini digunakan parameter algoritma yang telah ditentukan pada Subbab 4.1.1, yaitu learning rate sebesar 0,10, discount factor sebesar 0,90, dan epsilon decay sebesar 0,9995. Jumlah episode pelatihan divariasikan menjadi 1000, 1250, 1500, 1750, dan 2000 episode. Evaluasi dilakukan berdasarkan nilai Training Reward, Training Steps, Training Time, dan Testing Steps. Hasil pengujian ditunjukkan pada Tabel 4.5.

Tabel 4.5. Hasil Pengujian Episode

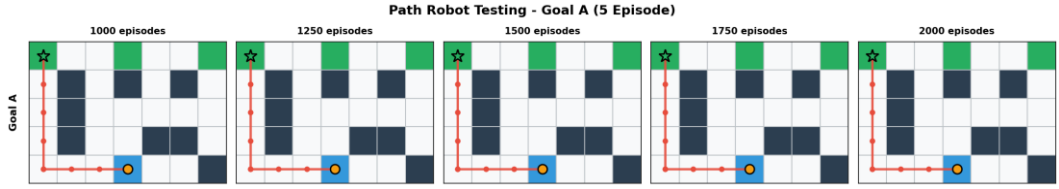
Episode	Training Reward	Training Steps	Training Time (s)	Testing Steps
1000	-3948,84	68,68	2,33	19
1250	-2854,36	59,54	2,65	19
1500	-2008,64	46,70	2,76	19
1750	-1303,92	40,90	2,94	19
2000	-869,80	36,00	3,22	19

Berdasarkan Tabel 4.5, peningkatan jumlah *episode* pelatihan memberikan pengaruh terhadap proses pembelajaran agen selama tahap *training*. Hal ini ditunjukkan oleh nilai *training reward* yang semakin meningkat (semakin mendekati nol), jumlah *training steps* yang semakin sedikit, serta waktu pelatihan yang bertambah seiring dengan meningkatnya jumlah *episode*. Kondisi tersebut menunjukkan bahwa agen semakin mampu menemukan jalur optimal selama proses pelatihan.

Meskipun demikian, peningkatan jumlah *episode* dari 1000 menjadi 2000 tidak memberikan peningkatan performa pada tahap *testing*. Seluruh model menghasilkan *Success Rate* sebesar 100% dengan rata-rata 19 langkah. Menurut penelitian [39], evaluasi algoritma *Q-Learning* dapat dilakukan berdasarkan *training reward*, *training steps*, kecepatan konvergensi, *testing success rate*, dan stabilitas hasil pengujian. Berdasarkan indikator tersebut, agen pada penelitian ini telah mencapai kondisi konvergen pada 1000 episode, sehingga penambahan jumlah *episode* tidak lagi meningkatkan kemampuan navigasi robot.

Selain performa pengujian, waktu pelatihan juga menjadi pertimbangan dalam menentukan jumlah *episode*. Pelatihan selama 1000 episode memerlukan waktu 2,33 detik, sedangkan pada 2000 episode meningkat menjadi 3,22 detik atau sekitar 38% lebih lama. Meskipun waktu pelatihan bertambah, hasil pengujian tetap menunjukkan *Success Rate* sebesar 100% dengan jumlah langkah yang sama. Oleh karena itu, penelitian ini menerapkan prinsip *early stopping*, yaitu menghentikan proses pelatihan ketika performa pengujian telah mencapai kondisi optimal dan stabil. Dengan demikian, 1000 episode dipilih sebagai parameter pelatihan karena

mampu memberikan keseimbangan antara kualitas pembelajaran, efisiensi waktu pelatihan, dan penggunaan sumber daya komputasi.

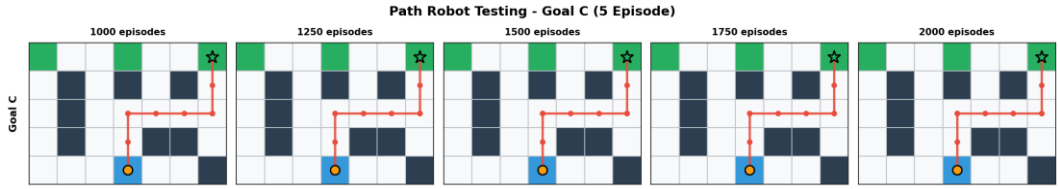


Gambar 4.1. Hasil *Testing* 5 Episode *Goal* A

Berdasarkan Gambar 4.1, seluruh pengujian menuju *Goal* A berhasil mencapai tujuan pada lima episode pengujian. Lintasan yang dihasilkan relatif sama pada setiap episode, menunjukkan bahwa agen mampu menerapkan kebijakan hasil pelatihan secara konsisten tanpa mengalami kegagalan mencapai tujuan.

Gambar 4.2. Hasil *Testing* 5 Episode *Goal* B

Berdasarkan Gambar 4.2, agen juga berhasil mencapai *Goal* B pada seluruh episode pengujian dengan lintasan yang konsisten. Tidak terdapat perubahan jalur yang signifikan antar episode, sehingga menunjukkan bahwa kebijakan yang diperoleh selama proses pelatihan telah mampu menghasilkan jalur navigasi yang stabil.



Gambar 4.3. Hasil *Testing* 5 Episode *Goal* C

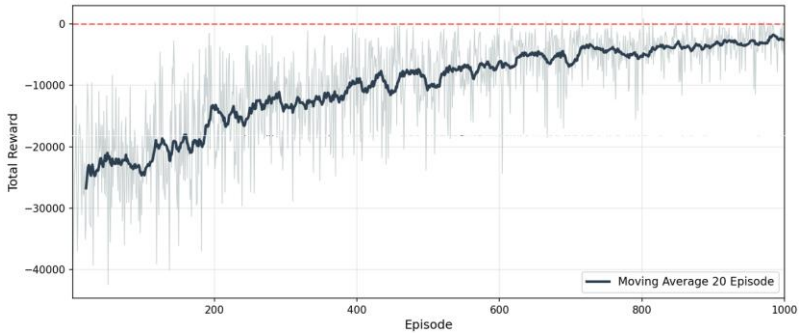
Berdasarkan Gambar 4.3, seluruh pengujian menuju *Goal* C berhasil mencapai tujuan dengan lintasan yang konsisten pada setiap episode. Hal ini menunjukkan bahwa agen mampu menentukan aksi yang tepat pada setiap *state* sehingga proses navigasi berlangsung sesuai dengan kebijakan hasil pelatihan.

Secara keseluruhan, hasil pengujian pada Goal A, Goal B, dan Goal C menunjukkan bahwa agen berhasil mencapai tujuan pada seluruh episode pengujian

dengan lintasan yang konsisten. Konsistensi tersebut menunjukkan bahwa kebijakan hasil pelatihan sebanyak 1000 episode telah konvergen dan mampu diterapkan secara stabil pada seluruh skenario navigasi yang digunakan dalam penelitian. Dengan demikian, jumlah pelatihan sebanyak 1000 episode dipilih sebagai parameter yang digunakan pada implementasi sistem.

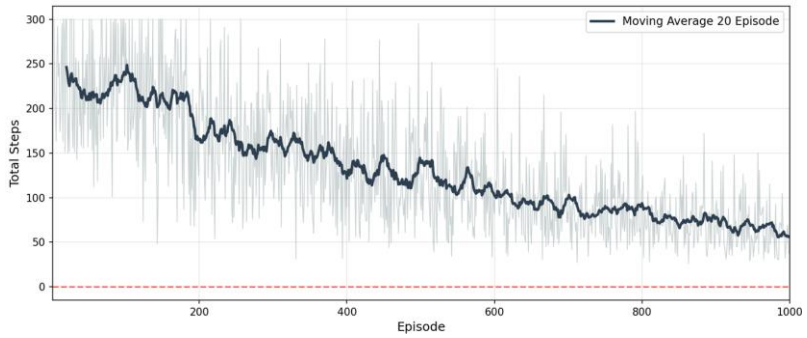
4.1.3 Analisis Proses Pembelajaran Algoritma

Pengujian ini dilakukan untuk menganalisis proses pembelajaran algoritma *Q-Learning* selama pelatihan menggunakan parameter terbaik yang telah diperoleh pada Subbab 4.1.1 serta jumlah pelatihan sebanyak 1000 episode sebagaimana ditentukan pada Subbab 4.1.2. Analisis dilakukan berdasarkan perkembangan nilai *reward* dan jumlah langkah (*step*) pada setiap episode. Selain data aktual pada setiap episode, ditampilkan pula *moving average* 20 episode untuk memperjelas kecenderungan (*trend*) proses pembelajaran.



Gambar 4.4. Grafik *Reward* terhadap Episode

Berdasarkan Gambar 4.4, nilai *reward* pada setiap episode mengalami fluktuasi yang cukup besar, terutama pada awal proses pelatihan. Fluktuasi tersebut menunjukkan bahwa agen masih berada pada tahap eksplorasi sehingga pemilihan aksi belum konsisten dan sering menghasilkan *reward* yang rendah akibat mengambil jalur yang kurang efisien, melakukan rotasi yang tidak diperlukan, maupun menabrak *obstacle*. Meskipun demikian, garis *moving average* menunjukkan kecenderungan nilai *reward* yang terus meningkat seiring bertambahnya jumlah episode. Hal ini menunjukkan bahwa agen secara bertahap mampu mempelajari aksi yang memberikan *reward* lebih tinggi sehingga kualitas *policy* yang dihasilkan semakin baik. Menjelang akhir proses pelatihan, garis *moving average* mulai mendatar yang menunjukkan bahwa proses pembelajaran telah mencapai kondisi yang relatif stabil.



Gambar 4.5. Grafik *Step* terhadap Episode

Berdasarkan Gambar 4.5, jumlah langkah (*step*) yang diperlukan agen untuk mencapai tujuan mengalami penurunan selama proses pelatihan. Pada episode awal, agen masih sering mencapai batas maksimum pelatihan karena belum memiliki *policy* yang mampu mengarahkan robot menuju tujuan secara efisien. Seiring bertambahnya jumlah episode, garis *moving average* menunjukkan tren penurunan jumlah langkah yang cukup signifikan. Meskipun data aktual masih berfluktuasi akibat proses eksplorasi, jumlah langkah yang diperlukan agen semakin kecil dan cenderung stabil pada episode-episode akhir. Kondisi tersebut menunjukkan bahwa agen telah mampu menemukan jalur yang lebih efisien untuk mencapai tujuan.

Untuk mengevaluasi hasil pelatihan secara kuantitatif, dilakukan analisis terhadap beberapa parameter kinerja algoritma Q-*Learning* sebagaimana ditunjukkan pada Tabel 4.6**.**

Tabel 4.6. Hasil Evaluasi *Training* Algoritma Q-*Learning*

Parameter	Hasil
Jumlah episode *training*	1000
Reward episode pertama	−38.153
Reward episode terakhir	−1.698
Reward maksimum	338 (Episode 976)
Reward minimum	−40.594 (Episode 48)
Rata-rata *reward* (200 episode terakhir)	−3.649,45
Step episode pertama	300
Step episode terakhir	38
Step minimum	29 (Episode 569)
Rata-rata *step* (200 episode terakhir)	73,94

Success rate (*greedy policy*, ε = 0)	100%
Rata-rata *step Goal* A	19
Rata-rata *step Goal* B	18
Rata-rata *step Goal* C	20

Berdasarkan Tabel 4.6, terlihat bahwa performa agen mengalami peningkatan selama proses pelatihan. Nilai *reward* meningkat dari −38.153 pada episode pertama menjadi −1.698 pada episode ke-1000. Meskipun nilai *reward* masih bernilai negatif, peningkatan tersebut menunjukkan bahwa agen semakin mampu mengurangi penalti akibat memilih jalur yang tidak efisien, melakukan rotasi yang tidak diperlukan, maupun bertabrakan dengan *obstacle*. Nilai *reward* maksimum sebesar 338 yang diperoleh pada episode ke-976 menunjukkan bahwa pada beberapa episode agen telah berhasil menemukan jalur yang memberikan hasil pembelajaran terbaik.

Selain peningkatan *reward*, jumlah langkah yang dibutuhkan agen juga mengalami penurunan yang signifikan. Pada awal pelatihan agen membutuhkan 150 langkah, yaitu sama dengan batas maksimum langkah (*maximum step*) yang ditetapkan. Setelah proses pembelajaran berlangsung, jumlah langkah pada episode terakhir menurun menjadi 38 langkah, sedangkan jumlah langkah minimum yang berhasil dicapai adalah 29 langkah pada episode ke-569. Penurunan jumlah langkah tersebut menunjukkan bahwa agen semakin mampu memilih jalur yang lebih pendek dan efisien untuk mencapai tujuan.

Evaluasi akhir dilakukan menggunakan greedy policy dengan nilai ε = 0, sehingga seluruh aksi ditentukan berdasarkan nilai Q terbesar tanpa melakukan eksplorasi. Hasil pengujian menunjukkan bahwa agen berhasil mencapai seluruh tujuan navigasi, yaitu *Goal* A, *Goal* B, dan *Goal* C, dengan success rate sebesar 100%. Rata-rata jumlah langkah yang diperlukan juga relatif konsisten, yaitu 19 langkah menuju *Goal* A, 18 langkah menuju *Goal* B, dan 20 langkah menuju *Goal* C. Hasil tersebut menunjukkan bahwa Q-table yang dihasilkan telah mampu merepresentasikan *policy* navigasi yang optimal sehingga siap diterapkan pada tahap implementasi robot mobile.

Berdasarkan keseluruhan hasil analisis dapat disimpulkan bahwa algoritma Q-*Learning* berhasil mempelajari kebijakan navigasi secara bertahap. Peningkatan

nilai *reward*, penurunan jumlah langkah, serta keberhasilan mencapai seluruh tujuan menggunakan *greedy policy* menunjukkan bahwa proses pelatihan telah menghasilkan *policy* yang stabil dan mampu digunakan sebagai dasar pengambilan keputusan pada sistem navigasi robot mobile.

4.1.4 Analisis *Policy* Hasil Pembelajaran

Pengujian ini dilakukan untuk menganalisis *policy* yang dihasilkan oleh algoritma *Q-Learning* setelah proses pelatihan selesai. Analisis dilakukan menggunakan parameter terbaik yang telah diperoleh pada Subbab 4.1.1 dan jumlah pelatihan sebanyak 1000 episode sebagaimana ditentukan pada Subbab 4.1.2. *Policy* divisualisasikan dalam bentuk *policy map* untuk menunjukkan aksi terbaik yang dipilih agen pada setiap state menuju masing-masing lokasi tujuan.

G	←	←	←	←	←	←
↑	X	↑	X	↑	X	↑
↑	X	↑	←	↑	←	↑
↑	X	↑	↑	X	X	↑
↑	←	↑	H	←	←	X

Gambar 4.6. *Policy Map Goal* B

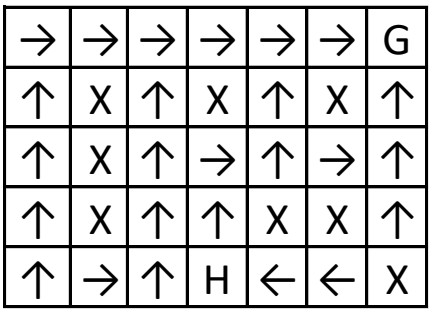
Berdasarkan **Gambar 4.6**, agen mampu menentukan jalur navigasi menuju *Goal* A melalui aksi yang dipilih pada setiap *state*. Arah panah pada setiap sel menunjukkan aksi dengan nilai *Q* tertinggi sehingga membentuk jalur yang mengarah ke tujuan dengan menghindari *obstacle*. Hal ini menunjukkan bahwa proses pembelajaran telah menghasilkan *policy* yang mampu mengarahkan robot menuju *Goal* A secara efisien.

→	→	→	G	←	←	←
↑	X	↑	X	↑	X	↑
↑	X	↑	←	↑	←	↑
↑	X	↑	↑	X	X	↑
↑	→	↑	H	←	←	X

Gambar 4.7. *Policy Map Goal* B

Berdasarkan Gambar 4.7, *policy* yang dihasilkan juga mampu mengarahkan agen menuju *Goal* B melalui jalur yang efisien. Aksi yang dipilih pada setiap *state*

menunjukkan konsistensi dalam menentukan arah pergerakan menuju tujuan tanpa melewati *obstacle* yang terdapat pada lingkungan navigasi. Hal tersebut menunjukkan bahwa algoritma berhasil membentuk *policy* yang sesuai untuk tujuan *Goal* B.



Gambar 4.8. *Policy Map Goal* C

Berdasarkan Gambar 4.8, agen mampu menentukan jalur menuju *Goal* C dengan memilih aksi yang menghasilkan lintasan terpendek sesuai kondisi lingkungan. Jalur yang terbentuk menunjukkan bahwa agen telah memanfaatkan hasil pembelajaran untuk menentukan aksi terbaik pada setiap *state* sehingga mampu mencapai tujuan secara efisien.

Berdasarkan hasil visualisasi *policy map* pada ketiga tujuan, algoritma *Q-Learning* berhasil menghasilkan *policy* yang mampu mengarahkan agen menuju setiap lokasi tujuan melalui jalur yang efisien. Meskipun tujuan navigasi berbeda, *policy* yang dihasilkan tetap menunjukkan konsistensi dalam memilih aksi terbaik pada setiap *state* serta menghindari *obstacle* sesuai dengan lingkungan yang digunakan pada proses pelatihan. Dengan demikian, *policy* yang dihasilkan tidak bersifat tetap, tetapi mampu beradaptasi terhadap perubahan tujuan navigasi sehingga siap diterapkan pada sistem navigasi robot mobile.

4.2 Pengujian Sensor

Pengujian sensor dilakukan untuk memastikan bahwa seluruh sensor yang digunakan pada robot mobile mampu memberikan data yang dibutuhkan selama proses navigasi. Pada penelitian ini digunakan tiga jenis sensor, yaitu *hall effect encoder* sebagai sensor posisi, IMU sebagai sensor orientasi, dan sensor proximity sebagai pendeteksi obstacle. Pengujian dilakukan terhadap masing-masing sensor untuk mengetahui kemampuan sensor dalam mendukung implementasi algoritma *Q-Learning* pada robot mobile.

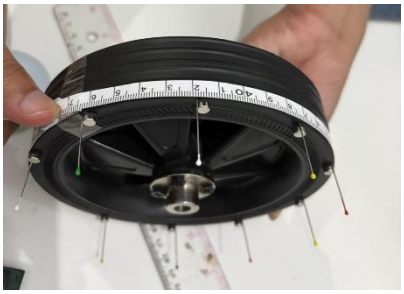
4.2.1 Pengujian Hall Effect Encoder

Pengujian Hall Effect Encoder dilakukan untuk mengevaluasi kemampuan sensor dalam mendeteksi putaran roda serta mengukur perpindahan robot pada setiap aksi *Forward*. Hall Effect Encoder berfungsi sebagai sensor odometri yang menghasilkan data jumlah pulsa (*tick*) berdasarkan putaran roda. Data tersebut digunakan sebagai dasar untuk memperkirakan jarak tempuh robot dan memperbarui posisi robot pada sistem koordinat grid yang digunakan dalam algoritma *Q-Learning*.

Berdasarkan perancangan sistem yang telah dijelaskan pada Bab II, robot menggunakan roda berdiameter 148 mm dengan 10 buah magnet yang dipasang secara merata pada keliling roda. Konfigurasi tersebut menghasilkan resolusi pembacaan sebesar 46,5 mm/tick, sehingga secara teoritis robot memerlukan sekitar 14 tick untuk berpindah sejauh satu grid (650 mm) dari center suatu grid menuju center grid berikutnya.

Gambar 4.9. Pengukuran Diameter Roda Robot

Pengukuran diameter roda dilakukan untuk memastikan dimensi roda yang digunakan sesuai dengan parameter perancangan pada Bab II. Diameter roda yang digunakan adalah 148 mm, yang menjadi dasar dalam penentuan resolusi Hall Effect Encoder.

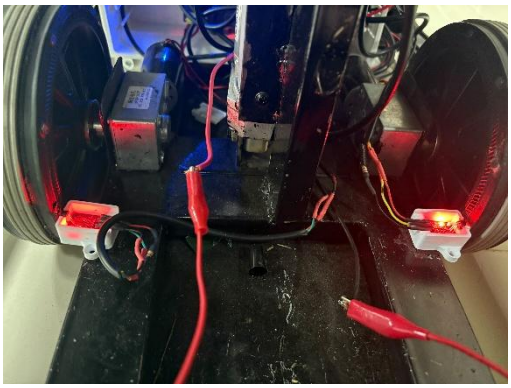


Gambar 4.10. Pemasangan Magnet pada Roda Robot

Gambar 4.10 menunjukkan konfigurasi akhir pemasangan magnet pada roda robot. Pemasangan magnet yang merata bertujuan menghasilkan pembacaan pulsa yang stabil sehingga setiap pulsa merepresentasikan perpindahan roda sebesar 46,5 mm sebagaimana telah dihitung pada Bab III.

Gambar 4.11. Konfigurasi Magnet pada Roda Robot

Sebanyak 10 buah magnet dipasang secara merata pada keliling roda robot. Setiap magnet yang melewati Hall Effect Encoder akan menghasilkan satu pulsa (*tick*), sehingga dalam satu kali putaran roda diperoleh 10 tick.



Gambar 4.12. Pembacaan Tick Hall Effect Encoder

Setelah proses pemasangan selesai, dilakukan pengujian untuk memastikan Hall Effect Encoder mampu membaca setiap magnet yang melewati sensor. Pengujian dilakukan dengan menggerakkan robot menggunakan satu aksi *Forward*. Selama roda berputar, nilai *tick* yang ditampilkan pada Serial Monitor bertambah secara bertahap sesuai dengan jumlah magnet yang melewati sensor. Untuk memberikan gambaran mengenai proses pembacaan Hall Effect Encoder selama satu aksi *Forward*, sebagian data telemetri ditunjukkan pada Tabel 4.7.

Tabel 4.7. Hasil Pengujian Hall Effect Encoder *Forward*

Tick *Left*	Tick *Right*	Grid X	Grid Y	*Action*
0	0	3	4	FWD

1	1	3	4	FWD
2	2	3	4	FWD
3	3	3	4	FWD
4	4	3	4	FWD
5	5	3	4	FWD
6	6	3	4	FWD
7	7	3	3	FWD
8	8	3	3	FWD
9	9	3	3	FWD
10	10	3	3	FWD
11	11	3	3	FWD
12	12	3	3	FWD
13	13	3	3	FWD
14	14	3	3	STOP

Berdasarkan Tabel 4.7 terlihat bahwa nilai Tick *Left* dan Tick *Right* meningkat secara bertahap mulai dari 0 hingga 14 selama robot melakukan satu aksi *Forward*. Ketika pembacaan encoder mencapai sekitar 7 tick, koordinat Grid X dan Grid Y mulai berubah sebagai indikasi bahwa robot telah melewati batas antargrid. Namun demikian, robot tetap melanjutkan pergerakan hingga pembacaan mencapai sekitar 13–14 tick, yang menunjukkan bahwa robot telah mencapai posisi center pada grid tujuan. Setelah posisi tersebut tercapai, aksi *Forward* dinyatakan selesai dan sistem melanjutkan ke aksi berikutnya sesuai hasil keputusan algoritma *Q-Learning*.

Untuk mengetahui tingkat akurasi Hall Effect Encoder, dilakukan pengukuran jarak perpindahan robot menggunakan meteran. Pengukuran dilakukan dari center grid awal menuju center grid tujuan setelah robot menyelesaikan satu aksi *Forward*.

Gambar 4.13. Pengukuran Jarak Perpindahan Robot Menggunakan Meteran

Hasil pengujian Hall Effect Encoder ditunjukkan pada Tabel 4.8.

Tabel 4.8. Hasil Pengujian Tick Hall Effect Encoder

Aksi	Avg Tick	Estimasi Encoder (mm)	Jarak Aktual (mm)	Encoder Error (%)
FWD-1	13,5	627,75	655	4,16
FWD-2	14,0	651,00	650	0,15
FWD-3	13,5	627,75	670	6,31
FWD-4	14,0	651,00	650	0,15
FWD-5	13,5	627,75	650	3,42
FWD-6	13,5	627,75	664	5,46
FWD-7	13,5	627,75	670	6,31
Rata-rata	13,64	634,93	658,43	3,71

Berdasarkan Tabel 4.8, Hall Effect Encoder menghasilkan pembacaan yang relatif konsisten pada kisaran 13,5–14,0 tick untuk setiap perpindahan satu grid. Rata-rata pembacaan encoder roda kiri sebesar 13,00 tick, sedangkan roda kanan sebesar 14,29 tick, sehingga diperoleh rata-rata keseluruhan sebesar 13,64 tick. Nilai tersebut mendekati hasil perhitungan teoritis yang telah diperoleh pada Bab III, yaitu sekitar 14 tick untuk perpindahan sejauh 650 mm.

Jumlah tick yang diperoleh pada setiap percobaan kemudian dikonversi menjadi jarak menggunakan resolusi encoder sebesar 46,5 mm/tick. Berdasarkan rata-rata pembacaan encoder sebesar 13,64 tick, diperoleh estimasi jarak sebesar 634,93 mm. Hasil estimasi tersebut kemudian dibandingkan dengan rata-rata jarak aktual hasil pengukuran menggunakan meteran sebesar 658,43 mm. Persentase galat dihitung menggunakan Persamaan (2.20), sehingga diperoleh:

$$\text{Error} = \frac{|\ 634{,}93 - 658{,}43\ |}{658{,}43} \times 100\% = 3{,}57\%$$

Hasil tersebut menunjukkan bahwa Hall Effect Encoder memiliki *encoder estimation error* sebesar 3,57% terhadap hasil pengukuran aktual. Nilai tersebut menunjukkan bahwa Hall Effect Encoder mampu memberikan estimasi jarak yang cukup baik, meskipun masih terdapat perbedaan terhadap hasil pengukuran aktual. Perbedaan tersebut dipengaruhi oleh resolusi encoder yang terbatas, kemungkinan kehilangan pembacaan pulsa (*missed pulse*), perubahan jarak antara magnet dan sensor (*air gap*), getaran mekanik, serta kemungkinan *wheel slip* selama robot bergerak.

Selain itu, dilakukan pula evaluasi terhadap ketepatan perpindahan robot menuju target sejauh 650 mm. Berdasarkan rata-rata jarak aktual sebesar 658,43 mm, diperoleh selisih perpindahan sebesar 8,43 mm terhadap target. Persentase penyimpangan tersebut dihitung menggunakan Persamaan (2.20), sehingga diperoleh *endpoint motion error* sebesar 1,30%. Nilai tersebut menunjukkan bahwa robot mampu mencapai posisi tujuan dengan deviasi yang relatif kecil meskipun masih dipengaruhi oleh karakteristik motor, inersia roda, dan *wheel slip*.

Berdasarkan hasil pengujian juga terlihat bahwa pembacaan encoder roda kiri berada pada kisaran 12–14 tick, sedangkan roda kanan berada pada kisaran 14–15 tick. Perbedaan maksimum sebesar 3 tick hanya terjadi pada beberapa percobaan dan masih menghasilkan rata-rata pembacaan yang mendekati nilai teoritis. Hal ini menunjukkan bahwa kedua encoder mampu memberikan data odometri yang cukup konsisten untuk memperkirakan perpindahan robot pada setiap aksi *Forward*.

Secara keseluruhan, Hall Effect Encoder mampu memberikan data odometri yang stabil untuk mendukung pembaruan posisi (*state*) pada algoritma Q-Learning. Hasil pengujian menunjukkan bahwa Hall Effect Encoder memiliki *encoder estimation error* sebesar 3,57%, sedangkan sistem menghasilkan *endpoint motion error* sebesar 1,30% terhadap target perpindahan satu grid. Adapun analisis mengenai perubahan orientasi robot selama proses perpindahan akan dibahas lebih lanjut pada pengujian sensor IMU.

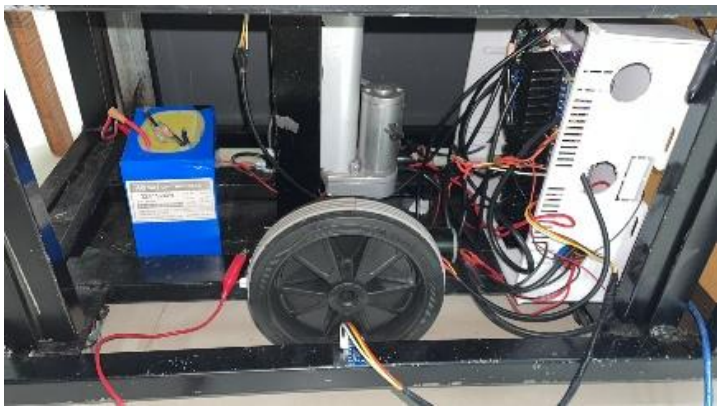
4.2.2 Pengujian IMU (Inertial Measurement Unit)

Pengujian Inertial Measurement Unit (IMU) dilakukan untuk mengevaluasi kemampuan sensor MPU6050 dalam mendeteksi perubahan orientasi robot selama proses navigasi. Pada penelitian ini, MPU6050 dimanfaatkan sebagai sensor umpan balik (*feedback*) yang memberikan informasi heading berdasarkan pembacaan gyroscope sumbu Z (*yaw*). Informasi heading tersebut digunakan oleh pengendali PID untuk mempertahankan arah robot saat bergerak lurus (*Forward*) serta menentukan penghentian rotasi ketika robot melakukan manuver belok (*Left* dan *Right*).

Sebelum dilakukan pengujian, sensor MPU6050 dikalibrasi pada kondisi robot diam untuk memperoleh nilai offset gyroscope sehingga pengaruh *drift* dapat diminimalkan. Selain itu, sistem juga menerapkan mekanisme *heading snap* setelah

robot menyelesaikan setiap aksi agar orientasi robot kembali ke arah kardinal (0°, 90°, 180°, atau 270°), sehingga kesalahan heading tidak terakumulasi pada aksi berikutnya.

Perlu diketahui bahwa pembacaan heading pada MPU6050 dan kompas digital menggunakan sistem referensi yang berbeda. Heading IMU mengacu pada sistem koordinat robot yang telah ditetapkan pada program, sedangkan kompas mengacu pada arah utara magnetik bumi (*magnetic north*). Oleh karena itu, kompas digunakan sebagai alat pembanding untuk memvalidasi perubahan orientasi robot selama proses navigasi, bukan untuk membandingkan nilai heading absolut secara langsung.



Gambar 4.14. Pemasangan Modul MPU6050

```
[16:24:53] RX: HOMING_NORTH,err=0.03
[16:24:53] RX: HOMING:COMPLETE
[16:24:53] RX: EVT,ACT_HOMED
[16:25:00] RX: OK:RST_YAW
[16:25:01] Pulse counter L/R reset ke 0
[16:25:01] RX: OK:RST_TICKS
[16:25:02] RX: OK:HOME
[16:25:02] RX: EVT,HOME_OK
```

Gambar 4.15. Proses Kalibrasi MPU6050

Pada saat robot bergerak lurus, heading hasil pembacaan MPU6050 digunakan sebagai umpan balik pengendali PID. Target heading dipertahankan pada 90°, sedangkan nilai heading aktual dibaca secara kontinu oleh MPU6050. Ketika heading mulai menyimpang dari target, pengendali PID secara otomatis mengubah nilai PWM motor kiri dan kanan sehingga robot kembali menuju arah yang diinginkan

Tabel 4.9. Hasil Pengujian Heading *Forward* Robot

No	Waktu (ms)	Target Heading IMU (°)	Heading IMU (°)	Sudut Aktual (Kompas) (°)	PWM *Left*	PWM *Right*
1	91073	90.00	90.00	207	220	220
2	91123	90.00	90.01	-	220	220
3	91173	90.00	90.11	-	220	220
4	91223	90.00	90.27	-	219	221
5	91273	90.00	90.59	-	217	223
6	91323	90.00	90.82	-	205	235
7	91373	90.00	91.13	-	205	235
8	91423	90.00	91.29	-	205	235
9	91473	90.00	91.41	-	205	235
10	91523	90.00	91.43	-	205	235
11	91573	90.00	91.37	-	205	235
12	91623	90.00	91.27	-	205	235
13	91673	90.00	91.04	-	205	235
14	91723	90.00	90.87	-	205	235
15	91773	90.00	90.66	-	217	223
16	91823	90.00	90.57	-	218	222
17	91873	90.00	90.50	-	218	222
18	91923	90.00	90.48	-	218	222
19	91973	90.00	90.45	203	218	222

Berdasarkan Tabel 4.9, target heading selama robot bergerak dipertahankan pada 90°, sedangkan heading aktual yang dibaca oleh MPU6050 berada pada rentang 90,00° hingga 91,43°. Selama proses perpindahan satu grid, heading mengalami perubahan kecil akibat pengaruh karakteristik motor, toleransi mekanik, serta kondisi permukaan lintasan. Meskipun demikian, penyimpangan tersebut masih dapat dikoreksi oleh pengendali PID sehingga orientasi robot tetap berada di sekitar nilai target.

Untuk mengetahui tingkat akurasi pembacaan heading, dilakukan perhitungan error heading menggunakan Persamaan 2.15. Berdasarkan data pada

Tabel 4.9, error maksimum terjadi ketika heading aktual mencapai 91,43°, sehingga diperoleh:

$$e_h = | 90{,}00 - 91{,}43 |$$

$$e_h = 1{,}43°$$

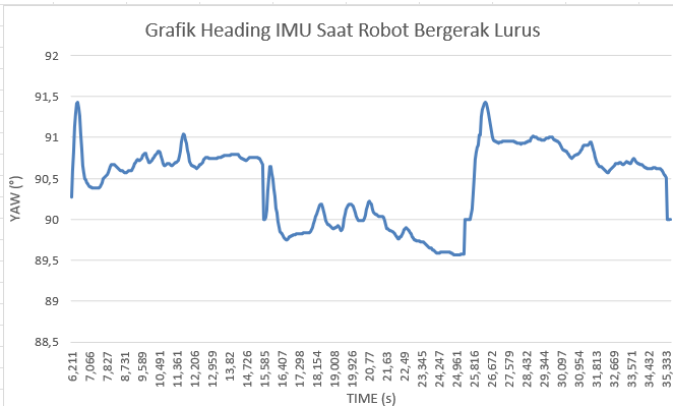
Selain menghitung error maksimum, tingkat akurasi sensor juga dievaluasi menggunakan Mean Absolute Error (MAE) berdasarkan Persamaan 2.16. Perhitungan dilakukan menggunakan seluruh 19 data heading hasil pengujian, sehingga diperoleh:

$$MAE = \frac{16{,}28}{19}$$

$$MAE = 0{,}86°$$

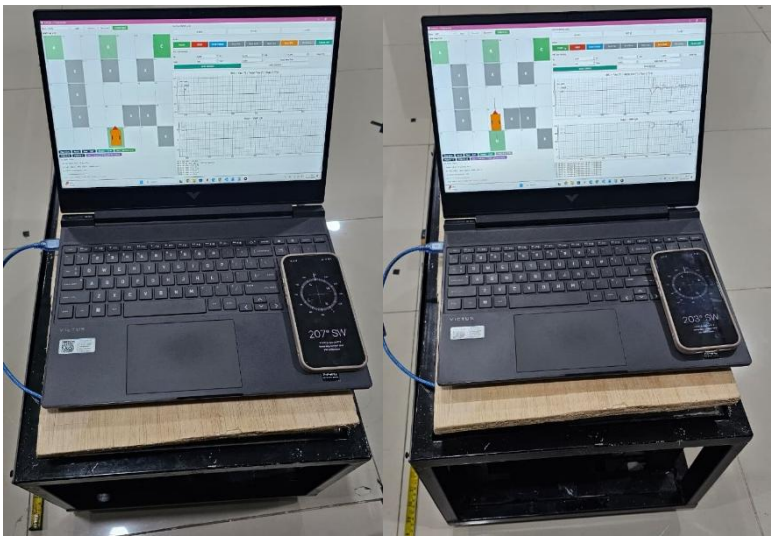
Nilai error heading maksimum sebesar 1,43° dan MAE sebesar 0,86° menunjukkan bahwa MPU6050 mampu mempertahankan orientasi robot dengan tingkat penyimpangan yang relatif kecil selama proses perpindahan satu grid. Hasil tersebut menunjukkan bahwa sensor memiliki kestabilan yang baik untuk digunakan sebagai sensor umpan balik pada sistem navigasi robot berbasis Q-*Learning*.

Hasil pengukuran menggunakan kompas menunjukkan orientasi robot berubah dari sekitar 207° pada kondisi awal menjadi 203° pada kondisi akhir pengujian. Perubahan sekitar 4° tersebut tidak menunjukkan adanya kesalahan pembacaan MPU6050, melainkan disebabkan oleh perbedaan sistem referensi antara heading IMU dan kompas digital. Oleh karena itu, kompas digunakan sebagai pembanding perubahan orientasi robot selama pengujian, bukan sebagai acuan nilai heading absolut.



Gambar 4.16. Grafik Heading IMU Saat Robot Lurus

Berdasarkan Gambar 4.16, heading robot berada di sekitar nilai target 90° selama proses perpindahan satu grid. Pada awal pergerakan terlihat adanya penyimpangan sesaat (*transient response*) sebelum heading menjadi stabil. Selama robot bergerak, heading mengalami fluktuasi kecil pada kisaran ±1° terhadap nilai target. Di sekitar pertengahan pengujian terlihat adanya perubahan heading yang relatif lebih besar, kemudian sistem kembali mengoreksi orientasi hingga heading kembali mendekati 90°. Kondisi tersebut menunjukkan bahwa pengendali PID mampu merespons perubahan orientasi secara cepat sehingga robot tetap mempertahankan arah geraknya selama proses perpindahan.



Gambar 4.17. Pengujian Robot Bergerak Lurus Menggunakan Kompas

Hasil pengukuran menggunakan kompas menunjukkan bahwa orientasi robot berubah dari sekitar 207° menjadi 203° selama robot bergerak lurus. Perubahan sebesar sekitar 4° tersebut menunjukkan adanya sedikit penyimpangan arah akibat pengaruh toleransi mekanik, ketidakseimbangan putaran motor, dan karakteristik sensor IMU. Meskipun demikian, penyimpangan tersebut masih dapat dikoreksi oleh sistem sehingga robot tetap berhasil mencapai posisi tujuan sesuai lintasan yang direncanakan.

Selanjutnya dilakukan pengujian ketika robot melakukan manuver belok kanan menggunakan metode rotate in place, yaitu kedua roda berputar dengan arah berlawanan sehingga robot berputar terhadap pusatnya. Berdasarkan hasil *tuning*, sistem menetapkan target penghentian rotasi sebesar 87° untuk mengompensasi efek inersia motor sebelum dilakukan proses *heading snap*. Data hasil pengujian ditunjukkan pada Tabel 4.10.

Tabel 4.10. Hasil Pengujian Heading *Right* Robot

Target Heading IMU (°)	Heading IMU (°)	Sudut Aktual (Kompas) (°)	PWM *Right*	PWM *Left*
3	90.00	112	-213	213
3	88.33	-	-210	210
3	82.26	-	-197	197
3	74.49	-	-181	181
3	65.71	-	-163	163
3	58.30	-	-147	147
3	51.93	-	-134	134
3	46.52	-	-123	123
3	33.49	-	-88	88
3	21.29	-	-65	65
3	9.90	-	-47	47
3	4.09	23	0	0

Berdasarkan Tabel 4.10, *heading* hasil pembacaan MPU6050 berubah secara bertahap dari 90,00° menjadi 4,09°, sedangkan hasil pengukuran menggunakan kompas berubah dari 112° menjadi 23°. Perubahan tersebut menunjukkan bahwa robot telah melakukan rotasi sekitar 90° ke arah kanan. Karena MPU6050 dan kompas menggunakan sistem referensi yang berbeda, perbandingan dilakukan berdasarkan besar perubahan sudut rotasi, bukan nilai *heading* absolut. Berdasarkan Persamaan (2.15), diperoleh perubahan sudut sebesar 85,91° pada MPU6050 dan 89° pada kompas, sehingga diperoleh error sebesar 3,09° atau 3,47%. Hasil tersebut menunjukkan bahwa MPU6050 memiliki tingkat akurasi yang baik dalam mengukur perubahan orientasi robot selama proses rotasi.

Selain akurasi pembacaan sensor, pengujian ini juga mengevaluasi ketelitian robot dalam menghentikan rotasi pada *heading* target. Berdasarkan Persamaan (2.15), diperoleh error penghentian rotasi sebesar 1,09° terhadap *heading* target 3°. Nilai *error* tersebut menunjukkan bahwa robot mampu menghentikan rotasi dengan

penyimpangan yang relatif kecil sehingga masih berada dalam batas toleransi sistem dan tidak memengaruhi proses navigasi pada satu kali manuver belok.

Selama proses rotasi, kedua motor berputar dengan arah yang berlawanan, ditunjukkan oleh nilai PWM motor kanan yang bernilai negatif dan PWM motor kiri yang bernilai positif dengan besar yang hampir sama. Pada awal rotasi kedua motor menerima PWM sebesar ±213, kemudian nilainya menurun secara bertahap hingga mencapai 0 ketika *heading* mendekati target. Penurunan PWM secara bertahap bertujuan mengurangi kecepatan sudut robot sehingga efek inersia dapat diminimalkan dan robot tidak mengalami *overshoot* yang berlebihan sebelum proses *heading snap* dilakukan. Dengan mekanisme tersebut, robot mampu berhenti pada orientasi yang mendekati target sehingga arah robot tetap sesuai dengan lintasan yang dihasilkan oleh algoritma *Q-Learning*.

Gambar 4.18. Pengujian Belok Kanan Menggunakan Kompas

Hasil pengujian menggunakan kompas menunjukkan bahwa orientasi robot berubah dari 112° menjadi 23°, yang mengonfirmasi bahwa robot telah melakukan rotasi sekitar 90° ke arah kanan. Dengan demikian, hasil pengujian menunjukkan bahwa MPU6050 mampu memberikan informasi heading yang cukup akurat untuk menentukan penghentian rotasi pada manuver belok kanan.

Tabel 4.11. Hasil Pengujian Heading *Left* Robot

Target Heading IMU (°)	Heading IMU (°)	Sudut Aktual (Kompas) (°)	PWM *Right*	PWM *Left*
87	0.00	20	213	-213

87	1.60	-	210	-210
87	8.16	-	196	-196
87	14.78	-	183	-183
87	28.13	-	152	-152
87	44.41	-	115	-115
87	60.19	-	84	-84
87	68.98	-	65	-65
87	80.29	-	46	-46
87	84.39	-	39	-39
87	85.84	289	0	0

Berdasarkan Tabel 4.11, heading hasil pembacaan MPU6050 meningkat secara bertahap dari 0,00° menjadi 85,84°, sedangkan hasil pengukuran menggunakan kompas berubah dari 20° menjadi 289°. Pada sistem kompas, perubahan tersebut tetap menunjukkan rotasi sebesar sekitar 90° karena kompas menggunakan rentang sudut 0°–359°. Selama robot berputar ke kiri, arah kompas melewati 0°, kemudian berlanjut menuju 359° hingga mencapai 289°, sehingga perubahan orientasi tetap merepresentasikan rotasi sebesar kurang lebih 90°.

Untuk mengetahui tingkat ketelitian penghentian rotasi, dilakukan perhitungan error heading menggunakan Persamaan 2.15. Berdasarkan data hasil pengujian, heading akhir yang diperoleh sebesar 85,84° terhadap target penghentian rotasi 87°, sehingga diperoleh:

$$e_h = | 87{,}00 - 85{,}84 |$$

$$e_h = 1{,}16^{\circ}$$

Nilai error sebesar 1,16° menunjukkan bahwa robot mampu menghentikan rotasi dengan penyimpangan yang relatif kecil terhadap target yang telah ditentukan. Nilai tersebut masih berada dalam batas toleransi sistem sehingga orientasi robot tetap sesuai untuk melanjutkan proses navigasi pada perpindahan grid berikutnya.

Selama proses rotasi, kedua motor berputar dengan arah yang berlawanan, ditunjukkan oleh nilai PWM motor kanan yang bernilai positif dan PWM motor kiri yang bernilai negatif dengan besar yang hampir sama. Pada awal rotasi kedua motor menerima PWM sebesar ±213, kemudian nilainya diturunkan secara bertahap hingga mencapai 0 ketika heading mendekati target. Penurunan PWM secara bertahap bertujuan mengurangi kecepatan sudut robot sehingga efek inersia dapat diminimalkan dan robot dapat berhenti lebih dekat dengan sudut yang diinginkan. Dengan mekanisme tersebut, proses rotasi berlangsung lebih stabil dan overshoot dapat dikurangi sebelum sistem melakukan heading snap.

Gambar 4.19. Pengujian Belok Kiri Menggunakan Kompas

Hasil pengujian menggunakan kompas menunjukkan bahwa orientasi robot berubah dari 20° menjadi 289°. Walaupun nilai awal dan akhir terlihat berbeda jauh secara numerik, perubahan tersebut tetap menunjukkan rotasi sekitar 90° karena sistem kompas menggunakan representasi sudut melingkar. Oleh karena itu, validasi dilakukan berdasarkan besar perubahan orientasi, bukan selisih langsung antara nilai awal dan nilai akhir. Hasil tersebut menunjukkan bahwa pembacaan MPU6050 memiliki kecenderungan yang sesuai dengan perubahan orientasi aktual robot selama melakukan manuver belok kiri.

Berdasarkan keseluruhan hasil pengujian, MPU6050 mampu memberikan pembacaan heading yang konsisten selama robot melakukan navigasi. Pada pengujian gerak lurus diperoleh error heading maksimum sebesar 1,43° dengan

Mean Absolute Error (MAE) sebesar 0,86°, sedangkan pada manuver belok kanan dan belok kiri diperoleh error rotasi masing-masing sebesar 1,09° dan 1,16°. Namun, pada pengujian navigasi Home–*Goal*–Home ditemukan bahwa error kecil pada setiap manuver belok dapat terakumulasi sehingga pada belokan keempat atau kelima penyimpangan orientasi aktual dapat mencapai sekitar 5°–10°.

Kondisi tersebut disebabkan oleh drift gyroscope, yaitu akumulasi kesalahan yang muncul akibat proses integrasi kecepatan sudut terhadap waktu. Karakteristik ini merupakan salah satu keterbatasan sensor MPU6050 berbasis MEMS. Penelitian [42] melaporkan bahwa performa MPU6050 cenderung mengalami penurunan seiring bertambahnya waktu operasi akibat akumulasi kesalahan pada data gyroscope, sehingga diperlukan mekanisme koreksi seperti *filtering* atau *sensor fusion* untuk mengurangi pengaruh drift.

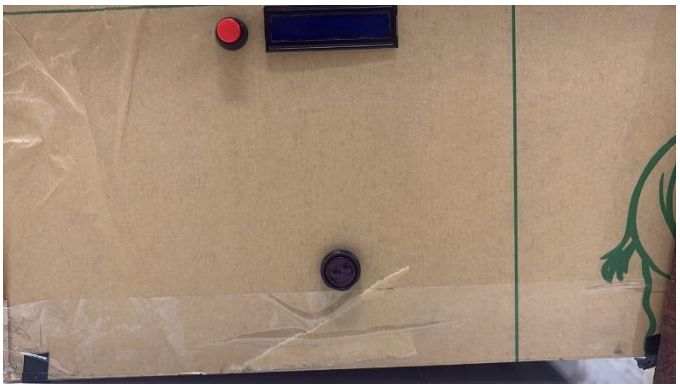
Pada penelitian ini, pengaruh drift diminimalkan melalui proses kalibrasi awal, Zero Velocity Update (ZUPT) ketika robot berhenti, serta mekanisme heading snap setelah setiap aksi selesai. Meskipun belum sepenuhnya menghilangkan drift, mekanisme tersebut mampu mengurangi akumulasi error sehingga robot tetap dapat mengikuti lintasan hasil algoritma Q-*Learning* dan mencapai tujuan navigasi dengan baik.

4.2.3 Pengujian Sensor Proximity

Pengujian sensor proximity dilakukan untuk mengevaluasi kemampuan sensor E18-D80NK dalam mendeteksi keberadaan objek pada jalur pergerakan robot. Pada penelitian ini, sensor proximity berfungsi sebagai sistem keselamatan (*safety system*) yang digunakan untuk mencegah terjadinya tabrakan ketika robot sedang menjalankan proses navigasi. Berbeda dengan sensor utama yang digunakan untuk menentukan posisi dan orientasi robot, sensor proximity hanya berperan sebagai pendeteksi halangan. Ketika sensor mendeteksi adanya objek di depan robot, sistem akan menghentikan pergerakan robot untuk sementara. Setelah objek dihilangkan, robot akan melanjutkan aksi berikutnya sesuai urutan navigasi yang telah ditentukan oleh algoritma *Q-Learning* tanpa melakukan perhitungan ulang jalur.

Sensor proximity dipasang pada bagian depan robot dengan posisi berada di tengah rangka dan menghadap ke arah lintasan. Posisi tersebut dipilih agar objek

yang berada tepat di depan robot dapat dideteksi sebelum terjadi kontak fisik. Jarak deteksi sensor diatur menggunakan *potensiometer* pada bagian belakang sensor. Berdasarkan hasil penyetelan, sensor memiliki jarak deteksi maksimum sekitar 50 cm.



Gambar 4.20. Pemasangan Sensor Proximity pada Robot

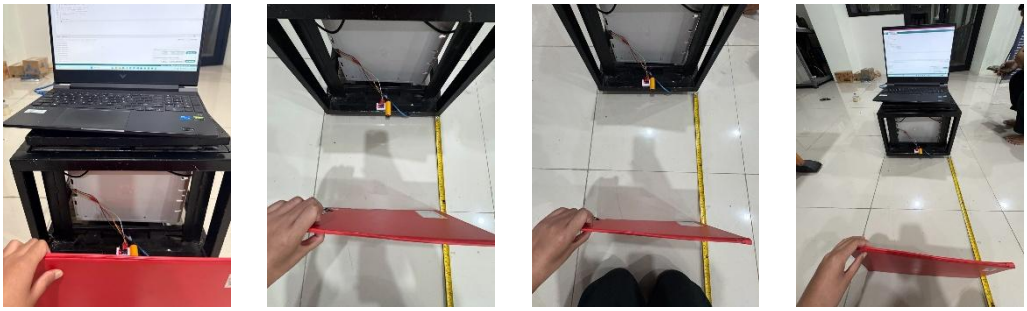
Setelah proses pemasangan selesai, dilakukan pengujian dengan meletakkan objek pada beberapa variasi jarak terhadap sensor. Setiap pengujian dilakukan dengan mengamati status keluaran sensor serta respons robot ketika objek berada pada area deteksi maupun di luar area deteksi. Pengujian ini bertujuan untuk memastikan bahwa sensor proximity mampu mendeteksi keberadaan objek sesuai dengan jarak deteksi yang telah ditentukan. Selain itu, hasil pengujian digunakan untuk memverifikasi bahwa robot dapat memberikan respons penghentian pergerakan ketika objek terdeteksi sebagai bagian dari sistem penghindaran rintangan. Hasil pengujian ditunjukkan pada Tabel 4.12.

Tabel 4.12. Hasil Pengujian Proximity

Jarak Objek (cm)	Status Sensor	Respon Robot
10	Terdeteksi	Robot berhenti
35	Terdeteksi	Robot berhenti
50	Terdeteksi	Robot berhenti
55	Tidak Terdeteksi	Robot Jalan

Berdasarkan Tabel 4.12 terlihat bahwa sensor proximity mampu mendeteksi objek secara konsisten hingga jarak sekitar 50 cm. Ketika objek berada di dalam area deteksi, keluaran sensor berubah menjadi aktif sehingga sistem segera

menghentikan kedua motor penggerak. Sebaliknya, ketika objek berada pada jarak lebih dari 50 cm, sensor tidak lagi mendeteksi keberadaan objek sehingga robot tetap menjalankan aksi navigasi sesuai dengan perintah yang diberikan oleh sistem.



(a) (b) (c) (d)

Gambar 4.21. (a) Obstacle pada jarak 10 cm, (b) obstacle pada jarak 35 cm, (c) obstacle pada jarak 50 cm, dan (d) obstacle pada jarak 55 cm.

Selain dipengaruhi oleh jarak, kemampuan sensor proximity juga dipengaruhi oleh posisi obstacle terhadap arah pancaran cahaya inframerah. Untuk mengevaluasi kondisi tersebut, dilakukan pengujian dengan menempatkan obstacle pada beberapa posisi terhadap sensor. Hasil pengujian ditunjukkan pada Gambar 4.21.

Berdasarkan Gambar 4.22 terlihat bahwa sensor memiliki area deteksi yang berpusat pada arah pancaran cahaya inframerah. Obstacle yang berada tepat di depan sensor dapat dideteksi dengan baik, sedangkan obstacle yang berada di luar area pancaran tidak menghasilkan sinyal deteksi. Pada penelitian ini sensor dipasang tepat di tengah rangka robot yang memiliki lebar sekitar 40 cm, sehingga area deteksi sensor hanya mencakup bagian tengah lintasan robot. Akibatnya, obstacle yang berada pada sisi kanan maupun sisi kiri robot hingga berada di luar jangkauan pancaran sensor tidak akan terdeteksi meskipun masih berada di depan robot.

(a) (b)

Gambar 4.22. (a) Diluar Jangkauan Sensor (b) Dalam Jangkauan Sensor

Kondisi tersebut merupakan salah satu keterbatasan sistem akibat penggunaan satu sensor proximity yang dipasang pada bagian tengah robot. Meskipun demikian, keterbatasan tersebut tidak memengaruhi hasil penelitian karena seluruh obstacle pada skenario pengujian ditempatkan tepat pada jalur pergerakan robot sehingga masih berada di dalam area deteksi sensor.

Selama proses pengujian, sensor proximity mampu memberikan respons yang konsisten. Ketika obstacle terdeteksi, robot secara otomatis menghentikan kedua motor penggerak hingga jalur kembali bebas. Setelah obstacle dipindahkan, robot melanjutkan aksi yang sedang dijalankan tanpa mengubah urutan aksi hasil algoritma *Q-Learning*. Dengan demikian, sensor proximity berfungsi sebagai sistem keselamatan (*interlock*) yang menghentikan sementara pergerakan robot tanpa memengaruhi proses navigasi.

Berdasarkan hasil pengujian dapat disimpulkan bahwa sensor proximity E18-D80NK mampu mendeteksi obstacle secara efektif hingga jarak 50 cm serta memberikan respons penghentian robot secara otomatis. Meskipun area deteksinya terbatas pada bagian tengah robot, sensor tetap mampu menjalankan fungsinya dengan baik pada seluruh skenario pengujian sehingga mendukung keamanan sistem navigasi robot berbasis *Q-Learning*.

4.3 Pengujian Pembentukan State Robot Mobile

Pengujian pembentukan *state* dilakukan untuk memastikan bahwa robot mobile mampu membentuk *state* sesuai dengan representasi yang digunakan pada algoritma *Q-Learning*. Pada penelitian ini, *state* direpresentasikan sebagai S = (X,Y,D,G), di mana X dan Y menyatakan posisi robot pada lingkungan grid, D menyatakan orientasi robot, sedangkan G merupakan tujuan navigasi. Posisi robot diperoleh berdasarkan data *hall effect encoder*, orientasi robot diperoleh dari sensor IMU, sedangkan tujuan navigasi ditentukan sebelum robot mulai bergerak. Seluruh komponen tersebut kemudian digabungkan menjadi *state* yang digunakan sebagai masukan dalam proses pemilihan aksi berdasarkan *Q-table*.

4.3.1 Pembentukan Posisi dan Orientasi Robot

Pembentukan posisi robot dilakukan menggunakan dua buah sensor *hall effect encoder* yang dipasang pada roda kiri dan roda kanan. Jumlah pulsa (*tick*) yang dihasilkan encoder digunakan untuk menghitung perpindahan robot, kemudian dikonversi menjadi koordinat grid dengan ukuran 1 grid = 650 mm. Sementara itu, orientasi robot diperoleh dari sensor IMU dalam bentuk nilai *yaw* yang selanjutnya dipetakan menjadi empat arah utama, yaitu North, East, South, dan West.

Tabel 4.13. Hasil Pembentukan Posisi dan Orientasi Robot

Tick Kiri	Tick Kanan	Yaw (°)	Grid (X,Y)	Orientasi
0	0	0,00	(3,4)	North
7	7	91,88	(2,4)	West
7	7	90,22	(1,4)	West
7	7	91,77	(0,4)	West
6	8	0,24	(0,3)	North
6	8	0,76	(0,2)	North
7	7	2,52	(0,1)	North
7	7	1,13	(0,0)	North

Berdasarkan Tabel 4.13, perubahan koordinat grid diperoleh dari hasil pembacaan *tick* encoder yang digunakan dalam proses odometri. Robot memulai navigasi dari posisi Home (3,4) dengan orientasi North, kemudian melakukan rotasi hingga orientasi berubah menjadi West sebelum bergerak menuju *Goal* A. Selanjutnya, koordinat grid berubah secara bertahap dari (3,4) menjadi (2,4), (1,4), (0,4), kemudian robot kembali menghadap North dan melanjutkan pergerakan hingga mencapai *Goal* A pada koordinat (0,0).

Terlihat bahwa perubahan koordinat grid terjadi ketika pembacaan encoder berada pada kisaran 6–8 tick, meskipun perpindahan dari pusat (*center*) suatu grid menuju pusat grid berikutnya memerlukan sekitar 14 tick. Kondisi tersebut terjadi karena robot memulai pergerakan dari pusat grid awal, sedangkan perubahan koordinat grid ditentukan ketika robot telah melewati batas antargrid yang berada di tengah dua grid. Dengan demikian, koordinat grid berubah terlebih dahulu sebelum robot mencapai pusat grid berikutnya. Setelah satu aksi perpindahan

selesai, sistem melakukan proses *snap to grid* sehingga posisi robot kembali berada pada pusat grid yang baru. Mekanisme tersebut bertujuan untuk mengurangi akumulasi kesalahan odometri selama proses navigasi.

Selain itu, nilai *yaw* yang diperoleh dari sensor IMU juga menunjukkan perubahan orientasi yang sesuai dengan arah pergerakan robot. Nilai *yaw* di sekitar 0° direpresentasikan sebagai North, sedangkan nilai *yaw* di sekitar 90° direpresentasikan sebagai West sesuai dengan sistem koordinat yang digunakan pada penelitian ini. Hasil tersebut menunjukkan bahwa data encoder dan IMU mampu membentuk komponen posisi dan orientasi yang digunakan dalam pembentukan *state*.

4.3.2 Pembentukan State

Setelah posisi dan orientasi robot diperoleh, kedua informasi tersebut dikombinasikan dengan tujuan navigasi (*goal*) sehingga membentuk *state* yang digunakan sebagai masukan algoritma *Q-Learning*. Pada penelitian ini, tujuan navigasi direpresentasikan sebagai *Goal* A, *Goal* B, *Goal* C, dan Home. Perubahan salah satu komponen penyusun akan menghasilkan *state* yang berbeda.

Tabel 4.14. Hasil Pembentukan State

Grid (X,Y)	Orientasi	*Goal*	State
(3,4)	North	A	(3,4,N,A)
(3,4)	West	A	(3,4,W,A)
(2,4)	West	A	(2,4,W,A)
(1,4)	West	A	(1,4,W,A)
(0,4)	West	A	(0,4,W,A)
(0,4)	North	A	(0,4,N,A)
(0,3)	North	A	(0,3,N,A)
(0,2)	North	A	(0,2,N,A)
(0,1)	North	A	(0,1,N,A)
(0,0)	North	A	(0,0,N,A)
(0,0)	North	Home	(0,0,N,H)

Berdasarkan Tabel 4.14, perubahan posisi maupun orientasi menyebabkan terbentuknya *state* yang berbeda selama proses navigasi. Pada awal navigasi, robot berada pada state (3,4,N,A). Setelah melakukan rotasi ke arah kiri, orientasi

berubah menjadi West tanpa mengubah posisi sehingga *state* berubah menjadi (3,4,W,A). Selanjutnya, setiap perpindahan satu grid menghasilkan perubahan koordinat posisi hingga robot mencapai *Goal* A pada koordinat (0,0). Setelah tujuan tercapai, sistem secara otomatis mengubah tujuan menjadi Home, sehingga *state* berubah menjadi (0,0,N,H) meskipun posisi dan orientasi robot tetap sama.

Hasil tersebut menunjukkan bahwa *state* yang dibentuk mampu merepresentasikan kondisi aktual robot berdasarkan kombinasi posisi, orientasi, dan tujuan navigasi sesuai dengan representasi S = (X,Y,D,G).

4.3.3 Pengujian Sinkronisasi State

Setelah state robot berhasil dibentuk berdasarkan informasi posisi, orientasi, dan tujuan navigasi, langkah berikutnya adalah melakukan sinkronisasi state antara Raspberry Pi dan ATmega2560 selama proses navigasi. Pada penelitian ini, Raspberry Pi berperan sebagai *High Level Controller* yang membentuk state, melakukan pencarian aksi pada Q-table, serta mengirimkan perintah aksi kepada ATmega2560 melalui komunikasi serial. Selanjutnya, ATmega2560 berfungsi sebagai *Low Level Controller* yang mengeksekusi aksi tersebut melalui pengendalian motor, pembacaan encoder, dan IMU. Setelah aksi selesai dijalankan, ATmega2560 mengirimkan konfirmasi kepada Raspberry Pi sehingga sistem dapat memperbarui state dan menentukan aksi berikutnya.

Pengujian ini bertujuan untuk memastikan bahwa proses sinkronisasi state berlangsung secara konsisten sehingga setiap aksi yang dijalankan robot selalu sesuai dengan state terbaru yang terbentuk. Hasil pengujian diperoleh dari data *event log* yang merekam proses pertukaran data antara Raspberry Pi dan ATmega2560 selama robot bergerak menuju *Goal* A.

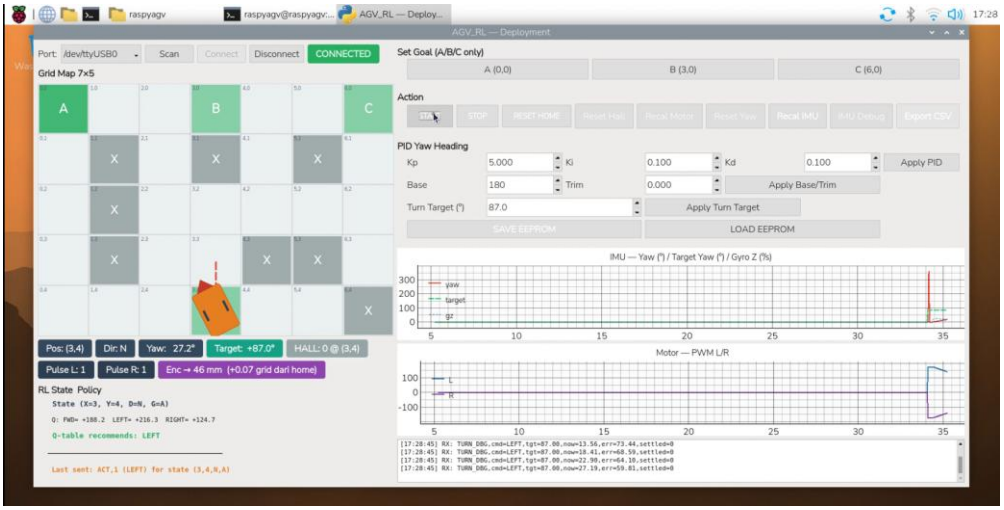
Tabel 4.15. Pengujian Sinkronisasi State antara Raspberry Pi dan ATmega2560

State Saat Ini	Waktu TX (s)	Perintah Raspberry Pi	Waktu RX (s)	Respons ATmega2560	State Berikutnya
3,4,N,A	0,109	ACT,1 (*LEFT*)	0,214	OK:ACT	3,4,W,A
3,4,W,A	6,065	ACT,0 (FWD)	6,213	OK:ACT	2,4,W,A
2,4,W,A	35,589	ACT,0 (FWD)	35,691	OK:ACT	1,4,W,A
1,4,W,A	63,612	ACT,0 (FWD)	63,723	OK:ACT	0,4,W,A

0,4,W,A	83,116	ACT,2 (*RIGHT*)	83,241	OK:ACT	0,4,N,A
0,4,N,A	89,454	ACT,0 (FWD)	89,573	OK:ACT	0,3,N,A
0,3,N,A	95,872	ACT,0 (FWD)	95,988	OK:ACT	0,2,N,A
0,2,N,A	102,341	ACT,0 (FWD)	102,460	OK:ACT	0,1,N,A
0,1,N,A	109,217	ACT,0 (FWD)	109,336	OK:ACT	0,0,N,A)
0,0,N,A	116,884	ACT,3 (STOP)	116,997	OK:STOP	*Goal* A

Berdasarkan Tabel 4.15, Raspberry Pi mengirimkan satu perintah aksi untuk setiap state yang terbentuk, kemudian ATmega2560 memberikan respons OK:ACT sebagai tanda bahwa perintah telah diterima dan siap dieksekusi. Setelah aksi selesai dijalankan, Raspberry Pi memperbarui state berdasarkan hasil pembacaan encoder, IMU, serta tujuan navigasi yang aktif. Dengan demikian, state berikutnya selalu dibentuk berdasarkan kondisi aktual robot setelah aksi sebelumnya selesai dilaksanakan.

Pada state awal (3,4,N,A), Raspberry Pi mengirimkan perintah ACT,1 (*LEFT*) untuk mengubah orientasi robot menuju arah barat. Setelah aksi tersebut selesai, state diperbarui menjadi (3,4,W,A). Selanjutnya Raspberry Pi mengirimkan beberapa perintah ACT,0 (FWD) sehingga robot berpindah secara berturut-turut dari koordinat (3,4) menuju (2,4), (1,4), dan (0,4). Ketika robot mencapai state (0,4,W,A), Raspberry Pi mengirimkan perintah ACT,2 (*RIGHT*) untuk mengubah orientasi robot kembali menghadap utara. Setelah orientasi sesuai, robot kembali menerima perintah *Forward* hingga mencapai state (0,0,N,A). Pada state tujuan tersebut, Raspberry Pi mengirimkan perintah ACT,3 (STOP) sebagai penanda bahwa *Goal* A telah berhasil dicapai.



Gambar 4.23. Log Raspberry Pi

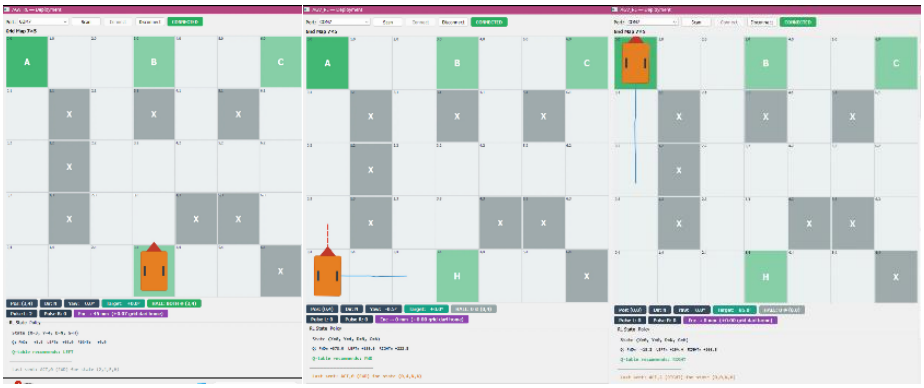
Hasil pengujian juga menunjukkan bahwa komunikasi antara Raspberry Pi dan ATmega2560 berlangsung secara dua arah (*closed-loop*). Raspberry Pi tidak mengirimkan seluruh rangkaian aksi sekaligus, melainkan menunggu konfirmasi dari ATmega2560 sebelum memperbarui state dan menentukan aksi berikutnya. Mekanisme tersebut memastikan bahwa setiap transisi state selalu didasarkan pada kondisi aktual robot setelah aksi sebelumnya selesai dieksekusi, sehingga mengurangi kemungkinan terjadinya ketidaksesuaian antara state pada Raspberry Pi dan kondisi fisik robot.

Selain itu, berdasarkan *timestamp* pada log komunikasi, selisih waktu antara data dikirim oleh Raspberry Pi dan diterima oleh ATmega2560 berada pada kisaran 100–150 ms. Waktu tersebut relatif kecil dibandingkan waktu eksekusi setiap aksi sehingga komunikasi serial tidak menjadi hambatan dalam proses navigasi. Dengan demikian, mekanisme sinkronisasi state melalui komunikasi dua arah mampu mendukung implementasi algoritma Q-*Learning* pada robot mobile secara real-time.

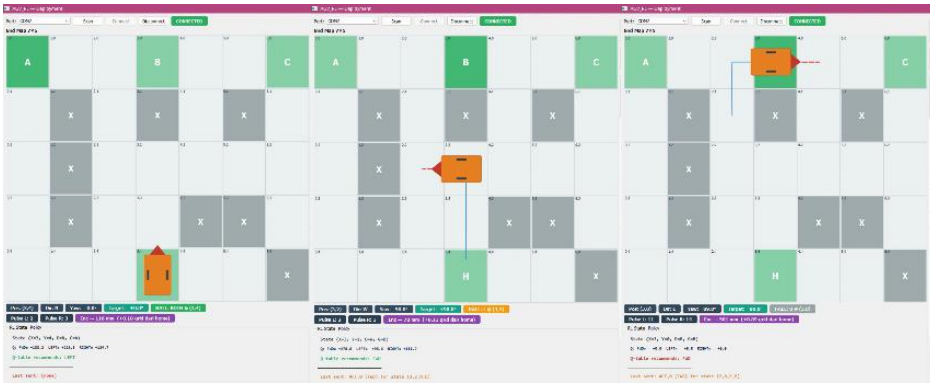
Berdasarkan keseluruhan hasil pengujian dapat disimpulkan bahwa proses sinkronisasi state antara Raspberry Pi dan ATmega2560 telah berjalan dengan baik. Setiap state berhasil menghasilkan aksi yang sesuai, setiap aksi berhasil diterima dan dieksekusi oleh ATmega2560, serta state berikutnya selalu diperbarui berdasarkan kondisi aktual robot. Hasil tersebut menunjukkan bahwa mekanisme sinkronisasi state mampu mendukung proses pengambilan keputusan berbasis Q-*Learning* secara konsisten selama robot melakukan navigasi menuju tujuan.

4.4 Pengujian Navigasi Robot Mobile

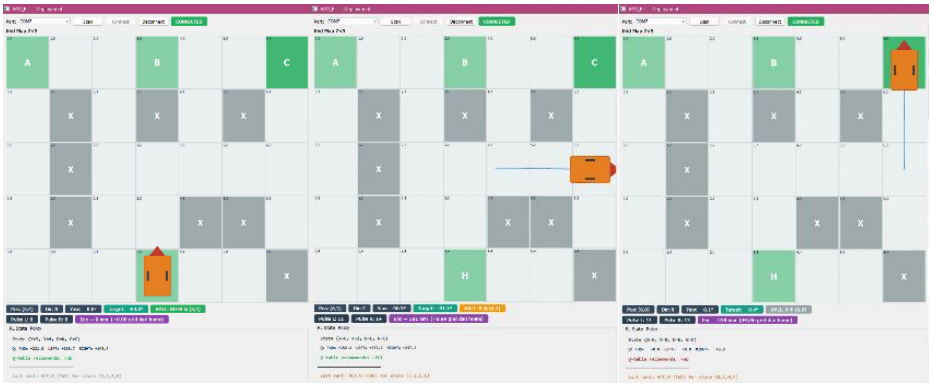
Pengujian navigasi robot mobile dilakukan untuk mengevaluasi implementasi *policy* hasil pembelajaran algoritma Q-*Learning* pada robot nyata. Sebelum dilakukan pengujian, *policy* hasil pembelajaran terlebih dahulu divisualisasikan menggunakan *Graphical User Interface* (GUI) yang dikembangkan pada penelitian ini. GUI menampilkan lintasan navigasi yang dihasilkan berdasarkan hasil *lookup* pada Q-Table untuk masing-masing tujuan, yaitu *Goal* A, *Goal* B, dan *Goal* C. Setiap lintasan menunjukkan urutan arah gerak (*direction*), perpindahan antargrid, serta perubahan orientasi robot yang akan digunakan sebagai acuan dalam pengujian pada robot nyata.



Gambar 4.24. Visualisasi Policy Hasil *Training* Menuju *Goal* A



Gambar 4.25. Visualisasi Policy Hasil *Training* Menuju *Goal* B



Gambar 4.26. Visualisasi Policy Hasil *Training* Menuju *Goal* C

Berdasarkan Gambar 4.24 hingga Gambar 4.26, setiap lintasan menunjukkan urutan aksi yang harus dijalankan robot untuk mencapai tujuan berdasarkan *policy* hasil pembelajaran algoritma Q-*Learning*. Perpindahan dari satu grid ke grid berikutnya merepresentasikan satu aksi *Forward* dengan target perpindahan 650 mm, sedangkan setiap aksi *Turn Left* dan *Turn Right* merepresentasikan perubahan orientasi sebesar 90°. Dengan demikian, visualisasi GUI digunakan sebagai acuan untuk mengevaluasi dua aspek implementasi navigasi pada robot nyata, yaitu kesesuaian urutan aksi terhadap *policy* dan kesesuaian posisi robot terhadap target navigasi.

Berdasarkan acuan tersebut, pengujian navigasi dibagi menjadi dua bagian. Pengujian pertama dilakukan untuk membandingkan urutan aksi yang dijalankan robot dengan *policy* hasil pembelajaran, sedangkan pengujian kedua dilakukan untuk mengevaluasi kesesuaian perpindahan posisi dan perubahan orientasi robot terhadap target navigasi berdasarkan data sensor dan hasil pengukuran aktual.

4.4.1 Pengujian Navigasi Berdasarkan *Policy*

Pengujian kesesuaian *policy* hasil *training* dilakukan untuk mengevaluasi apakah urutan aksi (*action sequence*) yang dihasilkan oleh algoritma Q-*Learning* pada tahap simulasi dapat diimplementasikan dengan benar pada robot nyata. Policy hasil pembelajaran yang telah divisualisasikan pada Gambar 4.24 hingga Gambar 4.26 digunakan sebagai acuan dalam pengujian ini. Setiap lintasan pada GUI menunjukkan urutan aksi *Forward* (FWD), *Turn Left* (*LEFT*), dan *Turn Right* (*RIGHT*) yang diperoleh dari hasil *lookup* pada Q-Table untuk masing-masing tujuan navigasi.

Perlu diketahui bahwa data telemetri direkam secara kontinu selama robot bergerak sehingga satu aksi dapat muncul pada beberapa sampel berturut-turut. Oleh karena itu, pada pengujian ini hanya digunakan perubahan *state* yang ditandai oleh perubahan koordinat grid, orientasi robot, atau aksi yang dijalankan. Dengan demikian, urutan aksi yang diperoleh merepresentasikan *policy* navigasi yang sebenarnya dijalankan robot.

a) Navigasi *Goal* A

Hasil implementasi navigasi menuju *Goal* A dibandingkan dengan policy yang ditunjukkan pada Gambar 4.24 dan disajikan pada Tabel 4.16.

Tabel 4.16. Hasil Navigasi Berdasarkan *Policy Goal* A

State (Grid, Arah)	Policy Hasil *Training*	*Action* Robot *Trial* 1	*Action* Robot *Trial* 2	Keterangan
(3,4,N)	*LEFT*	*LEFT*	*LEFT*	Berhasil
(3,4,W)	FWD	FWD	FWD	Berhasil
(2,4,W)	FWD	FWD	FWD	Berhasil
(1,4,W)	FWD	FWD	FWD	Berhasil
(0,4,W)	*RIGHT*	*RIGHT*	*RIGHT*	Berhasil
(0,4,N)	FWD	FWD	FWD	Berhasil
(0,3,N)	FWD	FWD	FWD	Berhasil
(0,2,N)	FWD	FWD	FWD	Berhasil
(0,1,N)	FWD	FWD	FWD	Berhasil

Berdasarkan Tabel 4.16 terlihat bahwa seluruh aksi yang dijalankan robot pada *Trial* 1 maupun *Trial* 2 identik dengan *policy* hasil *training*. Robot selalu memilih aksi yang sama dengan aksi yang memiliki nilai Q terbesar pada setiap *state*. Hasil tersebut menunjukkan bahwa implementasi *lookup* terhadap Q-table pada robot nyata telah berjalan dengan baik sehingga proses pengambilan keputusan tidak mengalami perubahan dibandingkan hasil simulasi.

b) Navigasi *Goal* B

Hasil implementasi navigasi menuju *Goal* B dibandingkan dengan policy pada Gambar 4.25 dan disajikan pada Tabel 4.17.

Tabel 4.17. Hasil Navigasi Berdasarkan *Policy Goal* B

State (Grid, Arah)	Policy Hasil *Training*	*Action* Robot *Trial* 1	*Action* Robot *Trial* 2	Keterangan
(3,4,N)	FWD	FWD	FWD	Berhasil
(3,3,N)	FWD	FWD	FWD	Berhasil
(3,2,N)	*LEFT*	*LEFT*	*LEFT*	Berhasil
(3,2,W)	FWD	FWD	FWD	Berhasil
(2,2,W)	*RIGHT*	*RIGHT*	*RIGHT*	Berhasil
(2,2,N)	FWD	FWD	FWD	Berhasil
(2,1,N)	FWD	FWD	FWD	Berhasil
(2,0,N)	*RIGHT*	*RIGHT*	*RIGHT*	Berhasil
(2,0,E)	FWD	FWD	FWD	Berhasil

Berdasarkan hasil pengujian *Goal* B, urutan aksi yang dijalankan robot pada kedua percobaan juga sesuai dengan *policy* hasil pembelajaran. Tidak ditemukan perubahan aksi maupun pemilihan jalur yang berbeda selama robot melakukan navigasi menuju tujuan. Hal tersebut menunjukkan bahwa implementasi algoritma Q-*Learning* mampu menghasilkan keputusan navigasi yang konsisten pada robot nyata.

c) Navigasi C

Hasil perbandingan *policy* hasil *training* dengan implementasi pada robot nyata pada *goal* A ditunjukkan pada Tabel 4.18.

Tabel 4.18. Hasil Navigasi Berdasarkan *Policy Goal* C

State (Grid, Arah)	Policy	*Trial* 1	*Trial* 2	Keterangan
(3,4,N)	FWD	FWD	FWD	Berhasil
(3,3,N)	FWD	FWD	FWD	Berhasil
(3,2,N)	*RIGHT*	*RIGHT*	*RIGHT*	Berhasil
(3,2,E)	FWD	FWD	FWD	Berhasil
(4,2,E)	FWD	FWD	FWD	Berhasil
(5,2,E)	FWD	FWD	FWD	Berhasil

(6,2,E)	*LEFT*	*LEFT*	*LEFT*	Berhasil
(6,2,N)	FWD	FWD	FWD	Berhasil
(6,1,N)	FWD	FWD	FWD	Berhasil
(6,0,N)	*LEFT*	*LEFT*	*LEFT*	Berhasil
(6,0,W)	FWD	FWD	FWD	Berhasil

Hasil pengujian *Goal* C menunjukkan bahwa robot mampu mengikuti seluruh urutan aksi sesuai *policy* hasil *training*. Aksi *RIGHT*, *LEFT*, maupun FWD dieksekusi pada *state* yang sama seperti hasil simulasi sehingga lintasan navigasi yang ditempuh robot identik dengan lintasan yang diperoleh pada tahap pembelajaran.

4.4.2 Pengujian Kesesuaian Posisi Robot terhadap Target Navigasi

Setelah kesesuaian *policy* hasil pembelajaran berhasil dibuktikan pada Subbab 4.4.1, pengujian selanjutnya dilakukan untuk mengevaluasi kesesuaian posisi robot terhadap target navigasi pada lingkungan nyata. Pengujian ini bertujuan untuk mengetahui apakah implementasi setiap aksi yang dihasilkan oleh algoritma Q-*Learning* mampu menghasilkan perpindahan posisi robot sesuai dengan target lintasan hingga mencapai tujuan.

Evaluasi dilakukan dengan membandingkan data telemetri yang diperoleh dari Hall Effect Encoder dan MPU6050 terhadap hasil pengukuran aktual pada arena pengujian. Hall Effect Encoder digunakan untuk mengukur perpindahan robot pada setiap aksi *Forward*, sedangkan MPU6050 digunakan untuk mengukur perubahan orientasi ketika robot melakukan aksi *Left* dan *Right*. Selanjutnya, hasil pembacaan sensor dibandingkan dengan kondisi aktual robot menggunakan pengukuran manual sehingga dapat diketahui tingkat kesesuaian implementasi navigasi pada robot nyata.

a) Navigasi *Goal* A

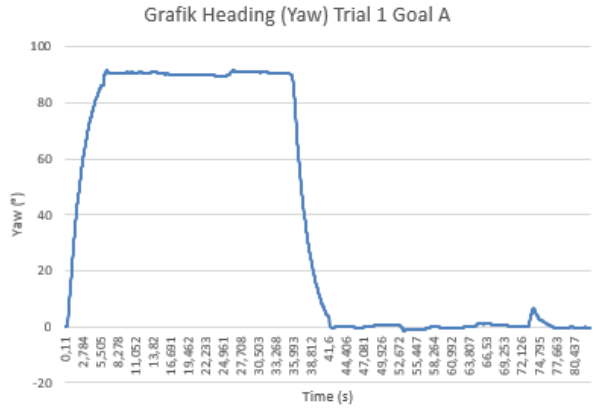
Pada *Trial* 1, robot memulai navigasi dari posisi Home (3,4) menuju *Goal* A (0,0) menggunakan *policy* hasil pembelajaran yang sama seperti pada simulasi.

Seluruh aksi berhasil dijalankan sesuai urutan yang telah ditentukan hingga robot mencapai *Goal* A. Dokumentasi hasil pengujian ditunjukkan pada Gambar 4.27.

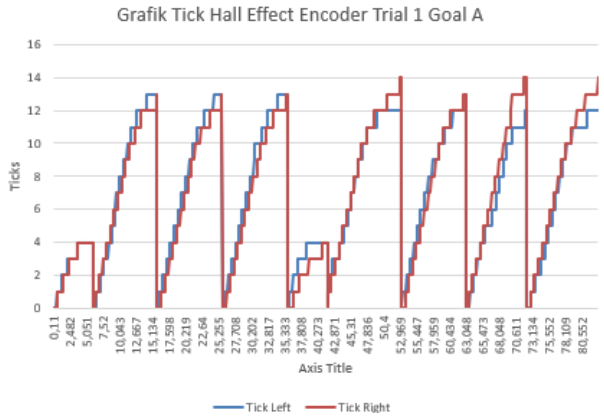
Gambar 4.27. Robot Berhasil Mencapai *Goal* A *Trial* 1

Tabel 4.19. Perbandingan Data Sensor dan Hasil Aktual *Trial* 1 *Goal* A

Grid	*Action*	Tick Sensor	Jarak Sensor (mm)	Jarak Aktual (mm)	Heading Sensor (Awal–Akhir)	Heading Aktual (Awal–Akhir)
(3,4)	*LEFT*	0 → 4	–	–	0,0° → 86,2°	0° → 88°
(3,4) → (2,4)	FWD	0 → 14	651	654	87,3° → 89,8°	88° → 90°
(2,4) → (1,4)	FWD	0 → 13	604,5	650	89,8° → 90,4°	90° → 90°
(1,4) → (0,4)	FWD	0 → 14	651	653	90,4° → 90,1°	90° → 91°
(0,4)	*RIGHT*	0 → 4	–	–	90,1° → 2,4°	91° → 0°
(0,4) → (0,3)	FWD	0 → 14	651	652	2,4° → 0,5°	0° → 0°
(0,3) → (0,2)	FWD	0 → 13	604,5	651	0,5° → −0,2°	0° → 0°
(0,2) → (0,1)	FWD	0 → 14	651	652	−0,2° → 0,8°	0° → 1°
(0,1) → (0,0)	FWD	0 → 14	651	655	0,8° → 0,4°	1° → 0°



Gambar 4.28. Grafik Heading (Yaw) *Trial* 1 *Goal* A



Gambar 4.29. Grafik Heading (Yaw) *Trial* 1 *Goal* A

Berdasarkan Tabel 4.19, setiap aksi yang dijalankan robot menghasilkan perubahan orientasi dan perpindahan posisi sesuai dengan target navigasi. Pada aksi *Left*, heading IMU berubah dari 0,0° menjadi 86,2°, sedangkan pengukuran aktual menunjukkan orientasi sekitar 88°, sehingga selisih yang terjadi masih relatif kecil. Selama aksi *Forward*, Hall Effect Encoder mencatat perpindahan sebesar 13–14 tick pada setiap perpindahan grid. Meskipun beberapa pengujian hanya menghasilkan 13 tick, robot tetap mampu menempuh jarak mendekati 650 mm akibat adanya momentum roda setelah motor dihentikan.

Grafik heading menunjukkan bahwa orientasi robot tetap stabil selama proses perpindahan. Penyimpangan heading relatif kecil sehingga pengendali PID mampu mempertahankan arah gerak robot sesuai heading target. Akibatnya, robot berhasil mencapai *Goal* A dengan posisi akhir yang masih berada di sekitar pusat grid. Berdasarkan pengamatan langsung, posisi robot hanya mengalami sedikit pergeseran ke arah kanan, namun belum memengaruhi keberhasilan navigasi maupun identifikasi posisi tujuan.

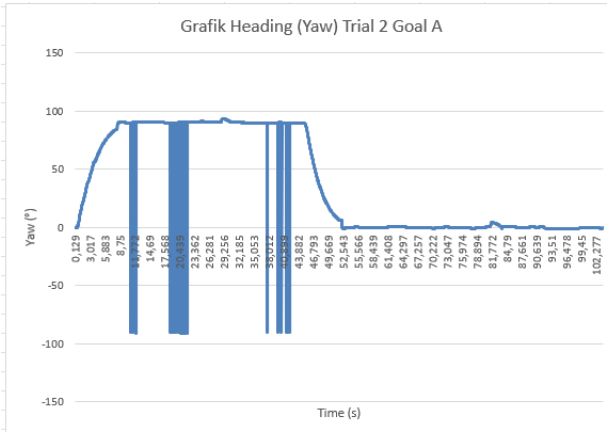
Pengujian *Trial* 2 dilakukan menggunakan *policy* hasil pembelajaran yang sama dengan *Trial* 1. Berdasarkan urutan aksi yang dijalankan, robot tetap berhasil mengikuti lintasan hasil algoritma Q-*Learning* dan mencapai *Goal* A. Akan tetapi, kondisi implementasi menunjukkan adanya penyimpangan posisi yang lebih besar dibandingkan *Trial* 1.

Gambar 4.30. Robot Mencapai *Goal* A *Trial* 2

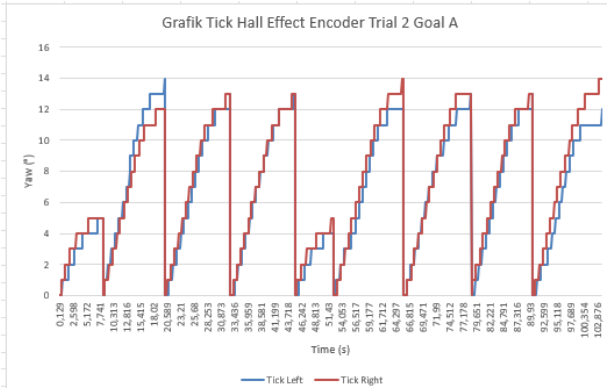
Tabel 4.20. Perbandingan Data Sensor dan Hasil Aktual *Trial* 2 *Goal* A

Grid	*Action*	Tick Sensor	Jarak Sensor (mm)	Jarak Aktual (mm)	Heading Sensor (Awal–Akhir)	Heading Aktual (Awal–Akhir)
(3,4)	*LEFT*	0 → 5	–	–	0,0° → 80,0°	0° → 82°
(3,4) → (2,4)	FWD	0 → 14	651	676	80,0° → 91,0°	82° → 93°
(2,4) → (1,4)	FWD	0 → 14	651	682	91,0° → 92,4°	93° → 94°
(1,4) → (0,4)	FWD	0 → 14	651	689	92,4° → 91,2°	94° → 93°
(0,4)	*RIGHT*	0 → 5	–	–	91,2° → 5,9°	93° → 8°
(0,4) → (0,3)	FWD	0 → 14	651	691	5,9° → 2,8°	8° → 5°
(0,3) → (0,2)	FWD	0 → 14	651	694	2,8° → 2,1°	5° → 4°
(0,2) → (0,1)	FWD	0 → 14	651	696	2,1° → 3,0°	4° → 3°

(0,1) → (0,0)	FWD	0 → 14	651	698	3,0° → 2,4°	3° → 2°



Gambar 4.31. Grafik Heading (Yaw) *Trial* 1 *Goal* A



Gambar 4.32. Grafik Heading (Yaw) *Trial* 1 *Goal* A

Berdasarkan urutan aksi yang dijalankan, robot tetap mengikuti *policy* hasil pembelajaran tanpa terjadi perubahan lintasan. Akan tetapi, grafik heading menunjukkan adanya fluktuasi pembacaan MPU6050 yang muncul secara tiba-tiba di tengah proses navigasi, meskipun robot sedang bergerak lurus dan tidak melakukan perubahan orientasi yang signifikan. Fluktuasi tersebut menyebabkan nilai heading yang digunakan sebagai umpan balik pengendali PID berubah secara tidak stabil.

Fluktuasi pembacaan heading menyebabkan sistem melakukan koreksi arah yang sebenarnya tidak diperlukan sehingga orientasi robot menyimpang secara bertahap dari heading target. Akibatnya, pada aksi *Forward* berikutnya arah gerak robot tidak lagi sejajar dengan lintasan yang direncanakan. Meskipun Hall Effect Encoder tetap mencatat perpindahan sekitar 14 tick setiap grid, jarak aktual yang

ditempuh mencapai 676–698 mm, lebih besar dari target 650 mm. Hal ini menunjukkan bahwa deviasi lintasan disebabkan oleh akumulasi kesalahan orientasi, bukan kesalahan pembacaan encoder. Robot tetap berhasil mencapai Grid *Goal* A, namun berhenti mendekati batas antara Grid (0,0) dan Grid (0,1). Dengan demikian, penyimpangan posisi pada *Trial* 2 terutama disebabkan oleh fluktuasi heading MPU6050 yang memengaruhi koreksi arah, sementara *policy* hasil Q-Learning tetap dijalankan dengan benar.

b) Navigasi *Goal* B

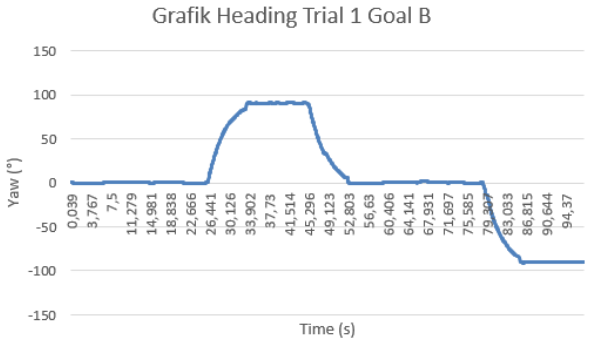
Pada *Trial* 1, robot memulai navigasi dari posisi Home (3,4) menuju *Goal* B (3,0) menggunakan *policy* hasil pembelajaran algoritma Q-*Learning*. Berdasarkan *policy* tersebut, robot bergerak lurus sebanyak dua grid, kemudian melakukan manuver *Left*, dilanjutkan dengan satu aksi *Forward*, satu aksi *Right*, dua aksi *Forward*, satu aksi *Right*, dan satu aksi *Forward* hingga mencapai *Goal* B. Seluruh urutan aksi tersebut sesuai dengan hasil pembelajaran yang telah dibahas pada Subbab 4.4.1.

Gambar 4.33. Robot Mencapai *Goal* B *Trial* 1

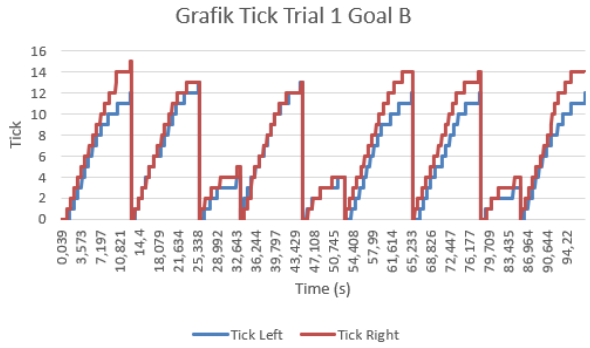
Tabel 4.21. Perbandingan Data Sensor dan Hasil Aktual *Trial* 1 *Goal* B

Grid	*Action*	Tick Sensor	Jarak Sensor (mm)	Jarak Aktual (mm)	Heading Sensor (Awal–Akhir)	Heading Aktual (Awal–Akhir)
(3,4)→ (3,3)	FWD	0→14	651	653	0,0°→0,4°	0°→0°
(3,3)→ (3,2)	FWD	0→13	604,5	649	0,4°→0,8°	0°→1°
(3,2)	*LEFT*	0→4	-	-	0,8°→87,6°	1°→89°

(3,2)→ (2,2)	FWD	0→14	651	652	87,6°→89,9°	89°→90°
(2,2)	*RIGHT*	0→4	-	-	89,9°→1,8°	90°→0°
(2,2)→ (2,1)	FWD	0→14	651	651	1,8°→0,5°	0°→0°
(2,1)→ (2,0)	FWD	0→13	604,5	650	0,5°→0,3°	0°→0°
(2,0)	*RIGHT*	0→4	-	-	0,3°→-90°	0°→270°
(2,0)→ (3,0)	FWD	0→14	651	654	-90°→-90,5°	270°→270°



Gambar 4.34. Grafik Heading *Trial* 1 *Goal* B



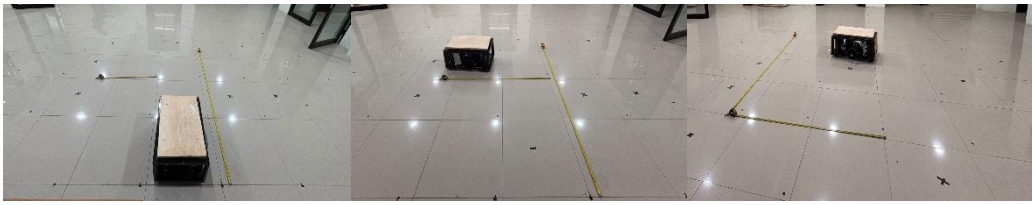
Gambar 4.35. Grafik Tick *Trial* 1 *Goal* B

Berdasarkan Tabel 4.21, robot berhasil mencapai *Goal* B dengan penyimpangan jarak dan sudut yang relatif kecil. Hasil pengukuran menunjukkan bahwa jarak perpindahan aktual pada setiap aksi *Forward* berada pada kisaran 649–654 mm, sehingga error maksimum terhadap target perpindahan 650 mm hanya sekitar 0,62%. Sementara itu, perubahan orientasi pada setiap aksi *Left* dan *Right*

menghasilkan penyimpangan sekitar 1–3° atau setara dengan 1,1–3,3% terhadap sudut target.

Nilai error tersebut masih berada dalam batas toleransi sistem sehingga tidak menyebabkan penyimpangan lintasan yang signifikan. Akibatnya, robot mampu mempertahankan arah navigasi dengan baik dan berhenti mendekati pusat *Goal* B sesuai dengan lintasan hasil *policy* Q-*Learning*.

Pada *Trial* 2, robot kembali menjalankan urutan aksi yang sama dengan *Trial* 1 sehingga secara algoritma implementasi *policy* tetap berjalan dengan benar. Namun, selama proses navigasi terlihat adanya fluktuasi pembacaan heading MPU6050 pada beberapa aksi *Forward*, sehingga orientasi robot tidak selalu kembali tepat pada heading target setelah melakukan manuver belok.

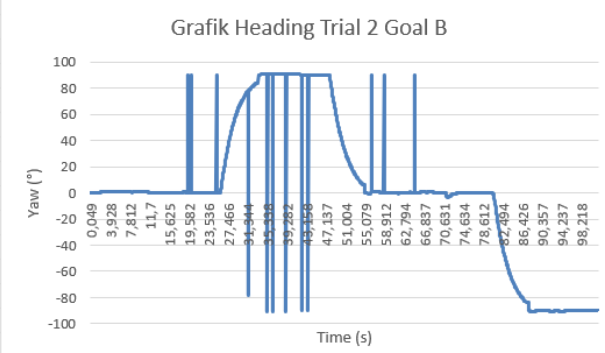


Gambar 4.36. Robot Mencapai *Goal* B *Trial* 2

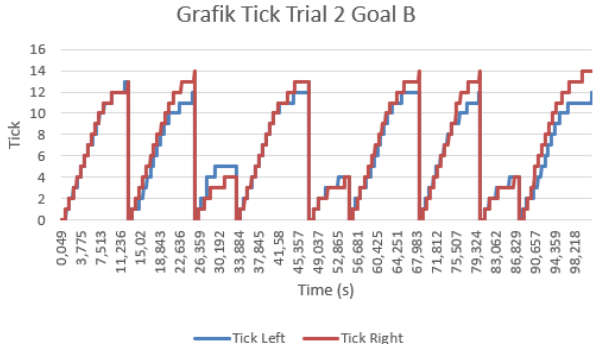
Tabel 4.22. Perbandingan Data Sensor dan Hasil Aktual *Trial* 2 *Goal* B

Grid	*Action*	Tick Sensor	Jarak Sensor (mm)	Jarak Aktual (mm)	Heading Sensor (Awal–Akhir)	Heading Aktual (Awal–Akhir)
(3,4)→ (3,3)	FWD	0→14	651	675	0,0°→1,5°	0°→2°
(3,3)→ (3,2)	FWD	0→14	651	680	1,5°→2,8°	2°→4°
(3,2)	*LEFT*	0→5	-	-	2,8°→81,4°	4°→83°
(3,2)→ (2,2)	FWD	0→14	651	687	81,4°→90,9°	83°→92°
(2,2)	*RIGHT*	0→5	-	-	90,9°→7,6°	92°→8°
(2,2)→ (2,1)	FWD	0→14	651	691	7,6°→3,1°	8°→5°

(2,1)→ (2,0)	FWD	0→14	651	694	3,1°→2,8°	5°→4°
(2,0)	*RIGHT*	0→5	-	-	2,8°→-90°	4°→280°
(2,0)→ (3,0)	FWD	0→14	651	698	-90°→ -91,1°	280°→ 278°



Gambar 4.37. Grafik Heading *Trial* 2 *Goal* B



Gambar 4.38. Grafik Heading *Trial* 2 *Goal* B

Berdasarkan Tabel 4.22, robot tetap berhasil mencapai *Goal* B dan menjalankan seluruh urutan aksi sesuai *policy* hasil pembelajaran algoritma Q-*Learning*. Namun demikian, hasil pengukuran menunjukkan bahwa jarak perpindahan aktual pada setiap aksi *Forward* berada pada kisaran 676–698 mm, sehingga diperoleh error jarak sekitar 4,0–7,4% terhadap target perpindahan 650 mm. Selain itu, perubahan orientasi pada beberapa aksi *Left* dan *Right* juga menunjukkan penyimpangan sekitar 8–10° terhadap sudut target.

Berdasarkan Gambar 4.37, nilai *heading* MPU6050 mengalami fluktuasi saat robot melakukan aksi *Forward*. Fluktuasi tersebut menyebabkan pengendali PID memberikan koreksi arah yang tidak diperlukan sehingga orientasi robot

menyimpang secara bertahap selama perpindahan antargrid. Akibatnya, lintasan aktual bergeser dari lintasan yang direncanakan. Meskipun Hall Effect Encoder tetap mencatat perpindahan sekitar 13–14 *tick* pada setiap aksi *Forward*, posisi robot tidak berhenti tepat di pusat setiap grid. Saat mencapai *Goal* B, posisi akhir robot mengalami deviasi beberapa sentimeter dari pusat grid tujuan, meskipun urutan aksi yang dijalankan tetap sesuai dengan *policy* hasil algoritma Q-Learning.

c) Navigasi *Goal* C

Pada *Trial* 1, robot memulai navigasi dari posisi Home (3,4) menuju *Goal* C (6,0) menggunakan *policy* hasil algoritma Q-*Learning*. Berdasarkan *policy* tersebut, robot melakukan aksi hingga mencapai *Goal* C. Seluruh aksi dieksekusi berdasarkan hasil *lookup* pada *Q-Table* yang telah dibahas pada Subbab 4.4.1.

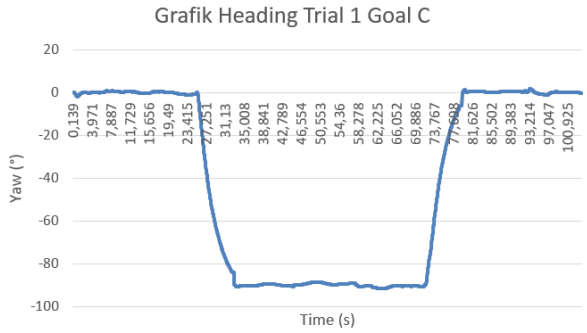
Untuk mengevaluasi kesesuaian implementasi navigasi, dilakukan perbandingan antara data telemetri yang diperoleh dari Hall Effect Encoder dan MPU6050 dengan hasil pengukuran aktual pada arena pengujian. Hall Effect Encoder digunakan untuk mengetahui perpindahan robot pada setiap aksi *Forward*, sedangkan MPU6050 digunakan untuk mengevaluasi perubahan orientasi pada aksi *Right* dan *Left*. Dokumentasi hasil pengujian ditunjukkan pada Gambar 4.39.

Gambar 4.39. Robot Berhasil Mencapai *Goal* C *Trial* 1

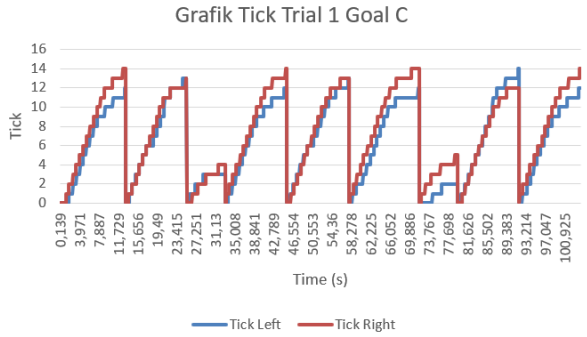
Tabel 4.23. Perbandingan Data Sensor dan Hasil Aktual *Trial* 1 *Goal* C

Grid	*Action*	Tick Sensor (Awal–Akhir)	Jarak Sensor (mm)	Jarak Aktual (mm)	Heading Sensor (Awal–Akhir)	Heading Aktual (Awal–Akhir)
(3,4) → (3,3)	FWD	0 → 14	651	652	0,0 → -0,6°	0° → 0°
(3,3) → (3,2)	FWD	0 → 13	604,5	649	-0,6°→ -0,8°	0° → -1°
(3,2)	*RIGHT*	0 → 4	–	–	-0,8°→ -89,4°	0° → 270°

(3,2) → (4,2)	FWD	0 → 14	651	653	-89,4° → -90,2°	270° → 270°
(4,2) → (5,2)	FWD	0 → 14	651	651	-90,2° → -89,8°	270° → 270°
(5,2) → (6,2)	FWD	0 → 13	604,5	650	-89,8°→ -90,4°	270° → 269°
(6,2)	*LEFT*	0 → 4	–	–	-90,4° → -0,5°	269° → 0°
(6,2) → (6,1)	FWD	0 → 14	651	652	-0,5° → 0,4°	0° → 0°
(6,1) → (6,0)	FWD	0 → 14	651	654	0,4° → 0,2°	0° → 1°



Gambar 4.40. Grafik Heading *Trial* 1 *Goal* C

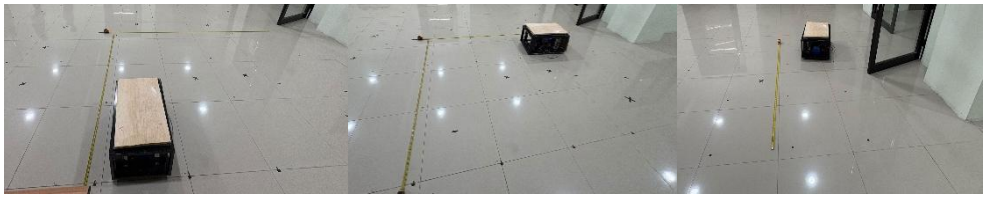


Gambar 4.41. Grafik Tick *Trial* 1 *Goal* C

Berdasarkan Tabel 4.23, robot berhasil mencapai *Goal* C dengan menjalankan seluruh urutan aksi sesuai *policy* hasil pembelajaran algoritma Q-*Learning*. Hasil pengukuran menunjukkan bahwa jarak perpindahan aktual pada setiap aksi *Forward* berada pada kisaran 649–654 mm, sehingga diperoleh error jarak maksimum sekitar 0,62% terhadap target perpindahan 650 mm. Selain itu,

perubahan orientasi pada aksi *Right* dan *Left* menghasilkan penyimpangan sekitar 1–3° terhadap sudut target 90°. Berdasarkan Gambar 4.40, nilai heading selama aksi *Forward* juga terlihat relatif stabil dengan fluktuasi yang kecil, sehingga tidak menyebabkan penyimpangan lintasan yang signifikan. Nilai error jarak dan sudut yang relatif kecil tersebut menunjukkan bahwa robot mampu mempertahankan arah navigasi dengan baik serta berhenti mendekati pusat *Goal* C, sehingga implementasi *policy* hasil algoritma Q-*Learning* pada *Trial* 1 dapat dikatakan berhasil dengan tingkat akurasi yang baik.

Pada *Trial* 2, robot kembali melakukan navigasi dari posisi Home (3,4) menuju *Goal* C (6,0) menggunakan *policy* hasil pembelajaran algoritma Q-*Learning* yang sama seperti pada *Trial* 1. Berdasarkan *policy* tersebut, robot menjalankan aksi hingga mencapai *Goal* C. Meskipun urutan aksi yang dijalankan sama dengan hasil pembelajaran, selama proses navigasi ditemukan adanya fluktuasi pembacaan heading MPU6050 pada beberapa aksi *Forward*. Kondisi tersebut menyebabkan pengendali PID melakukan koreksi arah yang tidak sesuai sehingga akurasi posisi robot mengalami penurunan dibandingkan *Trial* 1. Dokumentasi hasil pengujian ditunjukkan pada Gambar 4.42.

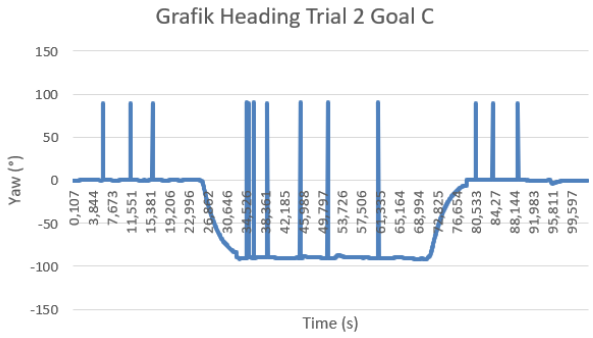


Gambar 4.42. Robot Berhasil Mencapai *Goal* C *Trial* 2

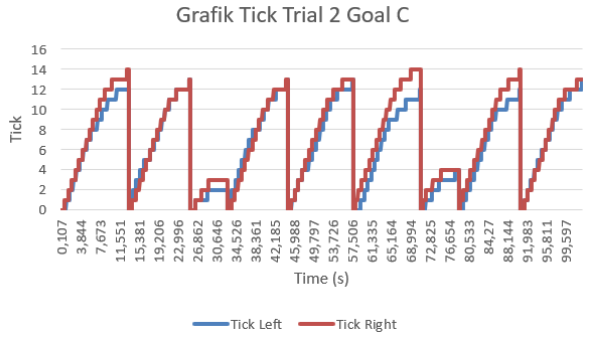
Tabel 4.24. Perbandingan Data Sensor dan Hasil Aktual *Trial* 2 *Goal* C

Grid	*Action*	Tick Sensor (Awal–Akhir)	Jarak Sensor (mm)	Jarak Aktual (mm)	Heading Sensor (Awal–Akhir)	Heading Aktual (Awal–Akhir)
(3,4) → (3,3)	FWD	0 → 14	651	676	0,0° → 0,3°	3° → 5°
(3,3) → (3,2)	FWD	0 → 14	651	684	0,3° → -1,2°	5° → 280°

(3,2)	*RIGHT*	0 → 5	–	–	-1,2° → -87,8°	280° → 276°
(3,2) → (4,2)	FWD	0 → 14	651	690	-87,8° → -91,6°	276° → 273°
(4,2) → (5,2)	FWD	0 → 14	651	694	-91,6° → -88,5°	273° → 270°
(5,2) → (6,2)	FWD	0 → 14	651	697	-88,5° → -92,7°	270° → 358°
(6,2)	*LEFT*	0 → 5	–	–	-92,7° → -2,1°	358° → 3°
(6,2) → (6,1)	FWD	0 → 14	651	693	-2,1° → 1,8°	3° → 5°
(6,1) → (6,0)	FWD	0 → 14	651	698	1,8° → 3,9°	3° → 5°



Gambar 4.43. Grafik Heading *Trial* 2 *Goal* C



Gambar 4.44. Grafik Tick *Trial* 2 *Goal* C

Berdasarkan Tabel 4.24, robot tetap berhasil mencapai *Goal* C dan menjalankan seluruh urutan aksi sesuai *policy* hasil pembelajaran algoritma Q-*Learning*. Namun, hasil pengukuran menunjukkan bahwa jarak perpindahan aktual

pada setiap aksi *Forward* berada pada kisaran 676–698 mm, sehingga diperoleh error jarak sekitar 4,0–7,4% terhadap target perpindahan 650 mm. Selain itu, perubahan orientasi pada aksi *Right* dan *Left* menunjukkan penyimpangan sekitar 8–10° terhadap sudut target. Berdasarkan Gambar 4.43, terlihat bahwa nilai heading mengalami fluktuasi ketika robot sedang melakukan aksi *Forward*. Seharusnya heading relatif konstan selama robot bergerak lurus, namun pada grafik terlihat beberapa perubahan heading yang terjadi secara tiba-tiba tanpa adanya aksi belok. Fluktuasi tersebut menyebabkan pengendali PID melakukan koreksi arah yang tidak diperlukan sehingga orientasi robot berubah secara bertahap selama proses navigasi. Akumulasi penyimpangan heading tersebut menyebabkan lintasan aktual robot bergeser dari lintasan yang direncanakan. Meskipun Hall Effect Encoder tetap menunjukkan pembacaan sekitar 13–14 tick pada setiap perpindahan grid, posisi akhir robot tidak berhenti tepat pada pusat *Goal* C, melainkan sedikit bergeser dari titik target. Hasil ini menunjukkan bahwa penyimpangan posisi pada *Trial* 2 lebih dipengaruhi oleh fluktuasi pembacaan sensor MPU6050 dibandingkan kesalahan implementasi *policy* hasil algoritma Q-*Learning*.

Berdasarkan hasil pengujian pada *Goal* A, *Goal* B, dan *Goal* C, robot mampu mengimplementasikan *policy* hasil pembelajaran algoritma Q-*Learning* secara konsisten pada lingkungan nyata. Seluruh urutan aksi yang dijalankan sesuai dengan hasil *Q-Table*, sehingga robot berhasil mencapai tujuan pada setiap percobaan. Meskipun demikian, pada beberapa *Trial* 2 terlihat peningkatan error jarak dan heading akibat fluktuasi pembacaan MPU6050 selama aksi *Forward*. Kondisi tersebut menyebabkan posisi akhir robot sedikit bergeser dari pusat grid tujuan, namun tidak memengaruhi keberhasilan navigasi secara keseluruhan. Hasil ini menunjukkan bahwa perbedaan antara hasil simulasi dan implementasi nyata lebih dipengaruhi oleh keterbatasan sistem sensor dan kendali dibandingkan algoritma Q-*Learning* yang digunakan.

4.5 Pengujian Respon Robot terhadap Obstacle

Pengujian respon robot terhadap obstacle dilakukan untuk mengetahui kemampuan robot mobile dalam merespons keberadaan halangan selama proses navigasi. Pada penelitian ini, deteksi obstacle dilakukan menggunakan sensor proximity yang berfungsi sebagai mekanisme keselamatan (*safety layer*). Informasi

obstacle tidak digunakan sebagai komponen *state* pada algoritma *Q-Learning*, melainkan digunakan untuk mengendalikan eksekusi aksi robot ketika terdapat halangan pada jalur navigasi.

Pengujian dilakukan dengan meletakkan obstacle pada lintasan robot ketika robot bergerak menuju *Goal* B. Selanjutnya diamati perubahan status pembacaan sensor dan respons robot selama obstacle berada pada area deteksi sensor hingga obstacle dipindahkan.

Tabel 4.25. Hasil Pengujian Terhadap Obstacle

Posisi Grid	*Goal*	Status Obstacle	Respon Robot	Hasil
(3,4)	B	0 (Tidak terdeteksi)	Robot bergerak menuju tujuan	Berhasil
(3,3)	B	1 (Terdeteksi)	Robot berhenti sementara (pause)	Berhasil
(3,3)	B	1 (Masih terdeteksi)	Robot tetap berhenti	Berhasil
(3,3)	B	0 (Obstacle hilang)	Robot melanjutkan navigasi	Berhasil

Berdasarkan Tabel 4.25, robot mampu memberikan respons yang sesuai terhadap perubahan kondisi obstacle selama proses navigasi. Pada kondisi awal ketika sensor proximity tidak mendeteksi obstacle (status = 0), robot bergerak menuju *Goal* B sesuai dengan jalur yang telah ditentukan oleh algoritma *Q-Learning*. Ketika robot mencapai posisi grid (3,3) dan sensor mendeteksi obstacle (status = 1), robot menghentikan sementara pergerakan (*pause*) sehingga tidak melanjutkan eksekusi aksi berikutnya.

Gambar 4.45. Obstacle Terdeteksi

Gambar 4.46. Obstacle Tidak Terdeteksi

Selama obstacle masih berada dalam area deteksi sensor, robot tetap berhenti sehingga tidak terjadi pergerakan pada jalur yang masih terhalang. Setelah obstacle dipindahkan dan status sensor kembali menjadi 0, robot secara otomatis melanjutkan navigasi dari posisi terakhir tanpa mengulang lintasan maupun melakukan inisialisasi ulang. Hasil pengujian menunjukkan bahwa mekanisme respons terhadap obstacle telah bekerja sesuai rancangan. Sensor proximity tidak memengaruhi proses pembentukan *state* maupun pemilihan aksi pada algoritma Q-Learning, tetapi hanya mengendalikan pelaksanaan aksi sebagai lapisan keselamatan (*safety layer*). Dengan demikian, robot tetap mengikuti *policy* hasil pembelajaran Q-Learning, sementara sensor proximity menghentikan sementara pergerakan ketika jalur terhalang dan mengizinkan robot melanjutkan navigasi setelah jalur kembali aman.