TUGAS AKHIR –TE 141599

PERANCANGAN SISTEM NAVIGASI DAN KENDALI ROBOT *QBOT* MENGGUNAKAN METODE JARINGAN SARAF TIRUAN DAN KONTROLER PID

Eva Sri Oktavia Ningrum
NRP 2214 105 094

Dosen Pembimbing
Ir. Rusdhianto Effendi. AK, MT.

JURUSAN TEKNIK ELEKTRO
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember
Surabaya 2016

FINAL PROJECT –TE 141599

DESIGN OF QBOT ROBOT NAVIGATION AND CONTROL SYSTEM USING NEURAL NETWORK METHOD AND PID CONTROLLER

Eva Sri Oktavia Ningrum
NRP 2214 105 094

Supervisor
Ir. Rusdhianto Effendi. AK, MT.

DEPARTMENT OF ELECTRICAL ENGINEERING
Faculty of Industrial Technology
Sepuluh Nopember Institute of Technology
Surabaya 2016

PERANCANGAN SISTEM NAVIGASI DAN KENDALI ROBOT *QBOT* MENGGUNAKAN METODE JARINGAN SARAF TIRUAN DAN KONTROLER PID

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan Untuk
Memperoleh Gelar Sarjana Teknik Elektro
Pada
Bidang Studi Teknik Sistem Pengaturan
Jurusan Teknik Elektro
Institut Teknologi Sepuluh Nopember

Menyetujui

Dosen Pembimbing

Ir. Rusdhianto Effendi A.K., M.T.
NIP. 1957 04 24 1985 02 1001

SURABAYA
JUNI, 2016

PERANCANGAN SISTEM NAVIGASI DAN KENDALI ROBOT *QBOT* MENGGUNAKAN METODE JARINGAN SARAF TIRUAN DAN KONTROLER PID

Nama	**: Eva Sri Oktavia Ningrum**
Nrp	**: 2214 105 094**
Dosen Pembimbing	**: Ir. Rusdhianto Effendie AK MT.**
Nip	**: 19570424 198502 1 001**

ABSTRAK

Sebagian besar aplikasi jaringan saraf tiruan pada robot digunakan sebagai sistem navigasi yaitu untuk perencanaan sebuah lintasan robot. Sistem navigasi pada *mobile robot* merupakan sistem yang dapat memandu robot untuk melakukan perpindahan dari posisi awal hingga ke posisi akhir yang diinginkan. Selain itu, sistem navigasi pada *mobile* robot harus mampu beradaptasi dengan lingkungan yang berubah-ubah. Jaringan saraf tiruan digunakan untuk memperoleh sebuah lintasan optimal yang bebas hambatan, yaitu lintasan dengan jarak tempuh terpendek tanpa menabrak *obstacle* yang terdapat pada area kerja robot.

Sedangkan kontroler PID digunakan untuk mengatur kecepatan roda kanan dan roda kiri pada robot ketika menetukan lintasan robot. Sehingga robot dapat berjalan sesuai dengan pengaturan kecepatan yang telah ditentukan oleh kontroler PID. Kemudian dilakukan tiga kali percobaan pada *mobile robot* tanpa menggunakan kontroler dengan jarak 350 cm ditempuh dalam waktu 52 detik, kemudian 48 detik dan 50 detik pada percobaan ketiga. Kemudian dilakukan percobaan pada *mobile robot* dengan jarak tempuh yang sama membutuhkan waktu 23 detik, kemudian 22 detik dan yang terakhir 25 di percobaan ketiga. Dapat disimpulkan jika kontroler memiliki respon yang lebih cepat jika dibandingkan dengan program tanpa kontroler.

Kata Kunci : Perancangan sistem navigasi**,** Jaringan saraf tiruan, *Qbot mobile robot*

DESIGN OF QBOT ROBOT NAVIGATION AND CONTROL SYSTEM USING NEURALNETWORK METHOD AND PID CONTROLLER

Name	**: Eva Sri Oktavia Ningrum**
Register Number	**: 2214 105 094**
Supervisor	**: Ir. Rusdhianto Effendie AK MT**
ID Number	**: 19570424 198502 1 001**

ABSTRACT

M*ost of applications of neural networks in robot used as a navigation system for planning a robot trajectory. The navigation system on the mobile robot is a system that can guide the robot to make the shift from the initial position to the final desired position. In addition, the navigation system on the mobile robot must be able to adapt to the changing environment. An artificial neural network is used to obtain an optimal barrier free path, ie the path with the shortest travel distance without hitting the obstacle that is contained in the robot work area.*

While the PID controller is used to control the speed of the right wheel and the left wheel on the robot when determining the trajectory of the robot. So that the robot can be run in accordance with the speed setting that has been determined by the PID controller. Then conducted three experiments on the mobile robot without using a controller with a distance of 350 cm to within 52 seconds, then 48 seconds and 50 seconds on the third try. Then conducted experiments on the mobile robot using a controller with the same distance takes 23 seconds, then 22 seconds and the last 25 at the third attempt. It can be concluded if the controller has a faster response when compared with programs without a controller.

Keyword **:** *The design of navigation systems, neural networks, Qbot mobile robot.*

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB 1
PENDAHULUAN

Pada bab satu akan dibahas mengenai latar belakang, permasalahan, tujuan, metodologi, sistematika, dan relevansi tugas akhir yang dikerjakan.

1.1 Latar Belakang

Autonomous Mobile Robot merupakan robot otomatis yang memiliki kemampuan dan kecerdasan dalam melakukan tugas tertentu. Selain itu, *mobile robot* juga banyak ditemukan di industri, militer, dan keamanan. Perkembangan *mobile robot* berlanjut pada tahun 1970-an, Johns Hopkins University membangun sebuah robot *"Beast",* robot ini melakukan gerakan dengan bantuan sensor ultrasonik [1].

Quanser Qbot merupakan *autonomous mobile robot* dengan penggerak dua buah roda (kanan dan kiri). *Qbot* dilengkapi dengan *Gumstix*, dimana robot dapat dikendalikan dengan menggunakan perangkat lunak *QUARC*. Perangkat lunak ini menyediakan fasilitas untuk mengembangkan dan menguji coba hasil rancangan kontroler pada komputer *host* dengan menggunakan antarmuka perangkat lunak Simulink MATLAB.

Seiring berkembangnya ilmu komputer dan kontrol, semakin banyak riset yang dilakukan dengan mencari metode terbaik untuk memecahkan sebuah permasalahan tertentu. Pengembangan mobile robot banyak dilakukan yaitu pada sistem navigasinya. Metode jaringan saraf tiruan adalah salah satu metode yang banyak digunakan dalam sistem navigasi pada *mobile robot*. Pada penelitian ini akan dibuat suatu sistem pengendali mobile robot menggunakan metode *neural network* dengan menggunakan umpan balik sensor. *Neural network* merupakan suatu sistem yang terdiri dari arsitektur jaringan syaraf dan metode pembelajaran. *Neural network* secara umum digunakan untuk menirukan sistem kerja otak yang memiliki kemampuan untuk belajar (beradaptasi atau mengikuti perubahan dan belajar atau menerima sesuatu yang baru). Dengan menggunakan metode *neural network* ini *mobile robot* akan melakukan proses pembelajaran tertentu bagaimana untuk bergerak maju, mundur, ke kiri, ke kanan atau kemungkinan lain berdasarkan pengalaman error yang terjadi. Kemudian kontroler PID digunakan

untuk mengontrol kecepatan roda kanan dan kiri robot ketika robot berjalan sesuai lintasan [2].

1.2 Tujuan

Tujuan tugas akhir ini adalah mendesain kontroler PID yang dapat diterapkan untuk implementasi pengaturan kecepatan *mobile robot Qbot* untuk bergerak dari titik awal menuju tujuan akhir dan dapat menghindari *obstacles* yang terdapat pada jalur dengan menggunakan metode jaringan saraf tiruan.

1.3 Metodologi

Metodologi yang digunakan pada penelitian tugas akhir ini adalah sebagai berikut:

1. **Tinjauan Pustaka**
   Dilakukan dengan mengumpulkan dan mempelajari literatur yang mendukung tugas akhir melalui *internet* dan media cetak berupa buku atau jurnal.
2. **Pengambilan Data dan Identifikasi**
   Dari hasil yang diperoleh dari studi literatur,identifikasi disesuaikan dengan keadaan *plant* sesungguhnya. Pada tahap ini akan dilakukan pengambilan data dari *Qbot Mobile robot*. Setelah memperoleh data yang dibutuhkan, akan dilakukan identifikasi dari *plant* untuk memperoleh parameter yang akan digunakan untuk mendesain kontroler sesuai yang diinginkan.
3. **Perancangan Kontroler**
   Pada tahap ini dibuat struktur kontroler PID untuk pengaturan kecepatan *Qbot mobile robot* dan implementasi jaringan saraf tiruan yang digunakan untuk sistem navigasi pada robot dalam menentukan jalurnya pada lintasan.
4. **Implementasi dan Uji Coba**
   Kontroler yang sudah dirancang lalu diimplementasikan pada *plant* berupa *Qbot Mobile robot* pada ruang yang sudah direncanakan jalurnya dan terdapat *obstacles.*
5. **Penulisan Buku Tugas Akhir**
   Penyusunan buku Tugas Akhir meliputi pendahuluan, teori dasar, perancangan sistem, implementasi, analisa serta penutup.

1.4 Sistematika

Pembahasan Tugas Akhir ini dibagi menjadi lima bab dengan sistematika sebagai berikut :

BAB 1 : Pendahuluan

Bab ini meliputi latar belakang, permasalahan, tujuan penelitian, dan sistematika penulisan.

BAB 2 : Teori Penunjang

Bab ini membahas tinjauan pustaka yang membantu penelitian, diantaranya teori *Qbot mobile robot*, teori sistem kontroler PID dan teori mengenai metode jaringan saraf tiruan.

BAB 3 : Perancangaan Sistem

Pada bab ini dijelaskan mengenai perancangan kontroler PID dan metode jaringan saraf tiruan yang diimplementasikan pada *Qbot mobile robot*.

BAB 4 : Hasil dan Analisa

Bab ini memuat hasil implementasi kontroler dan pengujiannya pada sistem.

BAB 5 : Penutup

Bab ini berisi kesimpulan dan saran dari hasil penelitian yang telah dilakukan.

1.5 Relevansi

Hasil yang diperoleh dari tugas akhir ini diharapkan menjadi referensi pengembangan teknologi dan perbandingan metode kontrol yang tepat untuk navigasi *Qbot mobile robot* di masa mendatang.

Halaman ini sengaja dikosongkan

BAB 2
TEORI PENUNJANG

Pada bab dua akan dibahas mengenai tinjauan pustaka dan teori-teori yang berhubungan dengan permasalahan yang dibahas pada bab sebelumnya.

2.1 Tinjauan Pustaka

Perkembangan *mobile robot* berkembang menjadi sebuah peralatan canggih untuk rumah tangga, militer, industri, dan pendidikan. Sebagian besar *mobile robot* dirancang untuk dapat beroperasi pada lingkungan kerja yang diinginkan, seperti robot permukaan, robot bawah air, dan robot udara. Robot permukaan dibagi berdasarkan pengeraknya seperti robot berkaki dan robot beroda. *Mobile robot* banyak dikembangkan untuk dunia pendidikan dan riset, dimana robot dilengkapi dengan perangkat keras, kemampuan, kontrol dan aplikasi. *Khepera III, Koala II Pioneer 3 DX*, dan *iRobot Create*. Robot tersebut merupakan robot yang banyak dikembangkan untuk riset dan aplikasi termasuk pemetan, pengintaian, operasi jarak jauh, dan navigasi otomatis. Penelitian tentang *mobile robot* telah banyak dilakukan, seperti pengembangan tentang metode kontrol *tracking*, navigasi dan algoritma untuk perencanaan lintasan [2]. Oleh karena itu, pada penelitian ini dilakukan perancangan sebuah lintasan menggunakan metode jaringan saraf tiruan untuk perencanaan lintasan dan kontrol PID yang digunakan sebagai kontrol kecepatan roda kanan dan kiri robot.

2.2 *Qbot Mobile Robot* [3]

Quanser mobile robot adalah sebuah robot *autonomous.* Robot ini merupakan *iRobot Create*, dimana ada beberapa sensor di yang terpasang pada robot ini. Sensor pada *Quanser mobile robot* diantaranya adalah sensor *infraredI,* sensor sonar dan kamera digital. Agar perangkat lunak *QUARC* dapat secara langsung di download pada robot ini maka dipasang sebuah QCM*(Quanser Controller Module)* pada komputer *Gumsti*x. Sehingga dengan begitu program dari Simulink Mathlab dapat secara langsung dieksekusi pada robot. Bentuk fisik dari *Qbot Mobile Robot* terdapat pada Gambar 2.1. Sedangkan di bawah ini akan disebutkan beberapa spesifikasi dari sensor dan aktuator yang terdapat pada *Quanser Mobile Robot :*

1. Motor servo dengan keluaran 8 PWM
2. 7 I/O digital yang dapat diatur ulang dan 1 keluaran digital berupa LED
3. 7 masukan analog, 12 bit, +5V, resolusi 6,2mV
4. 5 sensor inframerah (IR), jarak hingga 150cm
5. sensor sonar, jarak 15cm hingga 645cm, resolusi 1 *inch*
6. *axis magnetometer*, resulusi 0,77 milli-Gauss
7. Kamera USB hingga 9fps
8. Komunikasi *Wireless*

Gambar 2.1 Bentuk Fisik *Qbot Mobile Robot* [3]

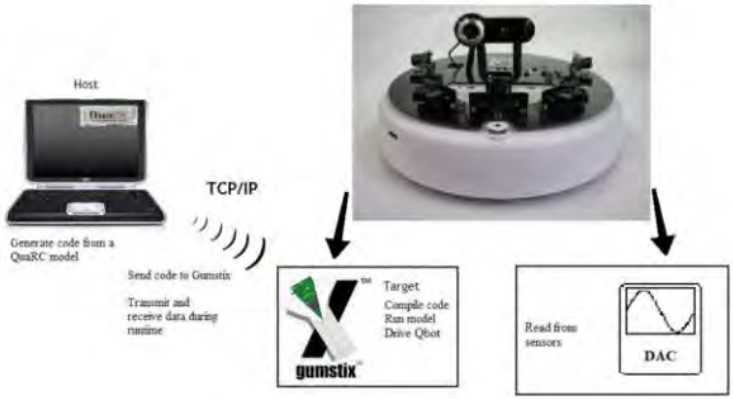
Berikut ini spesifikasi dan model parameter *Qbot mobile robot* dapat dilihat pada Tabel 2.1 berikut

Tabel 2.1 Spesifikasi Dan Parameter Model *Qbot Mobile Robot* [3]

Simbol	Deskripsi	Value	Unit
D	Diameter *Qbot mobile robot*	0,34	m
H	Tinggi *Qbot mobile robot* (dengan tambahan kamera)	0,19	m
Vmax	Kecepatan maksimum dari *Qbot mobile robot*	0,5	m/s
M	Berat total *Qbot mobile robot*	2,92	kg

Qbot mobile robot memiliki tiga buah blok yang berbeda fungsi, diantaranya yaitu *Roomba Block* yang digunakan sebagai pengendali gerakan pada *mobile robot.* Kemudian *Block Open CV* yang berfungsi untuk akses sensor kamera pada *mobile robot.* Dan yang terkahir ada pula *Block HIL* yang fungsinya berguna untuk membaca data yang

dihasilkan oleh sensor yang terpasang pada *Qbot Mobile Robot* serta berfungsi sebagai media yang memberikan data keluaran untuk pada motor servo. *Qbot mobile robot* ini dikendalikan melalui program yang dirancang menggunakan Simulink Mathlab. Pada *mobile robot* nantinya akan di implementasikan kontroler yang dirancang menggunakan Simulink dengan mathlab yang sudah berbasis *QUARC* pada komputer *host*. Kemudian kontroler ini dapat secara otomatis di *download* dan dapat langsung dijalankan atau di *compile* pula pada *Qbot Mobile Robot* langsung pada target *Gumstik.* Berikut di bawah pada Gambar 2.2 menggambarkan hirarki komunika *Qbot Mobile Robot* dengan komponen – komponen penunjangnya.

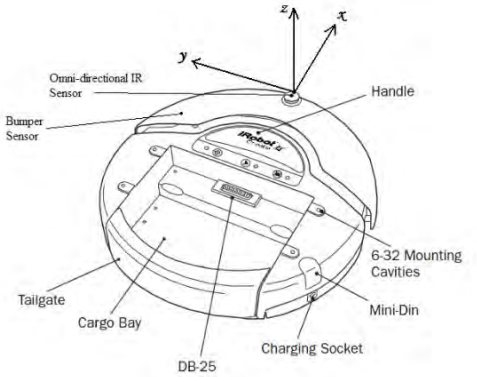


Gambar 2.2 Hirarki Komunikasi *Qbot Mobile Robot* [3]

2.2.1 Perangkat Keras *Qbot* [3]

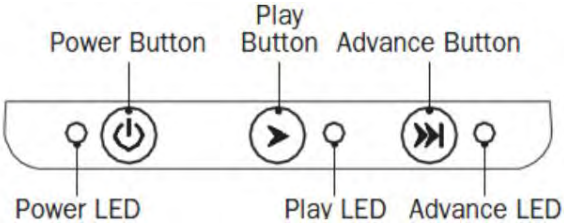
Pada *Qbot Mobile Robot* terdapat tiga koordinat yang dimiliki yang sesuai dengan *standart Quanser.* Tiga koordinat tersebut yaitu (x,y,z). Pada koordinat x menunjukkan pergerakan maju pada *Qbot mobile Robot,*kemudian koordinat y berada pada sebelah kiri pergerakan robot. Dan yang terakhir adalah koordinat z yang menunjukkan arah atas pada *mobile robot. Qbot Mobile Robot* ini memiliki diameter rangka sebesar 34 cm sedangkan tinggi yang terukur tanpa tinggi kamera adalah 7 cm. Pergerakan *Qbot Mobile Robot* ini digerakkan oleh dua roda yaitu roda kanan dan roda kiri robot. Keduanya bergerak dengan kemudi yang berbeda pula. Terdapat pula *bumper* dan sensor inframerah pada *Qbot*

Mobile Robot yang terpasang pada bagian depan robot. *QCN* dapat mengakses data yang dihasilkan oleh sensor pada robot ini. *Qbot Mobile Robot* menggunakan rangka dari *iRobot Create®.* Untuk penjelasan tentang rangka *Qbot Mobile Robot* terlihat pada Gambar 2.3 di bawah ini.



Gambar 2.3 Rangka *Qbot Mobile Robot* [3]

Sedangkan tombol – tombol yang digunakan pada *mobile robot* seperti yang terdapat pada Gambar 2.4 yaitu terdapat empat tombol. Tombol pertama dari sebelah kiri pada Gambar 2.4 yaitu tombol *Power* yang digunakan untuk menghidupkan / memulai menjalankan *Qbot mobile robot.* Serta terdapat *Led Indicator* disebelah kiri tombol *Power* yang menandakan robot sudah siap digunakan apabila *LED* sudah menyala. Kemudian tombol di bagian tengah adalah tombol *Play Button* yang berfungsi untuk mengoperasikan *Qbot mobile robot.* Dan yang terakhir adalah tombol *Advance* yang terdapat di sebelah kanan yang berguna untuk menjalankan demo untuk robot. Dan sama seperti tombol *Power* masing – masing tombol memiliki *LED Indicator* yang mendakan bahwa robot menjalankan perintah dari tombol yang sudah tersedia pada *Qbot mobile robot.* Di bawah ini pada Gambar 2.4 adalah penggambaran tombol yang terdapat pada *Qbot mobile robot.*



Gambar 2.4 Tombol Pada Rangka *Qbot*

2.2.2 Sensor Inframerah dan Sonar [3]

Qbot mobile robot memiliki sensor inframerah dan sensor sonar yang terpasang pada bagian depan robot. Terdapat lima sensor inframerah yang dapat digunakan pada *mobile robot* ini. Sensor inframerah ini dapat mendeteksi dengan jarak 20 – 150 cm. Sensor ini terhubung ke saluran *input* analog dari DAC *Qbot.* Kemudian blok HIL *Read Write* yang berfungsi untuk membaca sensor pada*mobile robot* akan membaca sinyal dari sensor inframerah tersebut. Sensor inframerah pada *mobile robot* ini adalah sensor inframerah *SHARP* 2Y0A02. Gambar 2.5 menunjukkan bentuk fisik dari sensor inframerah yang terpasang pada *Qbot Mobile Robot.*

Gambar 2.5 Sensor Inframerah *SHARP* 2Y0A02 [3]

MaxSonar-EZ0 mendeteksi objek dari 0 inci sampai 254 inci dengan resolusi 1 inci. Objek antara 0 inci dan 6 inci berkisar sebagai 6 inci. Terdapat tiga sensor pada *Qbot*. Sensor yang terhubung ke saluran masukan lain dari *Qbot* DAC, yang kemudian dapat dibaca dengan menggunakan blok HIL *Read Write.* Pada Gambar 2.6 adalah bentuk fisik dari sensor sonar yang terdapat pada *mobile robot.*

Gambar 2.6 Sensor sonar *MaxSonar-EZ0*

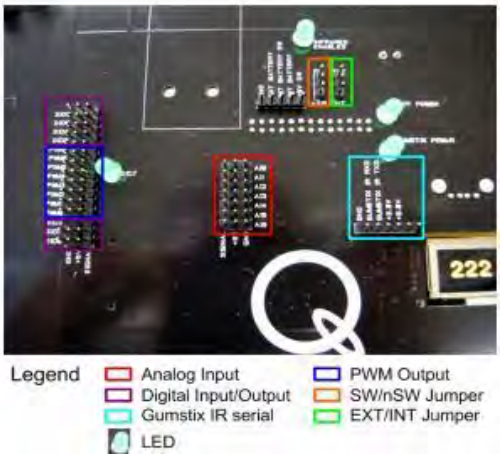
2.2.3 Kamera USB [3]

Kamera USB *Logitech Quickcam Pro* 9000 terpasang pada bagian atas *Qbot* seperti pada Gambar 2.7. Blok QuaRC menggunakan *Open Source Computer Vision* yang menyediakan pengguna untuk mengambil dan menampilkan gambar secara *real time*, memproses dan menyimpannya untuk analisis.

Gambar 2.7 Kamera USB *Logitech Quickcam* 9000 [3]

2.2.4 *Printed Circuit Board* (PCB) [3]

Kabel dan sirkuit untuk *Qbot* dalam PCB yang terletak dipenutup hitam *Qbot* atas komputer *Gumstix* dan DAC. Sensor dan kamera yang juga terpasang pada PCB. Gambar 2.8 menunjukkan pin diakses bagi pengguna. Secara khusus, DIO, *output* PWM, dan pin *input* analog telah diberi label untuk lebih jelas.



Gambar 2.8 PCB *Qbot* [3]

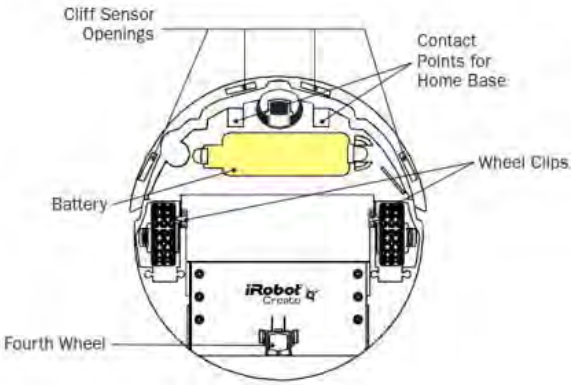
2.2.5 Baterai *Qbot* [3]

Qbot ini didukung oleh baterai *Advance Power System* (APS) yang disediakan oleh *iRobot* seperti pada Gambar 2.9. Baterai ini terletak di bawah *Qbot* dan dapat berlangsung terus menerus selam sekitar dua jam setelah terisi penuh.

Gambar 2.9 Baterai *Qbot* [3]

Baterai terletak dibagian bawah seperti pada Gambar 2.10.Lampu daya *Qbot* mengindikasikan tingkat daya baterai. Lampu hijau menandakan baterai masih terisi penuh, kemudian secara bertahap berubah menjadi merah bila baterai akan habis. Baterai memerlukan

waktu kurang dari 3 jam untuk pengisian baterai. Ketika pengisian, lampu daya akan berkedip perlahan dengan warna jingga dan akan berubah menjadi hijau ketika terisi penuh.



Gambar 2.10 Letak Baterai Pada *Qbot* [3]

2.2.6 *Pin Digital Input/Output* (DIO) [3]

Saluran DIO (0 sampai 6) ditetapkan sebgai *input* secara bawaan. Saluran DIO perlu dikonfigurasi baik sebagai *input* (atau *ouput*, tapi tidak keduanya) menggunakan *block HIL Initialize*. Jika *output* harus dalam keadaan yang dikenal *power up,* dianjurkan bahwa resistor 10K diletakkan dari I/O untuk 5V atau GND sesuai kebutuhan. Ada saluran digital akhir (7) yang merupakan *output* tetap, dan itu diwakili oleh *LED* berlabel DIO7.

2.2.7 *Pin Serial Gumstix IR* [3]

Qbot menyediakan serial koneksi TTL ke *port* serial *Gumstix* IR (*port* no 2). *Port* serial terdiri dari *ground* (GND), menerima *Gumstix* IR RXD, mengirimkan *Gumstix* IR TXD dan pin daya (+3.3V atau +5.0V). *Port* serial diakses melalui blok *QuaRC Stream* atau *Stream* API.

2.2.8 SW/nSW dan INT/EXT *Jumpers* [3]

INT/EXT *jumper switch Qbot* dari internal daya dari baterai *iRobot Create* (INT) dan eksternal baterai *power supply* (EXT). Tidak ada baterai eksternal yang disertakan pada *Qbot*, jadi jumper ini harus dibiarkan dalam posisi INT untuk daya *Qbot* tersebut. Saat *jumper*

power supply dalam posisi INT, *jumper* SW/ NSW menunjukkan apakah *iRobot Create* harus diaktifkan (SW) untuk *Qbot* menerima daya atau apakah *Qbot* harus selalu membutuhkan daya bahkan ketika *iRobot Create* dalam keadaan mati (nSW).

2.2.9 *Qbot Data Acquisition Card* (DAC) [3]

Qbot DAC adalah kartu data akuisisi, yang mampu menerima *input* analog dan *input* lainnya (untuk sensor sonar). *Qbot* DAC dapat juga membaca *output* PWM untuk servo aktuatuor. *Qbot* DAC terletak di bawah penutup hitam *Qbot*.

2.2.10 Gumstix [3]

Gumstix adalah *device* kecil pada *Qbot* robot yang berfungsi pada komputer sebagai *open source* dimana agar Simulink MATLAB secara langsung dapat diunduh, dikompilasi dan dijalankan melalui perangkat lunak QuaRC. *Motherboard Gumstix* terhubung secara langsung pada *Qbot* DAC. Pada *Gumstix* juga terdapat tambahan *Wifi* untuk menyediakan koneksi nirkabel antara target *Gumstix* dan komputer *host*.

2.3 QUARC [3]

Qbot Mobile Robot mempunyai sebuah perangkat lunak dengan nama QUARC *(Quanser Real-Time Control)*. QUARC ini berguna untuk menyediakan fasilitas pada penggunaan Simulink Mathlab dalam pengembangan dan pengujian kontroler yang nantinya akan digunakan pada *Qbot mobile robot.* Model kontroler yang dirancang pada Simulink Mathlab dapat dengan langsung diunduh serta dijalankan pada target *gumstik* secara *real time*. Sedangkan pada waktu yang bersamaan pula operator yang mengoperasikan *Qbot mobile robot* ini dapat memantau serta mengukur secara langsung parameter – parameter kontroler yang terpasang pada *Qbot mobile robot* secara langsung dan dengan waktu yang sama melalui komputer *host*. Kontroler yang telah dirancang tersebut dapat di unduh langsung dengan menggunakan fotur *wireless* pada robot secara *real time.*

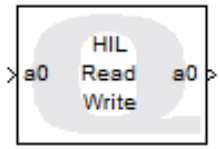
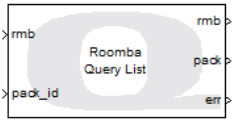
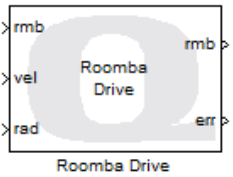
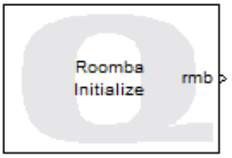
Pada Simulink mathlab terdapat beberapa blok yang sudah difasilitasi oleh QUARC *(Quanser Real-Time Control)* yang dapat digunakam untuk melakukan komunikasi secara *real time* . Di bawah ini adalah beberapa *block set* pada *mobile robot* yang akan dijelaskan pada Tabel 2.2 dan Tabel 2.3.

Tabel 2.2 *Blok Set Qbot* Untuk *Qbot* di *QUARC* [3]

Blok Set	**Deskripsi**
Aplikasi	*Blok set* ini memungkinkan pengguna untuk menerapkan algoritma navigasi menggunakan informasi sensorik yang tersedia. Blok ini hanya menggunakan data *encoder* roda dan *bump sensors* untuk navigasi dan menghindari rintangan.
Image Processing	*Blok set* ini mengimplementasikan akuisisi citra dan pengolahan fungsi menggunakan *Library Open CV*. Blok set pengolahan citra memungkinkan untuk menangkap gambar dari kamera USB, untuk proses *online* dan menyimpannya ke *disk* untuk analisis lebih lanjut.
Interface	Blok set ini mengimplementasikan dasar *Aplication Program Interfaces* (API) yang disediakan oleh *iRobot Create®*. API dapat dikategorikan berdasarkan fungsional berikut: • Sambungkan konfigurasi *serial* antara kontroler tingkat tinggi(misalnya, PC atau *Gumstix*) dan *Qbot*. • Pengaturan *mode* operasi *Qbot* • Mengakses informasi sensorik *Qbot* Konfigurasi *hardware* lainnya (misalnya, pengaturan *port* I/O digital dan analog, dan mengubah warna LED)

Tabel 2.3 Penjelasan Deskripsi *Blok QUARC* Pada *Qbot* [3]

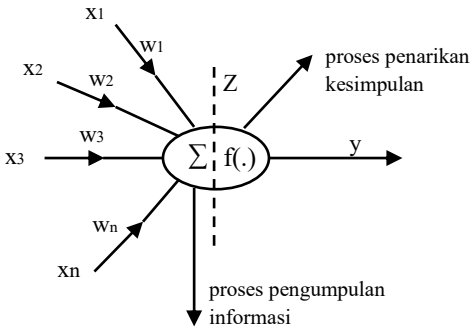
Blok	**Deskripsi**
	*Blok* inisialisasi yang diperlukan dalam semua model *QuaRC* menggunakan *HIL Card*. Hanya satu blok *HIL Initialize* yang dibutuhkan dalam model untuk mengatur satu kendaraan. Blok HIL *Read Write* harus referensi blok ini untuk menentukan *board* yang digunakan.

Blok	Deskripsi
 HIL Read Write (HIL-1)	Versi saat ini dari *Qbot* DAC yang dapat membaca dari saluran analog (untuk sensor infra merah) dan saluran sensor sonar. Saluran PWM yang didukung untuk operasi *write.*
 Roomba Query List	*Blok* ini mengambil berbagai data sensor dari *Qbot*. *Input* pack_id berkisar 7-42, dan setiap *input* akan menampilkan nilai dari *output pack*. Lihat halaman *Help* MATLAB untuk deskripsi lengkap untuk masing-masin paket ID. Nilai *pack_id* yang penting: 8 = Dinding (0 = tidak ada dinding, 1 = dinding terlihat) 19 = Jarak (Jarak yang *Qbot* telah melakukan perjalanan dalam *milimeter*) 20 = Sudut (Sudut dalam derajat *Qbot* berubah) 22 = Tegangan (Tegangan dari baterai *Qbot* dalam *milivolt*) 23 = Arus (Arus dalam *miliampere* mengalir atau keluar dari baterai *Qbot*)
 Roomba Drive	Blok ini menggerakkan *Qbot* dengan dua *input:* kecepatan dan radius. Untuk menggerakkan lurus, *input* radius mengambil 32768 atau 32767.
	Blok ini diperlukan untuk membuat sambungan serial ke *Qbot*. *Qbot* diidentifikasi oleh *Universal Resource Identifier* (URI), seperti *serial://localhost:1baud=57600,word=8, parity=none,stop=1*, di mana *port* komunikasi 1 dari target terhubung ke *port* serial *Qbot* dengan parameter yang ditentukan. Catatan: *Output* "rmb" dari blok ini harus terhubung ke *input* "rmb" dari blok di blok

Blok	Deskripsi
	Roomba yang ditetapkan dalam rangka untuk mengakses *Roomba*.

2.4 Jaringan Saraf Tiruan [4]

Jaringan syaraf *(Neural Network)* adalah merupakan salah satu representasi buatan dari otak manusia yang selalu mencoba untuk mensimulasikan proses pembelajaran pada otak manusia tersebut. Istilah buatan disini digunakan karena jaringan syaraf ini diimplementasikan dengan menggunakan program komputer yang mampu menyelesaikan sejumlah proses perhitungan selama proses pembelajaran. Pada Gambar 2.11 di bawah ini adalah struktur model pada metode jaringan syaraf tiruan yang digunakan pada sistem ini.



Gambar 2.11 Struktur Model Jaringan Saraf Tiruan [4]

2.4.1 Konsep Metode Jaringan Saraf Tiruan [5]

Jaringan syaraf tiruan terdiri dari beberapa *neuron* dan ada hubungan antar *neuron- neuron* seperti pada otak manusia. *Neuron* atau sel saraf adalah sebuah unit pemroses informasi yang merupakan dasar operasi jaringan syaraf tiruan.

Jaringan syaraf tiruan terdiri atas beberapa elemen penghitung tak *linier* yang masing-masing dihubungkan melalui suatu pembobot dan tersusun secara *paralel*. Pembobot inilah yang nantinya akan berubah (beradaptasi) selama proses pelatihan. Baik tidaknya suatu model JST ditentukan oleh:

1. Pola antar *neuron* (arsitekur jaringan)

2. Metode untuk menentukan dan mengubah bobot (disebut metode *learning*)
3. Fungsi aktivasi
4. JST disebut juga: *brain metaphor*, *computational neuroscience, parallel distributed processing*

2.4.2 Karakteristik Metode Jaringan Saraf Tiruan [5]

Baik tidaknya suatu model JST ditentukan oleh: Penyelesaian masalah dengan jaringan syaraf tiruan tidak memerlukan pemrograman. Jaringan syaraf tiruan menyelesaikan masalah melalui proses belajar dari contoh-contoh pelatihan yang diberikan. Biasanya pada jaringan syaraf tiruan diberikan sebuah himpunan pola pelatihan yang terdiri dari sekumpulan contoh pola. Proses belajar jaringan syaraf tiruan berasal dari serangkaian contoh-contoh pola yang diberikan.metode pelatihan yang sering dipakai adalah metode belajar terbimbing. Selama proses belajar itu pola masukan disajikan bersama-sama dengan pola keluaran yang diinginkan.

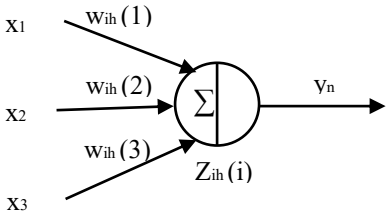
1. Faktor Bobot
   Bobot merupakan suatu nilai yang mendefinisikan tingkat atau kepentingan hubungan antara suatu node dengan *node* yang lain. Semakin besar bobot suatu hubungan menandakan semakin pentingnya hubungan kedua *node* tersebut. Bobot merupakan suatu hubungan berupa bilangan *real* maupun *integer*, tergantung dari jenis permasalahan dan model yang digunakan. Bobot-bobot tersebut bisa ditentukan untuk berada didalam *interval* tertentu. selama proses pelatihan, bobot tersebut dapat menyesuaikan dengan pola-pola *input*. Jaringan dengan sendirinya akan memperbaiki diri terus-menerus karena adanya kemampuan untuk belajar. Setiap ada suatu masalah baru, jaringan dapat belajar dari masalah baru tersebut, yaitu dengan mengatur kembali nilai bobot untuk menyesuaikan karakter nilai
2. Fungsi Aktifasi
   Setiap neuron mempunyai keadaan internal yang disebut level aktivasi atau level aktivitas yang merupakan fungsi *input* yang diterima. Secara tipikal suatu neuron mengirimkan aktivitasnya kebeberapa *neuron* lain sebagai sinyal. Yang perlu diperhatikan adalah bahwa *neuron* hanya dapat mengirimkan satu sinyal sesaat, walaupun sinyal tersebut

dapat dipancarkan ke beberapa neuron yang lain. Ada beberapa pilihan fungsi aktivasi yang digunakan dalam metode *backpropagation*, seperti fungsi *sigmoid biner*, dan *sigmoid bipolar*. Karakteristik yang harus dimiliki fungsi fungsi aktivasi tersebut adalah kontinue, *diferensiabel*, dan tidak menurun secara monoton. Fungsi aktivasi diharapkan dapat mendekati nilai-nilai maksimum dan minimum secara baik.

2.4.3 Model Pada Jaringan Syaraf Tiruan [6]

Pelatihan pada jaringan syaraf *backpropagation*, *feedfoward* (umpan maju) dilakukan dalam rangka perhitungan bobot sehingga pada akhir pelatihan akan diperoleh bobot-bobot yang baik. Selama proses pelatihan, bobot-bobot diatur secara *iteratif* untuk meminimumkan *error* (kesalahan) yang terjadi. *Error* (kesalahan) dihitung berdasarkan rata-rata kuadrat kesalahan (MSE). Rata-rata kuadrat kesalahan juga dijadikan dasar perhitungan unjuk kerja fungsi aktivasi. Sebagian besar pelatihan untuk jaringan *feedfoward* (umpan maju) menggunakan *gradien* dari fungsi aktivasi untuk menentukan bagaimana mengatur bobot-bobot dalam rangka meminimumkan kinerja. Gradien ini ditentukan dengan menggunakan suatu teknik yang disebut *backpropagation*. Berikut Penjelasannya :

1. Formulasi *Forward*
   Formulasi *forward* digunakan untuk menghitung keluaran dari setiap neuron pada jaringan saraf tiruan. Gambar 2.12 menunjukkan struktur formulasi *forward* pada jaringan saraf tiruan yang terdiri dari masukkan, bobot, fungsi aktivasi dan keluaran.



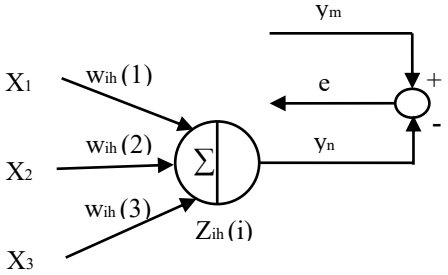
Gambar 2.12 Struktur Formulasi *Forward* [6]

Untuk formulasinya menggunakan Persamaan 2.1 dan Persamaan 2.2 di bawah ini :

$$z_{ih}(i) = \sum_{i=1}^{n} w_{ih}(i,j) \times x(i) \quad (2.1)$$

$$y_{ih}(i) = \lambda \times z_{ih} \quad (2.2)$$

2. Formulasi *Backward*



Gambar 2.13 Struktur Formulasi *Backward* [6]

Formulasi *backward* digunakan untuk merevisi bobot dari nilai *error* yang diperoleh dari proses adaptasi jaringan terhadap keluaran model yang diinginkan. Setiap ada *error* baru, jaringan dapat belajar dari *error* tersebut dengan merevisi nilai bobot untuk menyesuaikan karakter nilai. Gambar 2.13 menunjukkan struktur formulasi *backward*

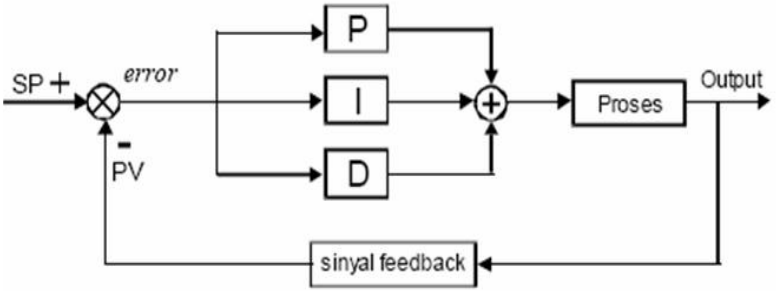
pada jaringan saraf tiruan. Persamaan 2.3 dan Persamaan 2.4 di bawah ini menunjukkan formulasi revisi bobot pada proses *backward.*

$$e = y_m - y_n \tag{2.3}$$

$$w^b = w^l + \alpha \times f'(z) \times x(i) \tag{2.4}$$

2.5 Kontroler PID

PID *(Proportional–Integral–Derivative controller)* merupakan kontroler untuk menentukan presisi suatu sistem instrumentasi dengan karakteristik adanya umpan balik pada sistem tesebut. Pengontrol PID adalah pengontrol konvensional yang banyak dipakai dalam dunia industri [7]. PID Blok Diagram dapat dilihat pada gambar di bawah :



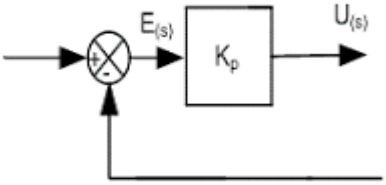
Gambar 2.14 Blok Diagram PID [7]

Menurut diagram blok diagram pada Gambar 2.14 keterangan SP adalah *Set point* adalah suatu prameter nilai acuan atau nilai yang diinginkan. PV adalah *Present Value*, adalah nilai bobot pembacaan sensor saat itu atau variabel terukur yang di umpan balikan oleh sensor (sinyal *feedback* dari sensor). *Error* artinya nilai kesalahan, adalah *deviasi* atau simpangan antar variabel terukur atau bobot sensor (PV) dengan nilai acuan (SP) [8].

2.5.1 Kontrol *Proportional* [9]

Pada pengendali jenis P *(proportional)* ini terdapat hubungan yang sebanding atau *proportional* antara keluaran terhadap kesalahan, secara lebih sederhana dapat dikatakan bahwa keluaran pengendali *proportional* merupakan perkalian antara konstanta *proportional* dengan

masukannya, di bawah ini adalah blok diagram untuk pengendali *proportional*, ditunjukkan pada Gambar 2.15 :



Gambar 2.15 Blok Diagram Untuk Pengendali *Proportional* [9]

Sedangkan untuk Persamaan 2.5 di bawah ini adalah persamaan matematis untuk pengendali proporsional :

$$U(t) = Kp\ e(t) \qquad (2.5)$$

Persamaan 2.6 di bawah ini adalah persamaan untuk fungsi alih untuk pengendali proporsional, yaitu :

$$\frac{(s)}{(s)} \quad p \qquad (2.6)$$

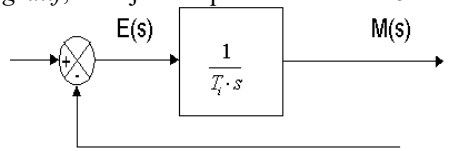
Pertambahan harga Kp akan menaikkan penguatan sistem sehingga dapat digunakan untuk memperbesar kecepatan tanggapan dan mengurangi *error steady state* (penyimpangan dalam keadaan mantap). Pemakaian alat kendali tipe proporsional ini sering tidak memuaskan karena penambahan Kp selain akan membuat sistem lebih sensitif tetapi juga cenderung mengakibatkan ketidakstabilan. Disamping itu penambahan harga Kp terbatas dan tidak cukup untuk mencapai tanggapan sampai suatu harga yang diinginkan. Kenyataannya dalam mengatur harga Kp terdapat keadaan-keadaan yang bertentangan . Di satu pihak diinginkan mengurangi ess sebanyak mungkin tetapi hal ini akan mengakibatkan osilasi bagi tanggapan yang berarti memperlama *"setting time"* sedangkan dipihak lain tanggapan terhadap setiap perubahan masukan harus terjadi secepat mungkin tetapi dengan lonjakan dan osilasi sekecil mungkin. Tanggapan yang cepat memang dapat diperoleh dengan memperbesar Kp tetapi hal ini juga akan mengakibatkan ketidakstabilan sistem.

2.5.2 Kontrol *Integratif* [9]

Kontroler *integral* berfungsi menghasilkan respon sistem yang memiliki kesalahan keadaan tunak nol. Kalau sebuah *plant* tidak

memiliki unsur integrator $\frac{1}{s}$, kontroller *proportional* tidak akan mampu menjamin keluaran sistem dengan kesalahan keadaan tunaknya nol. Dengan kontroler *integral*, respon sistem dapat diperbaiki, yaitu mempunyai kesalahan keadaan mantapnya nol. Kontroler *integral* memiliki karakteristik seperti halnya sebuah *integral*. Keluaran kontroler sangat dipengaruhi oleh perubahan yang sebanding dengan nilai sinyal kesalahan.

Keluaran kontroler ini merupakan jumlahan yang terus menerus dari perubahan masukannya. Kalau sinyal kesalahan tidak mengalami perubahan, keluaran akan menjaga keadaan seperti sebelum terjadinya perubahan masukan. Di bawah ini adalah diagram blok untuk pengendali *integratif*, ditunjukkan pada Gambar 2.16.



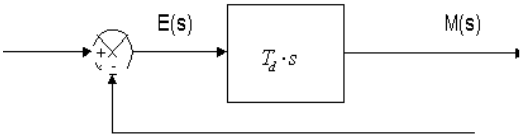
Gambar 2.16 Blok Diagram Untuk Pengendali *Integratif* [9]

Pemilihan Ki yang sangat tinggi justru dapat menyebabkan *output* berosilasi karena menambah *orde* sistem. Kontrol *integral* pada prinsipnya bertujuan untuk menghilangkan kesalahan keadaan tunak *(offset)* yang biasanya dihasilkan oleh kontrol *proportional*. Ketika digunakan, kontroler *integral* mempunyai beberapa karakteristik berikut ini:

1. Keluaran kontroler membutuhkan selang waktu tertentu, sehingga kontroler integral cenderung memperlambat respon.
2. Ketika sinyal kesalahan berharga nol, keluaran kontroler akan bertahan pada nilai sebelumnya.
3. Jika sinyal kesalahan tidak berharga nol, keluaran akan menunjukkan kenaikan atau penurunan yang dipengaruhi oleh besarnya sinyal kesalahan dan nilai Ki
4. Konstanta integral Ki yang berharga besar akan mempercepat hilangnya *offset*. Tetapi semakin besar nilai konstanta Ki akan mengakibatkan peningkatan osilasi dari sinyal keluaran kontroler.

2.5.3 Kontrol *Derivative* [9]

Keluaran kontroler *diferential* memiliki sifat seperti halnya suatu operasi *derivative*. Perubahan yang mendadak pada masukan kontroler, akan mengakibatkan perubahan yang sangat besar dan cepat. Gambar 2.17 menunjukkan blok diagram yang menggambarkan hubungan antara sinyal kesalahan dengan keluaran kontroller.



Gambar 2.17 Blok Diagram Untuk Pengendali *Derivative* [9]

Kontrol *Derivative* hanya berubah saat ada perubahan *error* sehingga saat *error* statis kontrol ini tidak akan bereaksi, hal ini pula yang menyebabkan kontroler *derivative* tidak dapat dipakai sendiri. Kontrol *derivative* dapat disebut pengendali laju, karena *output* kontroler sebanding dengan laju perubahan sinyal *error*. Kontrol *derivative* tidak akan pernah digunakan sendirian, karena kontroler ini hanya akan aktif pada periode peralihan. Pada periode peralihan, kontrol *derivative* menyebabkan adanya redaman pada sistem sehingga lebih memperkecil lonjakan. Seperti pada kontrol *proportional*, kontrol *derivative* juga tidak dapat menghilangkan *offset*. Karakteristik kontroler *derivative* adalah sebagai berikut:

1. Kontroler ini tidak dapat menghasilkan keluaran bila tidak ada perubahan pada masukannya (berupa sinyal kesalahan).
2. Jika sinyal kesalahan berubah terhadap waktu, maka keluaran yang dihasilkan kontroler tergantung pada nilai Td dan laju perubahan sinyal kesalahan.
3. Kontroler *derivative* mempunyai suatu karakter untuk mendahului, sehingga kontroler ini dapat menghasilkan koreksi yang signifikan sebelum pembangkit kesalahan menjadi sangat besar. Jadi kontroler *diferential* dapat mengantisipasi pembangkit kesalahan, memberikan aksi yang bersifat *korektif*, dan cenderung meningkatkan *stabilitas* sistem.

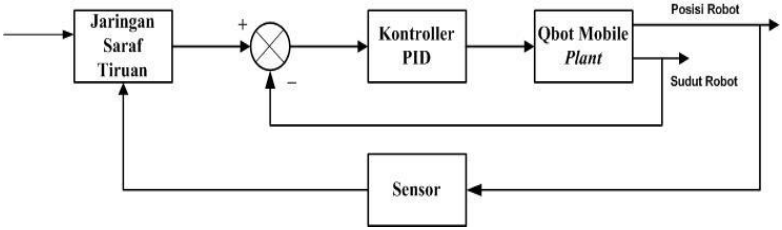
Halaman ini sengaja dikosongkan

BAB 3
PERANCANGAN SISTEM

Pada bab tiga ini akan dibahas mengenai proses perancangan implementasi perencanaan jalur *Qbot mobile robot* menggunakan metode jaringan saraf tiruan untuk penentuan jalurnya serta menggunakan kontroler PID untuk mengatur kecepatan robot.

3.1 Diagram Blok Sistem

Diagram blok pada Gambar 3.1 di bawah ini menunjukkan keseluruhan dari sistem perencanaan sistem navigasi dan kendali robot *Qbot* menggunakan metode jaringan saraf tiruan dan kontroler PID.



Gambar 3.1 Diagram Blok Sistem

Diagram blok diatas terdiri dari beberapa blok subsistem. Bagian target referensi merupakan data yang akan menjadi *input* untuk jaringan saraf tiruan. Target referensi terdiri dari data jarak ke target, sudut ke target, jarak ke *obstacle* dan sudut ke *obstacle*. Kemudian setelah diproses oleh jaringan saraf tiruan akan dikontrol kembali oleh kontoler PID untuk mendapatkan kecepatan roda kanan dan kiri robot.

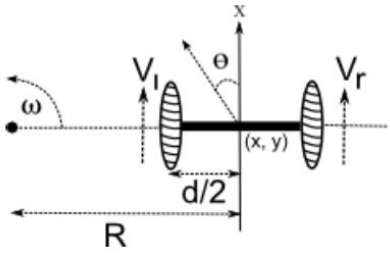
Sedangkan PID sendiri mendapat *input feedback* dari sudut robot yang tujuannya untuk mengetahui *error* dari sudut putar robot. Pada target referensi didapat dari data kecepatan robot yang sebelumnya dilakukan tahap *Forward propagation* dari data target referensi.

3.2 Prinsip Komputasi *Qbot Mobile robot* [10]

Berikut ini catatan yang diadopsi dari Dudek dan Jenkin mengenai prinsip komputasi *mobile robot.*

3.2.1 *Differential Drive Kinematics* [10]

Sebuah *differential drive mobile robot* terdiri dari dua roda yang terpasang pada aksis yang sama, dan tiap roda bisa dikontrol secara bebas untuk menggerakkan robot kearah depan atau ke belakang. Ketika kecepatan tiap roda dapat bervariasi untuk meraih gerakan melingkar, robot harus memutar sekitar titik yang disebut sebagai ICC (*Instantaneous Center of Curvature*), seperti yang ditunjukkan pada Gambar 3.2.



Gambar 3.2 Kinematik Dari *Differential* Robot [10]

Lintasan robot dapat dikendalikan dengan cara mengatur kecepatan pada tiap rodanya. Kecepatan rotasi ω sekitar ICC harus sama pada kedua roda. Maka, Persamaan 3.1 dan 3.2 berikut ini menyusun hubungan antara parameter gerak dari *differential drive mobile robot.*

$$\omega\left(R+\frac{d}{2}\right)=V_r \tag{3.1}$$

$$\omega\left(R-\frac{d}{2}\right)=V_l \tag{3.2}$$

Dimana d adalah jarak antara titik pusat dari kedua roda, V_r dan V_l adalah kecepatan roda kanan dan kiri saat menyusuri lantai,dan R adalah jarak dari ICC ke titik tengah antar roda.

Persamaan 3.1 dan 3.2 dapat diselesaikan pada setiap kasus waktu untuk R dan ω pada Persamaan 3.3 dan 3.4 sebagai berikut

$$R=\frac{d(V_r+V_l)}{2(V_r-V_l)} \tag{3.3}$$

$$\omega = \frac{V_r - V_l}{d} \quad (3.4)$$

3.3 Lokasi ICC [10]

Pada Gambar 3.2 diasumsikan bahwa robot berada di sebuah posisi (x,y) dan meuju kearah yang membuat sudut θ dengan X aksis. Dengan mengetahui kecepatan Vr, Vl dan menggunakan Persamaan 3.5 dan 3.6, lokasi ICC (ICC_x , ICC_y) dapat ditentukan sebagai berikut:

$$ICC_x = x - Rsin\theta \quad (3.5)$$

$$ICC_y = y + Rcos\theta \quad (3.6)$$

3.4 *Forward Kinematics* [10]

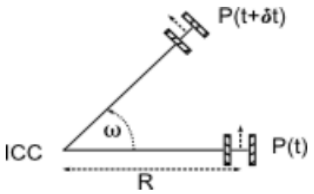
Berikut ini penjelasan permasalahan *forward kinematics* tentang bagaimana kecepatan yang diberikan pada roda dan konfigurasi posisi awal robot $(x, y, \theta)_{t=0}$ dapat menentukan posisi robot (x, y, θ).

3.4.1 *Pose Relative* Robot ke Lokasi ICC [10]

Diberikan sebuah kecepatan yang tidak bervariasi pada Vr dan Vl, lokasi ICC (ICC_x, ICC_y) akan menjadi posisi tetap. Karena itu , pada waktu t+δt pose robot akan menjadi Persamaan 3.7 berikut :

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} \cos(\omega \times \delta t) & -\sin(\omega \times \delta t) & 0 \\ \sin(\omega \times \delta t) & \cos(\omega \times \delta t) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x - ICC_x \\ y - ICC_y \\ \theta \end{bmatrix} + \begin{bmatrix} ICC_x \\ ICC_y \\ \omega \times \delta t \end{bmatrix} \quad (3.7)$$

Dimana (x, y, θ) dan (x', y', θ') adalah posisi robot pada waktu t dan t+δt, berturut-turut. Persamaan ini menggambarkan gerak berputar robot sekitar ICC nya dengan sebuah radius dari lengkungan R sebuah kecepatan sudut ω ,seperti pada Gambar 3.3.



Gambar 3.3 *Forward Kinematics Relative To ICC* [10]

3.4.2 *Pose Relative* Robot pada Posisi Awal Robot [10]

Persamaan gerak umum robot yang mampu bergerak dalam arah θ(t) tertentu pada kecepatan V(t) yang diberikan,dijelaskan pada Persamaan 3.8 di bawah ini :

$$\begin{aligned} x(t) &= \int_0^t V(t)\cos[\theta(t)]\,dt \\ y(t) &= \int_0^t V(t)\sin[\theta(t)]\,dt \\ \theta(t) &= \int_0^t \omega(t)dt \end{aligned} \tag{3.8}$$

Untuk sebuah robot differential drive seperti *Qbot,* Persamaan 3.8 menjadi Persamaan 3.9 di bawah ini :

$$\begin{aligned} x(t) &= \frac{1}{2}\int_0^t [V_r(t) + V_l(t)]\cos[\theta(t)]\,dt \\ y(t) &= \frac{1}{2}\int_0^t V(t)\sin[\theta(t)]\,dt \\ \theta(t) &= \frac{1}{d}\int_0^t \omega(t)dt \end{aligned} \tag{3.9}$$

Persamaan 3.8 dapat disederhanakan untuk menentukan *pose* robot saat V sebagai Persamaan 3.10 sebagai berikut :

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} x + V\cos(\theta)\delta t \\ y + V\sin(\theta)\delta t \\ \theta + \omega\delta t \end{bmatrix} \tag{3.10}$$

Demikian pula pada Persamaan 3.9 akan disederhanakan pada Persamaan 3.11 di bawah ini :

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} x + \frac{1}{2}[V_r + V_l]\cos(\theta)\,\delta t \\ y + \frac{1}{2}[V_r + V_l]\sin(\theta)\,\delta t \\ \theta + \frac{1}{d}[V_r - V_l]\delta t \end{bmatrix} \tag{3.11}$$

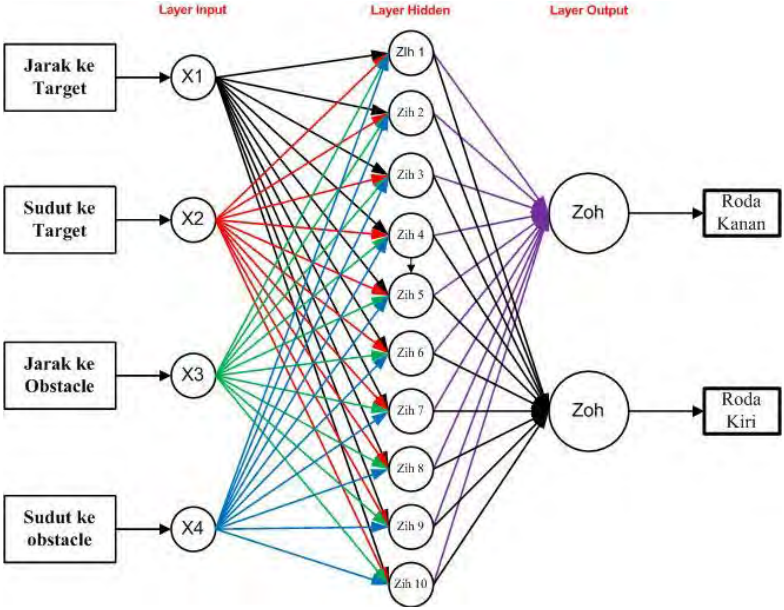
Dengan demikian, untuk kasus khusus dari $V_l=V_r=V$ dan $V_l=-V_r=V$, Persamaan 3.10 menjadi Persamaan 3.12 dan 3.13 berikut

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} x + V\cos(\theta)\delta t \\ y + V\sin(\theta)\delta t \\ \theta \end{bmatrix} \tag{3.12}$$

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} x \\ y \\ \theta + \frac{2V\delta t}{d} \end{bmatrix} \tag{3.13}$$

3.5 Perancangan Metode Jaringan Saraf Tiruan

Jaringan saraf tiruan pada *Qbot mobile robot* ini digunakan untuk navigasi robot untuk mencapai target posisi yang diinginkan. Pada Gambar 3.4 menunjukkan struktur metode jaringan saraf tiruan dengan 4 *input* variabel yaitu x1,x2,x3 dan x4. Kemudian jaringan saraf tiruan ini memiliki 10 *layer hidden* yang akan memproses *output* yaitu menghasilkan *theta* robot yang baru.

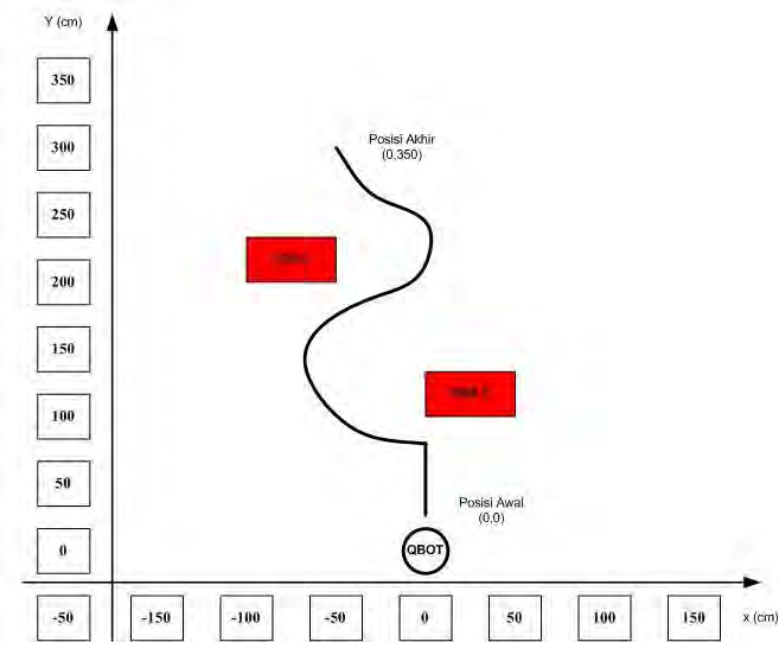


Gambar 3.4 Struktur Jaringan Saraf Tiruan

3.5.1 Tahap *Training* Pola

Pada proses *training* ini *Qbot* robot akan membuat pola jalur dari posisi awal menuju target dalam sebuah lintasan yang terdapat *obstacle*. Robot akan bergerak menuju target dengan dua dimensi koordinat yaitu (x,y) pada satuan centimeter. Pada proses *training* ini robot akan berjalan dari posisi awal menuju target tanpa menabrak *obstacle* yang telah disediakan. Sehingga pada proses akhirnya nanti akan didapatkan beberapa data yang akan dihasilkan dari proses training ini. Data tersebut diantaranya adalah jarak ke target, jarak ke *obstacle*, sudut ke target, sudut ke *obstacle*, kecepatan roda kanan dan kecepatan roda kiri. Dari data yang didapat ini akan digunakan pada proses selanjutnya yaitu proses *learning*.

Pada pola pertama posisi awal *Qbot mobile robot* berada pada koordinat (0 , 0)cm bergerak menuju posisi target (0 , 350)cm dan sudut awal sebesar 90° dengan pola *obstacles* seperti pada Gambar 3.5.



Gambar 3.5 Jalur *Qbot mobile robot*

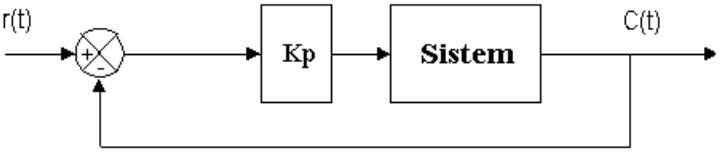
3.5.2 Tahap *Forward propagation*

Pada tahap ini dilakukan perhitungan untuk memperoleh nilai dari *theta robot.* Dalam perhitungan *forward propagation* ini nantinya akan memperoleh nilai bobot untuk roda kanan dan kiri robot yang baru yang nantinya akan dijadikan *input* pada sistem navigasi dan akan dikontrol ulang oleh kontroler PID.

1. ***Input Layer* :**
   Pada bagian *input layer* terdapat 4 *input* variable, diantaranya yaitu jarak target, sudut target, jarak *obstacles*, dan sudut *obstacles*. Keempat *input* variable tersebut yang nantinya akan diproses untuk mendapatkan data bobot baru untuk roda kanan dan kiri robot.
2. ***Hidden Layer* :**
   Hidden layer terdiri dari 10 node yang merupakan proses *input* dari jarak target, sudut target, jarak *obstacles*, dan sudut *obstacles.* Pada lapisan ini dilakukan proses menentukan basis aturan dan *rules*. Untuk proses perhitungannya menggunakan perhitungan *forward propagation*. Setelah itu barulah didapat data bobot baru untuk roda kanan dan kiri robot
3. ***Output Layer* :**
   Output layer merupakan hasil keluaran dari *hidden layer*, tahap ini melakukan proses relasi antara *hidden layer* dengan *output layer*. Pada sisitem ini *output layer* ini hanya ada dua *node* dengan keluaran bobot terbaru untuk roda kanan dan roda kiri pada robot. *Output* ini nantinyaakan dijadikan sebagai *input* untuk kontroler PID (*Proportional Integral Derivatif).*

3.6 Perancangan Kontroler PID

Pada perancangan implementasi PID kali ini sistem yang digunakan menggunakan sistem loop tertutup seperti pada blok berikut ini :



Gambar 3.6 Blok Sistem Loop Tertutup Pada Sistem Robot

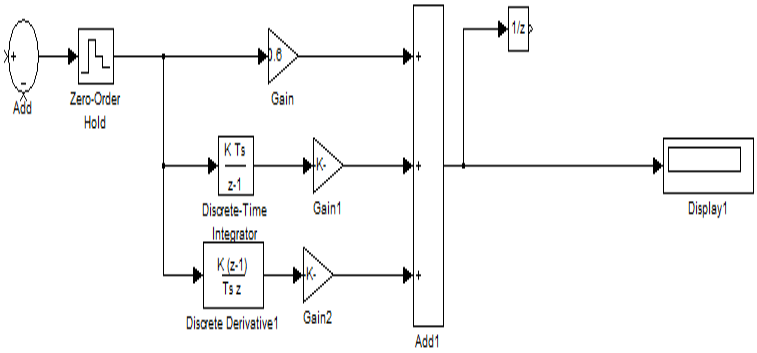
Permasalahan terbesar dalam desain kontroler PID adalah menentukan nilai Kp, Ki, Kd. Metode-metode *tuning* dilakukan berdasarkan model matematika system / *plant*. Jika model matematika tidak diketahui maka dilakukan dengan eksperimen terhadap sistem [11].

Metoda manual dengan *prosedural* MIT. Metode manual dengan *prosedural* MIT ini dilakukan berdasarkan perkiraan dan pengecekan. Pada metoda ini, pengontrolan *proprotional* sangat berperan, sedangkan *integral* dan *derivative* berfungsi memperbaiki *output* yang diinginkan. Ada beberapa nilai praktis untuk menentukan parameter kontrol Kc, Ti, dan Td yang diperoleh dari hasil *tuning*.

Langkah-langkah penalaan yang dilakukan dapat diuraikan sebagai berikut :

1. Jalankan robot dengan kontroler PID. Memberikan nilai Kp hingga kontroler masih menghasilkan keluaran dan *osilasi*.
2. Dengan mengamati keluaran kontroler. Mengatur nilai Kd dan Ki untuk mengurangi *osilasi* keadaan tunak.
3. Didapatkan dengan metode *tuning* manual *prosedural* MIT nilai Ki = 1, Kp = 0,0001 dan nilai Kd = 0,0006

Di bawah ini adalah perancangan kontroler PID *Tuning* pada sistem mobile robot :



Gambar 3.7 Blok Sistem Kontroler PID

Pada sistem ini kontroler PID mendapatkan *input* dari sistem navigasi robot yang sudah melewati proses *learning* pada metode jaringan saraf tiruan. Kemudian diproses oleh kontroler PID agar mendapatkan ketika robot melewati *obstacle* dapat dikontrol kecepatan roda kanan dan roda kiri robot.

Halaman ini sengaja dikosongkan

BAB 4
HASIL DAN ANALISA

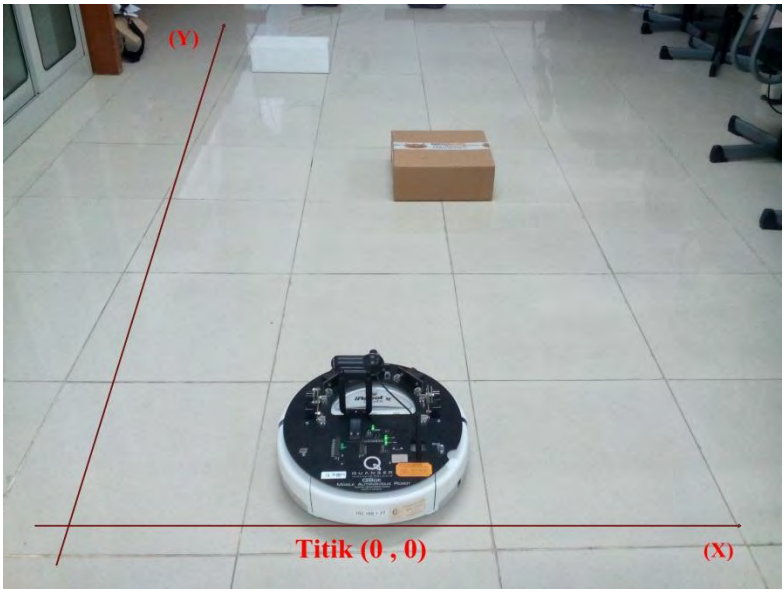
Bab ini membahas mengenai implementasi metode jaringan saraf tiruan dan implementasi controller PID pada *Qbot mobile robot*. Untuk implementasi kontroler PID menggunakan metode jaringan saraf tiruan ini dilakukan proses pengambilan data terlebih dahulu pada *Qbot* robot untuk dilakukan proses pada metode jaringan saraf tiruan. Metode jaringan saraf tiruan pada proses ini menggunakan penghitungan forward propagation. Pada metode jaringan saraf tiruan ini ada empat variabel sebagai masukan yang didapat dari target referensi robot, dantaranya adalah jarak ke target, jarak ke *obstacle*, sudut ke target dan sudut ke *obstacle*. Untuk *output* dari jaringan saraf tiruan ini adalah bobot baru untuk roda kanan dan roda kiri robot yang akan menentukan navigasi robot dari titik awal sampai dengan titik target robot. *Output* dari metode jaringan saraf tiruan inilah yang nantinya akan digunakan sebagai *input* pada kontroler PID dengan tujuan agar kecepatan roda kiri dan roda kanan yang dapat dikontrol.

4.1 Implementasi Perencanaan Jalur

Pada implementasi perencanaan jalur pola ini dilakukan beberapa tahap berikut diantaranya tahap *training*, *offline learning*, dan pengimplementasian kontroler PID dengan bobot baru seperti yang akan dijelaskan di bawah ini.

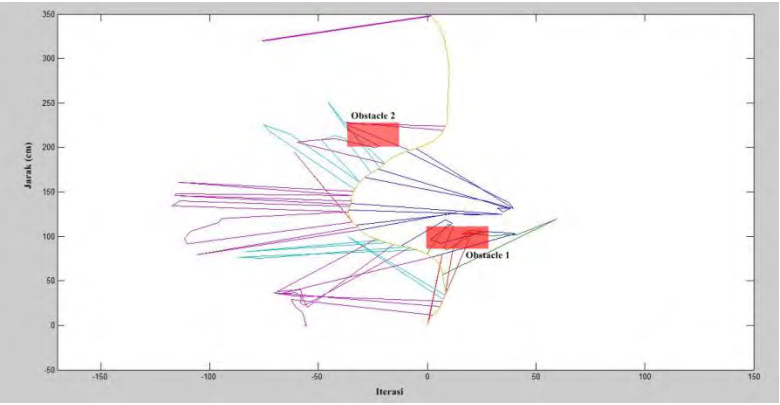
4.1.1 Tahap *Training* Pola

Pada proses *training* ini *Mobile Robot* akan dijalankan dengan menggunakan algoritma sederhana terlebih dahulu. Ditentukan titik awal dan titik target pada robot pada koordinat (x,y). Kemudian menentukan sudut robot sebesar 90 derajat. Setelah robot dijalankan sesuai program yang telah tersedia maka didapatlah empat data *training* yang dibutuhkan untuk diproses, yakni jarak ke target, jarak ke *obstacle*, sudut ke target dan sudut ke *obstacle*. Kemudian di dapat juga data kecepatan roda kanan dan roda kiri yang nantinya digunakan untuk menemukan *theta* robot yang baru untuk dijadikan *input* pada kontroler.



Gambar 4.1 *Qbot Mobile Robot* Pada Pola

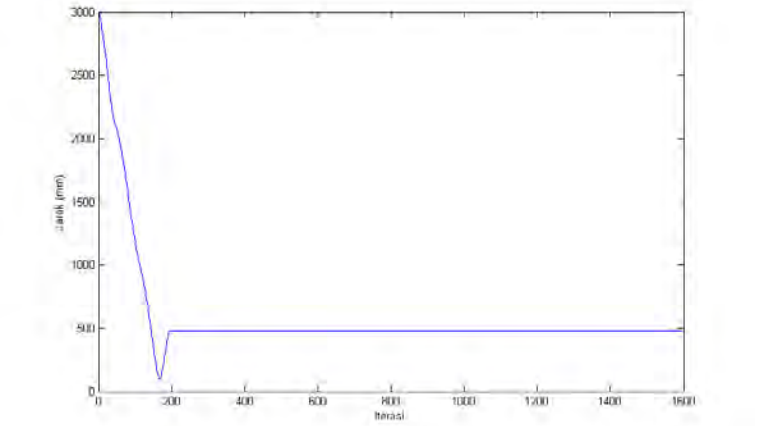
Pada tahap *training* ini robot akan diposisikan pada koordinat (0,0)cm kemudian ditetapkan untuk jarak targetnya yaitu pada koordinat (0,350)cm pada sudut awal 90°. Kemudian dengan menggunakan algoritma sederhana robot dijalankan dari posisi awal menuju target dengan gerakan melingkar ketika ada *obstacle*. Kemudian setelah robot sampai di titik koordinat target robot akan berhenti secara otomatis. Sehingga akan didapatkan beberapa data yang akan diperlukan untuk proses selanjutnya. Data yang didapat dari proses *training* ini diantaranya adalah jarak ke target, sudut ke target, jarak ke *obstacle*, sudut ke *obstacle*, kecepatan roda kanan dan roda kiri. Kemudian data terserbut yang akan digunakan untuk proses *offline learning*. Berikut adalah kurva proses pengambilan data *training* pada robot.



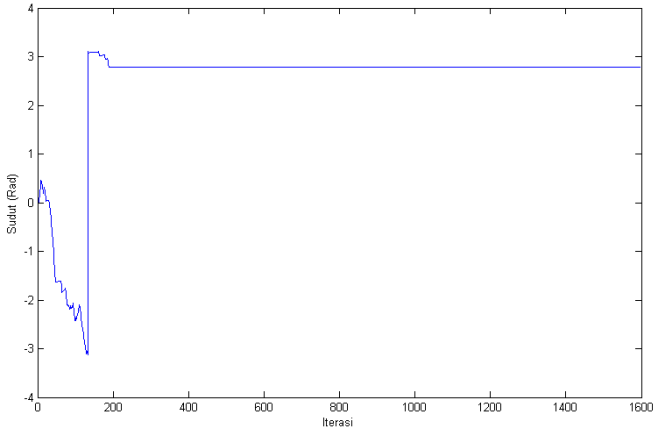
Gambar 4.2 *Training* Jalur *Qbot Mobile Robot* Pada Pola

Terlihat pada kurva robot berjalan pada titik koordinat awal yaitu (0,0) kemudian mengindari *obstacle* dengan gerakan melingkar. Terlihat *obstacle* tergambar pada kotak berwarna merah sehingga robot berjalan menjauhi *obstacle* dalam gerak yang melingkar. Kemudian pada jarak ke 350 cm robot berhenti bergerak.

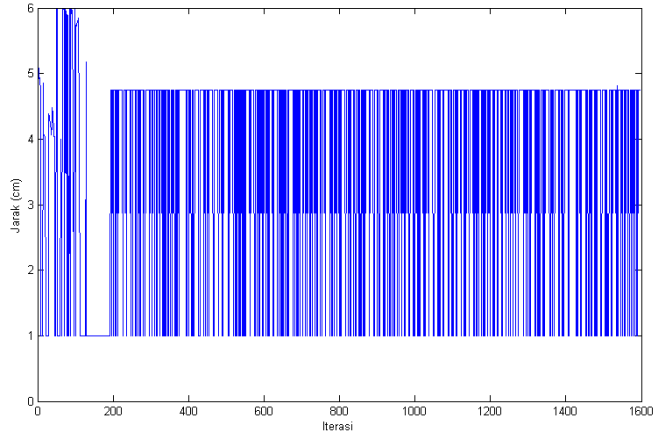
Dari hasil pergerakan *Qbot mobile robot* ini didapatkan data yang akan digunakan untuk proses *learning* seperti pada Gambar 4.3, Gambar 4.4, Gambar 4.5, dan Gambar 4.6.



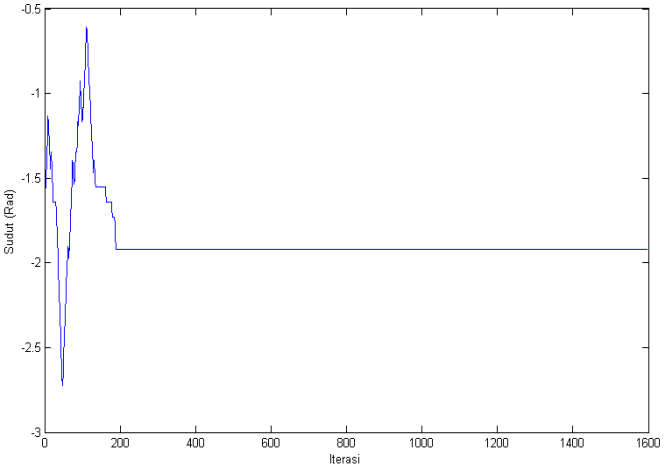
Gambar 4.3 *Output* Jarak Target



Gambar 4.4 *Output* Sudut Target



Gambar 4.5 *Output* Jarak *Obstacles*



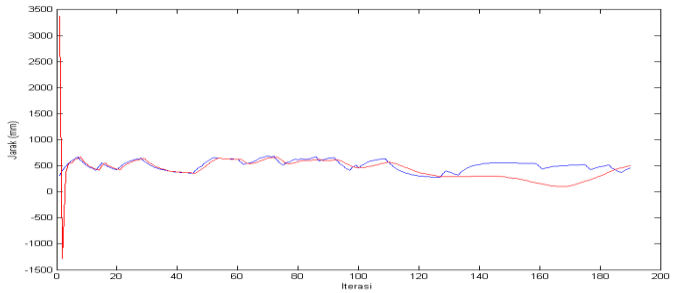
Gambar 4.6 *Output* Sudut *Obstacles*

Dari data diatas, Gambar 4.3 menunjukkan data jarak ke target yang diatur dari titik (0,0)cm menuju titik (0,350)cm. Perubahan grafik jarak target ini tidak *linear* dikarenakan pada saat *Qbot mobile robot* menuju target terdapat *obstacles* sehingga robot harus berbelok menghindar lalu kembali kejalur lintasan menuju target. Gambar 4.4 menunjukkan data sudut target dimana *Qbot mobile robot* mengukur sudut target yang akan dituju agar menjadi acuan robot menuju target. Gambar 4.5 menunjukkan data jarak *obstacles* dengan skala dari 1 sampai 6 cm. Data jarak obastacles ini menjadi acuan *Qbot mobile robot* untuk menghindar dari tabrakan dengan *obstacles*. Gambar 4.6 menunjukkan data sudut *obstacles* dimana data ini manjadi acuan *Qbot mobile robot* untuk mengetahui posisi sudut *obstacles* agar robot dapat menghindar dengan aman.

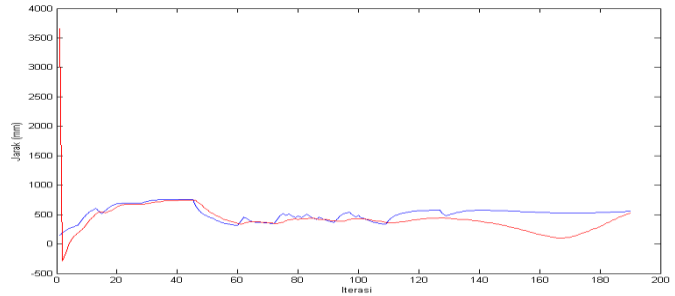
Keempat data gambar yang menunjukkan *output* di atas merupakan data masukan yang kemudian dilakukan komputasi sehingga menghasilkan sinyal kontrol yang akan digunakan untuk mengatur navigasi *Qbot mobile* robot.

4.1.2 Simulasi *Offline Learning*

Dari data yang diperoleh saat *training* dilakukan *learning* yang dilakukan secara *offline* pada data dengan menggunakan algoritma program *forward propagation* metode jaringan saraf tiruan. *Learning* dilakukan untuk mendapatkan *output* pada roda kanan dan roda kiri robot. Proses *learning* dilakukan sebanyak 50.000 *epoch*. Sehingga semakin banyak jumlah *epoch* maka hasil learing semakin baik pula. Di bawah ini adalah gambar hasil dari proses *learning* untuk roda kanan dan roda kiri.



Gambar 4.7 Proses *Learning* Sinyal *Output* Roda Kiri Robot



Gambar 4.8 Proses *Learning* Sinyal *Output* Roda Kanan Robot

Proses learning tersebut menghasilkan nilai bobot yang baru yang nantinya akan dijadikan *input* pada sistem navigasi kemudian akan dikontrol kecepatannya oleh kontroler PID. Nilai bobot yang baru pada kecepatan roda kiri yaitu,

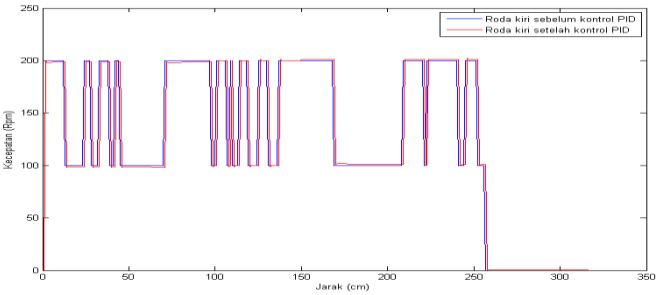
w = [-0,0116 0,4731 -0,0013 0,7767 -0,0096 0,5148 -0,0404 0,4772 0,5031 -0,0086
0,1528 0,6940 0,6396 0,8980 0,8943 0,6364 0,5198 0,7088 0,6098 0,3434
0,7033 0,7086 0,1959 0,8401 0,6590 0,3337 0,4992 0,6049 0,7773 0,6555
-0,5506 -0,4744 -0,7470 -0,6814 -0,1229 -0,4033 -0,3394 -0,7237 -0,2482 -0,2332]

Serta didapat pula hasil *learning* dengan bobot yang baru untuk roda kanan dengan nilai seperti berikut ini :

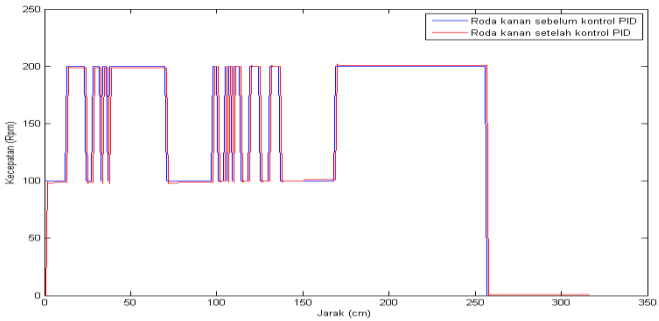
w = [0,0086 -0,0008 0,6496 -0,0006 -0,0002 0,6571 -0,0013 -0,0033 0,8082 -0,0007
0,6393 0,3432 0,8785 0,8628 0,5542 0,8185 0,6231 0,8845 0,8735 0,5519
0,4757 0,5152 0,7416 0,8441 0,3905 0,6205 -0,0774 0,6487 0,5729 0,3613
-0,3944 -0,6309 -0,8985 -0,2268 -0,1703 -0,7243 -0,7254 -0,8460 -0,5897 -0,3162]

4.1.3 Implementasi Pada *Qbot Mobile Robot*

Pada implementasi kontroler PID kali ini, PID mendapatkan *input* dari sistem navigasi robot. Sebelumnya sistem navigasi telah diproses untuk menggunakan jaringan saraf tiruan, dimana nilai bobot baru untuk roda kanan dan kiri sudah dimasukkan pada sistem navigasi. Sehingga *output* dari sistem navigasi tersebut yang akan diproses oleh PID untuk kecepatan roda kanan dan roda kiri robot. Berikut pada Gambar 4.9 dan Gambar 4.10 adalah hasil sinyal kontrol untuk kecepatan roda kanan dan roda kiri robot :

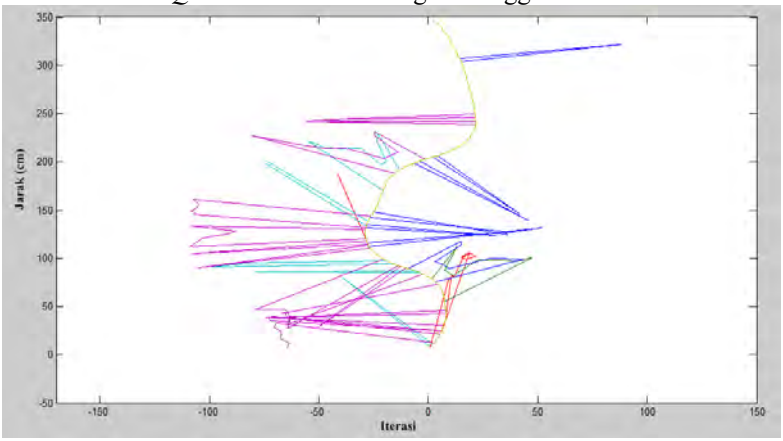


Gambar 4.9 Sinyal Kontrol Kecepatan Roda Kiri Robot



Gambar 4.10 Sinyal Kontrol Kecepatan Roda Kanan Robot

Hasil dari menjalankan *Qbot mobile robot* ini dapat dilihat pada Gambar 4.9 dan 4.12 sinyal kontrol PID *Qbot mobile robot* dari tiga kali percobaan mendekati sinyal kontrol data referensi dan waktu mencapai targetnya kontroler PID memiliki respon lebih cepat dan besar daripada data referensinya. Berikut pada Gambar 4.11 menunjukkan jalur yang dihasilkan oleh *Qbot mobile robot* dengan menggunakan kontroler PID.



Gambar 4.11 Jalur *Qbot* Robot Dengan Kontroler PID

Dari sinyal kontrol diperoleh kecepatan roda kanan dan roda kiri. Dari hasil perbandingan sinyal kontrol PID dan sinyal kontrol data referensi terlihat kecepatan roda kanan dan kecepatan roda kiri PID dari

tiga kali percobaan mendekati kecepatan roda kiri dan kecepatan roda kanan.

Dari hasil data kecepatan *Qbot mobile robot* diatas dapat diketahui perbandingan waktu yang dibutuhkan *Qbot mobile robot* untuk mencapai posisi target saat *training* dengan *Qbot mobile robot* yang menggunakan kontroler PID dengan dilakukaknnya tiga kali percobaan dapat dilihat pada Tabel 4.1.

Tabel 4.1 Perbandingan Waktu *Training* Dengan Kontroler PID

Data	**Koordinat awal *Qbot* (cm)**	**Koordinat target *Qbot* (cm)**	**Waktu (detik)**
Program Manual Per. 1	(0 , 0)	(0 , 350)	52
Program Manual Per. 2	(0 , 0)	(0 , 350)	48
Program Manual Per. 3	(0 , 0)	(0 , 350)	50
Percobaan 1 PID	(0 , 0)	(0 , 350)	23
Percobaan 2 PID	(0 , 0)	(0 , 350)	22
Percobaan 3 PID	(0 , 0)	(0 , 350)	25

Dari Tabel 4.1 tersebut terlihat bahwa respon kontroler PID lebih cepat dari data referensi hasil manual. Sehingga kontroler ini sangat baik untuk mempercepat respon dari model referensinya.

Halaman ini sengaja dikosongkan

BAB 5
PENUTUP

5.1 Kesimpulan

Dari hasil pengimplementasian perencanaan jalur pada *Qbot mobile robot* menggunakan metode jaringan saraf tiruan dengan kontroler PID dapat disimpulkan bahwa sistem kontroler PID ini dapat mengikuti data referensi dari hasil training yang telah dilakukan. Kontroler PID memiliki respon yang lebih cepat dibandingkan dengan respon dengan menggunakan program manual robot *Qbot*. Diketahui menggunakan program manual dari robot dengan target koordinat yang sama yaitu (0 3500) mm didapat hasil pada percobaan pertama yaitu 52 detik, pada percobaan kedua yaitu 48 detik dan pada percobaan ketika yaitu 50 detik. Dengan tiga kali percobaan menggunakan algoritma kontroler PID dan metode jaringan syaraf tiruan didapatkan waktu 23 detik pada percobaan pertama, 22 detik pada percobaan kedua, kemudian 25 pada percobaan ketiga dengan jarak target yang sama yaitu koordinat (0 , 3500) mm.

5.2 Saran

Untuk pengembangan penelitian mengenai *Qbot mobile robot* diharapkan pada penelitian selanjutnya dapat menggunakan metode lain untuk proses sistem navigasi robot. Selain itu diharapkan dapat mengembangkan pergerakan *Qbot mobile robot* ini menggunakan kontroler lain dengan dilengkapi komponen lain yang mendukung kinerja robot seperti menggunakan kamera *opti track* untuk proses *trayektori.*

DAFTAR PUSTAKA

[1] _____,"*Quanser Robot*". 21 Desember 2015. http: //www.quanser.com/products/qbot2

[2] Randy Reza Kautsar, A.R. Anom Besari. "Pengendalian Posisi Mobile Robot Menggunakan Metode Neural Network Dengan Umpan Balik Kamera Pemosisian Global", *Jurnal Electric Teknik Komputer*. Politeknik Elektronika Negeri Surabaya.

[3] _____, "*User Manual Quanser Qbot*" Document number 830.Revision 7, Inovative Educate, Quanser., 2012

[4] Pertiwi, Mentari Madi., "Implementasi Perancangan Kontroler Neuro Fuzzy Untuk Pengaturan Kecepatan Motor Spindle Pada CNC Jenis Milling", *Tugas Akhir* Teknik Elektro, Institut Teknologi Sepuluh Nopember, Surabaya, 2015.

[5] _____,"*Jaringan Syaraf Tiruan*". 08 Oktober 2014. https://rahmadya.com/2012/01/18/hidden-layer-pada-jaringan-syaraf-tiruan/

[6] Kuswadi, Son. *Kendali Cerdas Teori dan Aplikasi Praktisnya* 1st ed. Andi. Indonesia. 2007.

[7] _____,"*Kontroler PID*". 14 Mei 2015. http: // meriwardana.blogspot.co.id/2010/06/pengendali-pid-pid-controller-dengan_07.html

[8] Leotman, Baradiant Ivano., "Desain Kontroler Fuzzy PD untuk Pelacakan Objek pada Qbot Mobile Robot", *Tugas Akhir* Teknik Elektro, Institut Teknologi Sepuluh Nopember, Surabaya, 2014.

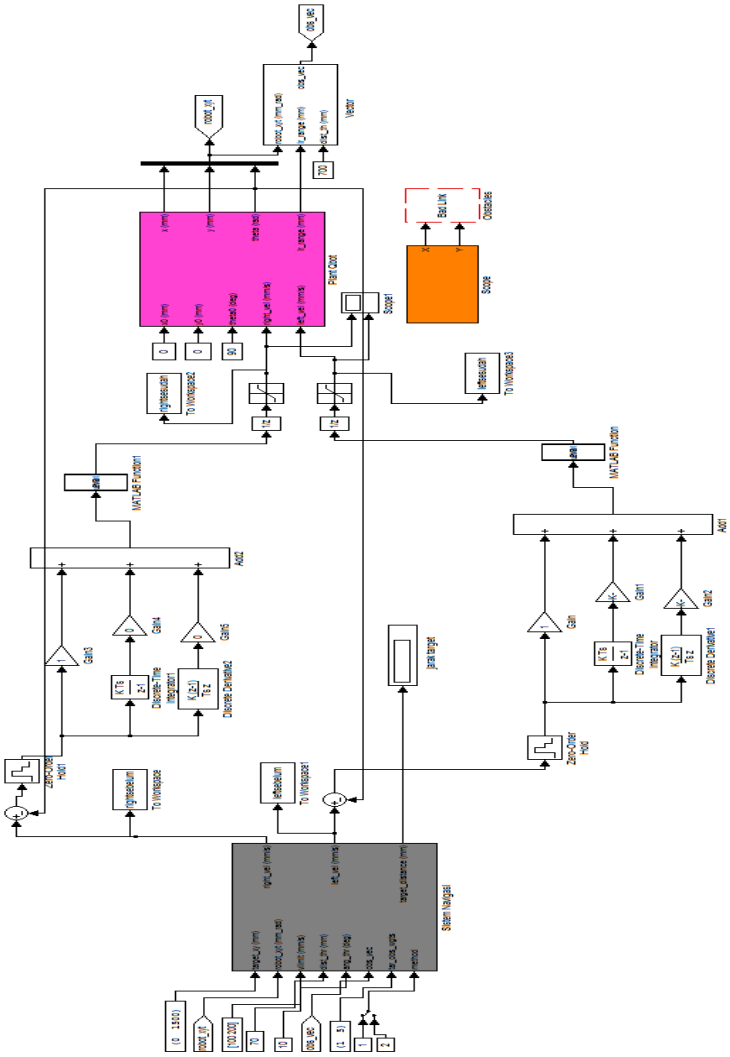
[9] Fahmizal., "Implementasi Sistem Navigasi Behavior Based Dan Kontroler PID Pada Manuver Robot Maze", *Tugas Akhir* Teknik Elektro, Institut Teknologi Sepuluh Nopember, Surabaya, 2011.

[10] _____, "*Kinematics Modeling*" Document number 830.Revision 7, Inovative Educate, Quanser., Agustus, 2012

[11] Arnas Elmiawan Akbar, Waru Djuriatno. "Implementasi Sistem Navigasi Wall Following Menggunakan Kontroler PID dengan Metode Tuning pada Robot Kontes Robot Cerdas Indonesia (KRCI) Divisi Senior Beroda", *Jurnal Electric Teknik Elektro*. Universitas Brawijaya.Malang

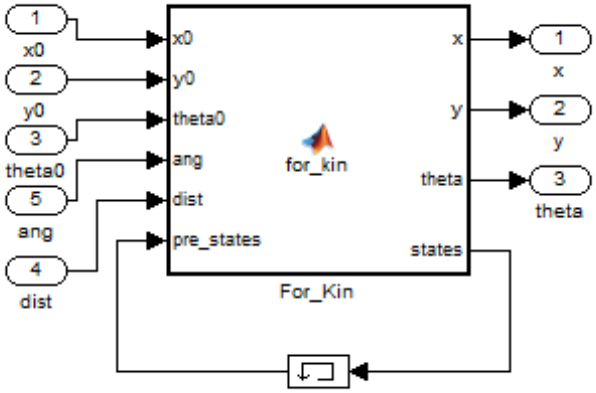
LAMPIRAN

1. LAMPIRAN A

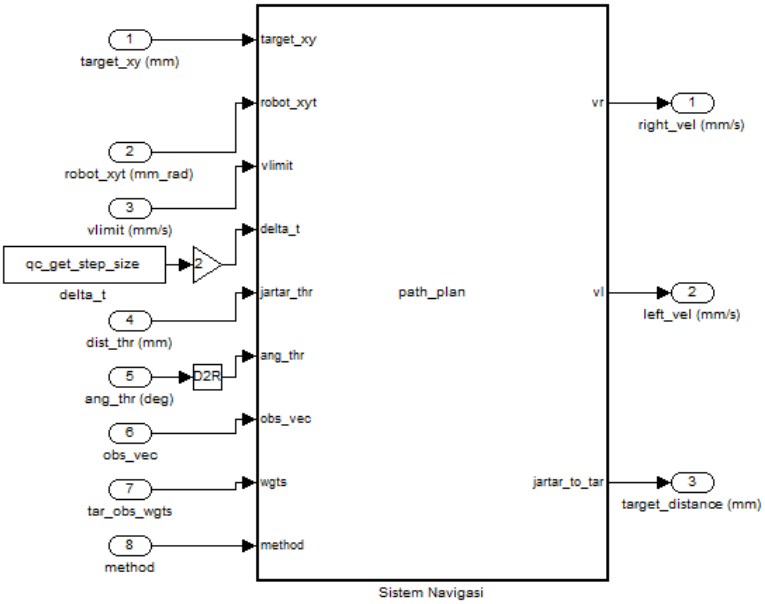


Gambar A1. Blok keseluruhan program

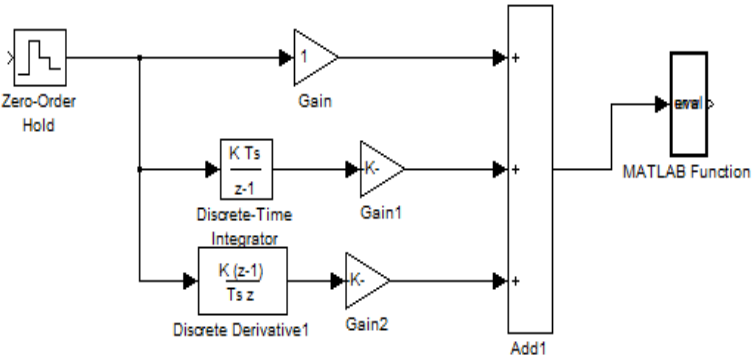
Gambar A2. Blok *Qbot*



Gambar A3. Blok forward kinematik



Gambar A4. Blok Sistem Navigasi



Gambar A5. Blok Kontroler PID

2. LAMPIRAN B

B1. Program *Forward Kinematics*

```
function [x, y, theta, states] = for_kin(x0, y0, theta0, ang, dist,
pre_states)
% This block supports an embeddable subset of the MATLAB language.
% See the help menu for details.

states = pre_states;

if states(1) == 0
    x = x0;
    y = y0;
    theta = check_angle(theta0);
    states(1) = 1;
else
    theta = check_angle(check_angle(states(4)) + check_angle(ang));
    x = states(2) + dist*cos(theta);
    y = states(3) + dist*sin(theta);
end

states(2) = x;
states(3) = y;
states(4) = theta;

return;

function y = check_angle(x)
y = x;
if y > pi
    y = x - 2*pi;
elseif y < -pi
    y = x + 2*pi;
end
return;
```

B2. Program Sistem Navigasi

```
function [vr, vl, dist_to_tar] = path_plan(target_xy,robot_xyt, ...
    vlimit, delta_t, dist_thr, ang_thr, obs_vec, wgts, method)
% This block supports an embeddable subset of the MATLAB language.
% See the help menu for details.

rx = robot_xyt(1);
ry = robot_xyt(2);
rtheta = robot_xyt(3);
tx = target_xy(1);
ty = target_xy(2);


dist_to_tar = find_dist(rx, ry, tx, ty);
if dist_to_tar < dist_thr
    vr = int16(0);
    vl = int16(0);
else
    ang_to_tar = find_theta(rx, ry, rtheta, tx, ty);
    obs_vec_abs = abs(obs_vec);
    if obs_vec_abs ~= 0
        tar_vec = complex((tx-rx), (ty-ry));
        tar_vec_abs = abs(tar_vec);
        if tar_vec_abs ~= 0
            tar_vec = tar_vec/tar_vec_abs;
        end
        vec = abs(wgts(1))*tar_vec - abs(wgts(2))*obs_vec;
        ang_to_Obs= check_angle(atan2(imag(vec), real(vec)) - rtheta);

    end
    if method == 2
        % Metode 2
        [vr, vl] = solve_inv_kin(dist_to_tar, ang_to_tar, vlimit, delta_t);
    else
        % metode 1 (JST)
        dist_to_Obs=abs(vec);
        vra = jstvr(dist_to_tar,ang_to_tar,dist_to_Obs,ang_to_Obs);
        vla = jstvl(dist_to_tar,ang_to_tar,dist_to_Obs,ang_to_Obs);
```

```
        vr=int16(vra);
        vl=int16(vla);


    end
end

return;


function dist = find_dist(rx, ry, tx, ty)
dist = sqrt((rx-tx)^2 + (ry-ty)^2);
return;

function theta = find_theta(rx, ry, rtheta, tx, ty)
X = tx - rx;
Y = ty - ry;
theta = atan2(Y, X);
theta = check_angle(theta - check_angle(rtheta));
return;

% function [vr, vl] = rotate_and_go(theta,vlimit, ang_thr)
%
% vhn=vlimit(2);
% vln=vlimit(1);
% if abs(theta) > ang_thr
%     if theta >= 0
%         vr = int16(vhn);
%         vl = int16(vln);
%     else
%         vr = int16(vln);
%         vl = int16(vhn);
%     end
% else
%     vr = int16(vhn);
%     vl = int16(vhn);
% end
%
% return;
```

```
function [vr, vl] = solve_inv_kin(dist,theta, vlimit, delta_t)
d = 252.5;
vmax = vlimit(2);
wmax = (2*vmax)/d;
w = theta/delta_t;
w_sign = sign(w);
if abs(w) > wmax
    w = w_sign*wmax;
end
v = dist/delta_t;
if v > vmax
    v = vmax;
end
vr_tmp = (2*v + d*w)/2;
vl_tmp = 2*v - vr_tmp;

max_of_vrvl = abs(max(vr_tmp, vl_tmp));
if max_of_vrvl > vmax
    vr_tmp = (vr_tmp/max_of_vrvl)*vmax;
    vl_tmp = (vl_tmp/max_of_vrvl)*vmax;
end
vr = int16(vr_tmp);
vl = int16(vl_tmp);


return;


function y = check_angle(x)
y = x;
if y > pi
    y = x - 2*pi;
elseif y < -pi
    y = x + 2*pi;
end
return;
function yol = jstvl(x1,x2,x3,x4)
```

```
%perhitungan forward output hiddentvl
wil1 = [-0.0116    0.4731   -0.0013    0.7767   -0.0096    0.5148   -0.0404
0.4772    0.5031   -0.0086];
wil2 = [ 0.1528    0.6940    0.6396    0.8980    0.8943    0.6364    0.5198
0.7088    0.6098    0.3434];
wil3 = [ 0.7033    0.7086    0.1959    0.8401    0.6590    0.3337    0.4992
0.6049    0.7773    0.6555];
wil4 = [-0.5506   -0.4744   -0.7470   -0.6814   -0.1229   -0.4033   -
0.3394   -0.7237   -0.2482   -0.2332];

wol = [-0.5598
        0.2752
        0.0017
        0.3618
       -0.1698
        0.2882
        0.0850
        0.2772
        0.2822
        0.0492];

 lamda=1;
 zhl1=(-0.0116*x1)+(0.1528*x2)+(0.7033*x3)+(-0.5506*x4);
 zhl2=(0.4731*x1)+(0.6940*x2)+(0.7086*x3)+(-0.4744*x4);
 zhl3=(-0.0013*x1)+(0.6396 *x2)+(0.1959*x3)+(-0.7470*x4);
 zhl4=(0.7767*x1)+(0.8980*x2)+(0.8401*x3)+ (-0.6814*x4);
 zhl5=(-0.0096*x1)+(0.8943*x2)+(0.6590*x3)+(-0.1229*x4);
 zhl6=(0.5148*x1)+(0.6364*x2)+(0.3337*x3)+(-0.4033*x4);
 zhl7=(-0.0404*x1)+(0.5198*x2)+(0.4992*x3)+(-0.3394*x4);
 zhl8=(0.4772*x1)+(0.7088*x2)+(0.6049*x3)+(-0.7237*x4);
 zhl9=(0.5031*x1)+(0.6098*x2)+(0.7773*x3)+(-0.2482*x4);
 zhl10=(-0.0086*x1)+(0.3434*x2)+(0.6555*x3)+(-0.2332*x4);


yol1=(wol(1)*zhl1)+(wol(2)*zhl2)+(wol(3)*zhl3)+(wol(4)*zhl4)+(wol(
5)*zhl5)+(wol(6)*zhl6)+(wol(7)*zhl7)+(wol(8)*zhl8)+(wol(9)*zhl9)+(
wol(10)*zhl10);
 yol= yol1;
```

```
return;
function yor = jstvr(x1,x2,x3,x4)
%perhitungan forward output hiddent vr
wir1 = [0.0086   -0.0008    0.6496   -0.0006   -0.0002    0.6571   -0.0013
-0.0033    0.8082   -0.0007];
wir2 = [0.6393    0.3432    0.8785    0.8628    0.5542    0.8185    0.6231
0.8845    0.8735    0.5519];
wir3 = [0.4757    0.5152    0.7416    0.8441    0.3905    0.6205   -0.0774
0.6487    0.5729    0.3613];
wir4 = [-0.3944   -0.6309   -0.8985   -0.2268   -0.1703   -0.7243   -
0.7254   -0.8460   -0.5897   -0.3162];

wor = [0.2125
      -0.1177
       0.5159
      -0.1400
      -0.1475
       0.4540
      -0.0348
       0.0351
       0.5528
      -0.0847];
 lamda=1;
 zhr1=(0.0086*x1)+(0.6393*x2)+(0.4757*x3)+(-0.3944*x4);
 zhr2=(-0.0008*x1)+( 0.3432*x2)+(0.5152*x3)+(-0.6309*x4);
 zhr3=(0.6496*x1)+(0.8785*x2)+(0.7416*x3)+(-0.8985*x4);
 zhr4=(-0.0006*x1)+(0.8628*x2)+(0.8441*x3)+(-0.2268*x4);
 zhr5=(-0.0002*x1)+(0.5542*x2)+(0.3905*x3)+(-0.1703*x4);
 zhr6=(0.6571*x1)+(0.8185*x2)+(0.6205*x3)+(-0.7243*x4);
 zhr7=(-0.0013*x1)+(0.6231*x2)+(-0.0774*x3)+(-0.7254*x4);
 zhr8=(-0.0033*x1)+(0.8845*x2)+(0.6487*x3)+(-0.8460*x4);
 zhr9=(0.8082*x1)+(0.8735*x2)+(0.5729*x3)+(-0.5897*x4);
 zhr10=(-0.0007*x1)+(0.5519*x2)+(0.3613*x3)+(-0.3162*x4);

yor1=(wor(1)*zhr1)+(wor(2)*zhr2)+(wor(3)*zhr3)+(wor(4)*zhr4)+(wo
r(5)*zhr5)+(wor(6)*zhr6)+(wor(7)*zhr7)+(wor(8)*zhr8)+(wor(9)*zhr9)
+(wor(10)*zhr10);
 yor= yor1;
return;
```

B3. Program Learning Jaringan Saraf Tiruan

```
clear all
close all
clc
load 'training.mat'
xx1=Jartar.signals.values
xx2=Sudtar.signals.values
xx3=tarobs.signals.values
xx4=Surobs.signals.values

vr=RodaKanan.signals.values
vl=RodaKiri.signals.values

jmd =length(xx1)
kk=0
a1=0.8;
b1=0.5;
for i=1:jmd
   vrobot=abs( (double(vr(i))+double(vl(i)))/2 );
   if vrobot>25
      kk=kk+1;
      % vin(kk)=vrobot;
      % th(kk)=asin((double(vr(i))-
double(vl(i)))/(double(vr(i))+double(vl(i))));
      uk=double(vl(i));
      if kk==1
         th(kk)=uk;
      else
         th(kk)=a1*th(kk-1)+b1*uk;
      end
      x1(kk)=xx1(i);
      x2(kk)=xx2(i);
      x3(kk)=xx3(i);
      x4(kk)=xx4(i);
   end
end

jumdat=kk;
```

```
jeh=10;% boleh diganti sembarang
wih=rand(4,jeh);
woh=rand(jeh,1);
wmax=0.9
lambda=1;
alpha=0.0000001;
thno=0

%pause()

for epp=1:500
   epp
   for d=1:jumdat
      tt(d)=d;
      %pause()
      %perhitungan forward output hiddent
      for j=1:jeh

zh(j)=x1(d)*wih(1,j)+x2(d)*wih(2,j)+x3(d)*wih(3,j)+x4(d)*wih(4,j);
         yh(j)=lambda*zh(j);
      end
      %perhitungan forward output neuro untuk sudut thn
      zo=0;
      for j=1:jeh
         zo=zo+woh(j)*yh(j);
      end
      thn=lambda*zo;

      %menghitung eror output etn
      etn=th(d)-thn;
      thno(d)=thn;
      %menghitung perambatan eror mundur ke level hiddent eth(j)
      for j=1:jeh
         if zo==0
            eth(j)=etn/jeh;
         else
            eth(j)=etn*woh(j)*yh(j)/zo;
         end
```

```
        end
        % Revisi BOBOT woh(j)
        for j=1:jeh
           woh(j)=woh(j)+alpha*etn*lambda*yh(j);
           if abs(woh(j))> wmax
              woh(j)=sign(woh(j))*rand();
           end
        end
         % Revisi BOBOT wih(i,j)

         for j=1:jeh
            wih(1,j)=wih(1,j)+alpha*eth(j)*lambda*x1(d);
            if abs(wih(1,j))> wmax
               wih(1,j)=sign(wih(1,j))*rand();
            end

            wih(2,j)=wih(2,j)+alpha*eth(j)*lambda*x2(d);
            if abs(wih(2,j))> wmax
               wih(2,j)=sign(wih(2,j))*rand();
            end
            wih(3,j)=wih(3,j)+alpha*eth(j)*lambda*x3(d);
            if abs(wih(3,j))> wmax
               wih(3,j)=sign(wih(3,j))*rand();
            end
            wih(4,j)=wih(4,j)+alpha*eth(j)*lambda*x4(d);
           if abs(wih(4,j))> wmax
               wih(4,j)=sign(wih(4,j))*rand();
            end
         end
    end
%     plot(tt,th,'b',tt,thno,'r');
%     pause(1000)
end
plot(tt,th,'b',tt,thno,'r')
th
thno
woh
wih
```

Halaman ini sengaja dikosongkan

RIWAYAT HIDUP PENULIS

Nama	:	Eva Sri Oktavia N
TTL	:	Pamekasan, 14 Oktober 1993
Jenis Kelamin	:	Perempuan
Agama	:	Islam
Alamat Rumah	:	JL. Purba 35 Pamekasan
Telp/HP	:	083850263211
E-mail	:	evasrioktavia@gmail.com

RIWAYAT PENDIDIKAN

- 1999 – 2005 : SDN Barurambat Kota 1 Pamekasan
- 2005 – 2008 : SMP Negeri 2 Pamekasan
- 2008 – 2011 : SMK Negeri 3 Pamekasan
- 2011 – 2014 : Bidang Studi Komputer Kontrol, Program D3 Teknik Elektro, ITS
- 2014 – Sekarang : Bidang Studi Teknik Sistem Pengaturan, S1 Teknik Elektro, ITS

PENGALAMAN KERJA

- Kerja Praktek di PLTU. IPMOMI Paiton, Probolinggo (1 Juli – 26 Juli 2013)
- PT. Infomedia Solusi Humanika (14 Oktober 2014 – 14 Februari 2015)
- Kerja Praktek di PT. Indonesia Power UPB Pesanggaran, Bali (27 Juli – 27 Agustus 2015)

PENGALAMAN ORGANISASI

- Sekretaris Departemen Hubungan Luar HIMAD3TEKTRO 2012 – 2013
- Kepala Departemen Hubungan Luar HIMAD3TEKTRO 2013 – 2014