IMPLEMENTASI ALGORITMA *BREADTH FIRST SEARCH* DAN *OBSTACLE DETECTION* DALAM PENELUSURAN LABIRIN DINAMIS MENGGUNAKAN ROBOT LEGO

Budianto[1], Lasguido[1], Fahry Fathurrahman[1], dan Adi Wibowo[2]

[1]Fakultas Ilmu Komputer, Universitas Indonesia, Kampus Baru UI Depok, Jawa Barat, 16424, Indonesia
[2]Program Studi Ilmu Komputer/Informatika, Universitas Diponegoro, Jl. Prof. Soedarta, S.H. kampus Tembalang, Semarang, Jawa Tengah, 50275, Indonesia

E-mail: budianto71@ui.ac.id

Abstrak

Dewasa ini perkembangan teknologi di dunia robot edukasi berkembang pesat. Robot-robot edukasi ini sering digunakan dalam riset penelitian karena kemudahan-kemudahan yang diberikannya dari segi perangkat keras. Salah satu contoh robot edukasi adalah robot LEGO Mindstorms NXT. Pada penelitian ini robot LEGO dibangun dalam bentuk robot *line follower*. Robot ini mampu menelusuri dan mencari jalan keluar dari labirin dinamis. Dalam menelusuri dan mencari jalan keluar, robot LEGO menggunakan algoritma *Breadth First Search* dan *Manhattan Distance* dalam memutuskan jalan mana yang harus diambil. Ketika menemui objek halangan, robot LEGO akan mengenali dan menghindari objek halangan tersebut dengan algoritma *Obstacle Detection* yang dimilikinya. Hasil implementasi membuktikan bahwa algoritma penelusuran labirin dinamis ini dapat diimplementasikan pada robot LEGO meskipun terdapat banyak keterbatasan dalam robot LEGO.

Kata Kunci: *breadth first search, labirin dinamis, LEGO mindstroms NXT, line following, obstacle detection*

Abstract

Nowadays, the development of technology in educational robots is rapidly evolving. Educational robots are often used in research studies because they provide convenience in terms of hardware. One example is the educational robot LEGO Mindstorms NXT robot. In this research, LEGO robots built in the form of line follower robot. Robot is able to browse and find a way out of the dynamic labyrinth. In track and find a way out, LEGO robot uses an algorithm Breadth First Search and Manhattan Distance in deciding which path to take. When encountering an obstacle object, LEGO robot will recognize and avoid that obstacle objects with Obstacle Detection algorithm. The results prove that the implementation of a dynamic maze search algorithm can be implemented on a LEGO robot even though there are many limitations in LEGO robot.

Keywords: *breadth first search, LEGO mindstroms NXT, line following, obstacle detection, the dynamic labyrinth*

1. Pendahuluan

Teknologi diciptakan oleh manusia untuk mempermudah dan meningkatkan kualitas hidup. Robot merupakan salah satu bentuk teknologi yang perkembangannya sangat pesat. Teknologi robot ini sudah semakin canggih dan hampir tidak bisa dipisahkan dalam berbagai proses industri maupun kegiatan akademis di dunia. Robotika merupakan cabang ilmu yang mempelajari tentang robot [1]. Cabang ilmu ini merupakan gabungan dari beberapa cabang ilmu pengetahuan seperti teknik mesin, teknik elektro, ilmu komputer, serta ilmu pengetahuan sosial dan kognitif.

Robot LEGO merupakan jenis robot yang dapat diprogram [2-5]. Robot ini dibuat oleh Lego Group. Robot ini terdiri dari banyak bagian-bagian kecil beserta sensor dan kabelnya. Robot ini pertama kali dirilis pada tahun 1998. Versi terakhir dari robot LEGO ini dirilis pada tahun 2009. Versi terakhir dari robot LEGO ini disebut dengan Lego Mindstorms NXT 2.0. Robot ini banyak digunakan di dalam penelitian maupun sebagai sarana edukasi bagi anak-anak atau orang-orang awam.

Suatu robot tidak terlepas dari algoritma kecerdasan buatan yang berada di belakangnya [6][7]. Dibalik kecanggihan teknologi suatu robot, robot tersebut tidak dapat berbuat banyak jika proses kecerdasan yang berada di dalamnya tidak dimaksimalkan dengan baik. Salah satu masalah yang dapat diselesaikan oleh robot adalah

masalah penelusuran labirin. Di bidang ilmu komputer, terdapat banyak algoritma yang dapat menyelesaikan permasalahan penelusuran labirin. Pada umumnya algoritma yang sering digunakan adalah algoritma yang memanfaatkan teori graf. Algoritma-algoritma tersebut biasanya diterapkan dalam penyelesaian masalah penelurusan labirin statis. Di aplikasi dunia nyata, labirin sering dijumpai bersifat dinamis.

Meskipun algoritma penelusuran graf ini sangat mudah diimplementasikan pada komputer, algoritma seperti ini cukup sulit untuk diimplementasikan pada robot asli, apalagi pada robot LEGO yang sensornya terbatas. Penelitian ini bertujuan untuk membuktikan kemampuan robot LEGO menjalankan algoritma penelurusan labirin dinamis yang kompleks dengan segala keterbasan robot LEGO.

2. Metodologi

Algoritma *Breadth First Search* (BFS) merupakan algoritma penelusuran graf [2][8]. *Breadth First Search* merupakan algoritma sederhana yang digunakan untuk melakukan penelusuran pada graf, algoritma ini merupakan algoritma yang menjadi pola dasar dan fundamental bagi banyak algoritma graf.

Algoritma ini menelusuri batas ambang antara *node* yang sudah dikunjungi dan *node* yang belum dikunjungi. Algoritma ini mengunjungi *node* sedemikian sehingga setiap *node* yang berjarak *s* dari *node* awal akan dikunjungi sebelum mengunjungi *node* yang berjarak *s*+1 dari *node* awal (*layer order*).

Penelusuran graf pada algoritma Breadth First Search dilakukan dengan menelusuri setiap *node* dari *node* awal. Struktur data antrian (*queue*) digunakan untuk melakukan proses ekspansi *Breadth First Search*. Struktur data antrian digunakan untuk menampung setiap *node* atau kemungkinan *node* yang valid untuk dilalui. Sifat dari *queue* adalah *First In First Out*. Oleh karena itu setiap *node* yang dipilih dan di-*expand* pertama kali oleh algoritma ini akan pertama kali juga diproses dan ditelusuri. Sifat dari *queue* tersebut mempertahankan sifat dari algoritma *Breadth First Search* itu sendiri, di mana algoritma ini menelusuri secara *layer order*. *Pseudocode* dari algoritma ini dapat dilihat pada gambar 1.

Setiap *node* yang telah ditelusuri oleh algoritma *Breadth First Search* tidak akan ditelurusi lagi. Untuk menandai *setiap node* yang sudah atau belum ditelusuri algoritma *Breadth First Search* menggunakan penanda berupa variabel tipe warna yang diberikan ke setiap *node*. Di mana dalam algoritma ini terdapat 3 jenis warna yaitu putih, abu-abu, dan hitam. Warna putih menandakan bahwa suatu *node* belum pernah ditelusuri atau di-*expand*. Warna abu-abu menandakan bahwa node tersebut telah di-*expand* dan berada di dalam antrian. *Node* warna hitam menandakan bahwa node tersebut telah ditelusuri dan telah berada di luar antrian.

```
BFS(G, s)
1 for each vertex u  V [G] - {s}
2     do color[u] ← WHITE
3         d[u] ← ∞
4         π[u] ← NIL
5 color[s] ← GRAY
6 d[s] ← 0
7 π[s] ← NIL
8 Q ← ∅
9 ENQUEUE(Q, s)
10 while Q ≠ ∅
11     do u ← DEQUEUE(Q)
12         for each v  Adj[u]
13             do if color[v] = WHITE
14                then color[v] ← GRAY
15                     d[v] ← d[u] + 1
16                     π[v] ← u
17                     ENQUEUE(Q, v)
18         color[u] ← BLACK
```

Gambar 1. *Pseudocode* algoritma BFS.

Masukan dalam algoritma ini adalah sebuah graf yang terdiri dari *nodes*, *edges*, *node* awal yang menjadi posisi awal, dan *node* akhir yang menjadi posisi tujuan dalam graf. Keluaran hasil pemrosesan algoritma *Breadth First Search* ini adalah jalur terpendek dari *node* awal hingga *node* tujuan yang dapat dilalui secara aman.

Algoritma *Obstacle Detection* yang digunakan dalam menangani penghalang membaca masukan dari sensor dan memperbaharui data labirin yang ada relatif terhadap letak-letak posisi objek-objek penghalang [7][9-11]. Objek-objek penghalang di labirin ini terdiri dari 3 macam yaitu objek halangan berupa Robot Si Kereti yang bergerak secara acak di dalam labirin, objek halangan statis yang diletakan secara acak di dalam labirin, dan objek halangan di pintu keluar (*node* posisi tujuan). Adanya sifat dari objek-objek yang perlu dihindari di dalam arena labirin ini membuat sifat labirin menjadi dinamis.

Dalam mendeteksi objek penghalang objek digunakan sensor berupa sensor ultrasonik dan sensor sentuh. Untuk menyederhanakan pola pelacakan, ketika pendeteksian objek berlangsung objek yang dideteksi tidak *akan* bergerak, dengan demikian asumsi yang digunakan dalam pendeteksian objek adalah *non-moving object* [3][12]. Untuk keperluan pendeteksian tersebut sensor ultrasonik berperan dalam mendeteksi objek yang berada jauh terhadap robot (berjarak satu *node* dari robot), sedangkan sensor sentuh

berperan dalam mendeteksi objek yang berada dekat terhadap robot. Berbeda dengan sensor *bluetooth*, pada algoritma deteksi penghalang, sensor *bluetooth* berperan dalam menangkap dan memberikan informasi dari objek dinamis yang bergerak (Robot Si Kereti).

Manhattan distance adalah formula untuk menghitung jarak antara dua titik. Perhitungan *Manhattan distance* untuk mencari jarak minimal dari dua buah titik (x1, y1) dan b(x2, y2) dapat dilakukan dengan menghitung nilai |x2-x1| + |y2-y1| [2][8].

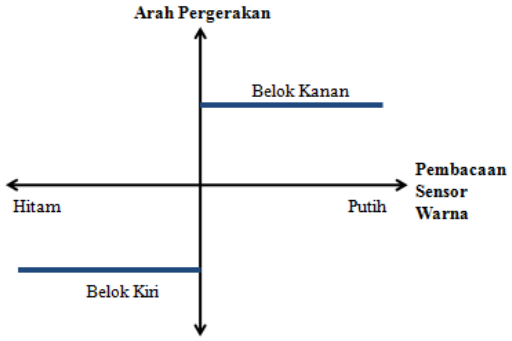
Nama Manhattan sendiri diambil dari daerah Manhattan suatu daerah kecil di Kota New York, yang memiliki jalan yang berbentuk kisi-kisi segi empat. Jarak antara dua lokasi yang berada di setiap kisi-kisi daerah Manhattan dapat diukur berdasarkan jalur horizontal dan vertikal yang terbentuk diantara kisi-kisi jalan tersebut. Perhitungan jarak antara dua lokasi dapat dihitung dengan dengan menggunakan perhitungan Phytagoras terhadap total jalur horizontal dan jalur vertikal yang terbentuk.

Algoritma *Line Following*, untuk mengikuti garis labirin, pada robot LEGO digunakan algoritma line following dua *state* (gambar 2). Algoritma ini memiliki dua *state*, yaitu apakah robot bergerak ke *kiri* atau ke kanan. Pergerakan ke kiri ataupun ke kanan tergantung pada nilai yang dibaca sensor warna. Dalam bahasa NXC (*Not-eXactly-C*), algoritma ini diimplementasikan dengan dua buah kondisional, yaitu kondisional ketika akan bergerak ke kiri dan sisanya kondisional untuk bergerak ke kanan.

Robot LEGO, dewasa ini, banyak dijumpai penggunaan robot LEGO dalam riset di bidang robotik. Banyak penelitian-penelitian yang memanfaatkan dan menggunakan robot lego untuk melakukan simulasi. Hal ini dikarenakan robot LEGO menawarkan stabilitas dan *reliabilitas* yang menyederhanakan dan memberikan pendekatan baru dalam bidang penelitian robotik [6][10].

Salah satu hal yang paling penting pada suatu sistem otomatis adalah mendapatkan informasi mengenai lingkungan [11-14]. Dalam mengambil informasi dari lingkungan digunakan berbagai macam jenis sensor. Informasi-informasi yang berguna dan bermanfaat dapat diperoleh dengan mengambil dan mengolah data-data yang diperoleh dari sensor [1].

Menurut kategorinya sensor dapat terbagi menjadi *proprioceptive sensors*, *exteroceptive sensors*, *passive sensors*, dan *active sensors*. *Proprioceptive sensors* adalah sensor yang mengukur nilai secara internal di dalam sistem suatu robot, contoh dari *proprioceptive sensors* ini adalah kecepatan motor atau status baterai robot. *Exteroceptive sensors* adalah sensor yang mengambil informasi dari keadaan lingkungan robot, contoh dari *exteroceptive sensors* ini adalah nilai sensor jarak atau intensitas cahaya. *Passive sensors* adalah sensor yang mengambil energi dari lingkungan. *Active sensors* adalah sensor yang mengeluarkan energi dan kemudian mengukur reaksi yang diciptakan dari energi tersebut, sensor ini memiliki dampak kepada lingkungan robot, dan performa *active sensors* ini jauh lebih baik daripada *passive sensors*. Sensor yang digunakan di dalam robot LEGO pada penelitian ini adalah sensor sentuh, warna, ultrasonik, dan sensor *bluetooth*.



Gambar 2. Algoritma *line following* dua *state.*

Gambar 3. Sensor sentuh.

Sensor Sentuh atau sensor sentuh (gambar 3) merupakan sensor yang tergolong ke dalam *exteroceptive sensors* dan *passive sensors*. Sensor ini menerima rangsangan fisik berupa sentuhan dari suatu objek. Ketika sensor ini menerima sentuhan, maka tombol yang berada di sensor ini akan tertekan, kemudian sensor ini akan mengirimkan impuls ke dalam robot (NXT Brick). Hal yang sama terjadi ketika kontak fisik yang terjadi pada sensor ini berakhir, ketika tombol yang berada di sensor ini terlepas, maka sensor juga akan mengirimkan impuls.

Gambar 4. Sensor warna.

Sensor Warna di Lego Mindstrom NXT, (gambar 4) merupakan sensor yang tergolong ke dalam *exteroceptive sensors*. Dalam penggunaannya sensor warna ini dapat berperan sebagai *active sensors*. Sebagai *active sensors* sensor cahaya ini mengemisikan sinar ke suatu objek, pantulan sinar ini kemudian akan ditangkap kembali oleh bagian sensor warna yang sensitif terhadap panjang gelombang cahaya.

Sensor cahaya ini mampu membedakan antara berbagai *macam* warna dengan intensitas RGB yang dapat berkisar masing-masing antara 0 sampai dengan 255. Sensor ini dapat membaca intensitas warna dengan mengenali gelombang cahaya yang dipantulkan dari suatu objek dan mengukur panjang gelombang yang dipantulkan dari permukaan objek tersebut.

Gambar 5. Sensor ultrasonik.

Sensor ultrasonik (gambar 5) tergolong ke dalam *exteroceptive sensors*. Sensor ini mengambil informasi dari keadaan lingkungan robot. Dalam *penggunaannya* sensor ini tergolong ke dalam *active* sensor. Sebagai *active* sensor sensor ultrasonik ini mengeluarkan gelombang ultrasonik yang memiliki frekuensi diantara 40 dan 80 kHz. Gelombang ultrasonik ini kemudian akan dipantulkan dan pantulan gelombang ultrasonik tersebut akan ditangkap lalu diukur oleh sensor ultrasonik itu sendiri.

Sensor ultrasonik ini digunakan untuk mengukur jarak antara robot dengan suatu objek yang berada di depan sensor ultrasonik ini. Sensor ultrasonik mampu mengukur jarak 0 sampai dengan 255 sentimeter dengan presisi ±3 cm [4][8].

Cara pengukuran jarak objek dari sensor ultrasonik ini menyerupai prinsip fisika dari kelelawar. Sebelumnya sensor ultrasonik telah mendapatkan jarak waktu tempuh gelombang ultrasonik dari sensor ultrasonik-objek-sensor ultrasonik, waktu tersebut kemudian dikalkulasi untuk mendapatkan jarak antara sensor ultrasonik dengan suatu objek. Perhitungan untuk mengukur jarak adalah $s= (t \times v)/2$, di mana s adalah jarak objek dengan robot, t adalah waktu tempuh gelombang ultrasonik, dan v adalah kecepatan gelombang ultrasonik. Penggunaan lebih dari satu sensor ultrasonik dapat menyebabkan *error* karena sensor ultrasonik yang satu dapat mempengaruhi sensor ultrasonik yang lainnya.

Sensor *bluetooth* (gambar 6) yang digunakan di dalam Lego Mindstroms NXT ini sudah terintegrasi di dalam NXT Brick. Sensor *bluetooth* ini berfungsi untuk komunikasi paket data antar NXT Brick. Jumlah maksimum komunikasi *bluetooth* yang dapat digunakan oleh suatu NXT Brick adalah empat, jadi satu NXT Brick bisa berkomunikasi melalui *bluetooth* maksimum ke empat NXT Brick yang berbeda.

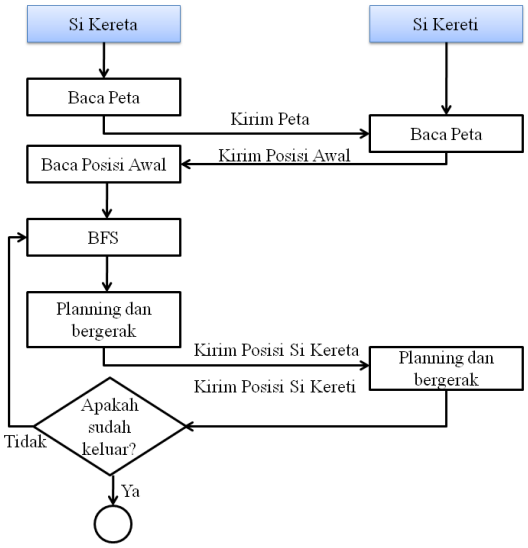
Gambar 6. Sensor *bluetooth* dalam NXT Brick.

Bricx Command Center (Not eXactly C), untuk mempermudah proses implementasi algoritma maka digunakan IDE Bricx *Command Center* untuk memfasilitasi pembuatan program pada robot Lego. Bahasa pemrograman yang digunakan untuk memrogram robot Lego adalah bahasa pemrograman *Not-eXactly-C* (NXC). NXC adalah bahasa pemrograman *high level* yang mampu berjalan di atas robot Lego Mindstrom NXT, merupakan gabungan dari bahasa NBC (*Next Byte Codes*) dan bahasa *assembly*. Bahasa NXC ini memiliki *syntax* seperti bahasa C.

Arsitektur dari algoritma penelusuran labirin ini dapat dilihat pada gambar 7. Robot pada awalnya sudah memiliki peta labirin yang akan ditelurusi, namun pada peta tersebut tidak ada informasi tentang halangan baik yang diam maupun yang bergerak. Si Kereta adalah robot yang akan menelusuri labirin untuk menemukan jalan keluar, sementara Si Kereti adalah penghalang dinamis yang akan bergerak secara acak di labirin. Dikarenakan robot LEGO memiliki keterbatasan dalam melakukan lokalisasi posisi robot dan posisi halangan dinamis, maka robot Si Kereta dan Si Kereti (gambar 8) akan berkomunikasi dengan *bluetooth* untuk menukar informasi posisi masing-masing.

Alur algoritma yang akan diimplementasikan tersebut adalah sebagai berikut: (a) Si Kereta membaca peta labirin dari berkas peta labirin yang memiliki ekstensi .map. Contoh berkas peta labirin dapat dilihat pada gambar 9. Angka 1 menunjukkan bahwa lokasi tersebut dapat dilewati, sementara angka 0 menunjukkan bahwa lokasi tersebut tidak dapat dilewati. Simbol ‘S’ pada peta menunjukkan pintu masuk labirin. Simbol ‘E’ pada peta menunjukkan pintu keluar labirin. Simbol ‘K’ pada peta menunjukkan kemungkinan posisi robot Si Kereti. (b) Si Kereta

akan mengirimkan peta labirin tersebut kepada Si Kereti dengan menggunakan *bluetooth*. Si Kereti lalu merespon dengan mengirimkan posisinya saat ini kepada Si Kereta. (c) Si Kereta akan menelurusi semua kemungkinan pintu keluar untuk mengetahui pintu keluar yang tepat.



Gambar 7. Arsitektur algoritma penelusuran labirin.

Gambar 8. Si Kereta dan Si Kereti.

Pada awalnya, semua kemungkinan pintu keluar tersebut akan diurutkan dengan menggunakan *Heuristic Manhattan Distance*. Jadi Si Kereta akan menelusuri pintu keluar yang jaraknya ke pintu masuk paling kecil berdasarkan *Manhattan Distance*. (d) Si Kereta dan Si Kereti akan berjalan bergantian. Sinkronisasi pergerakan ini dilakukan dengan menggunakan *bluetooth*. Si Kereta akan bergerak pada jalan terpendek ke pintu keluar, sementara Si Kereti akan bergerak secara acak. Ketika Si Kereta tidak dapat menemukan jalan ke pintu keluar (karena terhalang oleh Si Kereti), Si Kereta akan menunggu. (e) Perpindahan robot dari satu *node* ke *node* lainnya dilakukan dengan menggunakan algoritma *line following*. Persimpangan jalan ditandai dengan kotak warna biru. Kotak warna biru ini berfungsi sebagai *landmark* sehingga robot mengetahui bahwa dia sedang berada di persimpangan jalan.

```
8 5
1 0 1 1 1 1 1 0 1
1 0 0 0 1 0 1 0 1
1 1 1 1 1 0 1 1 1
0 0 0 0 1 0 1 0 1
E 0 1 1 1 1 K 1 1
1 0 0 0 1 0 1 0 1
1 1 1 1 1 0 1 1 1
0 0 1 0 0 0 0 0 0
1 1 1 1 1 0 1 1 1
1 0 1 0 1 0 1 0 1
1 1 K 0 1 1 1 1 K
0 0 1 0 1 0 1 0 1
1 1 1 1 1 1 1 1 1
1 0 0 0 1 0 0 0 1
1 1 E 0 S 0 E 1 1
```

Gambar 9. Contoh berkas peta labirin.

Pada pengiriman data menggunakan *bluetooth* pada robot LEGO, dapat terjadi penimpaan data seandainya robot A mengirim data ke robot B dan robot B belum sempat membacanya. Untuk mengatasi hal tersebut, kami membuat protokol komunikasi dengan menggunakan *acknowledgment* (ack) sehingga pesan yang dikirim pasti sampai dan terbaca dengan baik oleh robot tujuan. Gambaran fungsi *send* dan *receive* dalam bahasa NXC dapat dilihat pada gambar 10.

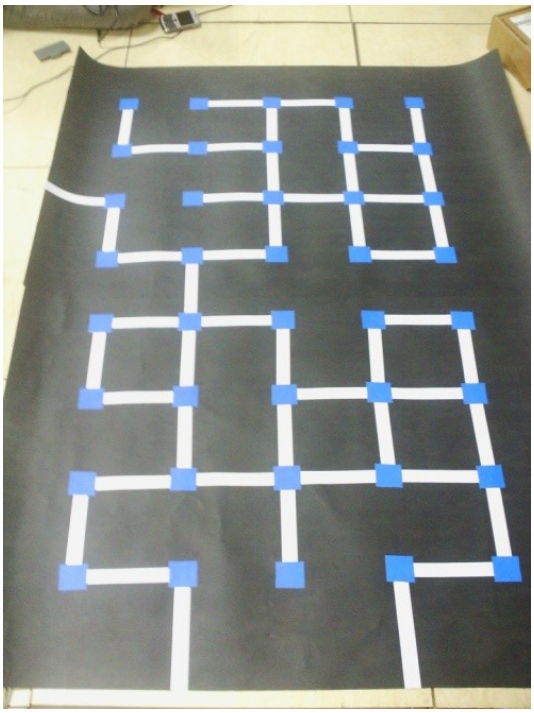
```
void receiveInt(int &ret){
    if(!use_bluetooth) return;

    ret = 0;
    while(ReceiveRemoteNumber(INBOX,true,ret)
!= NO_ERR){
        Wait(10);
    }
    SendRemoteNumber(BT_CONN,OUTBOX,0xFF);
}

void sendInt(int msg){
    if(!use_bluetooth) return;

    SendRemoteNumber(BT_CONN,OUTBOX,msg);
    int ack = 0;
    until(ack==0xFF) {

until(ReceiveRemoteNumber(INBOX,true,ack) ==
NO_ERR);
    }
}
```

Gambar 10. Protokol komunikasi *bluetooth* dalam bahasa NXC.

3. Hasil dan Pembahasan

Skenario uji coba digunakan untuk mengetahui performa dari algoritma yang telah dijelaskan sebelumnya. Uji coba algoritma penelurusan labirin ini dilakukan dalam dua skenario lapangan yang berbeda. Skenario Lapangan Kecil dilakukan pada lapangan kecil berukuran 8×5 kotak sementara Skenario Lapangan Besar dilakukan pada lapangan besar berukuran 8×6 kotak. Lapangan Kecil yang digunakan untuk uji coba dapat dilihat pada gambar 11.

Untuk setiap skenario akan dilakukan beberapa percobaan sebagai berikut. (a) Robot tunggal tanpa halangan di pintu keluar. (b) Robot tunggal dengan halangan di pintu keluar. (c) Robot tunggal dengan halangan di pintu keluar dan halangan statis pada labirin. (d) Dua robot, salah satu robot berperan sebagai halangan dinamis.



Gambar 11. Lapangan kecil.

Performa dari algoritma ini akan dibandingkan dengan algoritma tangan kanan. Algoritma tangan kanan tidak mencari jalan terpendek dari awal hingga tujuan melainkan dengan mengikuti arah kanan pada labirin.

Implementasi Algoritma *Breadth First Search* dan *Obstacle Detection*, pada setiap skenario uji coba digunakan algoritma *Breadth First Search* untuk mencari jalur yang akan dilalui robot. Jalur yang dicari oleh algoritma tersebut merupakan jalur dari *node* posisi awal hingga *node* posisi tujuan. Di setiap langkah pergerakan robot, robot selalu melakukan perhitungan keadaan labirin terhadap objek-objek penghalang yang ada [4][6][7][10][13-15]. Hasil dari perhitngan tersebut merupakan jalur yang aman untuk dilalui oleh robot. Tanpa adanya algoritma ini robot tidak akan mengetahui jalur efisien mana yang harus dilaluinya.

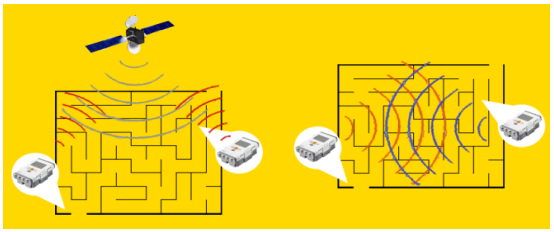
Pada skenario uji coba dengan halangan, jika robot berhasil mendeteksi adanya halangan statis dengan sensor ultrasonik, maka robot akan meng-*update* peta labirinnya. Robot akan mengubah informasi peta labirin tersebut dari bisa dilewati menjadi tidak bisa dilewati. Dengan peta labirin yang baru, maka robot harus kembali mencari jalur tependek untuk menuju *node* posisi tujuannya. Proses pencarian ulang *node* posisi tujuan dari robot ini dilakukan dan ditangani oleh algoritma *Breadth First Search*. Algoritma *Breadth First Search* ini akan menggunakan data peta labirin yang baru sebagai *input* dari perhitungannya. Karena kompleksitas algoritma ini linier terhadap jumlah *node* labirin, algoritma ini dapat dijalankan dengan cepat.

Pada skenario uji coba dengan dua robot, salah satu robot berperan sebagai halangan dinamis yang bergerak secara random diatas labirin. Pada skenario ini jika robot yang bertugas untuk mencari dan berjalan menuju *node* posisi tujuan menemui halangan berupa halangan statis ataupun halangan dinamis maka robot akan memperbaharui peta labirin yang dimilikinya menjadi tidak bisa dilewati. Karena jika robot tersebut tetap melalui jalan yang memiliki halangan objek, maka akan terjadi tabrakan yang tidak diinginkan.

Pada setiap skenario, untuk mendeteksi halangan statis robot menggunakan sensor ultrasonik.Sedangkan untuk mendeteksi halangan dinamis robot menggunakan sensor *bluetooth*. Sensor *bluetooth* ini sendiri merupakan representasi abstrak yang merupakan bentuk simplifikasi dari sensor GPS (gambar 12). Pada sensor GPS terdapat sistem sensor yang memantau posisi setiap objek yang ada di suatu peta. Sensor GPS kemudian akan mengambil informasi dari setiap objek yang ada di peta kemudian mengirimkan informasi tersebut kepada setiap objek yang ada di dalam peta.

Tujuan pemberian informasi tersebut adalah supaya setiap objek yang ada di dalam peta mengetahui posisi dirinya dan posisi objek lain yang berada di dalam peta labirin. Berbeda dengan sensor *bluetooth*, pada sensor ini, robot saling berkomunikasi antara satu dengan yang lainnya. Tujuan dari komunikasi tersebut adalah mengirimkan pesan lokasi yang sedang

dikunjungi. Informasi tersebut kemudian akan menjadi informasi lokasi yang dapat dimanfaatkan oleh robot lainnya. Informasi tersebut membuat robot yang lainnya untuk tidak melalui *node* yang sedang dikunjungi oleh robot yang memberikan pesan tersebut.



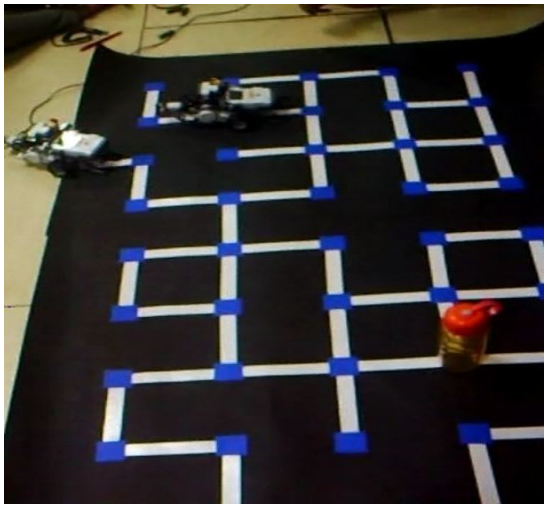
Gambar 12. Simplifikasi lokalisasi robot.

Untuk setiap skenario robot, terdapat beberapa (*multiple*) *node* posisi tujuan, disetiap *node* posisi tujuan tersebut diberikan objek penghalang yang menandakan posisi tujuan tersebut adalah pintu keluar yang benar atau tidak. Informasi mengenai objek halangan tersebut tidak diberikan, informasi tersebut hanya bisa dideteksi oleh sensor sentuh yang
terdapat pada robot. Ketika robot memasuki *node* posisi tujuan dan terhalang oleh objek halangan. Maka robot akan kembali mundur dan mencari jalan keluar berikutnya.

Dalam proses pencarian *node* jalan keluar yang ingin ditelusuri terlebih dahulu dilakukan dengan menggunakan teknik *Manhattan Distance*. Dengan teknik perhitungan *Manhattan Distance* setiap informasi kemungkinan *node* posisi tujuan akan diurutkan sesuai dengan jarak yang paling minimal. Tujuan pengurutan pintu-pintu keluar adalah agar robot menelusuri labirin dengan efisien dan tidak berputar-putar. *Node* posisi tujuan yang memiliki jarak paling kecil akan ditelusuri terlebih dahulu. Proses pencarian *node* posisi tujuan ini dilakukan secara terus menerus oleh robot sampai *node* posisi tujuan yang sebenarnya ditemukan. Dengan kata lain, robot akan berhenti dan selesai ketika robot tersebut mencapai *node* tempat tujuannya yang sebenarnya, yaitu *node* posisi tujuan yang tidak memiliki objek halangan.

Hasil uji coba untuk semua skenario berhasil dengan baik. Robot dapat menelusuri labirin dan menemukan jalan keluar tanpa menabrak halangan yang diam maupun halangan yang dinamis. Gambar 13 menunjukkan robot berhasil mencapai pintu keluar tanpa menabrak penghalang statis maupun penghalang dinamis.

Perbandingan performa antara algoritma BFS dan algoritma tangan kanan dapat dilihat pada tabel I. Secara umum, performa algoritma BFS lebih baik daripada algoritma tangan kanan. Algoritma BFS menelusuri *node* yang lebih sedikit daripada algoritma tangan kanan karena algoritma BFS mencari jalan terpendek, sementara algoritma tangan kanan hanya menelusuri arah kanan pada labirin. Akan tetapi, pada kasus labirin yang memiliki banyak pintu keluar seperti ini, ada kasus di mana algoritma tangan kanan memiliki performa yang lebih baik daripada algoritma BFS.



Gambar 13. Robot berhasil sampai ke pintu keluar.

Pada skenario keempat, terdapat tiga pintu keluar dengan penghalang di depan dua pintu keluar sehingga hanya ada satu pintu keluar yang benar. Algoritma BFS akan menelusuri pintu keluar yang terdekat berdasarkan *Manhattan Distance* sehingga robot sampai ke pintu yang salah dahulu baru kemudian menemukan pintu yang benar. Algoritma tangan kanan akan menelurusi labirin berdasarkan tangan kanan dan karena konfigurasi labirin yang seperti itu, robot langsung menemukan pintu keluar yang benar.

Sementara itu, pada algoritma BFS, jika jalan yang mengarah ke pintu keluar terhalangi oleh penghalang dinamis, robot akan menunggu sampai penghalang tersebut pindah dari jalan tersebut. Oleh karena itu, pada skenario kedelapan, terlihat bahwa waktu robot BFS lebih lama daripada algoritma tangan kanan, meskipun jumlah *node* yang ditelusuri algoritma BFS lebih sedikit.

Untuk labirin dinamis dengan banyak pintu keluar seperti pada penelitian ini, ada kasus algoritma tangan kanan lebih baik dibandingkan algoritma BFS meskipun secara umum algoritma BFS lebih baik.

TABEL I
PERBANDINGAN ALGORITMA BFS DAN ALGORITMA TANGAN KANAN

Skenario	Lapangan	Jumlah penghalang statis	Jumlah penghalang dinamis	Algoritma BFS		Algoritma Tangan Kanan	
				Node yang ditelusuri	Waktu (s)	*Node* yang ditelusuri	Waktu (s)
1	8×5	0	0	6	9	6	9
2	8×5	2	0	24	44	44	81
3	8×5	3	0	27	49	49	86
4	8×5	2	1	15	64	9	37
5	8×6	0	0	12	20	12	20
6	8×6	2	0	25	49	29	53
7	8×6	3	0	41	77	59	110
8	8×6	2	1	33	197	42	154

4. Kesimpulan

Algoritma BFS dapat diterapkan kepada robot LEGO untuk menelusuri labirin dinamis. Algoritma BFS ini berperan dalam pencarian jalan keluar dari suatu labirin. Kombinasi dari fungsi heuristik dan sensor-sensor dapat membuat robot LEGO mampu menelusuri labirin dinamis. Fungsi heuristik diterapkan pada pemilihan *node* titik tujuan dengan menggunakan *Manhattan Distance*. Sensor-sensor yang ada pada robot memberikan informasi mengenai lokasi-lokasi penghalang yang menghalangi jalan dari robot. Melalui penerapan dari Algoritma BFS, fungsi heuristik dan Algoritma *Object Detection* dengan menggunakan sensor-sensor yang ada pada robot, robot LEGO dapat keluar dari sebuah labirin yang dinamis.

Meskipun robot LEGO memiliki banyak keterbatasan dalam hal sensor, komunikasi, dan lainnya, robot LEGO dapat menjalankan algoritma yang kompleks dengan baik. Berdasarkan hasil penelitian ini, terlihat bahwa robot LEGO dapat digunakan untuk melakukan berbagai macam hal yang kompleks.

Referensi

[1] R. Siegwart & I.R. Nourbakhsh, *Introduction to Autonomous Mobile Robots*, The MIT Press, Massachusetts, 2004.

[2] J.F. Kelly, *Lego Mindstroms NXT 2.0: The King's Treasure*, Springer, New York, 2009.

[3] D.J. Perdue & L. Valk, *The Unofficial LEGO MINDSTROMS NXT 2.0 Inventor's Guide,* No Starch Press, San Francisco, 2011.

[4] J.D. Kelleher, Sensor Sensitivity, School of Computing, http://www.comp.dit.ie/jkelleher/appliedcomputing/week4/Sensors2.pdf, 2009, retrieved December 1, 2010.

[5] T. Braunl, *Embedded Robotics: Mobile Robot Design and Applications with Embedded Systems,* Springer, Berlin, 2008.

[6] L. House & J. Hill, *Designing Autonomous Mobile Robots: Inside the Mind of an Intelligent Machine*, Elsevier, New York, 2004.

[7] D. Lee, *The Map-Building and Exploration Strategies of a Simple Sonar-Equipped Mobile Robot,* Cambridge University Press, Cambridge, 2003.

[8] T.H. Cormen, C.E. Leiserson, R.L. Rivest, & C. Stein, *Introduction to Algorithm.* 2nd ed, The MIT Press, Massachusetts, 2001.

[9] X. Ming, *Fundamentals of Robotics: linking perception to action,* World Scientific Publishing, Singapore, 2003.

[10] S. Zhao, W. Tan, C. Wu, & C. Li, "Research on Robotic Popular Science System Based on LEGO Bricks" *International Conference on Computer Science and Software Engineering*, vol. 5, 741-744, 2008.

[11] S. Patnaik, L.C. Jain, S.G. Tzafestas, G. Resconi, & A. Konar, *Innovations in Robot Mobilty and Control,* Springer, Heidelberg, 2005.

[12] SwedeTrack System, Some Theory behind Obstacle Detection, http://www.swedetrack.com/obstact.htm, 2007, retrieved December 1, 2011.

[13] U. Nehmzow, *Mobile Robotics: A Practical Introduction*, Springer, London, 2003.

[14] G. Dudek & M. Jenkin, *Computational Principles of Mobile Robotics*, Cambridge University Press, Cambridge, 2010.

[15] W. Jacak, *Intelligent Robotic Systems: Design, Planning, and Control,* Kluwer Academic Publisher, New York, 1999.