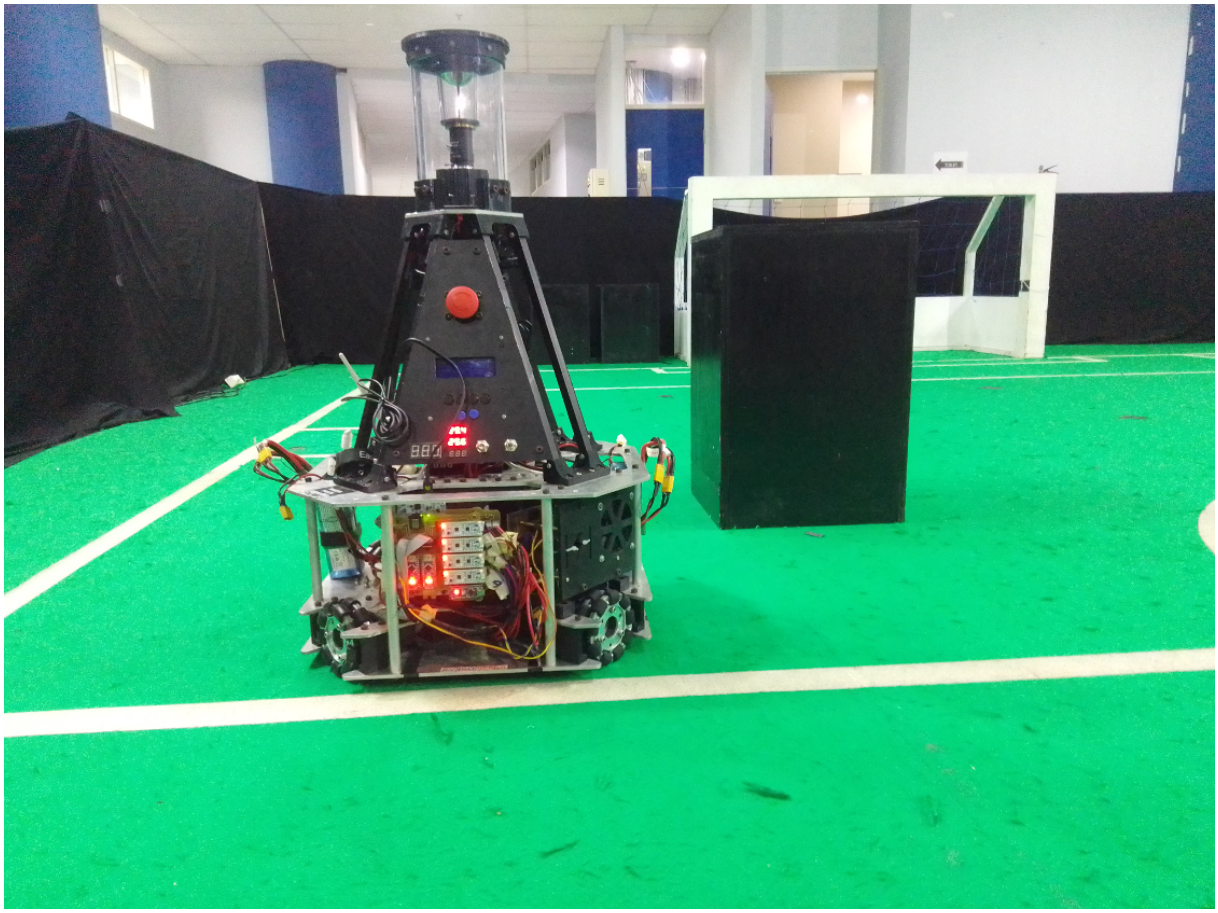
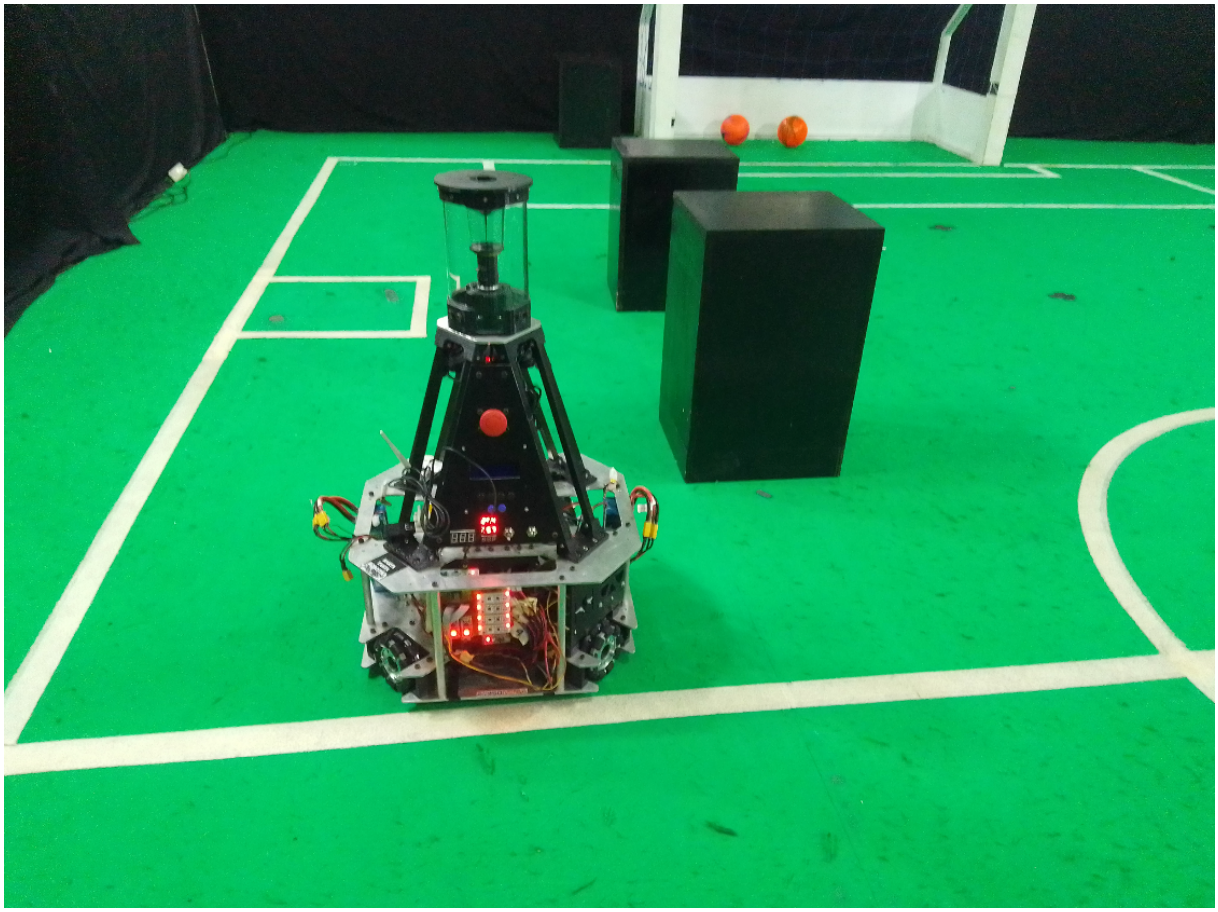
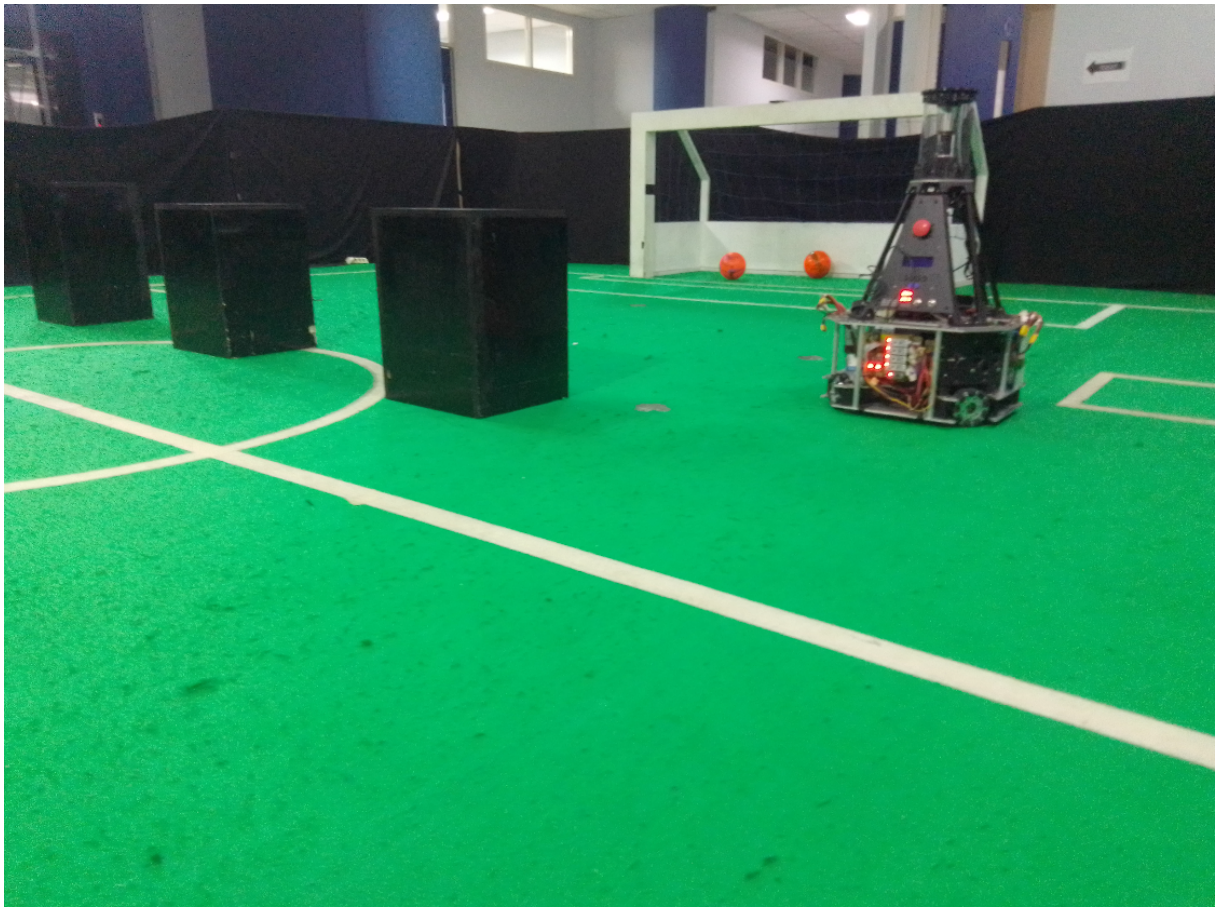
LAMPIRAN - LAMPIRAN

Lampiran 1. Dokumentasi Uji Aplikasi







Lampiran 2. Simulasi Perhitungan Metode

1. **Simulasi Perhitungan Metode**

Pada analisa permasalahan yang diterapkan pada penelitian ini, akan diberikan contoh perhitungan simulasi mengenai metode yang digunakan pada tahapan *pre-processing* dan *post-processing* dalam perencanaan rute robot pada sistem. Tahapan pertama yang dilakukan adalah melakukan pencarian rute berdasarkan algoritma A* yang direpresentasikan sebagai berikut:

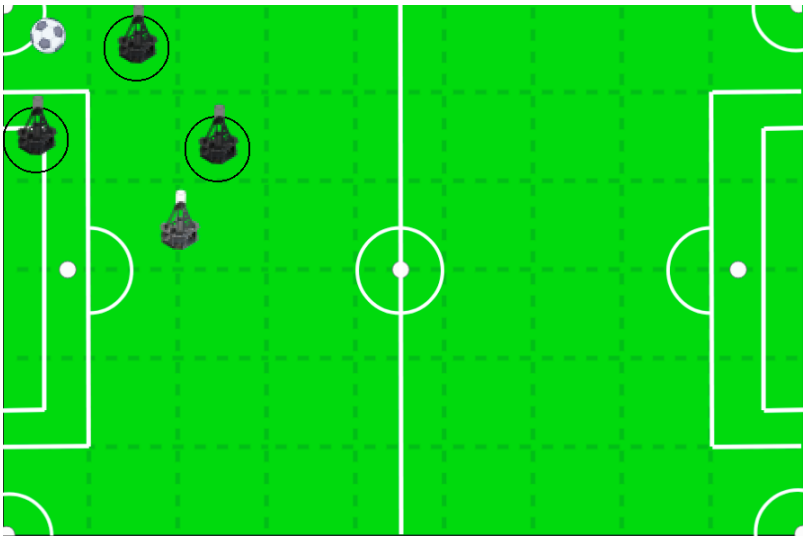
- **Simulasi Perhitungan Metode *A-star Search* (A*) dengan *Euclidean Distance***

Pada perencanaan rute menggunakan metode A* dilakukan beberapa tahapan yang akan direpresentasikan melalui langkah – langkah berikut ini, antara lain:

No	**Jenis *Node***	**Koordinat *Node***			**Keterangan *Node***
		x	***y***	***z***	
1.	*start node*	2.010	2.430	0º	Posisi awal robot saat ini berupa koordinat yang

					direpresentasikan dengan format ***x, y,*** dan ***z***.
2.	*end node*	0.540	0.380	0º	Posisi tujuan akhir dari robot berupa koordinat yang direpresentasikan dengan format ***x, y,*** dan ***z***.
3.	*obstacle node* 1	1.540	0.360	0º	Posisi robot lawan atau kawan yang diasumsikan sebagai halangan pada sistem berupa koordinat yang direpresentasikan dengan format ***x, y,*** dan ***z***.
4.	*obstacle node* 2	0.420	1.380	0º	
5.	*obstacle node* 3	2.440	1.480	0º	

Tabel di atas merupakan contoh permasalahan yang akan diselesaikan menggunakan algoritma A*, dimana ada 3 robot lawan sebagai halangan. Gambar dibawah merepresentasikan posisi koordinat setiap robot yang akan dilakukan simulasi perhitungan. Warna abu – abu merepresentasikan posisi koordinat robot asli, sedangkan warna hitam merepresentasikan posisi koordinat lawan.

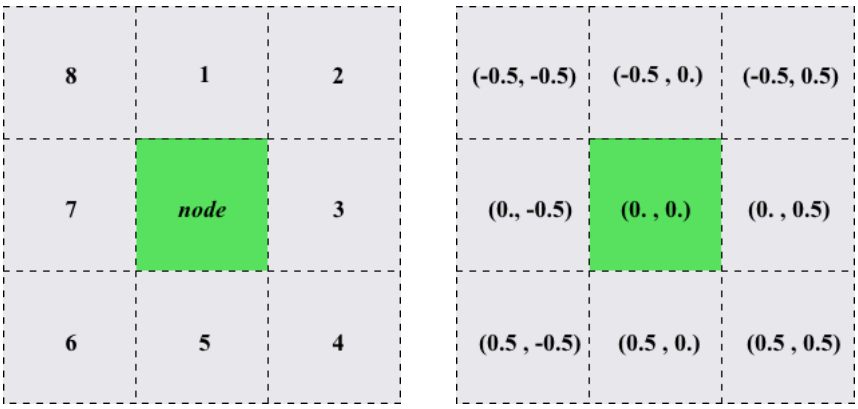


Setelah menentukan titik – titik koordinat. Langkah selanjutnya yaitu menentukan arah pencarian rute terhadap titik *start* pada algoritma A*. Nilai dari koordinat (***x***) akan ditambahkan sesuai indeks dengan arah baris, dan nilai dari koordinat (***y***) akan ditambahkan dengan arah

kolom sesuai indeks. Untuk setiap posisi koordinat (***x,y***) akan ditambahkan sesuai nilai yang direpresentasikan pada tabel dibawah ini:

No.	Arah Baris	Arah Kolom
1.	-0.5	0
2.	-0.5	+0.5
3.	0	+0.5
4.	+0.5	+0.5
5.	+0.5	0
6.	+0.5	-0.5
7.	0	-0.5
8.	-0.5	-0.5

Tabel perencanaan penentuan arah pada algoritma A* akan direpresentasikan pada gambar dibawah ini agar memudahkan pembaca dalam memahami permasalahan yang diberikan.



Langkah selanjutnya adalah menentukan jarak aman terhadap *obstacle* atau robot lawan, dan robot kawan untuk menghindari tabrakan saat pencarian rute menggunakan algoritma A*. Jarak aman terhadap tabrakan bisa berubah sesuai dengan kebutuhan yang ingin diterapkan pada robot. Tabel dibawah ini merepresentasikan jarak aman terhadap tabrakan robot.

No.	Jarak (meter)
1.	0.6375

Setelah menentukan jarak aman robot terhadap tabrakan, langkah selanjutnya adalah menentukan jarak minimal posisi koordinat yang dicari terhadap posisi tujuan akhir dalam perencanaan rute algoritma A*. Tabel dibawah ini merepresentasikan jarak minimum terhadap titik tujuan akhir.

No.	**Jarak (meter)**
1.	0.75

Langkah selanjutnya adalah melakukan perhitungan jarak *heuristic* terhadap posisi robot saat ini terhadap posisi *node* tetangga yang telah direpresentasikan pada arah pencarian rute algoritma A*. Fungsi *heuristic* yang dihitung adalah total dari kalkulasi persamaan ***f(n)*** *dan* ***g(n)*** yang direpresentasikan pada persamaan 2.1 atau dengan menggunakan fungsi *euclidean distance*. Data data pencarian posisi koordinat saat ini terhadap posisi *node* tetangga akan disimpan pada variabel *openSet* dan *closedSet. OpenSet* adalah variabel penampung terhadap *node* yang dikunjungi atau digunakan sebagai antrian dari *node* yang telah dikunjungi dengan melihat fungsi biaya ***f(n)*** paling kecil atau minimum, sedangkan *closedSet* adalah variabel penampung yang digunakan sebagai informasi bahwa *node* sudah dilakukan eksplorasi. Tabel dibawah ini merepresentasikan posisi koordinat awal robot yang ingin dilakukan pencarian.

No.	OpenSet ***f(n)*** minimum	
	x	*y*
1.	2.010	2.430

Dari hasil posisi koordinat awal kita melakukan kalkulasi fungsi *heuristic* jarak yang digunakan sebagai informasi utama dalam melakukan langkah – langkah selanjutnya dalam perencanaan rute dengan menggunakan algoritma A*. Tabel dibawah ini merepresentasikan output dari fungsi biaya terhadap *openSet* yang dieksplorasi.

No.	***f(cost)***	***h(n)***	***g(n)***
1.	2.523	2.523	0.000

Tabel dibawah ini akan merepresentasikan pencarian rute robot saat ini atau *openSet* dengan fungsi biaya paling minimum pada antrian terhadap *node* tetangga sesuai arah pencarian yang telah ditentukan.

(1.51, 1.93)	(1.51, 2.43)	(1.51, 2.93)
(2.01, 1.93)	(2.01, 2.43)	(2.01, 2.93)
(2.51, 1.93)	(2.51, 2.43)	(2.51, 2.93)

No.	A* *node*	
	x	*y*
1.	1.510	2.430
2.	1.510	2.930
3.	2.010	2.930
4.	2.510	2.930
5.	2.510	2.430
6.	2.510	1.930
7.	2.010	1.930
8.	1,510	1,930

Hasil dari pencarian terhadap posisi koordinat tetangga akan dilakukan perhitungan terhadap robot lawan atau obstacle apakah dianggap rute yang bisa dilalui robot atau tidak. Tabel berikut ini merepresentasikan jarak terhadap *obstacle* dan status rute yang dieksplorasi oleh *node.* Warna hijau pada tabel menyatakan *node* bersangkutan memiliki status aman, warna merah pada tabel menunjukkan bahwa *node* bersangkutan telah dilakukan eksplorasi, dan warna biru pada tabel menunjukkan bahwa *node* bersangkutan memiliki nilai fungsi biaya lebih besar dibandingkan *node* pada *openSet*.

No.	Jarak: *obstacle*		
1.	2.070	1.513	1.329

2.	2.570	1.895	1.723
3.	2.613	2.220	1.512
4.	2.747	2.602	1.452
5.	2.286	2.339	0.953
6.	1.845	2.161	0.455
7.	1.639	1.682	0.622
8.	1.570	1.221	1.033

No.	***f(cost)***	***h(n)***	***g(n)***	status *node*	Rank	Rank Terbaru
1.	2.768	2.268	0.500	aman	3	2
2.	3.435	2.728	0.707	aman	6	4
3.	3.443	2.943	0.500	aman	7	5
4.	3.929	3.222	0.707	aman	8	6
5.	3.343	2.843	0.500	aman	5	3
6.	3.214	2.507	0.707	tidak aman	-	-
7.	2.636	2.136	0.500	tidak aman	-	-
8.	2.536	1.828	0.707	aman	1	1
Total status *node* tidak aman				2		

Dari hasil eksplorasi *node* maka data – data koordinat bersangkutan akan dimasukkan ke dalam variabel *openSet* dan *closedSet*. Tabel dibawah ini merepresentasikan data *closedSet node* yang telah dilakukan eksplorasi dan tidak boleh dilewati kembali oleh algoritma A*.

ClosedSet	
x	***y***
2.010	2.430

Tabel dibawah ini akan merepresentasikan antrian dengan fungsi biaya jarak paling minimum pada variabel *openSet*. Data dengan fungsi biaya paling minimum akan dijadikan data acuan untuk melakukan pencarian terhadap *node* tetangga.

OpenSet		
x	***y***	***f(cost)***
1.510	1.930	2.536
1.510	2.430	2.768
2.510	2.430	3.343
1.510	2.930	3.435
2.010	2.930	3.443
2.510	2.930	3.929

Langkah selanjutnya yaitu melakukan pencarian rute berdasarkan antrian pada *openSet* dengan nilai fungsi biaya paling sedikit. Tabel dibawah ini merepresentasikan *node* yang menjadi acuan untuk merencanakan pencarian rute algoritma A*.

No.	OpenSet ***f(n)*** minimum	
	x	*y*
1.	1.510	1.930

Dari hasil posisi koordinat *openSet* dengan fungsi biaya paling minimum, langkah selanjutnya adalah menentukan fungsi biaya pada koordinat bersangkutan. Tabel dibawah ini merepresentasikan output dari fungsi biaya terhadap *openSet* yang dieksplorasi.

No.	***f(cost)***	***h(n)***	***g(n)***
1.	2.536	1.828	0.707

Langkah selanjutnya adalah melakukan pencarian *openSet* terhadap *node* tetangga sesuai dengan arah pencarian yang telah ditentukan. Tabel dibawah ini merepresentasikan pencarian rute bersangkutan.

(1.01, 1.43)	(1.01, 1.93)	(1.01, 2.43)
(1.51, 1.43)	(1.51, 1.93)	(1.51, 2.43)
(2.01, 1.43)	(2.01, 1.93)	(2.01, 2.43)

No.	A* *node*	
	x	*y*
1.	1.010	1.930
2.	1.010	2.430
3.	1.510	2.430
4.	2.010	2.430
5.	2.010	1.930
6.	2.010	1.430
7.	1.510	1.430
8.	1.010	1.430

Hasil dari pencarian terhadap posisi koordinat tetangga akan dilakukan perhitungan terhadap robot lawan atau obstacle apakah dianggap rute yang bisa dilalui robot atau tidak. Tabel berikut ini merepresentasikan jarak terhadap *obstacle* dan status rute yang dieksplorasi oleh *node.*

No.	Jarak: *obstacle*		
1.	1.657	0.807	1.499
2.	2.137	1.204	1.717
3.	2.070	1.513	1.329
4.	2.123	1.905	1.043
5.	1.639	1.682	0.622
6.	1.169	1.591	0.433
7.	1.070	1.091	0.931

8.	1.194	0.592	1.431

No.	*f(cost)*	*h(n)*	*g(n)*	status *node*	Rank	Rank Terbaru
1.	2.827	1.620	1.207	aman	3	2
2.	3.517	2.103	1.414	aman	7	3
3.	3.475	2.268	1.207	aman	-	-
4.	3.937	2.523	1.414	aman	-	-
5.	3.343	2.136	1.207	tidak aman	-	-
6.	3.221	1.806	1.414	tidak aman	-	-
7.	2.637	1.429	1.207	aman	2	1
8.	2.565	1.150	1.414	tidak aman	-	-
Total status *node* tidak aman				3		

Langkah selanjutnya adalah melakukan perubahan data pada variabel *closedSet* dan *openSet*. Tabel dibawah ini merepresentasikan perubahan data yang terjadi pada variabel *closedSet* dan *openSet*.

ClosedSet	
x	***y***
2.010	2.430
1.510	1.930

OpenSet		
x	***y***	***f(cost)***
1.510	1.430	2.637
1.510	2.430	2.768
1.010	1.930	2.827

2.510	2.430	3.343
1.510	2.930	3.435
2.010	2.930	3.443
1.010	2.430	3.517
2.510	2.930	3.929

Langkah selanjutnya yaitu melakukan pencarian rute berdasarkan antrian pada *openSet* yang terbaru dengan nilai fungsi biaya paling sedikit. Tabel dibawah ini merepresentasikan *node* yang menjadi acuan untuk merencanakan pencarian rute algoritma A*.

No.	OpenSet ***f(n)*** minimum	
	x	*y*
1.	1.510	1.430

Dari hasil posisi koordinat *openSet* dengan fungsi biaya paling minimum, langkah selanjutnya adalah menentukan fungsi biaya pada koordinat bersangkutan. Tabel dibawah ini merepresentasikan output dari fungsi biaya terhadap *openSet* yang dieksplorasi.

No.	***f(cost)***	***h(n)***	***g(n)***
1.	2.637	1.429	1.207

Langkah selanjutnya adalah melakukan pencarian *openSet* terhadap *node* tetangga sesuai dengan arah pencarian yang telah ditentukan. Tabel dibawah ini merepresentasikan pencarian rute bersangkutan.

(1.01, 0.93)	(1.01, 1.43)	(1.01, 1.93)
(1.51, 0.93)	(1.51, 1.43)	(1.51, 1.93)
(2.01, 0.93)	(2.01, 1.43)	(2.01, 1.93)

No.	A* *node*	
	x	*y*

1.	1.010	1.430
2.	1.010	1.930
3.	1.510	1.930
4.	2.010	1.930
5.	2.010	1.430
6.	2.010	0.930
7.	1.510	0.930
8.	1.010	0.930

Hasil dari pencarian terhadap posisi koordinat tetangga akan dilakukan perhitungan terhadap robot lawan atau obstacle apakah dianggap rute yang bisa dilalui robot atau tidak. Tabel berikut ini merepresentasikan jarak terhadap *obstacle* dan status rute yang dieksplorasi oleh *node.*

No.	Jarak: *obstacle*		
1.	1.194	0.592	1.431
2.	1.657	0.807	1.499
3.	1.570	1.221	1.033
4.	1.639	1.682	0.622
5.	1.169	1.591	0.433
6.	0.739	1.652	0.698
7.	0.571	1.179	1.080
8.	0.778	0.742	1.532

No.	***f(cost)***	***h(n)***	***g(n)***	status *node*	Rank	Rank Terbaru
1.	2.857	1.150	1.707	tidak aman	-	-
2.	3.534	1.620	1.914	aman	-	-

3.	3.536	1.828	1.707	aman	-	-
4.	4.050	2.136	1.914	tidak aman	-	-
5.	3.514	1.806	1.707	tidak aman	-	-
6.	3.484	1.570	1.914	aman	4	2
7.	2.822	1.115	1.707	tidak aman	-	-
8.	2.638	0.723	1.914	aman	1	1
Total status *node* tidak aman				4		

Langkah selanjutnya adalah melakukan perubahan data pada variabel *closedSet* dan *openSet*. Tabel dibawah ini merepresentasikan perubahan data yang terjadi pada variabel *closedSet* dan *openSet*.

ClosedSet	
x	***y***
2.010	2.430
1.510	1.930
1.510	1.430

OpenSet		
x	***y***	***f(cost)***
1.010	0.930	2.638
1.510	2.430	2.768
1.010	1.930	2.827
2.510	2.430	3.343
1.510	2.930	3.435
2.010	2.930	3.443
2.010	0.930	3.484
1.010	2.430	3.517
2.510	2.930	3.929

Langkah selanjutnya yaitu melakukan pencarian rute berdasarkan antrian pada *openSet* yang terbaru dengan nilai fungsi biaya paling sedikit. Tabel dibawah ini merepresentasikan *node* yang menjadi acuan untuk merencanakan pencarian rute algoritma A*.

No.	OpenSet ***f(n)*** minimum	
	x	*y*
1.	1.010	0.930

Dari hasil posisi koordinat *openSet* dengan fungsi biaya paling minimum, langkah selanjutnya adalah menentukan fungsi biaya pada koordinat bersangkutan. Tabel dibawah ini merepresentasikan output dari fungsi biaya terhadap *openSet* yang dieksplorasi.

No.	***f(cost)***	***h(n)***	***g(n)***
1.	2.638	0.723	1.914

Langkah selanjutnya adalah melakukan pencarian *openSet* terhadap *node* tetangga sesuai dengan arah pencarian yang telah ditentukan. Tabel dibawah ini merepresentasikan pencarian rute bersangkutan.

(0.51, 0.43)	(0.51, 0.93)	(0.51, 1.43)
(1.01, 0.43)	(1.01, 0.93)	(1.01, 1.43)
(1.51, 0.43)	(1.51, 0.93)	(1.51, 1.43)

No.	A* *node*	
	x	*y*
1.	0.510	0.930
2.	0.510	1.430
3.	1.010	1.430
4.	1.510	1.430
5.	1.510	0.930
6.	1.510	0.430

7.	1.010	0.430
8.	0.510	0.430

Hasil dari pencarian terhadap posisi koordinat tetangga akan dilakukan perhitungan terhadap robot lawan atau obstacle apakah dianggap rute yang bisa dilalui robot atau tidak. Tabel berikut ini merepresentasikan jarak terhadap *obstacle* dan status rute yang dieksplorasi oleh *node.*

No.	Jarak: *obstacle*		
1.	1.177	0.459	2.007
2.	1.485	0.103	1.931
3.	1.194	0.592	1.431
4.	1.070	1.091	0.931
5.	0.571	1.179	1.080
6.	0.076	1.446	1.403
7.	0.535	1.118	1.774
8.	1.032	0.954	2.197

No.	***f(cost)***	***h(n)***	***g(n)***	status *node*	Rank	Rank Terbaru
1.	2.965	0.551	2.414	tidak aman	-	-
2.	3.672	1.050	2.621	tidak aman	-	-
3.	3.565	1.150	2.414	tidak aman	-	-
4.	4.051	1.429	2.621	aman	-	-
5.	3.529	1.115	2.414	tidak aman	-	-
6.	3.593	0.971	2.621	tidak aman	-	-
7.	2.887	0.473	2.414	tidak aman	-	-
8.	2.680	0.058	2.621	aman	1	1

Total status *node* tidak aman	6

Langkah selanjutnya adalah melakukan perubahan data pada variabel *closedSet* dan *openSet*. Tabel dibawah ini merepresentasikan perubahan data yang terjadi pada variabel *closedSet* dan *openSet*.

ClosedSet	
x	***y***
2.010	2.430
1.510	1.930
1.510	1.430
1.010	0.930

OpenSet		
x	***y***	***f(cost)***
0.510	0.430	2.680
1.510	2.430	2.768
1.010	1.930	2.827
2.510	2.430	3.343
1.510	2.930	3.435
2.010	2.930	3.443
2.010	0.930	3.484
1.010	2.430	3.517
2.510	2.930	3.929

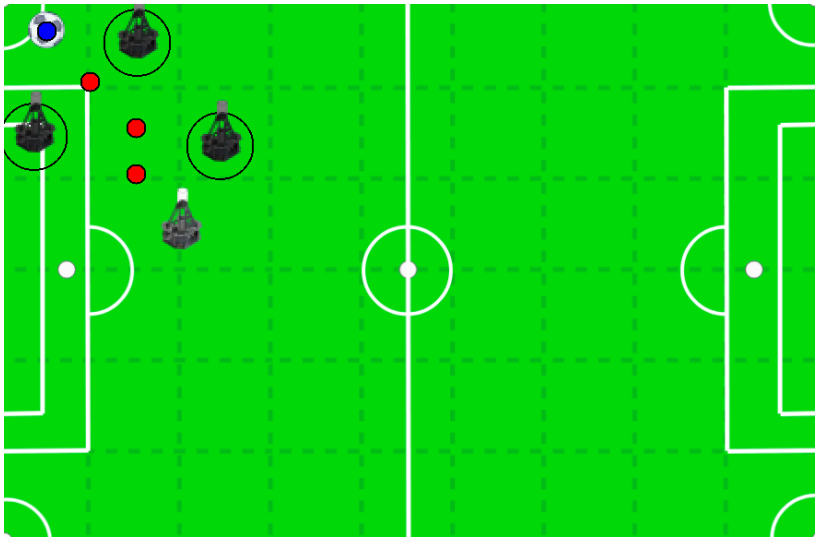
Langkah selanjutnya yaitu menentukan apakah koordinat dengan nilai paling minimum pada variabel *openSet* memiliki jarak kurang dari sama dengan terhadap jarak *threshold* titik tujuan akhir yang ditentukan. Tabel di bawah ini merepresentasikan output jarak *openSet* terhadap titik tujuan akhir.

Jarak Akhir	
Distance	0.058
Status	Pencarian Selesai

Dikarenakan jarak terhadap titik tujuan akhir telah masuk kedalam *threshold* yang ditentukan, maka langkah selanjutnya adalah mengurutkan rute perjalanan yang telah diolah oleh algoritma A* dari titik koordinat awal hingga menuju titik koordinat akhir tujuan robot. Tabel dibawah ini merepresentasikan output perencanaan rute algoritma A*.

No.	A* *node*	
	x	*y*
1.	1.51	1.93
2.	1.51	1.43
3.	1.1	0.93
4.	0.54	0.38

Hasil dari output perencanaan rute menggunakan metode A* akan direpresentasikan pada gambar dibawah ini agar memudahkan pembaca.



- **Simulasi Perhitungan Metode *Improved A-star Search* (A*) hasil *Euclidean Distance***

Simulasi Perhitungan Metode *Improved A-star Search* (A*) adalah tahapan *post-processing*. Tahapan ini adalah tahap akhir yang dilakukan pada penelitian ini dengan menyederhanakan rute perencanaan yang telah didapatkan oleh algoritma A*. Metode *Improved* A* membutuhkan kombinasi algoritma *Digital Differential Analyzer*. Langkah pertama yang dilakukan dalam simulasi ini adalah penentuan jarak aman terhadap tabrakan *obstacle* yang direpresentasikan pada tabel dibawah ini.

No.	**Jarak (meter)**
1.	0.6375

Selanjutnya adalah membuat variabel penampung terhadap rute perencanaan robot, nilai variabel penampung berisi posisi *start* dari robot yang direpresentasikan pada tabel dibawah ini.

Rute Normalisasi	
x	*y*
2.010	2.430

Setelah melakukan penambahan rute perjalanan pada variabel penampung maka langkah selanjutnya adalah melakukan pencarian rute berdasarkan algoritma DDA yang direpresentasikan pada tabel dibawah ini.

No.	**Node**	**Koordinat Posisi**	
		x	***y***
1.	*start point*	2.010	2.430
2.	*end point*	1.510	1.930

Setelah melakukan penentuan rute yang akan dicari menggunakan algoritma DDA. Selanjutnya adalah melakukan perhitungan untuk pencarian rute yang dilewati berdasarkan persamaan rumus 2.4, 2.5, 2.6, dan 2.7. Tabel dibawah ini merepresentasikan perhitungan persamaan DDA.

Pencarian Rute DDA	
dx	-0.500
dy	-0.500
step	1.000
x	-0.500
y	-0.500

Dari *step* yang dihasilkan, maka dilakukan perhitungan dengan menjumlahkan nilai koordinat (x, y) titik awal atau *start* point ditambahkan dengan koordinat (x, y) sesuai yang dihasilkan pada tabel di atas. Tabel dibawah ini merepresentasikan output perhitungan *step* algoritma DDA.

No	Koordinat	
	x	*y*
1.	1.510	1.930

Selanjutnya adalah melakukan perhitungan terhadap jarak *obstacle* untuk pengecekan apakah *node* bersangkutan aman atau tidak aman. Tabel dibawah ini merepresentasikan status jarak *node* terhadap *obstacle.*

No.	Jarak: *obstacle*			status *node*
1.	1.570	1.221	1.033	aman

Langkah selanjutnya adalah merencanakan kembali koordinat rute dari hasil algoritma A* terhadap *node* selanjutnya untuk normalisasi rute. Tabel dibawah ini menunjukkan pencarian algoritma DDA terhadap *node* selanjutnya.

No.	**Node**	**Koordinat Posisi**

		x	y
1.	*start point*	2.010	2.430
2.	*end point*	1.510	1.430

Dari hasil perencanaan *node* maka dilakukan proses algoritma DDA sesuai dengan data yang direpresentasikan pada tabel dibawah ini.

Pencarian Rute DDA	
dx	-0.500
dy	-1.000
step	1.000
x	-0.500
y	-1.000

Dari *step* yang dihasilkan maka data posisi koordinat (x, y) akan dilakukan kalkulasi penjumlahan agar mencapai titik target tujuan. Tabel dibawah ini merepresentasikan *step* yang terjadi pada algoritma DDA.

No.	Koordinat	
	x	*y*
1.	1.510	1.430

Selanjutnya adalah melakukan perhitungan terhadap jarak *obstacle* untuk pengecekan apakah *node* bersangkutan aman atau tidak aman. Tabel dibawah ini merepresentasikan status jarak *node* terhadap *obstacle.*

No.	Jarak: *obstacle*			status *node*
1.	1.070	1.091	0.931	aman

Langkah selanjutnya adalah merencanakan kembali koordinat rute dari hasil algoritma A* terhadap *node* yang terbaru untuk normalisasi rute. Tabel dibawah ini menunjukkan pencarian algoritma DDA terhadap *node* selanjutnya.

No.	**Node**	**Koordinat Posisi**	
		x	***y***
1.	*start point*	2.010	2.430
2.	*end point*	1.100	0.930

Dari hasil perencanaan *node* maka dilakukan proses algoritma DDA sesuai dengan data yang direpresentasikan pada tabel dibawah ini.

Pencarian Rute DDA	
dx	-0.910
dy	-1.500
step	2.000
x	-0.455
y	-0.750

Dari *step* yang dihasilkan maka data posisi koordinat (x, y) akan dilakukan kalkulasi penjumlahan untuk mencapai titik tujuan. Tabel dibawah ini merepresentasikan proses kalkulasi *step* pada *node*.

No.	Koordinat	
	x	*y*
1.	1.555	1.680
2.	1.100	0.930

Selanjutnya adalah melakukan perhitungan terhadap jarak *obstacle* untuk pengecekan apakah *node* bersangkutan aman atau tidak aman. Tabel dibawah ini merepresentasikan status jarak *node* terhadap *obstacle.*

No.	Jarak: *obstacle*			status *node*
1.	1.320	1.174	0.907	aman
2.	0.720	0.815	1.448	aman

Langkah selanjutnya adalah melanjutkan proses penyederhanaan rute dengan menggunakan algoritma DDA terhadap *node* yang terbaru. Tabel dibawah ini merepresentasikan pencarian rute dari titik tujuan awal terhadap titik tujuan akhir.

No.	**Node**	**Koordinat Posisi**	
		x	***y***
1.	*start point*	2.010	2.430
2.	*end point*	0.540	0.380

Dari hasil perencanaan *node* maka dilakukan proses algoritma DDA sesuai dengan data yang direpresentasikan pada tabel dibawah ini.

Pencarian Rute DDA	
dx	-1.470
dy	-2.050
step	2.000
x	-0.735
y	-1.025

Selanjutnya adalah melakukan perhitungan terhadap jarak *obstacle* untuk pengecekan apakah *node* bersangkutan aman atau tidak aman. Tabel dibawah ini merepresentasikan status jarak *node* terhadap *obstacle.*

No.	Koordinat	
	x	*y*
1.	1.275	1.405
2.	0.540	0.380

Selanjutnya adalah melakukan perhitungan terhadap jarak *obstacle* untuk pengecekan apakah *node* bersangkutan aman atau tidak aman. Tabel dibawah ini merepresentasikan status jarak *node* terhadap *obstacle.*

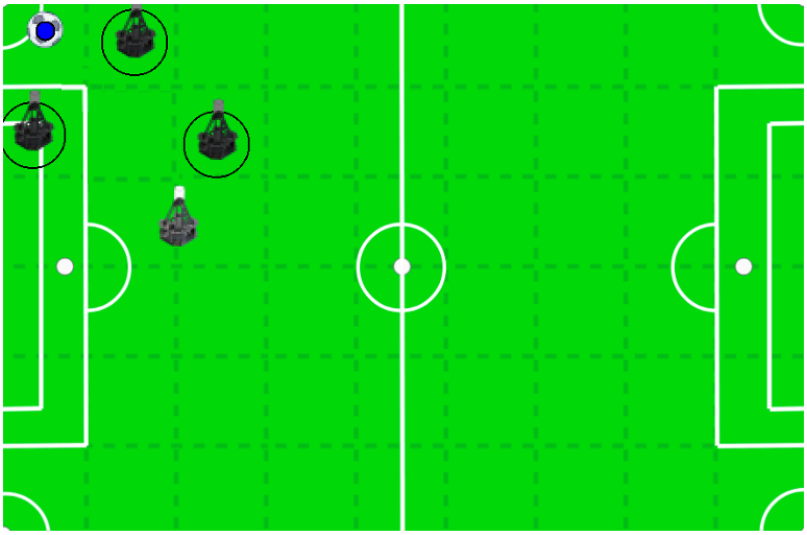
No.	Jarak: *obstacle*			status *node*
1.	1.078	0.855	1.167	aman
2.	1.000	1.007	2.195	aman

Dikarenakan output dari perencanaan rute dari hasil algoritma A* titik koordinat awal hingga titik koordinat tujuan akhir robot setelah dilakukan penyederhanaan rute tidak mengalami tabrakan terhadap *obstacle* atau dianggap rute yang paling efektif. Sebelumnya dilakukan proses untuk menghapus posisi koordinat pada variabel Rute Normalisasi indeks pertama. Hasil dari output penyederhanaan rute menggunakan algoritma *Improved* A* dengan kombinasi algoritma DDA adalah sebagai berikut.

Rute Normalisasi	
x	*y*
2.010	2.430

Rute Normalisasi Terbaru	
x	*y*
0.540	0.380

Hasil akhir dari perencanaan rute algoritma *Improved* A* direpresentasikan pada gambar dibawah ini agar mudah dipahami oleh pembaca. Output yang dihasilkan dari perencanaan *Improved* A* lebih sedikit dibandingkan output dari perencanaan rute dengan menggunakan algoritma A*. Hal ini menyebabkan pergerakan dari robot akan lebih efisien dan efektif terhadap titik tujuan yang dilalui hingga mencapai titik koordinat akhir dari tujuan robot yang diinginkan.



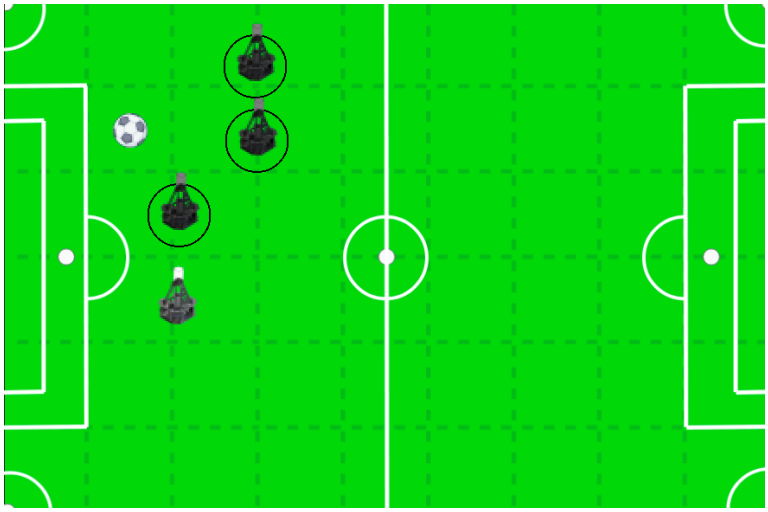
- **Simulasi Perhitungan Metode *A-star Search* (A*) dengan *Manhattan Distance***

Pada perencanaan rute menggunakan metode A* dilakukan beberapa tahapan yang akan direpresentasikan melalui langkah – langkah berikut ini, antara lain:

No	**Jenis *Node***	**Koordinat *Node***			**Keterangan *Node***
		x	***y***	***z***	
1.	*start node*	2.050	3,430	0º	Posisi awal robot saat ini berupa koordinat yang direpresentasikan dengan format ***x, y,*** dan ***z***.

2.	*end node*	1.500	1.520	0º	Posisi tujuan akhir dari robot berupa koordinat yang direpresentasikan dengan format ***x, y,*** dan ***z***.
3.	*obstacle node* 1	2.980	1.480	0º	Posisi robot lawan atau kawan yang diasumsikan sebagai halangan pada sistem berupa koordinat yang direpresentasikan dengan format ***x, y,*** dan ***z***.
4.	*obstacle node* 2	2.960	0.620	0º	
5.	*obstacle node* 3	2.,080	2.340	0º	

Tabel di atas merupakan contoh permasalahan yang akan diselesaikan menggunakan algoritma A*, dimana ada 3 robot lawan sebagai halangan. Gambar dibawah merepresentasikan posisi koordinat setiap robot sesuai informasi koordinat yang ada.

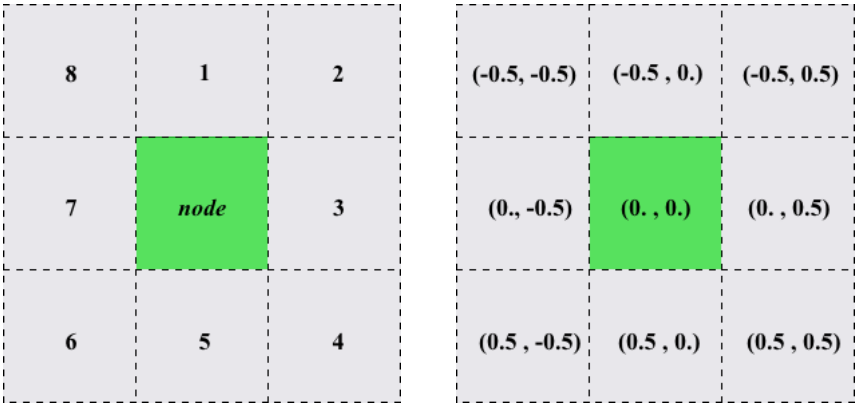


Setelah menentukan titik – titik koordinat. Langkah selanjutnya yaitu menentukan arah pencarian rute terhadap titik *start* pada algoritma A*. Nilai dari koordinat (***x***) akan ditambahkan sesuai indeks dengan arah baris, dan nilai dari koordinat (***y***) akan ditambahkan dengan arah kolom sesuai indeks. Untuk setiap posisi koordinat (***x,y***) akan ditambahkan sesuai nilai yang direpresentasikan pada tabel dibawah ini:

No.	**Arah Baris**	**Arah Kolom**
1.	-0.5	0

2.	-0.5	+0.5
3.	0	+0.5
4.	+0.5	+0.5
5.	+0.5	0
6.	+0.5	-0.5
7.	0	-0.5
8.	-0.5	-0.5

Tabel perencanaan penentuan arah pada algoritma A* akan direpresentasikan pada gambar dibawah ini agar memudahkan pembaca dalam memahami permasalahan yang diberikan.



Langkah selanjutnya adalah menentukan jarak aman terhadap *obstacle* atau robot lawan, dan robot kawan untuk menghindari tabrakan saat pencarian rute menggunakan algoritma A*. Jarak aman terhadap tabrakan bisa berubah sesuai dengan kebutuhan yang ingin diterapkan pada robot. Tabel dibawah ini merepresentasikan jarak aman terhadap tabrakan robot.

No.	**Jarak (meter)**
1.	0.65

Setelah menentukan jarak aman robot terhadap tabrakan, langkah selanjutnya adalah menentukan jarak minimal posisi koordinat yang dicari terhadap posisi tujuan akhir dalam perencanaan rute algoritma A*. Tabel dibawah ini merepresentasikan jarak minimum terhadap titik tujuan akhir.

No.	**Jarak (meter)**
1.	0.75

Langkah selanjutnya adalah melakukan perhitungan jarak *heuristic* terhadap posisi robot saat ini terhadap posisi *node* tetangga yang telah direpresentasikan pada arah pencarian rute algoritma A*. Fungsi *heuristic* yang dihitung adalah total dari kalkulasi persamaan ***f(n)*** *dan* ***g(n)*** yang direpresentasikan pada persamaan 2.2 atau dengan menggunakan fungsi *manhattan distance*. Data data pencarian posisi koordinat saat ini terhadap posisi *node* tetangga akan disimpan pada variabel *openSet* dan *closedSet. OpenSet* adalah variabel penampung terhadap *node* yang dikunjungi atau digunakan sebagai antrian dari *node* yang telah dikunjungi dengan melihat fungsi biaya ***f(n)*** paling kecil atau minimum, sedangkan *closedSet* adalah variabel penampung yang digunakan sebagai informasi bahwa *node* sudah dilakukan eksplorasi. Tabel dibawah ini merepresentasikan posisi koordinat awal robot yang ingin dilakukan pencarian.

No.	OpenSet ***f(n)*** minimum	
	x	*y*
1.	2.050	3.430

Dari hasil posisi koordinat awal kita melakukan kalkulasi fungsi *heuristic* jarak yang digunakan sebagai informasi utama dalam melakukan langkah – langkah selanjutnya dalam perencanaan rute dengan menggunakan algoritma A*. Tabel dibawah ini merepresentasikan output dari fungsi biaya terhadap *openSet* yang dieksplorasi.

No.	***f(cost)***	***h(n)***	***g(n)***
1.	2.460	2.460	0.000

Tabel dibawah ini akan merepresentasikan pencarian rute robot saat ini atau *openSet* dengan fungsi biaya paling minimum pada antrian terhadap *node* tetangga sesuai arah pencarian yang telah ditentukan.

(1.55, 2.93)	(1.55, 3.43)	(1.55, 3.93)
(2.05, 2.93)	(2.05, 3.43)	(2.05, 3.93)
(2.55, 2.93)	(2.55, 3.43)	(2.55, 3.93)

No.	A* *node*	
	x	*y*
1.	1.550	3.430
2.	1.550	3.930
3.	2.050	3.930
4.	2.550	3.930
5.	2.550	3.430
6.	2.550	2.930
7.	2.050	2.930
8.	1.550	2.930

Hasil dari pencarian terhadap posisi koordinat tetangga akan dilakukan perhitungan terhadap robot lawan atau obstacle apakah dianggap rute yang bisa dilalui robot atau tidak. Tabel berikut ini merepresentasikan jarak terhadap *obstacle* dan status rute yang dieksplorasi oleh *node.* Warna hijau pada tabel menyatakan *node* bersangkutan memiliki status aman, warna merah pada tabel menunjukkan bahwa *node* bersangkutan telah dilakukan eksplorasi, dan warna biru pada tabel menunjukkan bahwa *node* bersangkutan memiliki nilai fungsi biaya lebih besar dibandingkan *node* pada *openSet*.

No.	Jarak: *obstacle*		
1.	2.418	3.144	1.212
2.	2.837	3.598	1.676
3.	2.621	3.433	1.590
4.	2.487	3.335	1.658
5.	1.997	2.840	1.187
6.	1.512	2.346	0.754
7.	1.723	2.483	0.591
8.	2.037	2.706	0.793

No.	*f(cost)*	*h(n)*	*g(n)*	status *node*	Rank	Rank Terbaru
1.	2.460	1.960	0.500	aman	1	1
2.	3.460	2.460	1.000	aman	4	3
3.	3.460	2.960	0.500	aman	4	4
4.	4.460	3.460	1.000	aman	8	7
5.	3.460	2.960	0.500	aman	4	5
6.	3.460	2.460	1.000	aman	4	6
7.	2.460	1.960	0.500	tidak aman	-	-
8.	2.460	1.460	1.000	aman	1	2
Total status *node* tidak aman				1		

Dari hasil eksplorasi *node* maka data – data koordinat bersangkutan akan dimasukkan ke dalam variabel *openSet* dan *closedSet*. Tabel dibawah ini merepresentasikan data *closedSet node* yang telah dilakukan eksplorasi dan tidak boleh dilewati kembali oleh algoritma A*.

ClosedSet	
x	***y***
2.050	3.430

Tabel dibawah ini akan merepresentasikan antrian dengan fungsi biaya jarak paling minimum pada variabel *openSet*. Data dengan fungsi biaya paling minimum akan dijadikan data acuan untuk melakukan pencarian terhadap *node* tetangga.

OpenSet		
x	***y***	***f(cost)***
1.550	3.430	2.460
1.550	2.930	2.460
1.550	3.930	3.460

2.050	3.930	3.460
2.550	3.430	3.460
2.550	2.930	3.460
2.550	3.930	4.460

Langkah selanjutnya yaitu melakukan pencarian rute berdasarkan antrian pada *openSet* dengan nilai fungsi biaya paling sedikit. Tabel dibawah ini merepresentasikan *node* yang menjadi acuan untuk merencanakan pencarian rute algoritma A*.

No.	OpenSet ***f(n)*** minimum	
	x	*y*
1.	1.550	3.430

Dari hasil posisi koordinat *openSet* dengan fungsi biaya paling minimum, langkah selanjutnya adalah menentukan fungsi biaya pada koordinat bersangkutan. Tabel dibawah ini merepresentasikan output dari fungsi biaya terhadap *openSet* yang dieksplorasi.

No.	***f(cost)***	***h(n)***	***g(n)***
1.	2.460	1.960	0.500

Langkah selanjutnya adalah melakukan pencarian *openSet* terhadap *node* tetangga sesuai dengan arah pencarian yang telah ditentukan. Tabel dibawah ini merepresentasikan pencarian rute bersangkutan.

(1.05, 2.93)	(1.05, 3.43)	(1.05, 3.93)
(1.55, 2.93)	(1.55, 3.43)	(1.55, 3.93)
(2.05, 2.93)	(2.05, 3.43)	(2.05, 3.93)

No.	A* *node*	
	x	*y*
1.	1.050	3.430
2.	1.050	3.930
3.	1.550	3.930
4.	2.050	3.930
5.	2.050	3.430
6.	2.050	2.930
7.	1.550	2.930
8.	1.050	2.930

Hasil dari pencarian terhadap posisi koordinat tetangga akan dilakukan perhitungan terhadap robot lawan atau obstacle apakah dianggap rute yang bisa dilalui robot atau tidak. Tabel berikut ini merepresentasikan jarak terhadap *obstacle* dan status rute yang dieksplorasi oleh *node.*

No.	Jarak: *obstacle*		
1.	2.744	3.398	1.500
2.	3.119	3.822	1.894
3.	2.837	3.598	1.676
4.	2.621	3.433	1.590
5.	2.160	2.954	1.090
6.	1.723	2.483	0.591
7.	2.037	2.706	0.793
8.	2.414	2.997	1.187

No.	***f(cost)***	***h(n)***	***g(n)***	status *node*	Rank	Rank Terbaru

1.	3.360	2.360	1.000	aman	3	2
2.	4.360	2.860	1.500	aman	7	7
3.	3.460	2.460	1.000	aman	6	4
4.	4.460	2.960	1.500	aman	8	5
5.	3.460	2.460	1.000	aman	-	-
6.	3.460	1.960	1.500	tidak aman	-	-
7.	2.460	1.460	1.000	aman	1	1
8.	3.360	1.860	1.500	aman	2	3
Total status *node* tidak aman				1		

Langkah selanjutnya adalah melakukan perubahan data pada variabel *closedSet* dan *openSet*. Tabel dibawah ini merepresentasikan perubahan data yang terjadi pada variabel *closedSet* dan *openSet*.

ClosedSet	
x	***y***
2.050	3.430
1.550	3.430

OpenSet		
x	***y***	***f(cost)***
1.550	2.930	2.460
1.050	3.430	3.360
1.050	2.930	3.360
1.550	3.930	3.460
2.050	3.930	3.460
2.550	3.430	3.460
2.550	2.930	3.460

1.050	3.930	4.360
2.550	3.930	4.460

Langkah selanjutnya yaitu melakukan pencarian rute berdasarkan antrian pada *openSet* yang terbaru dengan nilai fungsi biaya paling sedikit. Tabel dibawah ini merepresentasikan *node* yang menjadi acuan untuk merencanakan pencarian rute algoritma A*.

No.	OpenSet ***f(n)*** minimum	
	x	*y*
1.	1.050	3.430

Dari hasil posisi koordinat *openSet* dengan fungsi biaya paling minimum, langkah selanjutnya adalah menentukan fungsi biaya pada koordinat bersangkutan. Tabel dibawah ini merepresentasikan output dari fungsi biaya terhadap *openSet* yang dieksplorasi.

No.	***f(cost)***	***h(n)***	***g(n)***
1.	2.460	1.460	1.000

Langkah selanjutnya adalah melakukan pencarian *openSet* terhadap *node* tetangga sesuai dengan arah pencarian yang telah ditentukan. Tabel dibawah ini merepresentasikan pencarian rute bersangkutan.

(1.05, 2.43)	(1.05, 2.93)	(1.05, 3.43)
(1.55, 2.43)	(1.55, 2.93)	(1.55, 3.43)
(2.05, 2.43)	(2.05, 2.93)	(2.05, 3.43)

No.	A* *node*	
	x	*y*
1.	1.050	2.930

2.	1.050	3.430
3.	1.550	3.430
4.	2.050	3.430
5.	2.050	2.930
6.	2.050	2.430
7.	1.550	2.430
8.	1.050	2.430

Hasil dari pencarian terhadap posisi koordinat tetangga akan dilakukan perhitungan terhadap robot lawan atau obstacle apakah dianggap rute yang bisa dilalui robot atau tidak. Tabel berikut ini merepresentasikan jarak terhadap *obstacle* dan status rute yang dieksplorasi oleh *node.*

No.	Jarak: *obstacle*		
1.	2.414	2.997	1.187
2.	2.744	3.398	1.500
3.	2.418	3.144	1.212
4.	2.160	2.954	1.090
5.	1.723	2.483	0.591
6.	1.329	2.026	0.095
7.	1.717	2.294	0.538
8.	2.151	2.631	1.034

No.	***f(cost)***	***h(n)***	***g(n)***	status *node*	Rank	Rank Terbaru
1.	3.360	1.860	1.500	aman	1	2
2.	4.360	2.360	2.000	aman	1	1

3.	3.460	1.960	1.500	aman	-	-
4.	4.460	2.460	2.000	aman	-	-
5.	3.460	1.960	1.500	tidak aman	-	-
6.	3.460	1.460	2.000	tidak aman	-	-
7.	2.460	0.960	1.500	tidak aman	-	-
8.	3.360	1.360	2.000	aman	1	3
Total status *node* tidak aman				3		

Langkah selanjutnya adalah melakukan perubahan data pada variabel *closedSet* dan *openSet*. Tabel dibawah ini merepresentasikan perubahan data yang terjadi pada variabel *closedSet* dan *openSet*.

ClosedSet	
x	***y***
2.050	3.430
1.550	3.430
1.550	2.930

OpenSet		
x	***y***	***f(cost)***
1.050	3.430	3.360
1.050	2.930	3.360
1.050	2.430	3.360
1.550	3.930	3.460
2.050	3.930	3.460
2.550	3.430	3.460
2.550	2.930	3.460
1.050	3.930	4.360

2.550	3.930	4.460

Langkah selanjutnya yaitu melakukan pencarian rute berdasarkan antrian pada *openSet* yang terbaru dengan nilai fungsi biaya paling sedikit. Tabel dibawah ini merepresentasikan *node* yang menjadi acuan untuk merencanakan pencarian rute algoritma A*.

No.	OpenSet ***f(n)*** minimum	
	x	*y*
1.	1.050	3.430

Dari hasil posisi koordinat *openSet* dengan fungsi biaya paling minimum, langkah selanjutnya adalah menentukan fungsi biaya pada koordinat bersangkutan. Tabel dibawah ini merepresentasikan output dari fungsi biaya terhadap *openSet* yang dieksplorasi.

No.	***f(cost)***	***h(n)***	***g(n)***
1.	3.360	2.360	1.000

Langkah selanjutnya adalah melakukan pencarian *openSet* terhadap *node* tetangga sesuai dengan arah pencarian yang telah ditentukan. Tabel dibawah ini merepresentasikan pencarian rute bersangkutan.

(0.55, 2.93)	(0.55, 3.43)	(0.55, 3.93)
(1.05, 2.93)	(1.05, 3.43)	(1.05, 3.93)
(1.55, 2.93)	(1.55, 3.43)	(1.55, 3.93)

No.	A* *node*	
	x	*y*
1.	0.550	3.430
2.	0.550	3.930
3.	1.050	3.930

4.	1.550	3.930
5.	1.550	3.430
6.	1.550	2.930
7.	1.050	2.930
8.	0.550	2.930

Hasil dari pencarian terhadap posisi koordinat tetangga akan dilakukan perhitungan terhadap robot lawan atau obstacle apakah dianggap rute yang bisa dilalui robot atau tidak. Tabel berikut ini merepresentasikan jarak terhadap *obstacle* dan status rute yang dieksplorasi oleh *node.*

No.	Jarak: *obstacle*		
1.	3.116	3.702	1.879
2.	3.451	4.094	2.207
3.	3.119	3.822	1.894
4.	2.837	3.598	1.676
5.	2.418	3.144	1.212
6.	2.037	2.706	0.793
7.	2.414	2.997	1.187
8.	2.830	3.338	1.640

No.	***f(cost)***	***h(n)***	***g(n)***	status *node*	Rank	Rank Terbaru
1.	4,360	2.860	1.500	aman	4	3
2.	5,360	3.360	2.000	aman	8	6
3.	4,360	2.860	1.500	aman	4	2
4.	4,460	2.460	2.000	aman	7	5
5.	3,460	1.960	1.500	aman	-	-

6.	3,460	1.460	2.000	aman	-	-
7.	3,360	1.860	1.500	aman	1	1
8.	4,360	2.360	2.000	aman	4	4
Total status *node* tidak aman				0		

Langkah selanjutnya adalah melakukan perubahan data pada variabel *closedSet* dan *openSet*. Tabel dibawah ini merepresentasikan perubahan data yang terjadi pada variabel *closedSet* dan *openSet*.

ClosedSet	
x	***y***
2.050	3.430
1.550	3.430
1.550	2.930
1.050	3.430

OpenSet		
x	***y***	***f(cost)***
1.050	2.930	3.360
1.050	2.430	3.360
1.550	3.930	3.460
2.050	3.930	3.460
2.550	3.430	3.460
2.550	2.930	3.460
1.050	3.930	4.360
0.550	3.430	4.360
0.550	2.930	4.360
2.550	3.930	4.460

0.550	3.930	5.360

Langkah selanjutnya yaitu melakukan pencarian rute berdasarkan antrian pada *openSet* yang terbaru dengan nilai fungsi biaya paling sedikit. Tabel dibawah ini merepresentasikan *node* yang menjadi acuan untuk merencanakan pencarian rute algoritma A*.

No.	OpenSet ***f(n)*** minimum	
	x	*y*
1.	1.050	2.930

Dari hasil posisi koordinat *openSet* dengan fungsi biaya paling minimum, langkah selanjutnya adalah menentukan fungsi biaya pada koordinat bersangkutan. Tabel dibawah ini merepresentasikan output dari fungsi biaya terhadap *openSet* yang dieksplorasi.

No.	***f(cost)***	***h(n)***	***g(n)***
1.	3.360	1.860	1.500

Langkah selanjutnya adalah melakukan pencarian *openSet* terhadap *node* tetangga sesuai dengan arah pencarian yang telah ditentukan. Tabel dibawah ini merepresentasikan pencarian rute bersangkutan.

(0.55, 2,43)	(0.55, 2,93)	(0.55, 3.43)
(1.05, 2,43)	(1.05, 2,93)	(1.05, 3.43)
(1.55, 2,43)	(1.55, 2,93)	(1.55, 3.43)

No.	A* *node*	
	x	*y*
1.	0,550	2.930
2.	0.550	3.430
3.	1.050	3.430

4.	1.550	3.430
5.	1.550	2.930
6.	1.550	2.430
7.	1.050	2.430
8.	0.550	2.430

Hasil dari pencarian terhadap posisi koordinat tetangga akan dilakukan perhitungan terhadap robot lawan atau obstacle apakah dianggap rute yang bisa dilalui robot atau tidak. Tabel berikut ini merepresentasikan jarak terhadap *obstacle* dan status rute yang dieksplorasi oleh *node.*

No.	Jarak: *obstacle*		
1.	2.830	3.338	1.640
2.	3.116	3.702	1.879
3.	2.744	3.398	1.500
4.	2.418	3.144	1.212
5.	2.037	2.706	0.793
6.	1.717	2.294	0.538
7.	2.151	2.631	1.034
8.	2.609	3.014	1.533

No.	***f(cost)***	***h(n)***	***g(n)***	status *node*	Rank	Rank Terbaru
1.	4.360	2.360	2.000	aman	4	2
2.	5.360	2.860	2.500	aman	8	3
3.	4.360	2.360	2.000	aman	-	-
4.	4.460	1.960	2.500	aman	-	-
5.	3.460	1.460	2.000	aman	-	-

6.	3.460	0.960	2.500	tidak aman	-	-
7.	3.360	1.360	2.000	aman	1	1
8.	4.360	1.860	2.500	aman	4	2
Total status *node* tidak aman				1		

Langkah selanjutnya adalah melakukan perubahan data pada variabel *closedSet* dan *openSet*. Tabel dibawah ini merepresentasikan perubahan data yang terjadi pada variabel *closedSet* dan *openSet*.

ClosedSet	
x	***y***
2.050	3.430
1.550	3.430
1.550	2.930
1.050	3.430
1.050	2.930

OpenSet		
x	***y***	***f(cost)***
1.050	2.430	3.360
1.550	3.930	3.460
2.050	3.930	3.460
2.550	3.430	3.460
2.550	2.930	3.460
1.050	3.930	4.360
0.550	3.430	4.360
0.550	2.930	4.360
0.550	2.430	4.360

2.550	3.930	4.460
0.550	3.430	5.360

Langkah selanjutnya yaitu melakukan pencarian rute berdasarkan antrian pada *openSet* yang terbaru dengan nilai fungsi biaya paling sedikit. Tabel dibawah ini merepresentasikan *node* yang menjadi acuan untuk merencanakan pencarian rute algoritma A*.

No.	OpenSet ***f(n)*** minimum	
	x	*y*
1.	1.050	2.430

Dari hasil posisi koordinat *openSet* dengan fungsi biaya paling minimum, langkah selanjutnya adalah menentukan fungsi biaya pada koordinat bersangkutan. Tabel dibawah ini merepresentasikan output dari fungsi biaya terhadap *openSet* yang dieksplorasi.

No.	***f(cost)***	***h(n)***	***g(n)***
1.	3.360	1.360	2.000

Langkah selanjutnya adalah melakukan pencarian *openSet* terhadap *node* tetangga sesuai dengan arah pencarian yang telah ditentukan. Tabel dibawah ini merepresentasikan pencarian rute bersangkutan.

(0.55, 1.93)	(0.55, 2.43)	(0.55, 2.93)
(1.05, 1.93)	(1.05, 2.43)	(1.05, 2.93)
(1.55, 1.93)	(1.55, 2.43)	(1.55, 2.93)

No.	A* *node*	
	x	*y*
1.	0.550	2.430
2.	0.550	2.930

3.	1.050	2.930
4.	1.550	2.930
5.	1.550	2.430
6.	1.550	1.930
7.	1.050	1.930
8.	0.550	1.930

Hasil dari pencarian terhadap posisi koordinat tetangga akan dilakukan perhitungan terhadap robot lawan atau obstacle apakah dianggap rute yang bisa dilalui robot atau tidak. Tabel berikut ini merepresentasikan jarak terhadap *obstacle* dan status rute yang dieksplorasi oleh *node.*

No.	Jarak: *obstacle*		
1.	2.609	3.014	1.533
2.	2.830	3.338	1.640
3.	2.414	2.997	1.187
4.	2.037	2.706	0.793
5.	1.717	2.294	0.538
6.	1.499	1.925	0.670
7.	1.982	2.316	1.109
8.	2.471	2.743	1.584

Nomor	***f(cost)***	***h(n)***	***g(n)***	status *node*	Rank	Rank Terbaru
1.	4.360	1.860	2.500	aman	5	2
2.	5.360	2.360	3.000	aman	8	-
3.	4.360	1.860	2.500	aman	-	-
4.	4.460	1.460	3.000	aman	-	-

5.	3.460	0.960	2.500	tidak aman	-	-
6.	3.460	0.460	3.000	aman	2	3
7.	3.360	0.860	2.500	aman	1	1
8.	4.360	1.360	3.000	aman	4	4
Total status *node* tidak aman				1		

Langkah selanjutnya adalah melakukan perubahan data pada variabel *closedSet* dan *openSet*. Tabel dibawah ini merepresentasikan perubahan data yang terjadi pada variabel *closedSet* dan *openSet*.

ClosedSet	
x	***y***
2.050	3.430
1.550	3.430
1.550	2.930
1.050	3.430
1.050	2.930
1.050	2.430

OpenSet		
x	***y***	***f(cost)***
1.050	1.930	3.360
1.550	3.930	3.460
2.050	3.930	3.460
2.550	3.430	3.460
2.550	2.930	3.460
1.550	1.930	3.460
1.050	3.930	4.360

0.550	3.430	4.360
0.550	2.930	4.360
0.550	2.430	4.360
0.550	1.930	4.360
2.550	3.930	4.460
1.550	2.930	4.460

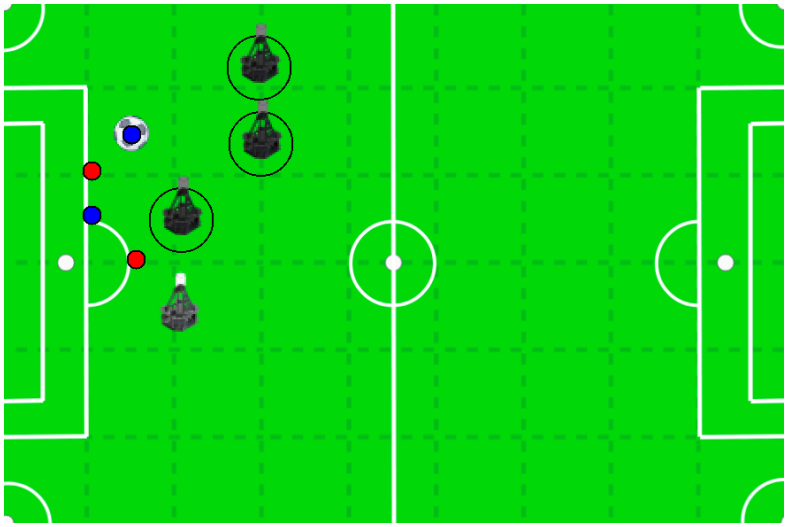
Langkah selanjutnya yaitu menentukan apakah koordinat dengan nilai paling minimum pada variabel *openSet* memiliki jarak kurang dari sama dengan terhadap jarak *threshold* titik tujuan akhir yang ditentukan. Tabel di bawah ini merepresentasikan output jarak *openSet* terhadap titik tujuan akhir.

Jarak Akhir	
Distance	0.609
Status	Pencarian Selesai

Dikarenakan jarak terhadap titik tujuan akhir telah masuk kedalam *threshold* yang ditentukan, maka langkah selanjutnya adalah mengurutkan rute perjalanan yang telah diolah oleh algoritma A* dari titik koordinat awal hingga menuju titik koordinat akhir tujuan robot. Tabel dibawah ini merepresentasikan output perencanaan rute algoritma A*.

No.	A* *node*	
	x	*y*
1.	1.550	2.930
2.	1.050	2.430
3.	1.050	1.930
4.	1.500	1.520

Hasil dari output perencanaan rute menggunakan metode A* akan direpresentasikan pada gambar dibawah ini agar memudahkan pembaca.



- **Simulasi Perhitungan Metode *Improved A-star Search* (A*) hasil *Manhattan Distance***

Simulasi Perhitungan Metode *Improved A-star Search* (A*) adalah tahapan *post-processing*. Tahapan ini adalah tahap akhir yang dilakukan pada penelitian ini dengan menyederhanakan rute perencanaan yang telah didapatkan oleh algoritma A*. Metode *Improved* A* membutuhkan kombinasi algoritma *Digital Differential Analyzer*. Langkah pertama yang dilakukan dalam simulasi ini adalah penentuan jarak aman terhadap tabrakan *obstacle* yang direpresentasikan pada tabel dibawah ini.

No.	**Jarak (meter)**
1.	0.65

Selanjutnya adalah membuat variabel penampung terhadap rute perencanaan robot, nilai variabel penampung berisi posisi *start* dari robot yang direpresentasikan pada tabel dibawah ini.

Rute Normalisasi	
x	*y*
2.050	3.430

Setelah melakukan penambahan rute perjalanan pada variabel penampung maka langkah selanjutnya adalah melakukan pencarian rute berdasarkan algoritma DDA yang direpresentasikan pada tabel dibawah ini.

No.	**Node**	**Koordinat Posisi**

		x	***y***
1.	*start point*	2.050	3.430
2.	*end point*	1.550	2.930

Setelah melakukan penentuan rute yang akan dicari menggunakan algoritma DDA. Selanjutnya adalah melakukan perhitungan untuk pencarian rute yang dilewati berdasarkan persamaan rumus 2.4, 2.5, 2.6, dan 2.7. Tabel dibawah ini merepresentasikan perhitungan persamaan DDA.

Pencarian Rute DDA	
dx	-0.500
dy	-0.500
step	1.000
x	-0.500
y	-0.500

Dari *step* yang dihasilkan, maka dilakukan perhitungan dengan menjumlahkan nilai koordinat (x, y) titik awal atau *start* point ditambahkan dengan koordinat (x, y) sesuai yang dihasilkan pada tabel di atas. Tabel dibawah ini merepresentasikan output perhitungan *step* algoritma DDA.

No	Koordinat	
	x	*y*
1.	1.550	2.930

Selanjutnya adalah melakukan perhitungan terhadap jarak *obstacle* untuk pengecekan apakah *node* bersangkutan aman atau tidak aman. Tabel dibawah ini merepresentasikan status jarak *node* terhadap *obstacle.*

No.	Jarak: *obstacle*	status *node*

1.	2.037	2.706	0.793	aman

Dikarenakan jalur bersangkutan aman dari tabrakan, maka langkah selanjutnya adalah merencanakan kembali koordinat rute dari hasil algoritma A* terhadap *node* selanjutnya untuk normalisasi rute. Tabel dibawah ini menunjukkan pencarian algoritma DDA terhadap *node* selanjutnya.

No.	**Node**	**Koordinat Posisi**	
		x	***y***
1.	*start point*	2.050	3.430
2.	*end point*	1.050	2.430

Dari hasil perencanaan *node* maka dilakukan proses algoritma DDA sesuai dengan data yang direpresentasikan pada tabel dibawah ini.

Pencarian Rute DDA	
dx	-1.000
dy	-1.000
step	1.000
x	-1.,000
y	-1.000

Dari *step* yang dihasilkan maka data posisi koordinat (x, y) akan dilakukan kalkulasi penjumlahan agar mencapai titik target tujuan. Tabel dibawah ini merepresentasikan *step* yang terjadi pada algoritma DDA.

No.	Koordinat	
	x	*y*
1.	1.050	2.430

Selanjutnya adalah melakukan perhitungan terhadap jarak *obstacle* untuk pengecekan apakah *node* bersangkutan aman atau tidak aman. Tabel dibawah ini merepresentasikan status jarak *node* terhadap *obstacle.*

No.	Jarak: *obstacle*			status *node*
1.	2.151	2.631	1.034	aman

Dikarenakan jalur bersangkutan aman dari tabrakan, maka langkah selanjutnya adalah merencanakan kembali koordinat rute dari hasil algoritma A* terhadap *node* selanjutnya untuk normalisasi rute. Tabel dibawah ini menunjukkan pencarian algoritma DDA terhadap *node* selanjutnya.

No.	**Node**	**Koordinat Posisi**	
		x	***y***
1.	*start point*	2.050	3.430
2.	*end point*	1.050	1.930

Dari hasil perencanaan *node* maka dilakukan proses algoritma DDA sesuai dengan data yang direpresentasikan pada tabel dibawah ini.

Pencarian Rute DDA	
dx	-1.000
dy	-1.500
step	2.000
x	-0.500
y	-0.750

Dari *step* yang dihasilkan maka data posisi koordinat (x, y) akan dilakukan kalkulasi penjumlahan untuk mencapai titik tujuan. Tabel dibawah ini merepresentasikan proses kalkulasi *step* pada *node*.

No.	Koordinat	
	x	*y*
1.	1.550	2.680
2.	1.050	1.930

Selanjutnya adalah melakukan perhitungan terhadap jarak *obstacle* untuk pengecekan apakah *node* bersangkutan aman atau tidak aman. Tabel dibawah ini merepresentasikan status jarak *node* terhadap *obstacle.*

No.	Jarak: *obstacle*			status *node*
1.	1.867	2.496	0.630	tidak aman
2.	1.982	2.316	1.109	aman

Dikarenakan pencarian yang dilakukan oleh algoritma DDA mengalami status jalur yang tidak aman terhadap tabrakan *obstacle*, maka langkah yang dilakukan adalah mengurangi indeks rute pencarian saat ini dengan nilai 1 atau menjadikan *node* rute pencarian sebelumnya sebagai titik tujuan awal yang akan dieksplorasi oleh algoritma DDA. Tabel dibawah ini merepresentasikan penyimpanan rute pencarian normalisasi DDA dan penentuan titik tujuan awal, dan titik tujuan akhir yang akan dieksplorasi oleh algoritma DDA.

Rute Normalisasi	
x	*y*
2.050	3.430
1.050	2.430

No.	**Node**	**Koordinat Posisi**	
		x	***y***

1.	*start point*	1.050	2.430
2.	*end point*	1.050	1.930

Dari hasil perencanaan *node* maka dilakukan proses algoritma DDA sesuai dengan data yang direpresentasikan pada tabel dibawah ini.

Pencarian Rute DDA	
dx	0.000
dy	-0.500
step	1.000
x	0.000
y	-0.500

Selanjutnya adalah melakukan perhitungan terhadap jarak *obstacle* untuk pengecekan apakah *node* bersangkutan aman atau tidak aman. Tabel dibawah ini merepresentasikan status jarak *node* terhadap *obstacle.*

No.	Koordinat	
	x	*y*
1.	1.050	1.930

Selanjutnya adalah melakukan perhitungan terhadap jarak *obstacle* untuk pengecekan apakah *node* bersangkutan aman atau tidak aman. Tabel dibawah ini merepresentasikan status jarak *node* terhadap *obstacle.*

No.	Jarak: *obstacle*			status *node*
1.	1.982	2.316	1.109	aman

Dikarenakan jalur bersangkutan aman dari tabrakan, maka langkah selanjutnya adalah merencanakan kembali koordinat rute dari hasil algoritma A* terhadap *node* selanjutnya untuk normalisasi rute. Tabel dibawah ini menunjukkan pencarian algoritma DDA terhadap *node* selanjutnya.

No.	**Node**	**Koordinat Posisi**	
		x	***y***
1.	*start point*	1.050	2.430
2.	*end point*	1.500	1.520

Dari hasil perencanaan *node* maka dilakukan proses algoritma DDA sesuai dengan data yang direpresentasikan pada tabel dibawah ini.

Pencarian Rute DDA	
dx	0.450
dy	-0.910
step	1.000
x	0.450
y	-0.910

Dari *step* yang dihasilkan maka data posisi koordinat (x, y) akan dilakukan kalkulasi penjumlahan untuk mencapai titik tujuan. Tabel dibawah ini merepresentasikan proses kalkulasi *step* pada *node*.

No.	Koordinat	
	x	*y*
1.	1.500	1.520

Selanjutnya adalah melakukan perhitungan terhadap jarak *obstacle* untuk pengecekan apakah *node* bersangkutan aman atau tidak aman. Tabel dibawah ini merepresentasikan status jarak *node* terhadap *obstacle.*

No.	Jarak: *obstacle*			status *node*
1.	1.481	1.715	1.004	aman

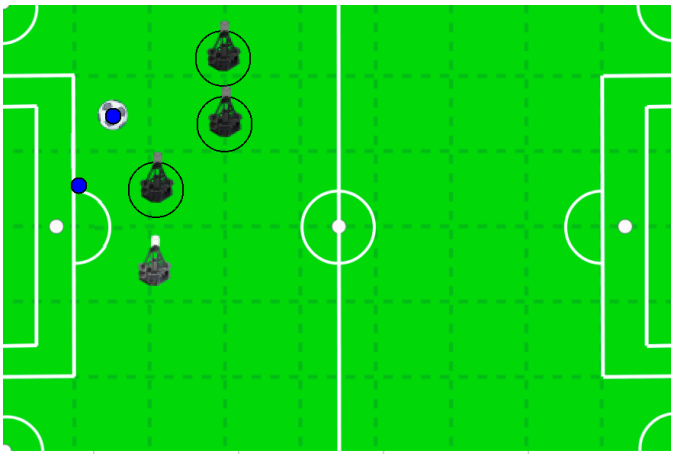
Dikarenakan output dari perencanaan rute dari hasil algoritma A* titik koordinat awal hingga titik koordinat tujuan akhir robot setelah dilakukan penyederhanaan rute tidak mengalami tabrakan terhadap *obstacle* atau dianggap rute yang paling efektif. Sebelumnya dilakukan proses untuk menghapus posisi koordinat pada variabel Rute Normalisasi indeks pertama. Hasil dari output penyederhanaan rute menggunakan algoritma *Improved* A* dengan kombinasi algoritma DDA adalah sebagai berikut.

Rute Normalisasi	
x	*y*
2.050	3.430
1.050	2.430
1.500	1.520

Rute Normalisasi Terbaru	
x	*y*
1.050	2.430
1.500	1.520

Hasil akhir dari perencanaan rute algoritma *Improved* A* direpresentasikan pada gambar dibawah ini agar mudah dipahami oleh pembaca. Output yang dihasilkan dari perencanaan *Improved* A* lebih sedikit dibandingkan output dari perencanaan rute dengan menggunakan algoritma A*. Hal ini menyebabkan pergerakan dari robot akan lebih efisien dan efektif

terhadap titik tujuan yang dilalui hingga mencapai titik koordinat akhir dari tujuan robot yang diinginkan.



Lampiran 3. Biodata Penulis

DATA PRIBADI

Nama	:	Muhammad Saifuddin Jazuli
NIM	:	1741720146
Tempat / Tanggal	:	Malang, 27 Juni 1998
Jenis Kelamin	:	Laki - laki
Agama	:	Islam
Jurusan / Prodi	:	Teknologi Informasi / DIV – Teknik Informatika
Alamat	:	Perumahan Saptoraya Blok TT. 12, RT03 RW13, Desa Saptorenggo, Kecamatan Pakis, Kabupaten Malang
Nomor Telepon	:	081233861720
Email	:	jazulienux@gmail.com

RIWAYAT PENDIDIKAN

2017 - 2021	:	Politeknik Negeri Malang
2014 - 2017	:	SMK Negeri 6 Malang
2011 - 2014	:	SMP Negeri 24 Malang
2005 - 2011	:	SD Negeri Polehan 2 Malang

Lampiran 4. Verifikasi Penulisan Laporan Skripsi

KEMENTERIAN PENDIDIKAN, KEBUDAYAAN
RISET DAN TEKNOLOGI
POLITEKNIK NEGERI MALANG
JURUSAN TEKNOLOGI INFORMASI
PROGRAM STUDI TEKNIK INFORMATIKA
JL. Soekarno Hatta PO Box 04 Malang Telp. (0341) 404424 pes. 1122

No. Skripsi : 596

FORM VERIFIKASI

ABSTRAK BAHASA INGGRIS DAN TATA TULIS BUKU LAPORAN SKRIPSI

Nama Mahasiswa : Muhammad Saifuddin Jazuli **NIM** : 1741720146
Tanggal Ujian : Jum'at, 11 Juni 2021
Judul : Penerapan Algoritma *Path Planning* Pada Robot Sepak Bola Beroda Politeknik Negeri Malang

NO	BAGIAN YANG DIVERIFIKASI	NAMA VERIFIKATOR	TANGGAL VERIFIKASI	TTD
1	Abstrak Berbahasa Inggris	Satrio Binusa Suryadi, S.S, M.Pd	Rabu, 15 September 2021	
2	Tata Tulis Buku Laporan Skripsi	Arie Rachmad Syulistyo, S.Kom., M.Kom	Jum'at, 17 September 2021