UNIVERSITAS INDONESIA

RANCANG BANGUN PENGENDALIAN DAN KOMUNIKASI *MOBILE ROBOT*

SKRIPSI

YUNIKE LEVINA

1306410805

FAKULTAS TEKNIK

PROGRAM STUDI TEKNIK ELEKTRO

DEPOK

JUNI 2017

UNIVERSITAS INDONESIA

RANCANG BANGUN PENGENDALIAN DAN KOMUNIKASI *MOBILE ROBOT*

SKRIPSI

Diajukan sebagai salah satu syarat untuk memperoleh gelar Sarjana Teknik

YUNIKE LEVINA

1306410805

FAKULTAS TEKNIK

KEKHUSUSAN TEKNIK KENDALI

DEPOK

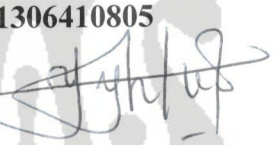
JUNI 2017

HALAMAN PERNYATAAN ORISINALITAS

Skripsi ini adalah hasil karya saya sendiri,

dan semua sumber baik yang dikutip maupun dirujuk

telah saya nyatakan dengan benar.

Nama	**: Yunike Levina**
NPM	**: 1306410805**
Tanda Tangan	**:** 
Tanggal	**: 21 Juni 2017**

HALAMAN PENGESAHAN

Skripsi ini diajukan oleh	:
Nama	: Yunike Levina
NPM	: 1306410805
Program Studi	: Teknik Elektro
Judul Skripsi	: Rancang Bangun Pengendalian dan Komunikasi *Mobile Robot*

Telah berhasil dipertahankan di hadapan Dewan Penguji dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Sarjana Teknik pada Program Studi Teknik Elektro, Fakultas Teknik, Universitas Indonesia.

DEWAN PENGUJI

Pembimbing	: Dr. Abdul Muis, S.T., M.Eng	(.....)
Penguji	: Dr. Ir. Aries Subiantoro, M.SEE.	(.....)
Penguji	: Ir. Wahidin Wahab, M.Sc., Ph.D	(.....)

Ditetapkan di : Depok

Tanggal : 21 Juni 2017

KATA PENGANTAR

Puji syukur kepada Tuhan Yang Mahaesa atas segala karunia-Nya serta berkat-Nya sehingga penulis dapat menyelesaikan skripsi ini. Penulisan skripsi dilakukan dalam rangka memenuhi salah satu syarat untuk mendapatkan gelar Sarjana Teknik Program Studi Teknik Elektro, Fakultas Teknik, Universitas Indonesia.

Penyusunan skripsi ini tidak lepas dari banyaknya dukungan dari berbagai pihak sehingga laporan ini dapat diselesaikan dengan baik dan tepat pada waktunya. Secara khusus penulis ingin mengucapkan terima kasih kepada pihak-pihak yang telah membantu selama proses penulisan skripsi ini di antaranya :

1. Dr. Abdul Muis, S.T., M.Eng selaku dosen pembimbing yang telah meluangkan waktu, pikiran, serta tenaga untuk membantu penulis menyelesaikan skripsi;
2. Bapak dan Ibu yang turut memberikan motivasi serta menyediakan fasilitas agar penulis dapat menyelesaikan tugas akhir, begitu juga dengan kakak-kakak.
3. Anthony, Darwin, dan Suwandi, teman satu bimbingan yang turut membantu dalam perancangan sistem multi mobile robot.

Akhir kata, semoga semua kebaikan yang telah dilakukan pihak-pihak di atas dibalas oleh Tuhan Yang Mahaesa. Semoga skripsi ini dapat menambah wawasan serta bermanfaat bagi penulis dan para pembaca.

Depok, 2 Juni 2017

Penulis

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

Sebagai sivitas akademik Universitas Indonesia, saya yang bertanda tangan di bawah ini:

Nama : Yunike Levina
NPM : 1306410805
Program Studi : Teknik Elektro
Departemen : Teknik Elektro
Fakultas : Teknik
Jenis Karya : Skripsi

demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Indonesia **Hak Bebas Royalti Noneksklusif (*Non-exclusive Royalty-Free Right*)** atas karya ilmiah saya yang berjudul:

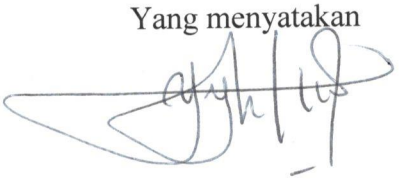
"Rancang Bangun Pengendalian dan Komunikasi *Mobile Robot*"

beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Noneksklusif ini Universitas Indonesia berhak menyimpan, mengalihmedia/format-kan, mengelola dalam bentuk pangkalan data (*database*), merawat, dan memublikasikan tugas akhir saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Depok
Pada tanggal : 21 Juni 2017

Yang menyatakan



Yunike Levina

ABSTRAK

Nama : Yunike Levina
Program Studi : Teknik Elektro
Judul : Rancang Bangun Pengendalian dan Komunikasi *Mobile Robot*

Dalam kehidupan sehari-hari tak jarang ditemukan pekerjaan yang membutuhkan lebih dari satu orang dalam penyelesaiannya. Konsep tersebut diadaptasikan ke penyelesaian tugas kompleks untuk sistem otonomi dengan lebih dari satu mobile robot atau disebut juga mobile robot kooperatif. Dalam mengakomodasi sistem mobile robot kooperatif yang baik, beberapa aspek perlu diperhatikan terutama komunikasi antar anggotanya.

Pada skripsi ini, *mobile robot* akan dirancang dengan menggunakan trayektori linier dan sinusoidal sebelum antar robotnya dikomunikasikan untuk bertukar informasi. Sistem meggunakan protokol komunikasi nirkabel *internet socket* sebagai media pertukaran informasi antar robotnya sehingga pengujian terhadap komunikasi juga perlu dilakukan.

Berdasarkan hasil pengujian dapat diketahui bahwa setiap mobile robot mempunyai karakteristik dan pergerakan yang berbeda satu sama lain tetapi masih dapat dikendalikan dengan menggunakan nilai pengendali yang sama. Hasil pengujian juga menunjukkan bahwa komunikasi dengan *internet socket* sudah dapat digunakan dalam aplikasi *mobile robot* komunikatif

Kata Kunci: *mobile robot, pengendali posisi, pengendali orientasi, pengendali kecepatan, odometri, filter low pass, perancangan trayektori, pemrograman socket, internet socket, robot komunikatif.*

ABSTRACT

Name : Yunike Levina

Major : Electrical Engineering

Title : Design and Implementation of Mobile Robot Control and Inter-Robot Communication

In daily life, a lot of tasks need more than one people to complete because of it complexity. The concept of using more hand to complete a complex problems is adapted in autonomous system that used more than one robot which often defined as cooperative robot. In order to accommodate a good cooperative mobile robot system, interrobot communication should be carefully designed.

In this script, the mobile robot would be design while using linear and sinusoidal trajectory to test whether before being communicated between each other. The system using wireless internet socket communication protocol as the information exchange's media between the robots, therefore an experiment need to be done to test the communication as well.

Accordintg to experiment done, the result show that each robot has its own characteristic and movement dyamics. However, the differences are still tolerable and still can be controlled using the same controllers' constans.The experiment also show that internet socket communication is proven to be able implemented in communicative mobile robots.

Keywords: *mobile robot, position control, orientation control, speed control, odometry, low pass filter, trajectory planning, socket programming, internet socket, communicative robots.*

DAFTAR ISI

DAFTAR GAMBAR

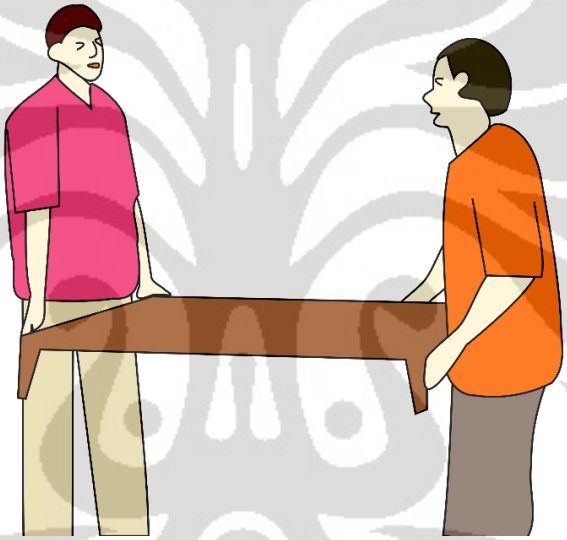
DAFTAR TABEL

BAB 1

PENDAHULUAN

1.1 Latar Belakang

Dalam kehidupan sehari-hari, manusia kerap kali dihadapkan dalam situasi yang pelaksanaannya membutuhkan bantuan orang lain, seperti memindahkan barang yang besar dan berat, maupun menarik barang berat ataupun distribusi barang dalam jumlah besar. Hal-hal diatas merupakan sebagian kecil dari pekerjaan yang dilakukan sehari-hari namun tidak dapat dilakukan seorang diri, sehingga dalam pengerjaannya dilakukan bersama orang lain secara sinergis.



Gambar 1.1 Ilustrasi Pengangkutan Meja yang Dilakukan dengan Bantuan Orang Lain

Konsep pengerjaan suatu hal yang cukup berat untuk dilakukan sendiri sehingga butuh dilakukan oleh lebih dari satu orang tersebutlah yang diadaptasi dalam sistem otomasi, yaitu penggunaan banyak robot yang bekerja secara kooperatif dalam melakukan suatu pekerjaan yang kompleks. Beberapa robot diatur agar dapat bekerja secara kolaboratif dalam mencapai tujuan yang sama, yaitu menyelesaikan pekerjaan kompleks yang dimaksudkan di atas. Multi-robot kooperatif tersebut biasanya digunakan untuk membantu dalam transportasi barang dalam ukuran maupun jumlah yang besar.

Seiring dengan berjalannya waktu, penelitian mengenai multi-robot kooperatif dalam bidang transportasi semakin marak dikembangkan mengingat

keandalannya dalam menyelesaikan suatu masalah yang kompleks. Pengendalian dan perancangan sistem multi robot yang merupakan bagian vital dari keseluruhan penelitian multi-robot kooperatif pun tak luput gencar dikembangkan.

Penggunaan multi-robot kooperatif dalam transportasi barang besar menggunakan berbagai variasi bentuk robot platform *mobile*. Oleh karena itu, tak jarang ditemukan multi-*mobile robot* yang bekerja dalam menyelesaikan suatu permasalahan sistem transportasi yang kompleks. Hal tersebut mendorong tingginya tingkat ketertarikan masyarakat mengenai topik koordinasi antar robot dan sistem multi-robot mobil selama beberapa tahun ini terus bertambah dikarenakan sistem multi-robot mobil lebih efisien daripada penggunaan robot tunggal.

Dalam mencapai kemampuan untuk dapat bekerja secara kooperatif, komunikasi antar robot merupakan aspek yang sangat penting dalam sistem multi-robot. Oleh karena itu, sifat komunikatif antar robot sangat diperlukan. Sifat komunikatif dalam konteks ini merupakan kemampuan robot untuk saling bertukar informasi keadaan robot tersebut dalam saat tertentu ke robot/ objek lainnya. Sehingga pada skripsi ini akan dirancang mobile robot yang dapat saling berkomunikasi mengenai posisi nya satu sama lain.

Untuk menunjang data/informasi yang dikomunikasikan valid, maka robot perlu diberikan algoritma pengendalian yang sesuai, baik secara keseluruhan maupun untuk masing-masing unitnya. Pengendalian unit yang baik akan menunjang tercacpainya sistem yang semakin dapat mencapai tujuan pengimplementasiannya. Oleh karena itu, pada skripsi ini akan dibahas mengenai pengujian setiap unit beserta pengendalian aktuator masing-masing unitnya sebelum pada akhirnya diimplementasikan ke dalam sistem *mobile robot* komunikatif.

1.2 Perumusan Masalah

Perumusan masalah dalam penulisan skripsi ini adalah sebagai berikut:

- Untuk mendapatkan hasil pengendalian *mobile robot* yang baik, perlu ditunjang oleh pengendalian aktuator, yaitu motor, yang mampu

menghasilkan respon sesuai spesifikasi sehingga perlu dilakukan perancangan kendali motor

- Sistem multi-robot mobil komunikatif terdiri dari beberapa *mobile robot* sehingga perlu dilakukan pengujian kemampuan masing-masing mobile robot untuk mengikuti trayektori yang diberikan sebelum kemudian dikomunikasikan.
- Dalam melakukan suatu pergerakan kooperatif, dibutuhkan komunikasi yang baik antr robotnya agar setiap robot dapat mempertahankan pergerakan yang sesuai dengan pergerakan sistem

1.3 Tujuan Penelitian

Tujuan penelitian pada skripsi ini adalah :

- Mengembangkan sistem pengendalian dua *mobile robot* dengan menggunakan odometri dan dengan mempertimbangkan karakteristik motor untuk mengikuti trayektori linier dan sinusoidal.
- Mengembangkan komunikasi *internet socket* sebagai sarana pertukaran informasi posisi antar *mobile robot* untuk diimplementasikan ke robot kooperatif.

1.4 Batasan Masalah

Penulis membatasi masalah pada perancangan ini kepada sistem yang menggunakan mobile robot non-holonomik dengan dua roda independen. Pada skripsi ini akan dibahas pengendalian *mobile robot,* yaitu mencakup pengendalian kecepatan motor *mobile robot* dengan menggunakan referensi kecepatan motor serta pengendalian posisi dan orientasi *mobile robot*. Posisi *mobile robot* diestimasi dengan menggunakan *dead reckoning* berbasis odometri. Komunikasi antar unit robot dilakukan dengan menggunakan teknologi Wi-Fi melalui *UDP socket*.

1.5 Metodologi Penelitian

Dalam penulisan skripsi ini, metodologi penulisan yang dilakukan adalah sebagai berikut :

1. Studi literatur dengan membaca hasil penelitian yang telah dibuat sebelumnya serta beberapa jurnal referensi yang membahas mengenai pengendalian.
2. Perancangan dan perakitan sistem *mobile robot* untuk pengujian
3. Inisiasi pengujian *mobile robot* seperti pengecekan putaran motor dan encoder.
4. Pengujian terhadap parameter motor DC, konstanta pengendali motor DC, gain filter *low-pass*, serta konstanta pengendali posisi dan orientasi *mobile robot*.
5. Menganalisa data hasil pengujian dan menentukan hasil terbaik serta langkah perbaikan apabila perlu.
6. Mereview ulang dan menarik kesimpulan

1.6 Sistematika Penulisan

Skripsi ini ditulis dalam 5 (lima) bab yang terdiri dari dengan isi yang berbeda untuk masing-masing bab. Berikut merupakan penjelasan untuk masing-masing babnya. Pendahuluan yang terdiri dari latar belakang, tujuan penulisan, pembatasan masalah, metodologi penulisan, serta sistematika penulisan. Kemudian dilanjutkan dengan dasar teori yang berisi teori mengenai pengantar *mobile robot*, model *mobile robot* dan kinematiknya, dasar pengendalian motor DC, sistem robot kooperatif, serta komunikasi antar robot.

Kemudian dilanjutkan dengan perancangan sistem *mobile robot* komunikatif. Bagian ini terdiri dari perancangan pengujian motor beserta dan algoritma trayektori serta perancangan pengujian pengendalian *mobile robot* maupun sistem robot komunikatif. Dilanjutkan dengan implementasi dan hasil pengujian yang berisikan hasil pengujian yang telah dilakukan beserta analisa untuk masing-masing pengujian. Penulisan diakhiri dengan penutup yang berisi kesimpulan dalam rancang bangun *mobile robot* komunikatif.

BAB 2

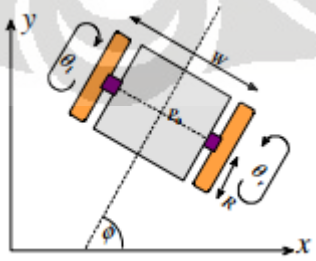
DASAR TEORI

2.1 Model Mobile Robot Non-Holonomik

Mobile robot adalah robot yang dapat bergerak secara otonomi, memiliki navigasi, dan pergerakannya yang tidak tetap tergantung dari ruang kerjanya masing-masing. *Mobile robot* non-holonomik merupakan *mobile robot* yang mempunyai pergerakan dengan batasan holonomic, yaitu pergerakan *mobile robot* dimana dalam pergerakkannya *mobile robot* mempunyai keterbatasan yang tidak dapat langsung bergerak kearah normal/orthogonal roda-rodanya. Oleh karena itu, untuk sistem non-holonomik *mobile robot* perlu melakukan perubahan orientasinya terlebih dahulu.

Selanjutnya akan dibahas mengenai permodelan *mobile robot* non-holonomik dengan dua roda independen. Sebuah *mobile robot* dapat dimodelkan secara kinematika dan juga terhadap elektromekaniknya. Pada permodelan kinematika, *mobile robot* akan dimodelkan berdasarkan pergerakan *mobile robot* itu sendiri. Permodelan secara elektromekanik dapat direpresentasikan dengan memodelkan aktuator yang digunakan oleh *mobile robot* sehingga dapat ditemukan pada subbab berikutnya.

Berikut merupakan diagram skematik yang menggambarkan representasi posisi *mobile robot* terhadap diagram kartesian sebuah *mobile robot* dengan x dan y sebagai posisi kartesiannya dan θ sebagai sudut orientasi *mobile robot* terhadap sumbu referensi x.



Gambar 2.1 Mobile Robot Non-Holonomik Dua Roda

Berdasarkan gambar diatas dapat diturunkan rumus kinematika dari

perputaran kedua roda sebagai berikut[2]

$$\dot{x}_0 \cos\phi + \dot{y}_0 \sin\phi + \frac{W}{2}\dot{\phi} = R\dot{\theta}_r \quad (2.1)$$

$$\dot{x}_0 \cos\phi + \dot{y}_0 \sin\phi - \frac{W}{2}\dot{\phi} = R\dot{\theta}_l \quad (2.2)$$

jika dilakukan eliminasi maka akan didapatkan kedua persamaan sebagai berikut (dengan penjumlahan dan pengurangan)

$$\dot{x}_0 \cos\phi + \dot{y}_0 \sin\phi = \frac{R}{2}(\dot{\theta}_r + \dot{\theta}_l) \quad (2.3)$$

$$W\dot{\phi} = R(\dot{\theta}_r - \dot{\theta}_l) \quad (2.4)$$

$$\phi = \frac{R}{W}(\theta_r - \theta_l) \quad (2.5)$$

Kemudian hubungan antara kecepatan linier dan proyeksinya terhadap sumbu kartesian x dan y didapatkan sebagai berikut.

$$v = \dot{x}_0 \cos\phi + \dot{y}_0 \sin\phi \quad (2.6)$$

sehingga ketika dibandingkan dengan hasil eliminasi ϕ pada kinematika mobile robot saat kedua roda berputar dapat diketahui bahwa

$$v = \frac{R}{2}(\dot{\theta}_r + \dot{\theta}_l) \quad (2.7)$$

Setelah mendapatkan nilai hubungan kecepatan linier (v) dengan kecepatan putar masing-masing roda *mobile robot* saat bergerak, maka dapat dicari nilai proyeksi kecepatan linier terhadap sumbu kartesian x dan y terhadap kecepatan putar kedua roda dengan memasukkan nilai v yang diperoleh ke dalam rumus proyeksi kecepatan linier terhadap sumbu x dan y di bawah.

$$\dot{x}_0 = v\cos\phi \text{ dan } \dot{y}_0 = v\sin\phi \quad (2.8)$$

sehingga persamaan menjadi

$$\dot{x}_0 = \frac{R}{2}\cos\phi\ (\dot{\theta}_r + \dot{\theta}_l) \quad \text{dan} \quad \dot{y}_0 = \frac{R}{2}\sin\phi\ (\dot{\theta}_r + \dot{\theta}_l) \quad (2.9)$$

Berdasarkan penurunan rumus di atas dapat dimodelkan kinematika *mobile robot* non-holonomik dengan representasi state space.

$$\begin{bmatrix} \dot{x}_0 \\ \dot{y}_0 \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} \frac{R}{2}\cos\phi & \frac{R}{2}\cos\phi \\ \frac{R}{2}\sin\phi & \frac{R}{2}\sin\phi \\ \frac{R}{W} & -\frac{R}{W} \end{bmatrix} \begin{bmatrix} \dot{\theta}_r \\ \dot{\theta}_l \end{bmatrix} \quad (2.10)$$

Matriks yang menghubungkan antara turunan posisi kartesian dan turunan posisi sudut kedua motor disebut dengan matriks Jacobian. Sedangkan yang menghubungkan turunan posisi sudut kedua motor dengan turunan posisi kartesian merupakan matriks Invers Jacobian. Berikut merupakan matriks jacobian dan invers jacobian.

$$Jacobian = \begin{bmatrix} \frac{R}{2}cos\phi & \frac{R}{2}cos\phi \\ \frac{R}{2}sin\phi & \frac{R}{2}sin\,\phi \\ \frac{R}{W} & -\frac{R}{W} \end{bmatrix} \quad (2.11)$$

Dikarenakan matriks jacobian yang didapatkan tidak berbentuk matriks persegi, maka untuk mendapatkan matriks invers jacobian dilakukan dengan menggnakan *pseudo-invers* matriks Jacobian dengan rumus sebagai berikut dengan J merupakan matriks jacobian.

$$J^+ = (J^T J)^{-1} J^T \quad (2.12)$$

Sehingga didapatkan nilai matriks invers jacobian (J^+) sebagai berikut

$$J^+ = \frac{1}{R}\begin{bmatrix} \cos\phi & \sin\phi & \frac{W}{2} \\ \cos\phi & \sin\phi & -\frac{W}{2} \end{bmatrix} \quad (2.13)$$

Matriks jacobian dan invers jacobian digunakan untuk merepresentasikan hubungan kinematika dari kecepatan sudut *joint mobile robot* dengan kecepatan proyeksi kartesiannya dan sebaliknya. Jacobian digunakan untuk mentransformasikan kecepatan sudut kedua roda/*joint mobile robot* dengan

kecepatan proyeksi kartesiannya.

Secara sederhana, matriks jacobian dan invers jacobian merupakan sarana untuk mengubah perhitungan yang berdasarkan *joint* ke perhitungan yang berdasarkan koordinat kartesian global. Nilai hasil matriks yang sudah ditransformasi kemudian digunakan untuk perhitungan referensi motor maupun perhitungan referensi kartesian.

Pada skripsi ini estimasi posisi *mobile robot* dilakukan dengan algoritma dead reckoning berbasis odometri. Dead reckoning sendiri merupakan prosedur matematika sederhana untuk menentukan posisi saat ini dari sebuah kendaraan/*mobile robot* dengan mempertimbangkan posisi sebelumnya melalui informasi orientasi dan kecepatan yang diberikan dalam jangkauan waktu tertentu[3]. Odometri sendiri merupakan bentuk *dead reckoning* sederhana dimana perpindahan *mobile robot* sepanjang haluan diturunkan dari odometer yang terpasang. Odometer yang digunakan merupakan sensor yang mengestimasi pergerakan relatif untuk mendapatkan posisi absolut dari *mobile robot*[4].

Berikut merupakan rumus algoritma *dead reckoning* yang digunakan pada skripsi ini. [2]

$$v_k = v_{x\,(k)} \cos \phi_k + v_{y\,(k)}\, sin\phi_k \tag{2.14}$$

Persamaan tersebut merupakan persamaan perantara untuk menghitung kecepatan linier dari sistem mobile robot dengan menggunakan proyeksi kecepatan sesaat yang diproyeksikan di sumbu kartesian. Nilai kecepatan sesaat pada sumbu didapatkan melalui hasil perkalian antara matriks jacobian dengan kecepatan masing-masing roda yang dibaca oleh sensor.

$$\phi_k = \frac{R}{W}\left(\theta_{r\,(k)} - \theta_{l\,(k)}\right) + \phi_{init} \tag{2.15}$$

Kemudian melalui hasi pembacaan posisi sudut roda kanan dan roda kiri beserta orientasi awal dapat dicari nilai orientasi saat ini sehingga melalui perubahan orientasi terhadap orientasi sebelumnya dapat diketahui perubahan posisi x dan posisi y yang dapat dihitung dengan rumus berikut.

$$x_k = x_{k-1} + v_k \cos\left(\frac{\phi_k + \phi_{k-1}}{2}\right).\Delta t \quad (2.16)$$

$$y_k = y_{k-1} + v_k \sin\left(\frac{\phi_k + \phi_{k-1}}{2}\right).\Delta t \quad (2.17)$$

Berdasarkan persamaan di atas dapat dilihat bahwa posisi *mobile robot* saat ini dapat diestimasi dengan menggunakan nilai sebelumnya dan kecepatan dan orientasi *mobile robot* yang dapat diperoleh melalui sensor yang terpasang pada *mobile robot*. Selanjutnya akan dibahas mengenai sistem kendali *mobile robot* secara keseluruhan, meliputi unit pengendali, aktuator, dan sensor.

2.2 Sistem Kendali Mobile Robot

Komponen perangkat keras sistem pengendali tersusun dari divais baik mekanik maupun elektris yang mempunyai fungsionalitas sebagai aktuator, sensor, dan pengendali.[5]. Pada sistem *mobile robot* yang digunakan ketiga komponen perangkat keras tersebut adalah motor DC, enkoder, dan mikrokontroler secara berurutan.

Ketiga komponen tersebut sangat mempunyai peranan penting sehingga perlu dibahas secara khusus setiap komponennya. Oleh karena itu, pada bagian ini akan dibahas masing-masing komponen dengan diawali dari pembahasan komponen pengendali dan diikuti dengan penjelasan mengenai komponen sensor dan aktuator secara berurutan.

2.2.1 Unit Pengendali

Pengendalian *mobile robot* ini dilakukan dengan menggunakan unit pengendali mikrokontroler Arduino sehingga pengendali pada *mobile robot* berupa pengendali diskrit. Pada pengendali diskrit, kecepatan sampling time sangat berpengaruh terhadap keseluruhan hasil pengendalian. Oleh karena itu, dibutuhkan sistem pengendalian secara *real time*.

Sistem *real-time* adalah sistem perangkat lunak dimana fungsi dari sistem, tergantung pada hasil yang dihasilkan oleh sistem waktu[6]. Secara keseluruhan sistem *real time* yang dimaksud adalah sistem yang dapat menyelesaikan semua tugas dan menghasilkan respons yang sesuai dalam rentang waktu yang sudah

ditentukan.

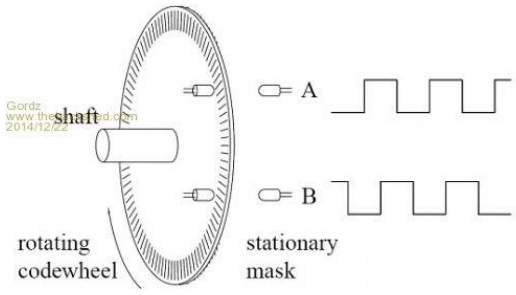
Pada skripsi ini untuk menjaga pengendalian yang dilakukan bersifat *real time* maka dilakukan penggunaan *timer interrupt* untuk waktu sampling yang diinginkan dengan mempertimbangkan kesesuaian nilai waktu cuplik. *Timer interrupt* sendiri merupakan fungsi untuk memanggil suatu rutin dalam jangka waktu yang spesifik[7] .

Waktu yang dapat ditunjang oleh *timer interrupt* bergantung terhadap banyaknya bit timer yang digunakan, seperti untuk Arduino timer1 merupakan timer 16-bit dan timer2 merupakan timer 8-bit. Dalam pelaksanaan interrupt perlu diperhatikan bahwa dalam mengeksekusi rutin yang terlalu kompleks frekuensi interrupt yang dilakukan harus disesuaikan agar tidak terjadi kodisi '*lock up'* ketika CPU mengakses fungsi loop utama. Selanjutnya akan dibahas mengenai komponen perangkat keras berikutnya yaitu sensor.

2.2.2 Sensor

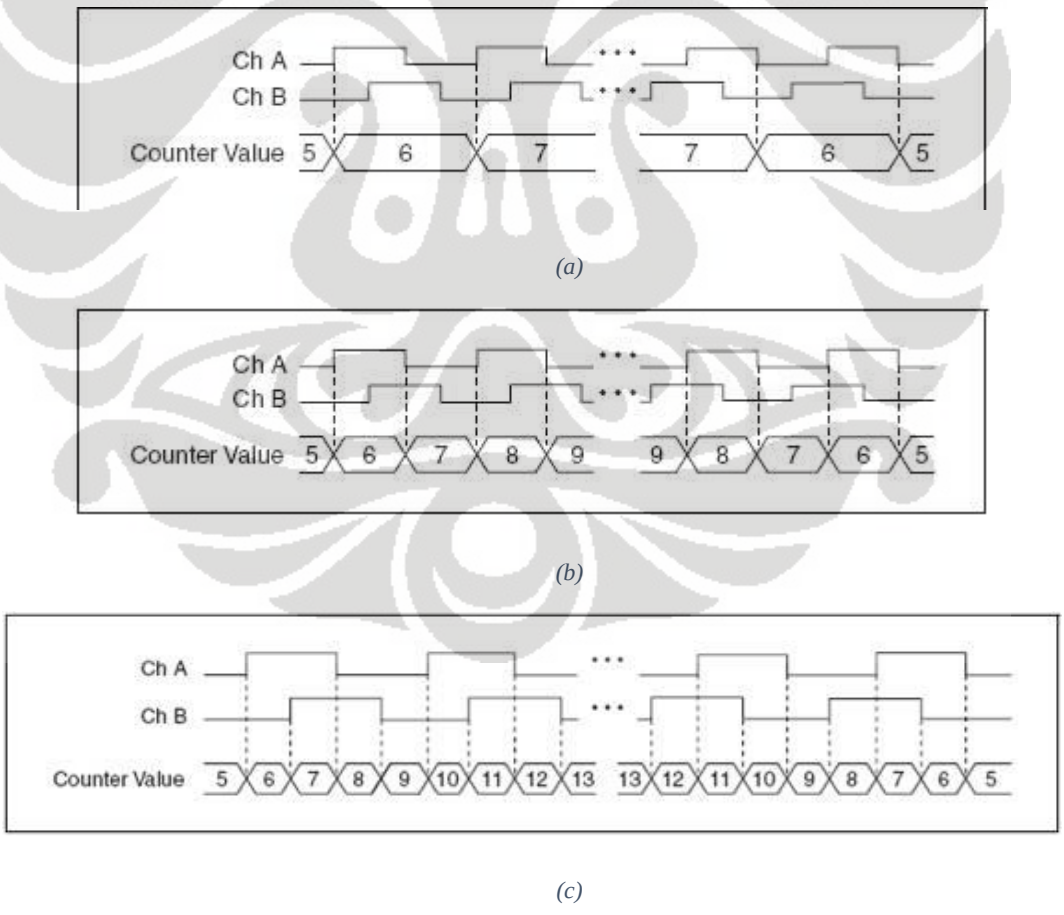
Mayoritas mobile robot mengandalkan pergerakan, yaitu pergerakan roda dan jalur. Pemahaman dasar mengenai sensor yang secara akurat memberikan informasi mengenai posisi angular dan kecepatan motor merupakan prasyarat yang penting dalam *dead reckoning* berbasis odometri. Sensor yang paling sering digunakan untuk mobile robot merupakan jenis *rotary encoder* dengan *incremental* ataupun *absolute optical* encoder.

Pada skripsi ini, akan digunakan sensor *quadrature encoder* dalam pembacaan posisi putaran rodanya. *Quadrature encoder* sendiri merupakan salah satu jenis dari *incremental encoder* yang mempunyai dua track kode dengan sektor yang diposisikan orthogonal terhadap fasa. Kedua keluaran channel, yaitu channel A dan channel B akan mengindikasikan posisi dan arah perputaran dari roda. Berikut merupakan ilustrasi dari quadrature encoder.



Gambar 2.2 Ilustrasi Keluaran Quadrature Encoder

Dari Gambar 2.2 apabila channel A diketahui mendahului channel B, maka dapat dianalisa bahwa arah perputaran roda merupakan searah jarum jam. Dengan memonitoring kedua pulsanya dan fase relatif sinyal A dan sinyal B, dapat diketahui posisi sudut dan arah rotasi. Melalui perhitungan pulsa pergerakan yang didapatkan dapat dikonversikan menjadi posisi sudut dari roda. Dalam perhitungannya, encoder dapat dibaca dengan menggunakan 1x pembacaan, 2x pembacaan, dan 4x pembacaan seperti ilustrasi berikut.[8]



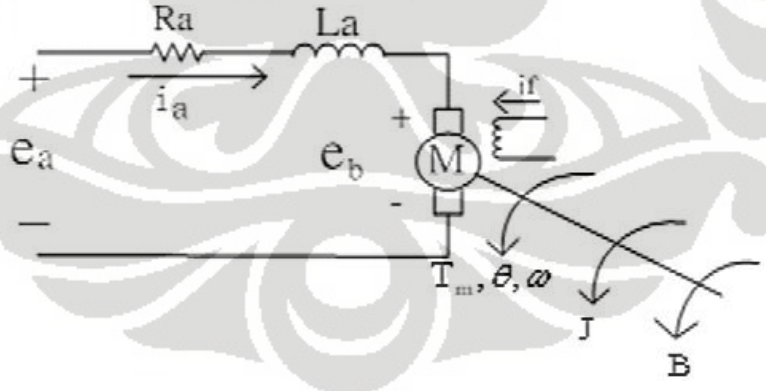
Gambar 2.3 Ilustrasi Pembacaan Quadrature Encoder: (a) 1x Pembacaan; (b) 2x Pembacaan; (c) 4x Pembacaan

Banyaknya pembacaan yang digunakan akan mempengaruhi tingkat ketelitian/akurasi pembacaan yang dilakukan dimana pembacaan semakin besar maka akurasi akan semakin baik. Perhitungan banyak pulsa dapat dilakukan dengan menggunakan algoritma pada pemrograman ataupun menggunakan counter. Pada skripsi ini, perhitungan banyak pulsa dilakukan dengan menggunakan *counter quadrature*, yaitu HCTL-2032.

HCTL-2032 merupakan IC CMOS yang dapat digunakan untuk *quadrature decoder, counter,* dan *bus interface*.[9] HCTL-2032 didesain untuk menghasilkan performa sistem *motion control closed loop* yang baik dengan *counter up/down* biner. HCTL-2032 yang digunakan diintegrasikan pada sisi elektris yang terhubung dengan Arduino. Setelah membahas komponen sensor, maka selanjutnya akan dibahas mengenai komponen aktuator yang akan digunakan.

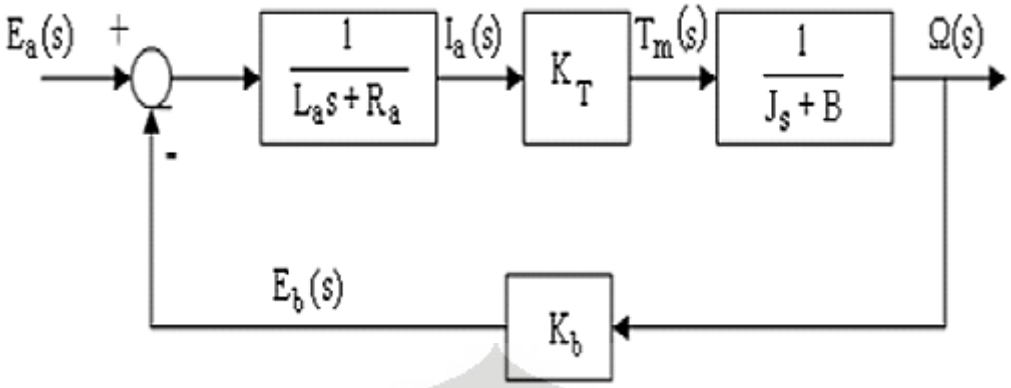
2.2.3 Aktuator

Dalam pengoperasiannya, *mobile robot* bergerak berdasarkan aktuatornya, yaitu motor DC. Oleh sebab itu, akan dibahas secara singkat mengenai model motor DC dan dasar pengendalian motor DC yang mencakup pemberian PWM dan rangkaian H-Bridge. Berikut merupakan ilustrasi rangkaian dalam elektromekanik dari motor DC.



Gambar 2.4 Ilustrasi Rangkaian Elektromekanik Motor DC

Berdasarkan rangkaian tersebut dapat dibuat blok diagram hubungan elektromekanik untuk masing-masing motor DC sebagai berikut.

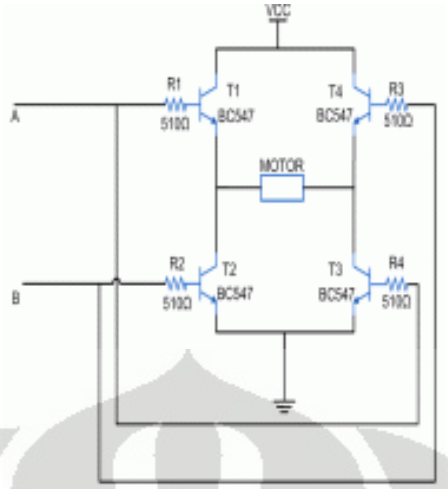


Gambar 2.5 Blok Diagram Elektromekanik Rangkaian Motor DC

Pada blok diagram diatas, model mekanik motor yang digunakan merupakan model mekanik motor yang paling sederhana, yaitu hanya mempunyai inersia tanpa konstanta damper.

Berdasarkan block diagram di atas dapat dilihat bahwa dengan mengatur tegangan maupun arus armature pada rangkaian elektris motor, maka kita dapat mempengaruhi putaran pada motor. Oleh sebab itu, pengaturan akan kecepatan dan posisi motor dapat dilakukan dengan memberikan sinyal pengendali berupa tegangan armature dengan menyesuaikan fungsi alih sesuai dengan motor yang digunakan.

Pada aplikasi riil, arus keluaran dari mikrokontroler yang merupakan pusat komputasi dan pengendalian mobile robot relatif lebih kecil dari rating arus untuk menjalankan motor sehingga perlu diteruskan ke driver motor. Driver motor yang umum digunakan adalah rangkaian H-Bridge yang terintegrasi ke dalam IC L298 atau L293. Berikut merupakan ilustrasi gambar rangkaian H-Bridge.



Gambar 2.6 Rangkaian H-Bridge

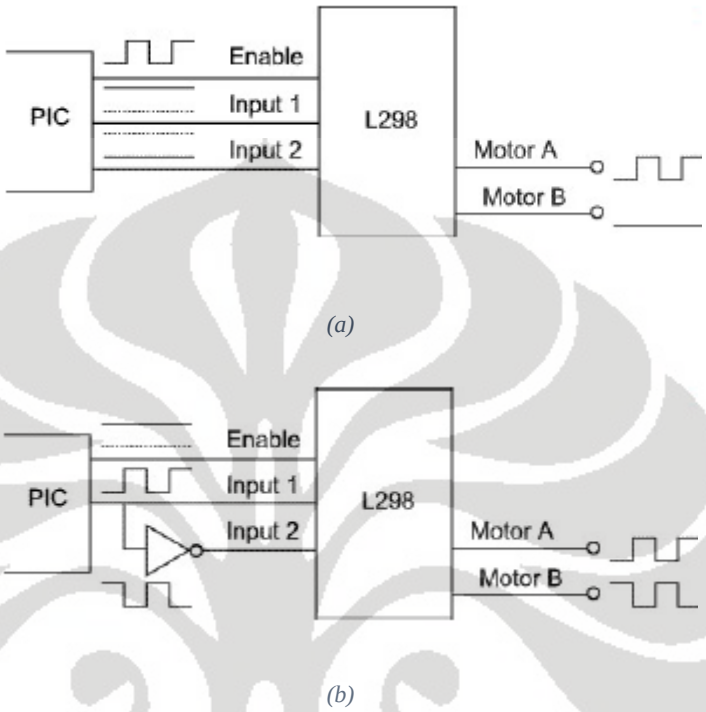
Rangkaian H-Bridge ini berguna sebagai perantara motor dengan mikrokontroler dalam pengendalian besar arus yang dilewatkan ke dalam motor DC. Rangkaian H-bridge juga berguna dalam pengaturan arah putaran motor DC berdasarkan input yang diberikan.

Prinsip kerja H-Bridge adalah dengan melakukan penentuan arah rangkaian dengan *swiching.* Apabila input A diberikan tegangan yang lebih besar dari input B, maka arus akan mengalir dari Vcc melewati T1, motor, dan T3 sampai ke ground. Sebaliknya, apabila input B diberikan tegangan yang lebih besar dibandinngkan input A, maka arus akan mengalir dari Vcc melewati T4, motor, dan T2 sampai ke ground. Perbedaan kutub positif dan kutub negatif dari motor seiring dengan perubahan pemberian input/ aliran arus tersebutlah yang akan merubah arah perputaran dari motor yang digunakan.

Kemudian, dalam pengaturan posisi dan kecepatan motor DC dapat dilakukan dengan memberikan variasi tegangan seperti yang ditunjukkan pada blok diagram motor DC pada subbab sebelumnya. Pemberian variasi tegangan kepada motor berupa variasi *duty cycle* pada *Pulse Width Modulation* (PWM). Sinyal PWM dibentuk melalui proses switching transistor dengan lama switching untuk setiap transistornya.

Pemberian PWM ke motor DC dilewatkan melalui motor *driver* yang berupa H-Bridge. Terdapat dua metode pemberian PWM ke motor melalui *driver*, baik

dengan memberikan PWM pada *enable* IC L298, maupun dengan menggunakan PWM pada salah satu input IC L298 seperti yang direpresentasikan pada gambar di bawah.[2]



Gambar 2.7 Kedua Jenis Pemberian Sinyal PWM ke Driver Motor (a) Metode 1 : dengan Memberikan PWM pada Enable dan; (b) Metode 2: dengan Membeikan PWM ke Salah Satu Input dan Memberikan Invers PWM ke Input Lainnya.

Pada metode pertama, sinyal PWM diberikan pada *enable* dan kedua input (input 1 dan input 2) digunakan untuk mengendalikan arah putaran motor saja sehingga diberikkan sinyal "HIGH" dan "LOW" secara berlawanan sesuai dengan arah putaran motor yang diinginkan. Pengendalian motor dengan metode ini dinilai mempunyai performa yang tidak sebaik metode kedua, tetapi lebih hemat energi dibandingkan metode kedua.

Pada metode kedua, *enable* dibuat untuk selalu menyala dan PWM dari mikrokontroler dimasukkan ke salah satu input, yaitu input 1. Input 2 merupakan negasi dari input 1, sehingga *duty cycle* PWM yang masuk ke dalam input 2 merupakan lama waktu *off* dari sinyal PWM yang diberikan pada input 1. Hal ini mengakibatkan kedua jalur motor akan terbuka sesuai dengan besarnya tegangan yang diberikan untuk masing-masing transistor dan motor akan berputar dengan *duty cycle* yang merupakan selisih kedua tegangan/ *duty cycle* yang diberikan.

Berikut merupakan rumus untuk konversi perhitungan pemberian *duty cycle* PWM pada input 1 dengan metode 2 menjadi seperti metode 1 jika diketahui *duty cycle* yang ingin diberikan kepada.

$$IN1 = \frac{duty\ cycle + 100\%}{2} \tag{2.18}$$

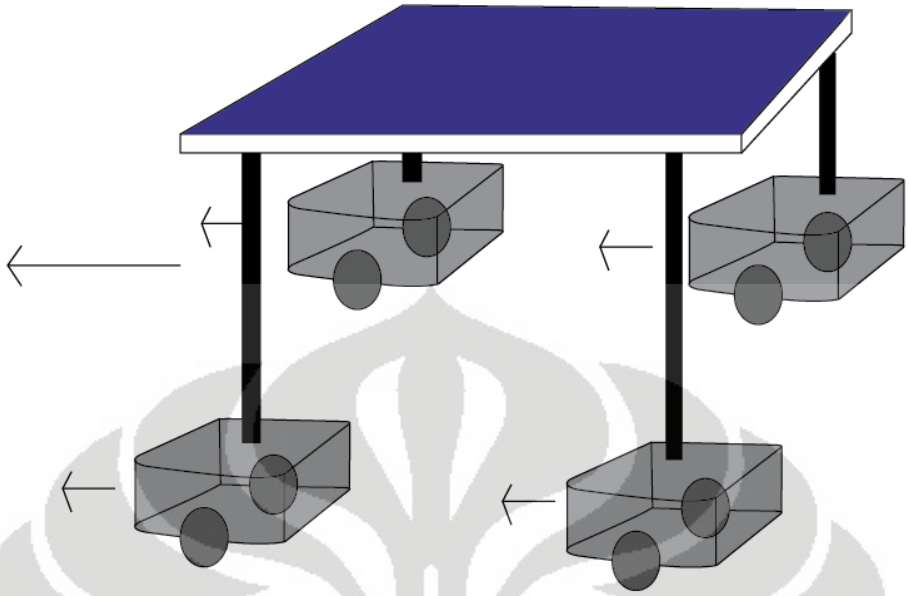
dan input 2 merupakan negasi dari *duty cycle* yang didapatkan pada input 1 sehingga motor akan berputar mengikuti arah tegangan/*duty cycle* salah satu input yang lebih besar dengan *duty cycle* yang merupakan selisih kedua input.

Perbedaan keduanya selain metode pemberian PWMnya adalah aplikasi dan daya yang digunakan. Pada metode pertama akan didapatkan keluaran dengan hasil yang sesuai untuk aplikasi yang membutuhkan pergerakan ke satu arah seperti motor yang hanya membutuhkan pergerakan maju. Sedangkan metode kedua akan menghasilkan keluaran yang cukup sesuai untuk aplikasi yang membutuhkan pergantian dua arah, yaitu maju dan mundur, dengan cukup sering.

Daya yang digunakan untuk kedua metode pun berbeda dimana penggunaan daya untuk metode pertama lebih sedikit dibandingkan metode kedua seperti yang telah disinggung sebelumnya. Hal ini disebabkan karena untuk metode pertama hanya membutuhkan satu PWM pada enable nya dan input "HIGH" dan "LOW" untuk kedua inputnya, sedangkan metode kedua membutuhkan *enable* yang selalu "HIGH" dengan PWM untuk kedua inputnya. Oleh karena itu, dapat dilihat pemakaian daya metode kedua lebih besar dibandingkan metode pertama.

2.3 Robot Kooperatif

Robot kooperatif didefinisikan sebagai sekelompok robot yang bekerja sebagai sebuah tim untuk mencapai tujuan bersama. Robot kooperatif dinilai meningkatkan efisiensi dan memungkinkan penyelesaian tugas yang cukup kompleks yang tidak dapat dikerjakan sendiri. Selain itu, penggunaan beberapa robot kooperatif untuk melakukan tugas yang kompleks dinilai lebih sederhana dan mura dibandingkan menggunakan sebuah robot yang kompleks dikarenakan fleksibel. Berikut merupakan ilustrasi sederhana untuk tugas robot kooperatif



Gambar 2.8 Ilustrasi Sistem Multi-Robot Kooperatif

Berdasarkan ilustrasi di atas dapat dilihat bahwa sekumpulan mobile robot yang sama sedang bekerja secara kooperatif untuk mengangkut meja ke suatu tujuan bersama. Berdasarkan keberagaman robot dalam sistemnya, multi-robot kooperatif dapat diklasifikasikan menjadi homogen dan heterogen. Multi-robot dikatakan homogen jika anggota sistem tersebut bersifat sejenis atau sama dan heterogen jika anggotanya merupakan robot yang berbeda. Sistem di atas dapat diklasifikasikan sebagai sistem multi-robot kooperatif homogen dikarenakan menggunakan tipe robot yang sejenis dalam aplikasinya, yaitu mobile robot identik.

Robot yang berkerja dalam sebuah kooperasi dengan robot yang lain memiliki beberapa karakteristik antara lain mengetahui robot lain yang di kelompoknya sehingga untuk dapat bekerja secara kooperatif dibutuhkan kesamaan tujuan [10]. Beberapa penilitian mengklasifikasikan sistem multi-robot kooperatif menjadi dua kategori, yaitu aktif dan pasif. Pada sistem aktif, masing-masing robot berkomunikasi satu sama lain dalam rangka pertukaran informasi. Robot-robot tersebut dapat mengatur pekerjaan masing-masing dan mampu mengambil keputusan. Sedangkan pada sistem pasif, tidak ada hubungan

komunikasi antar robot yang membuat sistem menjadi mudah didesain dan robust. Pada sistem ini, robot tidak berbagi informasi dan membuat keputusan bersama. Skripsi ini akan membahas mengenai sistem kooperatif aktif sehingga diperlukan komunikasi antar robotnya yang akan dibahas pada subbab berikutnya.

2.4 Komunikasi Antar Robot

Agar robot dapat mengetahui informasi robot lainnya dalam sistem kooperatif aktif diperlukan komunikasi antar robot. Secara umum, berdasarkan media yang digunakan, komunikasi dibagi menjadi dua kategori, yaitu komunikasi kabel (*wired communication*) dan komunikasi nirkabel (*wireless communication*). Komunikasi dengan kabel dinilai cukup sederhana dan lebih murah untuk aplikasi yang mempunyai jarak dekat dan komponen tetap. Akan tetapi, komunikasi dengan menggunakan kabel dinilai kurang sesuai dalam aplikasi yang membutuhkan mobilitas besar seperti *mobile robot*. Oleh karena itu, pada skripsi ini digunakan komunikasi nirkabel dalam pertukaran informasi antar robotnya.

Sistem komunikasi nirkabel sendiri memiliki beberapa jenis teknologi yang dapat digunakan untuk aplikasi *mobile robot*, antara lain infra-merah(*infrared/IR*), *bluetooth*, *ZigBee*, dan *Wi-Fi*. Setiap teknologi nirkabel memiliki keunggulannya masing-masing akan tetapi pada skripsi ini digunakan teknologi *Wi-Fi* dikarenakan mobilitasnya yang tinggi dan mampu mengirim data dengan waktu yang relatif cepat. Berikut akan dibahas

2.4.1 Protokol Komunikasi

Selain teknologi atau media yang digunakan untuk komunikasi, perlu juga diperhatikan ketentuan dan cara bagaimana data atau informasi yang ingin dipertukarkan yang disebut sebagai protokol komunikasi. Protokol komunikasi secara keseluruhan dibagi menjadi dua jenis secara garis besar, yaitu protokol komunikasi UDP (*User Datagram Protocol*) dan TCP (*Transmisson Control Protocol*). Kedua protokol komunikasi berkerja di atas IP (*Internet Protocol)* dan berada pada layer *Transport* pada OSI. Berikut akan dibahas mengenai layer *transport* sebelum masuk ke pembahasan mengenai UDP dan TCP.

Layer Transport merupakan layer yang berfungsi mempersiapkan data aplikasi dengan cara melakukan segmentasi data untuk di*transport* melalui *network,* dan memproses data *network* dengan cara menyusun kembali potongan-potongan data tersebut untuk digunakan oleh application layer. Transport layer juga mengidentifikasi setiap *application* yang antara *source* dan *destination,* setiap *application* diberikan nomor port unik untuk identifikasi ini. Pada setiap potongan data, terdapat nomor port untuk mengidentifikasikan protokol dan aplikasi data tersebut.

Layer Transport memiliki beberapa fungsi utama untuk, membagi data yang diterima dari sebuah aplikasi menjadi beberapa bagian dan menambahkan kop (header) berisi berbagai informasi untuk identifikasi dan manajemen setiap segment. Kemudian layer transport menyusun kembali segmen-segmen data berdasarkan informasi pada header dan pada akhirnya menyalurkan data yang telah disusun kembali ke aplikasi yang benar.

Dalam transport layer kita mengetahui bahwa setiap protokol memiliki fungsi yang digunakan membentuk berbagai bentuk komunikasi, Untuk mengidentifikasi komunikasi apa yang sedang berlangsung maka dibutuhkan sebuah *identifier* agar koneksi bisa terhubung tanpa ada masalah. Dimana *identifier* tersebut merupakan *Port Numbers.*

Port numbers ditentukan dengan berbagai cara tergantung dari pesan tersebut merupakan sebuah permintaan (*request*) atau jawaban (*response*). Ketika *server* memproses menggunakan *static port numbers* yang telah ditentukan, maka *client* secara dinamis menentukan sebuah *port number* untuk setiap komunikasi.

Pada Protokol Layer Transport, khususnya pada TCP/IP, ada dua protokol yang paling sering digunakan, yakni User Datagram Protocol (UDP) dan Transport Control Protocol (TCP). Masing-masing dari protocol ini dapat melakukan komunikasi dengan banyak aplikasi dan yang menjadi perbedaannya ialah mereka di desain untuk fungsi spesifik yang berbeda untuk setiap pengimplementasiannya. Keduanya memiliki aplikasi yang berbeda dimana TCP lebih *session-oriented* dan UDP lebih ke *data-oriented.*

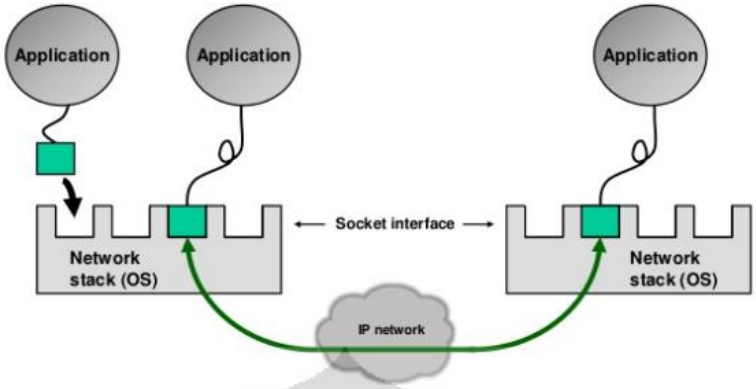
Secara umum, protokol TCP lebih banyak digunakan karena dinilai handal dalam mengatur data dalam jumlah banyak dengan memecah data tersebut menjadi paket individual dan mengecek serta mengurutkan paket dalam urutan yang benar. Akan tetapi, akibat kehandalan tersebut, dibutuhkan data tambahan yang dikirim dan juga latensi yang lebih besar.

Berbeda dengan TCP, komunikasi UDP hanya mengirimkan paket sehingga bandwidth data tambahan dan latensi lebih kecil dibandingkan dengan TCP. Akan tetapi, komunikasi dengan menggunakan UDP kurang handal dikarenakan paket yang dikirim dapat hilang atau diterima dengan urutan yang berantakan.

UDP merupakan protokol komunikasi yang ideal untuk aplikasi yang membutuhkan latensi rendah dan tidak terlalu terpengaruh oleh beberapa data yang hilang. Pada aplikasi ini dibutuhkan pertukaran data dengan sangat cepat agar data yang didapatkan mampu mengejar waktu cuplik sistem yang digunakan. Pada skripsi ini digunakan protokol komunikasi UDP dengan menggunakan model *socket client-server*.

Socket adalah sebuah file (*file descriptor*) yang digunakan sebagai sebuah titik untuk berkomunikasi dengan proses lain melalui jaringan (atau dalam suatu sistem operasi yang sama, namun berbeda proses) [11]. Penggunaan *socket* memperbolehkan komunikasi dua arah atau *bidirectional* antara *client* dengan *server* dengan mengurangi *traffic* yang redundan.

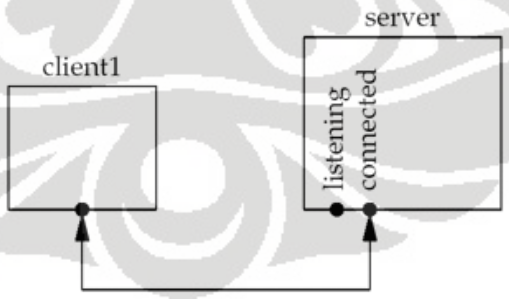
Fitur utama dari *socket* adalah terakomodasinya kebutuhan komunikasi dua arah dengan *server* yang tidak perlu membuka beberapa koneksi lapisan di atasnya dan juga kecepatan transfer data yang cepat[13]. Oleh karena itu, pada skripsi ini komunikasi didesain dengan menggunakan socket sebagai media pertukaran data antar robot yang digunakan. Berikut merupakan gambar ilustrasi mengenai pengertian socket dalam layer komunikasi.



Gambar 2.9 Ilustrasi Socket pada Komunikasi

Socket berdasarkan gambar 2.9 merupakan *stack* layer *network* dalam kernel yang berupa ruang-ruang *file*. Socket dapat diisikan data yang sesuai dengan data yang diinginkan dari lapisan atasnya, yaitu layer aplikasi. Aplikasi akan mengisikan data ke ruang yang tersedia pada socket. Ketika kedua socket yang berbeda sudah dihubungkan oleh suatu jaringan lokal/ jaringan yang sama, maka keduanya dapat saling bertukar data secara bidirectional melalui jalur yang telah terbentuk ketika kedua socket diikat/dihubungkan.

Ketika kedua socket sudah terhubung, maka kedua socket tersebut dapat bertukar informasi secara leluasa. Pertukaran informasi yang terjadi bersifat terikat dan server hanya akan terdedikasi untuk mengirimkan data ke client yang sudah terikat seperti gambar di bawah ini.



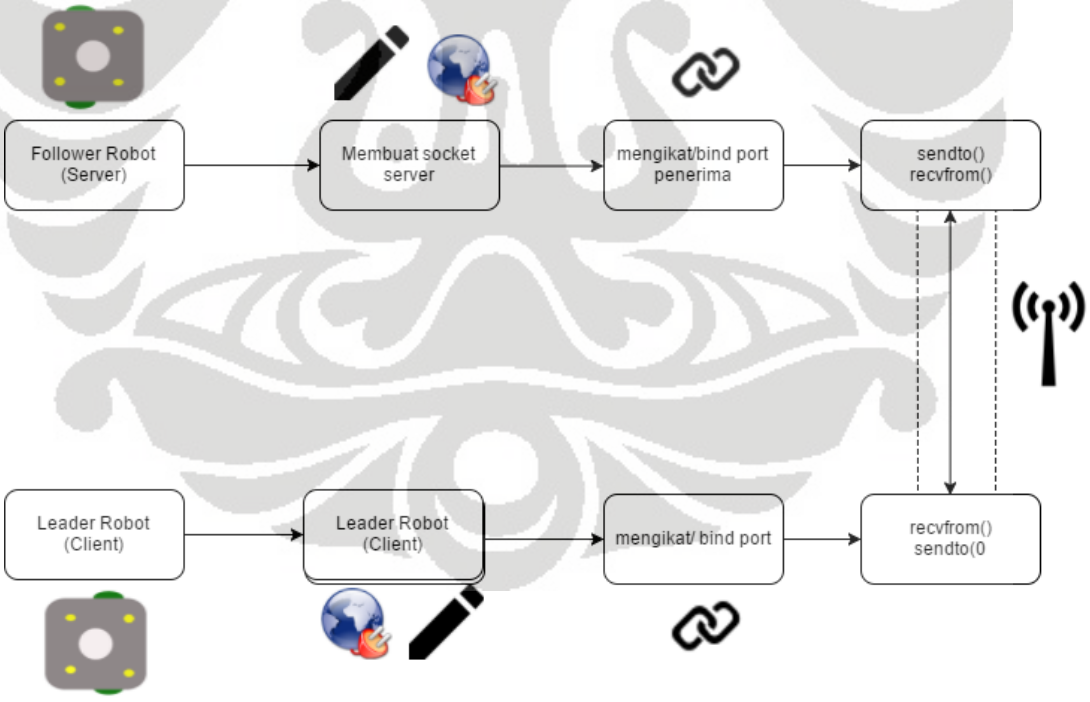
Gambar 2.10 Ilustrasi Model Server Client Bidirectional

Berdasarkan gambar 2.10 dapat dilihat bahwa ketika server masih mendengar *request* maka client apapun yang melakukan request koneksi akan diterima. Koneksi yang sudah diterima akan membentuk suatu ikatan/jalur

komunikasi antara server dan client yang bersifat bidirectional dan terdedikasi hanya untuk client tersebut.

Ada dua jenis dari *Internet socket*, yang pertama adalah *Stream Socket* dan yang kedua adalah *Datagram Socket*. *Stream Socket* adalah jenis *socket* yang digunakan untuk komunikasi yang bersifat *reliable*, seperti yang ditemukan pada komunikasi berbasis TCP/IP. *Datagram socket* digunakan untuk komunikasi yang bersifat *connectionless*, seperti yang ditemukan pada komunikasi berbasis UDP [12]. Pada skripsi ini digunakan *internet socket* jenis *datagram socket*. Berikut merupakan ilustrasi *datagram socket* pada Gambar 2.11.

Pada saat memprogram dengan *socket*, umumnya dibentuk model *client-server*, dimana *client* bertindak sebagai yang meminta data ataupun memberi data dan bersifat aktif, sedangkan *server* menunggu koneksi dari *client* secara pasif. Kemudian ketika *client* sudah selesai meminta pertukaran data maka *client* akan menutup koneksi *socket* dan mennyambungkan lagi ketika ingin meminta pertukaran data seperti yang diilustrasikan melalui Gambar 2.11.



Gambar 2.11 Ilustrasi Alur Komunikasi Datagram Socket

Dalam aplikasinya, *socket* biasa disusun dengan menggunakan *socket programming* yang akan dijelaskan pada bagian berikutnya.

2.4.2 Socket Programming

Socket programming merupakan algoritma yang bertujuan agar suatu program dapat berinteraksi dengan program lainnya dalam satu jaringan. Pada skripsi ini digunakan socket dengan model client-server. Model client-server ini membutuhkan kedua proses berada di jaringan yang sama dimana client perlu mengetahui ketersediaan dan alamat dari server. Sedangkan server tidak perlu mengetahui alamat client sebelum komunikasi terbentuk. Setelah koneksi terbentuk, kedua sisi dapat bertukar informasi secara terdedikasi.

Baik client maupun server harus membentuk socket agar dapat terhubung melalui layer transport. Ketika socket terbentuk, program harus mendefinisikan domain address dan tipe socket. Koneksi antar socket hanya dapat terbentuk jika kedua socket berada di domain yang sama dan meggunakan tipe socket yang sama juga.

Address domain socket terbagi menjadi dua, yaitu internet domain socket dan unix domain socket. Proses pada internet domain berlangsung pada kedua host yang berkomunikasi melalui internet. Sedangkan pada unix-domain kedua proses berbagi file system yang sama dalam berkomunikasi. Oleh karena itu, format address pada internet domain terdiri dari alamat IP dan pada unix domain terdiri dari entry file system tersebut.

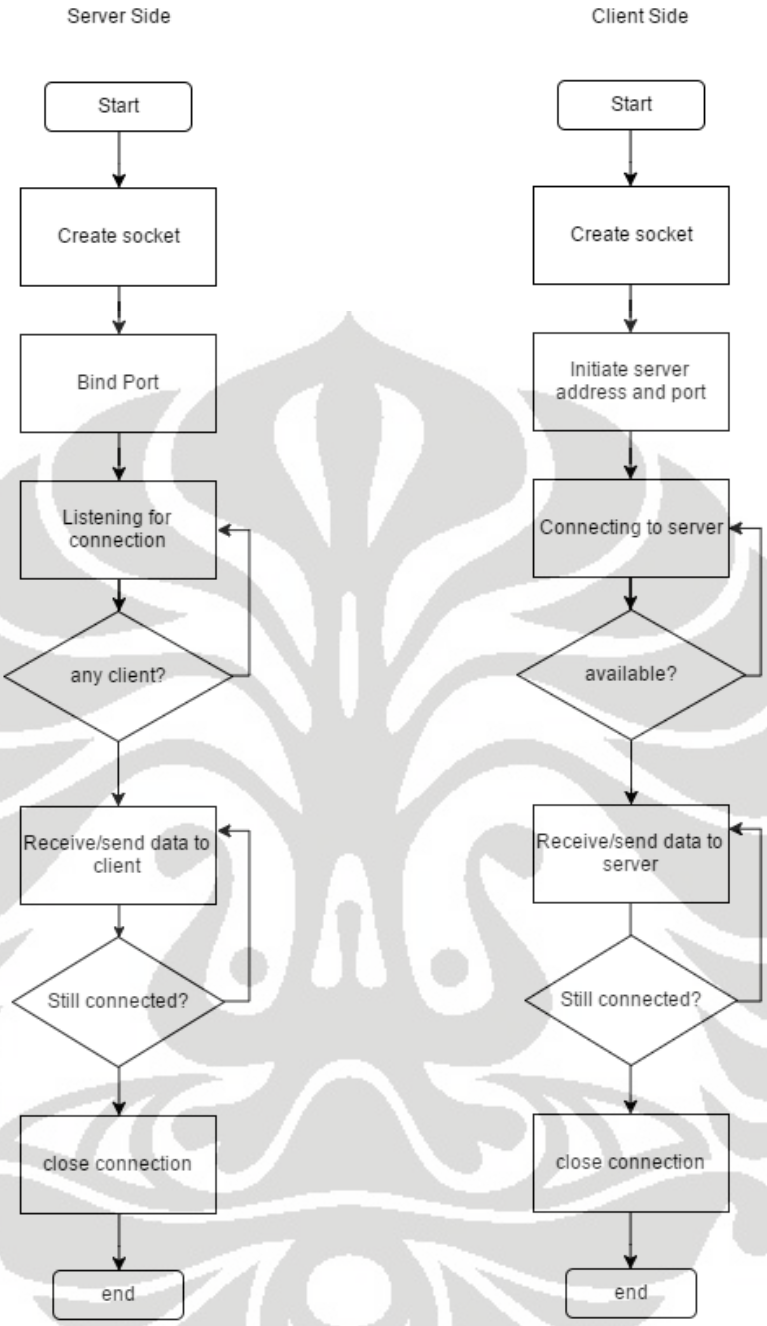
Kedua domain socket mempunyai nomor port unik yang merupakan 16-bit unsigned int. Port yang digunakan biasanya bernilai di atas 2000 karena port bernilai rendah seperti 22 sudah dipakai untuk standart service. Oleh karena itu dianjurkan untuk menggunakan nomor port dengan nilai yang besar untuk memungkinkan komunikasi yang bersifat terdedikasi dan unik.

Setelah mengetahui persyaratan dalam pembentukan komunikasi, maka dilanjutkan dengan membuat socket dan menghubungkan kedua socket yang diinginkan dengan pemrograman. Secara umum, perlu dibuat program untuk client dan juga server yang tentunya tidak identik. Oleh karena itu, berikut akan ditampilkan ilustrasi mengenai algoritma koneksi socket melalui flow chart.

Berdasarkan gambar 2.12 dapat dilihat bahwa kedua sisi baik client maupun server mempunyai algoritma yang cukup mirip tetapi berbeda dalam pembentukkan saluran komunikasinya dimana server cenderung menunggu dan menangkap permintaan pembuatan koneksi dari client yang ingin melakukan pertukaran informasi dengan server dan memberikan respon kepada permintaan yang sesuai.

Sedangkan, pada bagian client butuh dilakukan konfigurasi alamat server beserta port server yang ingin menjadi tujuan untuk diberikan permintaan untuk pembuatan koneksi. Kemudian, permintaan tersebut akan direspon oleh server dan akan dibentuk koneksi bidireksional jika server tidak terikat dengan client lain. Apabila server sudah menerima permintaan dari client untuk membuat jalur komunikasi antar keduanya, maka server akan terikat terhadap client tersebut pada port yang bersangkutan.

Setelah jalur komunikasi antara server dan client terbentuk, maka kedua sisi dapat bertukar informasi dengan jalur yang tetap terbentuk secara terus-menerus selama client masih meminta pertukaran informasi. Akan tetapi, sampai ketika client memutuskan koneksi yang terbentuk, maka server akan menutup koneksinya terhadap client. Kemudian kedua koneksi akan tertutup dan program berhenti.



Gambar 2.12 Flow Chart Pemrograman Socket untuk Server dan Client

BAB 3

PERANCANGAN SISTEM MOBILE ROBOT KOMUNIKATIF

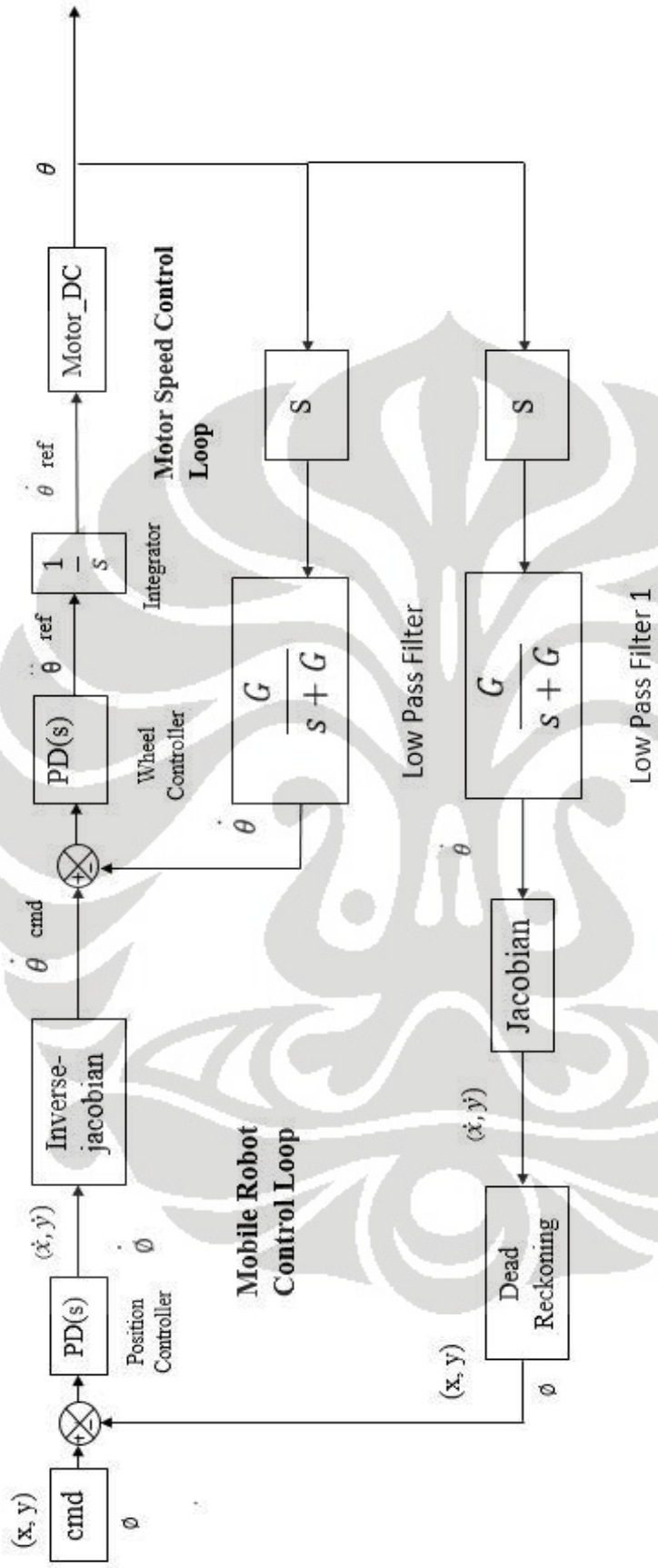
Agar memperoleh hasil sistem *mobile robot* komunikatif yang baik perlu dilakukan perancangan/desain secara keseluruhan yang baik dengan tetap mempertimbangkan mengenai bagian yang perlu dilakukan pengujian. Setelah mempunyai perancangan secara keseluruhan yang baik, untuk menunjang pengendalian sistem *mobile robot* komunikatif yang baik juga perlu memperhatikan mengenai komunikasi dan juga pengiriman paket data yang merupakan aspek penting dimana pada skripsi ini.

3.1 Desain Pengendalian Unit Mobile Robot

Berikut merupakan skematik blok diagram yang merepresentasikan rancangan sistem *closed loop mobile robot* yang ingin diraih dimana loop pengendalian dibuat berbentuk dua pengendalian *cascade* seperti yang diilustrasikan oleh gambar 3.1 di halaman berikutnya.

Secara umum, fungsi alih *closed loop* dibagi menjadi dua buah bagian loop, yaitu *inner loop* dan *outer loop* sesuai dengan arsitektur hirarkial *inner-outer loop*. Arsitektur hirarkial *inner-outer loop* sudah umum digunakan oleh berbagai bidang aplikasi yang memiliki dua jenis respons yang berbeda kecepatannya tetapi masih saling memiliki korelasi dimana respons sistem *inner loop* mempengaruhi respons dari sistem *outer loop*.

Pada sistem *mobile robot, inner loop* yang digunakan adalah pengendali kecepatan motor dan outer loop adalah pengendali posisi trayektori dari *mobile robot.* Pemisahan loop ini dilakukan karena terdapat dua buah sistem, yaitu pengendali kecepatan motor yang mempunyai respons yang cukup cepat dibandingkan dengan perubahan posisi *mobile robot* yang relatif lebih lama. Penjelasan masing-masing komponen dalam block diagram akan dijelaskan pada subbab yang spesifik menjelaskan loop yang bersangkutan.



Gambar 3.1 Blok Diagram Closed Loop Mobile Robot Sederhana

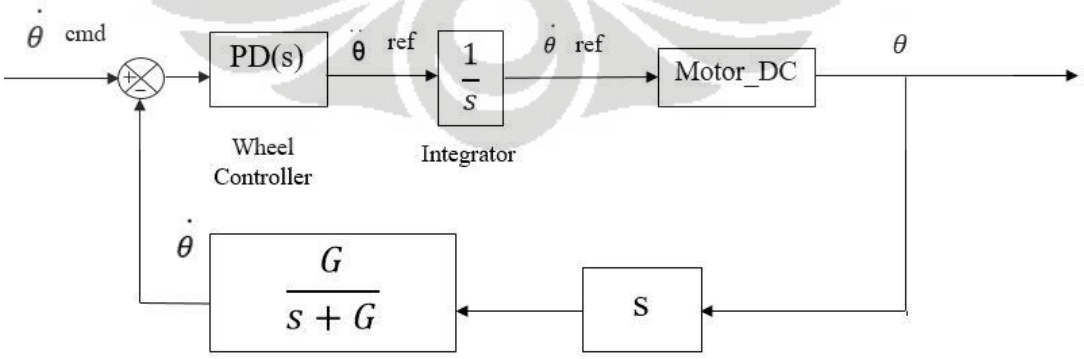
Pemilihan waktu sampling untuk kedua loop diberikan berbeda sesuatu dengan karakteristik kecepatan loop tersebut. Pada aplikasi ini, loop pengendali kecepatan motor diberikan *sampling time* yang lebih cepat dimana loop ini diharapkan memiliki waktu cuplik yang minimal 2x lebih cepat dibandingkan loop pengendali posisi yang merupakan loop lapisan atas dari pengendali kecepatan motor.

Hal ini dilakukan agar motor sebagai aktuator dari mobile robot sudah dapat mengeksekusi dan mejalankan perintah kecepatan roda kanan dan kiri yang diberikan dari layer pengendali posisi sebelum perintah tersebut berganti seiring dengan bergantinya nilai referensi posisi yang didapatkan pada loop pengendali posisi. Oleh karena itu, dilakukan pemberian waktu cuplik yang lebih cepat ke loop pengendali kecepatan motor .

Pengendalian kecepatan motor dan posisi *mobile robot* sangat penting sehingga dinilai butuh dilakukaan pemisahan *inner loop* dan *outer loop* agar didapatkan sistem yang lebih baik. *Inner loop* yang didesain dengan baik dapat memperbaiki dan menyederhanakan sistem dengan lebih efisien. Berikut akan dibahas mengenai masing-masing loop kendali dan juga perancangan pengujian kendali masing-masing loop.

3.1.1 Desain Pengendalian Motor Berbasis Kecepatan

Berikut merupakan skematik blok diagram pengendali kecepatan yang ingin dicari parameter pengendalian dan juga gain yang baik untuk filter.



Gambar 3.2 Blok Diagram Loop Pengendali Kecepataan Motor Mobile Robot

Pada loop pengendali kecepatan ini terdiri dari blok motor dc, pengendali *Proporsional-plus-Derivative* (PD), blok *low-pass filter* dan *integrato*r untuk menentukan kecepatan sudut referensi yang akan diberikan ke motor. Blok motor sendiri pun terdiri dari rangkaian elekris dan rangkaian mekanik motor seperti yang sudah disinggung pada dasar teori.

Berikut akan dibahas perancangan dasar untuk setiap komponen loop pengendali kecepatan yang telah dilakukan yang mencakup parameter motor, *low pass filter*, dan pengendali kecepatan PD yang digunakan.

Komponen pengendali roda (*wheel controller)* digunakan untuk mengendalikan respon motor agar mencapai nilai perintah yang diberikan oleh lapisan yang lebih luar, yaitu pengendali posisi mobile robot. Pengendali yang digunakan merupakan pengendali PD yang akan menghasilkan keluaran berupa percepatan referensi. Percepatan referensi akan diintegralkan menjadi kecepatan referensi untuk dimasukkan ke motor dengan memberikan referensi kecepatan yang akan dikonversikan menjadi duty cycle PWM untuk mengendalikan motor DC yang digunakan seperti yang sempat dijelaskan pada dasar teori. Nilai referensi kecepatan yang diinputkan ke motor DC melalui PWM akan mengendalikan motor DC untuk berputar sesuai dengan besarnya duty cycle yang diberikan.

Hasil dari perputaran motor DC kemudian akan dibaca oleh sensor kuadratur enkoder dan dikembalikan sebagai *feedback* negatif dari sistem. Hasil pembacaan berupa posisi sudut yang memiliki banyak noise sehingga untuk diumpan balikkan menjadi kecepatan sudut perlu dilakukan penurunan terlebih dahulu. Kemudian, hasil tersebut dimasukkan ke filter low pass untuk menyaring nilai noise yang tercampur dikarenakan pembacaan sensor. Penjelasan mengenai filter akan difokuskan lebih dalam pada pembahasan selanjutnya.

Pengendali kecepatan motor yang digunakan seperti yang sempat disinggung sebelumnya adalah pengendali *Proportional-plus-Derrivative*. Pengendali ini akan mendapatkan masukkan *error* antara kecepatan sebenarnya

dengan kecepatan command yang diberikan oleh loop di atasnya, yaitu loop pengendali *mobile robot*.

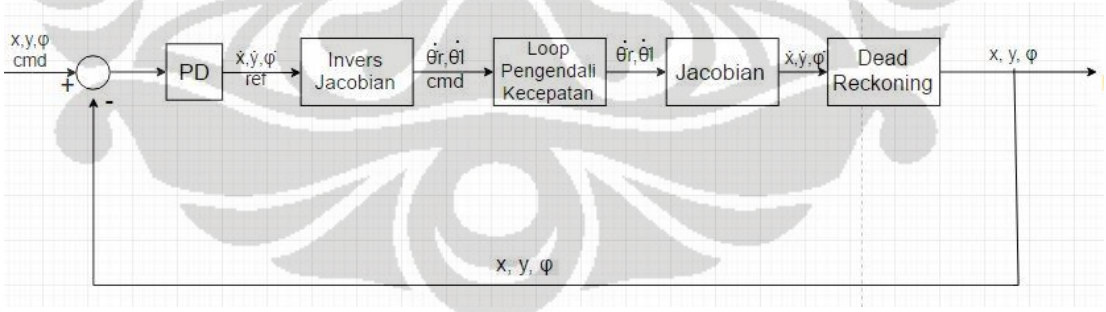
Pengendali ini membutuhkan konstanta pengendali Kp dan Kv yang sesuai agar memperbaiki respons sistem yang pada awalnya tidak dapat mengejar nilai input referensi yang diberikan. Pengujian nilai konstanta pengendali yang digunakan untuk loop kecepatan dilakukan dengan melakukan *tuning* manual dan menurunkan nilai dari fungsi alih.

Parameter respons yang ingin didapatkan dengan melakukan proses pengujian nilai pengendali adalah sistem dengan respon yang mampu mencapai nilai referensi yang diberikan dengan relatif cepat tetapi memiliki *overshoot* yang rendah, dan juga memiliki osilasi yang tidak terlalu besar pada nilai setimbangnya.

Setelah mendapatkan respons motor yang sesuai untuk setiap unitnya, maka langkah selanjutnya akan dibahas mengenai desain pengendalian posisi dan orientasi mobile robot untuk setiap unitnya.

3.1.2 Desain Pengendalian Posisi dan Orientasi Mobile Robot

Berikut merupakan skematik blok diagram loop pengendalian *mobile robot* yang akan dicari parameter pengendali posisi *mobile robot* dan orientasinya.



Gambar 3.3 Blok Diagram Loop Pengendalian Mobile Robot

Pada loop pengendalian *mobile robot*, variable yang akan dikendalikan adalah posisi *mobile robot*, yaitu posisi kartesian (x,y) dari *mobile robot* dan juga orientasi (φ) dimana keduanya masing-masing menggunakan pengendali Proporsional-plus-Derivative(PD). Dalam loop pengendalian mobile robot ini

terdapat beberapa block yang sempat dibahas sebelumnya baik pada dasar teori maupun subbab perancangan yang sebelumnya. Oleh karena itu, akan dibahas secara singkat mengenai komponen di dalam loop ini sebelum akhirnya masuk ke bagian perancangan.

Pada loop ini, setelah perintah didapatkan, maka akan masuk ke pengendali PD yang akan memberikan referensi posisi maupun orientasi. Nilai posisi dan orientasi tersebut masih merupakan referensi dalam koordinat global sehingga untuk digunakan sebagai referensi perputaran sendi, yaitu roda mobile robot masih perlu dilakukan transformasi dengan menggunakan invers jacobian sehingga didapatkan nilai referensi dalam sudut roda mobile robot. Hasil keluaran dari invers jacobian merupakan kecepatan sudut perintah roda kanan dan roda kiri yang kemudian akan menjadi masukkan dari loop di dalamnya, yaitu loop pengendali kecepatan motor DC yang telah dibahas pada subbab sebelumnya.

Keluaran dari motor DC, yang berupa kecepatan sudut roda kanan dan roda kiri kemudian diumpan balikkan ke loop pengendali *mobile robot*. Umpan balik yang masih berupa kecepatan sudut roda kanan dan roda kiri ditransformasikan dengan matriks jacobian untuk didapatkan kecepatan sesaat terhadap sumbu x dan sumbu y yang diperlukan dalam mencari kecepatan linier *mobile robot* saat itu seperti yang telah dijelaskan pada dasar teori. Nilai kecepatan linier dan juga perubahan orientasi kemudian akan digunakan untuk mengestimasi posisi kartesian mobile robot dengan menggunakan algoritma *dead reckoning*. Nilai posisi kartesian dan orientasi yang telah didapatkan akan menjadi umpan balik negatif terhadap perintah posisi yang telah diberikan sebelumnya.

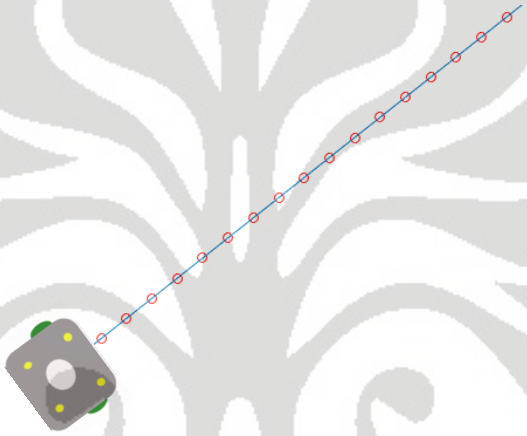
Kemudian, setelah membahas secara singkat masing-masing komponen pada loop pengendali mobile robot ini, perlu dibahas mengenai perancangan pengendalian mobile robot secara keseluruhan. Berikut akan dibahas secara spesifik mulai dari perancangan algoritma trayektori *mobile robot* awal, pengendali posisi, dan pengendali orientasi *mobile robot*.

Pada skripsi ini, akan dilakukan percobaan *mobile robot* dengan dua skema gerakan/ trayektori yang berbeda. Kedua trayektori tersebut antara lain trayektori

linier dan trayektori sinusoidal. Berikut akan dibahas mengenai kedua trayektori tersebut.

a. Trayektori Linier

Pada trayektori linier, *mobile robot* bergerak dari titik awal ke titik akhir dengan pergerakan garis lurus. Nilai x dan y dari *mobile robot* akan bertambah secara linier sehingga akan menghasilkan pergerakan yang seperti garis lurus. Berikut merupakan illustrasi dari pergerakan *mobile robot* dengan trayektori linier.



Gambar 3.4 Ilustrasi Trayektori Linier

b. Trayektori Sinusoidal

Pada trayektori sinusoidal, mobile robot diharapkan bergerak dengan bentuk sinusoidal dimana sumbu x dianggap sebagai waktu sehingga pada sumbu x merupakan trayektori linier sedangkan sumbu y berupa persamaan sinusoidal terhadap x. Persamaan sinusoidal yang digunakan dapat menggunakan persamaan sinusoidal biasa seperti y(t)=Asin(ωt) sehingga akan didapatkan pergerakan *mobile robot* berbentuk gelombang sinusoidal.

Berikut merupakan rumus posisi x dan y trayektori sinusoidal untuk masing-masing pencuplikannya.

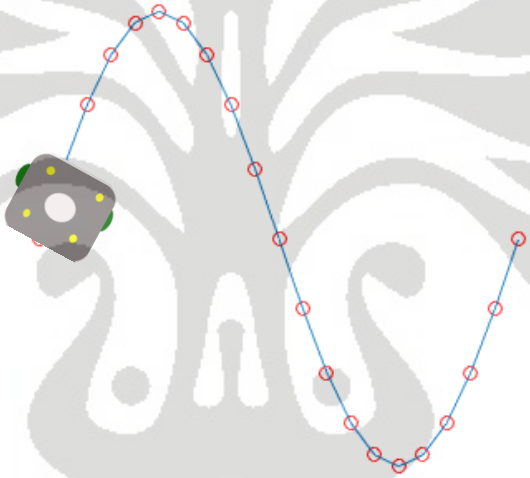
$$x[n] = \frac{(x_{final} - x_{init}) * n}{N} \quad (3.1)$$

$$y[n] = A * \sin(\omega * x[n]) \quad (3.2)$$

dimana

x[n]	=	Tujuan posisi x saat pencuplikan ke-n	(m)
y[n]	=	Tujuan posisi y saat pencuplikan ke-n	(m)
x_{final}	=	Posisi akhir x robot yang diinginkan	(m)
x_{init}	=	Posisi awal x mobile robot	(m)
n	=	Pencuplikan saat ini	
N	=	Total banyak pencuplikan	
A	=	Amplitudo gelombang sinusoidal	
ω	=	Frekuensi gelombang sinusoidal	

Berikut merupakan ilustrasi dari trajectory sinusoidal.



Gambar 3.5 Ilustrasi Trayektori Sinusoidal

Agar *mobile robot* dapat mencapai trayektori yang diinginkan, maka perlu dilakukan pengendalian posisi koordinat kartesian *mobile robot* terhadap posisi yang merupakan target/ referensi yang ingin dicapai oleh *mobile robot*. Pengendalian tersebut menggunakan metode RMRC *(Resolved Motion Rate Control*) sehingga titik koordinat dibagi menjadi beberapa bagian yang kecil untuk setiap samplingnya.

Pengendali yang digunakan untuk mengendalikan posisi *mobile robot* adalah pengendali *Proporsional-plus-Derivative* (PD). Masukkan dari pengendali ini adalah error yang dihasilkan dari *trajectory planning* dengan posisi *mobile robot* saat ini yang didapatkan melalui dead reckoning. Sama seperti pengendali PD

pada pengendalian kecepatan motor, diperlukan kedua parameter konstanta pengendali Kp dan Kv yang masing-masing merupakan konstanta untuk *proporsional* dan *derivative* secara berurutan.

Parameter respons yang ingin didapatkan dengan melakukan proses pengujian nilai pengendali adalah sistem dengan respon yang mampu mencapai nilai referensi yang diberikan dan tidak memiliki *offset* yang besar pada nilai setimbangnya. Oleh karena itu, perlu dilakukan pengujian untuk mencari nilai parameter konstanta pengendali yang sesuai untuk mendapatkan respons yang diinginkan.

Kemudian, setelah mencari nilai pengendali posisi, perlu juga dicari nilai parameter pengendali orientasi *mobile robot*. Nilai pengendali orientasi biasanya dibedakan terhadap nilai pengendali posisi untuk beberapa kasus yang membutuhkan pengendali orientasi yang berbeda, seperti pada kasus trayektori linier bergradien dan trayektori sinusoidal.

Akan tetapi, untuk beberapa aplikasi untuk trayektori linier yang hanya bergerak sepanjang sumbu x saja, tak jarang ditemukan penggunaan nilai parameter pengendali orientasi yang sama dengan nilai konstanta pengendali posisi. Hal ini dinilai kurang baik dalam pengendalian *mobile robot* dikarenakan orientasi dengan posisi *mobile robot* memiliki batasan yang berbeda sehingga perlu dilakukan pemisahan nilai pengendali, walau jenis pengendali yang digunakan sama, yaitu *Proporsional-plus-Derivative*(PD).

Oleh karena itu, perlu juga dicari kombinasi antara Kp dan Kv posisi *mobile robot* dengan Kp dan Kv orientasi *mobile robot* yang sesuai agar *mobile robot* dapat bergerak mengikuti trayektori yang diberikan secara baik.

3.1.3 Perancangan Filter Low-Pass untuk Mengatasi Derivative Noise

Kemudian komponen lain yang juga penting adalah fiter *low-pass*. Pada loop pengendali kecepatan motor output dari motor yang berupa putaran akan dibaca oleh sensor posisi sudut yang digunakan, yaitu enkoder quadratur. Hasil dari enkoder kuadratur yang berupa posisi akan dibaca dan diumpan balikkan

sebagai keluaran sebenarnya yang akan terus dikendalikan agar mengikuti posisi/ kecepatan yang diinginkan.

Pada inner loop yang merupakan pengendali kecepatan motor, umpan balik yang diharapkan ialah kecepatan sudut juga. Sedangkan, keluaran dari enkoder kuadratur merupakan posisi motor sehingga perlu hasil sensor perlu diturunkan. Setelah diturunkan akan didapatkan kecepatan angular yang akan dimasukkan ke dalam *low pass filter* yang berguna untuk meredam efek dari *noise*.

Pada setiap *derivative*/ penurunan secara umum akan memberikan efek banyaknya *noise* sehingga tidak jarang ditemukan banyak model *derivative* yang ditambahkan *low pass filter* untuk mengurangi *noise* yang didapatkan. Pada aplikasi pengendalian kecepatan motor pun juga digunakan filter *low pass* untuk mengurangi efek *noise* yang didapatkan dari derivative sehingga pada setiap penurunan suatu variable akan dilewatkan *low pass filter* terlebih dahulu. Berikut merupakan fungsi alih *low pass filter* yang digunakan untuk *derivative*.

$$T(s) = \frac{G}{s+G} \tag{3.3}$$

Berikut merupakan blok diagram *low pass filter* untuk *derivative*.

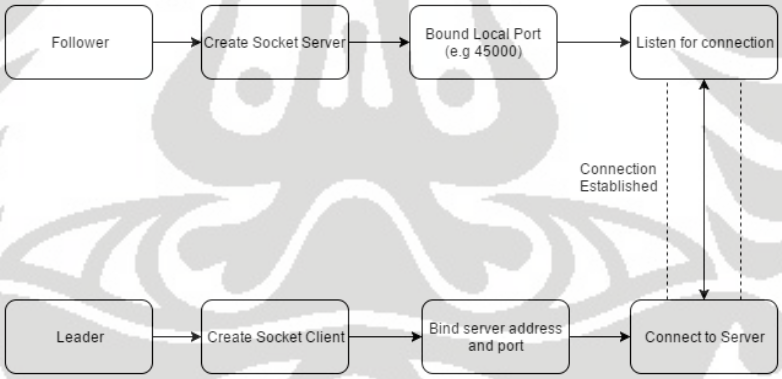
Gambar 3.6 Blok Diagram Low Pass Filter dan Derivative.

3.2 Perancangan Komunikasi Antar *Mobile Robot*

Pada skripsi ini komunikasi antara kedua *mobile robot* dilakukan dengan menggunakan komunikasi nirkabel *Wi-Fi* melalui *internet socket*. Komunikasi dilakukan dengan menggunakan *Wi-Fi* terintegasi mikrokontroler yang berada pada masing-masing robot. Sebelumnya alamat IP dari masing-masing mikrokontroler robot, perlu dibuat menjadi *static* dan unik satu sama lain sehingga tidak terjadi kesalahan pengikatan koneksi.

Perlu diperhatikan untuk dapat berkomunikasi kedua robot harus terhubung dengan jaringan local yang sama, atau secara sederhana dapat dikatakan kedua robot harus terkoneksi *Wi-Fi* yang sama. Setelah terhubung ke jaringan lokal yang sama dan mempunyai alamat IP yang unik, maka saluran untuk berkomunikasi dapat dibuat antar kedua robot. Pada pertukaran data dengan menggunakan *internet socket* perlu ditentukan robot yang akan menjadi *server* dan *client*.

Hal ini ditentukan melalui perilaku dan kebutuhan robot itu sendiri, dimana untuk robot yang perlu menerima data secara terus menerus ditetapkan sebagai *server* dan robot yang memberikan informasi ditetapkan sebagai *client*. Pada aplikasi ini, diharapkan robot penerima dapat mengikuti pergerakan pengirim sehingga pengirim akan terus memberikan posisinya secara terus-menerus ke penerima. Berdasarkan spesifikasi tersebut, maka robot penerima ditetapkan menjadi *server* dan robot pengirim ditetapkan menjadi client. Berikut merupakan ilustrasi sederhana mengenai bagaimana koneksi terbentuk antara *server* dan *client*.



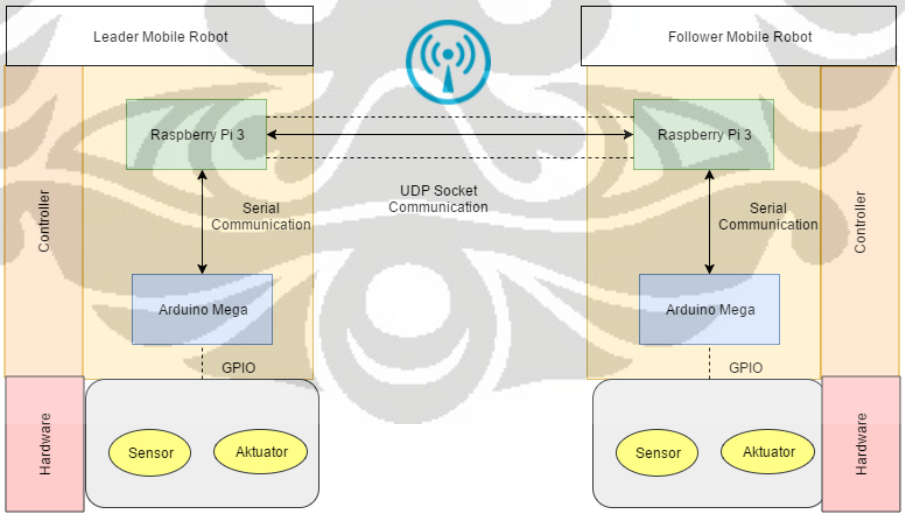
Gambar 3.7 Ilustrasi Pembentukan Koneksi Socket

Berdasarkan ilustrasi tersebut dapat dilihat bahwa *server*, yaitu robot penerima, akan menunggu permintaan koneksi dari *client*, yaitu pengirim, dan ketika permintaan untuk membuat sambungan diterima oleh penerima, maka koneksi akan terhubung dan pengikatan *port server* dan *client* terjadi sehingga koneksi komunikasi *bidirectional* terbentuk. Setelah koneksi terbentuk maka pertukaran informasi dapat terjadi.

Pertukaran data pada skripsi ini dirancang satu arah, yaitu dari pengirim menuju penerima dimana informasi yang diberikan adalah informasi posisi dan orientasi pengirim yang akan diikuti oleh penerima. Hal ini dinilai lebih mudah diimplementasikan walaupun tingkat keberhasilan pergerakan bertumpu kepada robot pengirim.

Kemudian, pada skripsi ini skema komunikasi yang dirancang yaitu dengan menggunakan *socket datagram* diimplementasikan menggunakan Raspberry Pi 3 yang terintegrasi dalam mobile robot. Setiap unit Raspberry Pi 3 pada mobile robot dapat terkoneksi Wi-Fi dan saling berkomunikasi satu sama lain. Socket dibuat melalui program yang sekaligus menggabungkan data yang diterima melalui serial komunikasi dari Arduino yang merupakan mikrokontroler yang berhubungan langsung dengan sensor dan aktuator.

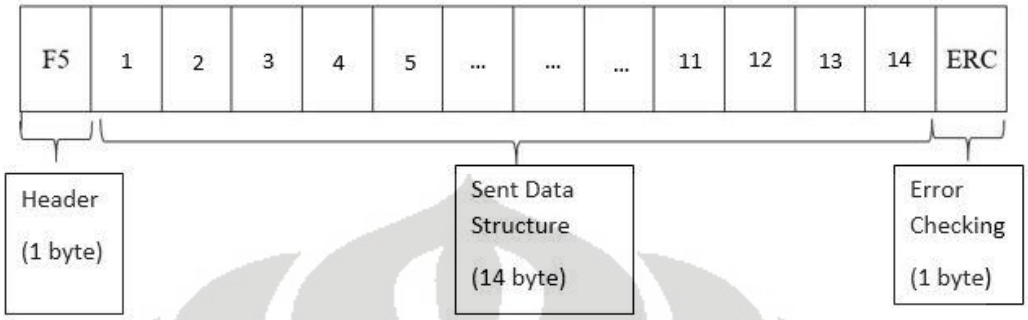
Data yang didapatkan melalui komunikasi serial dari Arduino kemudian akan diteruskan Raspberry Pi 3 menjadi paket data yang akan dikirim melalui *socket datagram* ke mobile robot lainnya. Kemudian, pada skripsi ini akan dilakukan pengujian terhadap implementasi komunikasi antar robot yang menggunakan *socket datagram* yang akan dibahas pada bab selanjutnya.



Gambar 3.8 Ilustrasi Komunikasi Antar Robot

Pada skripsi ini akan dikirimkan data berupa *struct* yang terdiri dari *16 byte* dengan *1 byte header* dan *1 byte error checking* yang diletakkan masing-masing

di awal dan di akhir data yang ingin dikirim. Penggunaan *struct* dilakukan untuk meningkatkan efisiensi jumlah data yang dikirim. Berikut merupakan format paket yang dikirimkan.



Gambar 3.9 Format Paket yang Dikirim

Sesuai dengan gambar di atas, paket yang dikirim diberikan header terlebih dahulu sebagai penanda bahwa data yang akan ditangkap merupakan data yang benar. Nilai header diusahakan unik sehingga digunakan nilai F5 sebagai header dikarenakan keunikan nilai dan juga simbol ASCII dari nilai F5 sendiri. Kemudian, setelah header ditangkap maka data akan dibuka dan dibaca nilai berikutnya sampai data terakhir. Data yang dikirimkan pada aplikasi ini, sebanyak 14 *byte* data yang merupakan gabungan dari perintah *keyboard* dan posisi serta orientasi dari pengirim. Perincian data yang dikirim akan dijelaskan melalui tabel berikut.

Tabel 3.1 Rincian Data yang Dikirimkan Beserta Ukurannya

No	Data	Tipe Data	Ukuran
1	keyboard command	integer	2 byte
2	x	float	4 byte
3	y	float	4 byte
4	ϕ	float	4 byte
Total Data			14 byte

Berdasarkan tabel di atas dapat dilihat bahwa data yang dikirimkan untuk data posisi dan orientasi berupa float dikarenakan nilai yang didapatkan bukanlah bilangan bulat melainkan nilai masih berupa bilangan riil dengan sisa pecahan. Kemudian untuk keyboard command yang berupa nilai karakter ascii, penggunaan nilai integer dinilai cukup sehingga digunakan tipe data integer.

Setelah semua data dipindahkan maka akan dilakukan pengecekan error checking dengan menggunakan XOR data-data yang diterima dengan nilai error checking yang dikirim pada akhir paket. Hal ini dilakukan untuk validasi data yang telah diterima dan mengetahui kesesuaian jumlah data yang diterima terhadap jumlah data yang dikirimkan.

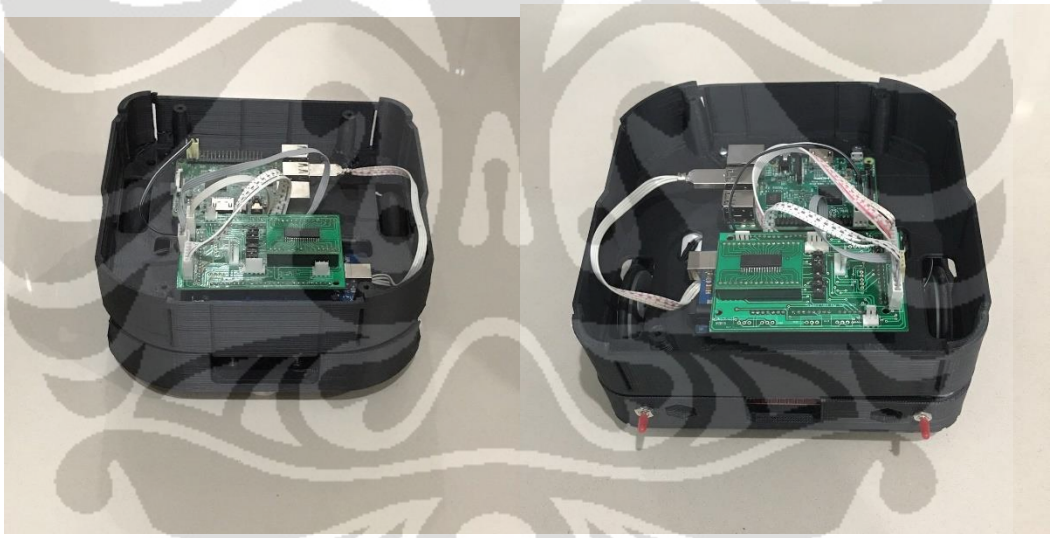
BAB 4

IMPLEMENTASI DAN ANALISA

Sistem *mobile robot* komunikatif sangat bergantung pada pengendalian setiap anggotanya dimana setiap unitnya memiliki pengendalian motor dan pengendalian mobile robot. Untuk itu langkah pertama yang dilakukan adalah menguji pengendalian masing-masing unit mobile robot yang digunakan..

4.1 Implementasi Pengujian Setiap Unit Mobile Robot

Setelah dilakukan perancangan untuk pengujian setiap unit mobile robot, perlu dilakukan implementasi pengujia setiap unit *mobile robot*. *Mobile robot* yang akan diuji berjumlah dua dan masing-masing sudah ditetapkan *mobile robot* pengirim dan *mobile robot* penerimanya. Berikut merupakan gambar kedua *mobile robot* yang digunakan tersebut



Gambar 4.1 Kedua Mobile Robot yang Digunakan untuk Implementasi Sistem Komunikatif

Kedua *mobile robot* yang digunakan didesain identik dengan perbedaan roda dan posisi penempatan mikrokontroler saja. Akan tetapi, walaupun identik keduanya dapat mempunyai karaktersitik respon yang berbeda sehingga dibutuhkan pengujian untuk masing-masing robot sehingga didapatkan karakteristik masing-masing robot. Berikut akan dibahas mengenai pengujian masing-masing unit *mobile robot.*

Setiap unitnya dilakukan pengujian pengendalian motor dan juga pengendalian posisi serta orientasi *mobile robot.* Dalam implementasinya, digunnakan waktu cuplik loop pengendalian motor yang lebih cepat dibandingkan loop pengendalian posisi untuk menunjang tercapainya perintah yang diberikan. Pada implementasi skripsi ini, digunakan waktu cuplik loop pengendali motor sebesar 4 ms dan loop pengendali posisi dan orientasi mobile robot sebesar 20 ms seperti yang ditunjukkan pada tabel berikut.

Tabel 4.1 Tabel Waktu Cuplik yang Digunakan

No	Nama Loop	Sampling time (ms)
1	Pengendali Kecepatan Motor	4
2	Pengendali Posisi dan Orientasi Mobile Robot	20

Selanjutnya akan dibahas mengenai hasil pengujian dan analisa setiap unit robot dengan menggunakan waktu cuplik yang sama untuk kedua robotnya.

4.2 Hasil Pengujian dan Analisa Setiap Unit Mobile Robot

Sebelum memulai sistem *mobile robot* kooperatif, perlu ditinjau respons dan kemampuan masing-masing unit dalam mengikuti perintah trayektori yang diberikan. Oleh sebab itu, akan dibahas mengenai pengujian dan analisa hasil pengujian untuk setiap robot yang digunakan dimulai dari pengendalian motor sampai ke pengendalian posisi dan orientasi mobile robot masing-masing unitnya. Sebelum masuk ke pembahasan mengenai pengendali motor setiap unitnya, berikut akan dibahas pengaruh gain low-pass filter terhadap respon sistem secara keseluruhan.

4.2.1 Pengaruh Gain Low-Pass Filter terhadap Respon Sistem

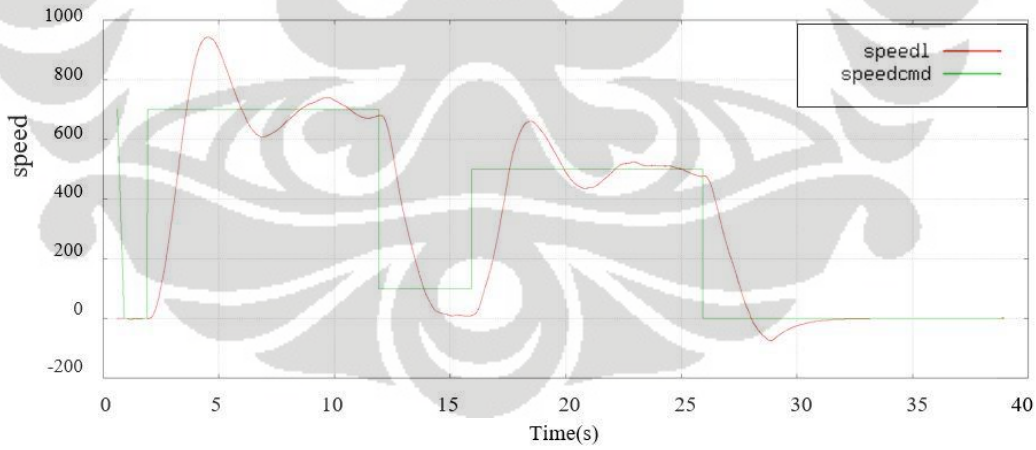
Implementasi program yang dilakukan untuk merealisasikan *low pass filter* untuk penurunan menggunakan algoritma yang didekatkan dengan model persamaan kontinu. Sedangkan, implementasi program dilakukan dengan menggunakan mikrokontroler yang merupakan pengendali diskrit sehingga terlihat terdapat kemungkinan ketidaksesuaian pada saat implementasi. Akan tetapi, ketika membandingkan hasil penyaringan *low pass filter* dengan simulasi MATLAB didapatkan bahwa dengan penggunaan algoritma filter yang seperti

kontinu nilai frekuensi *cut-off* asli yang berbeda dengan nilai gain model *low-pass filter* yang digunakan.

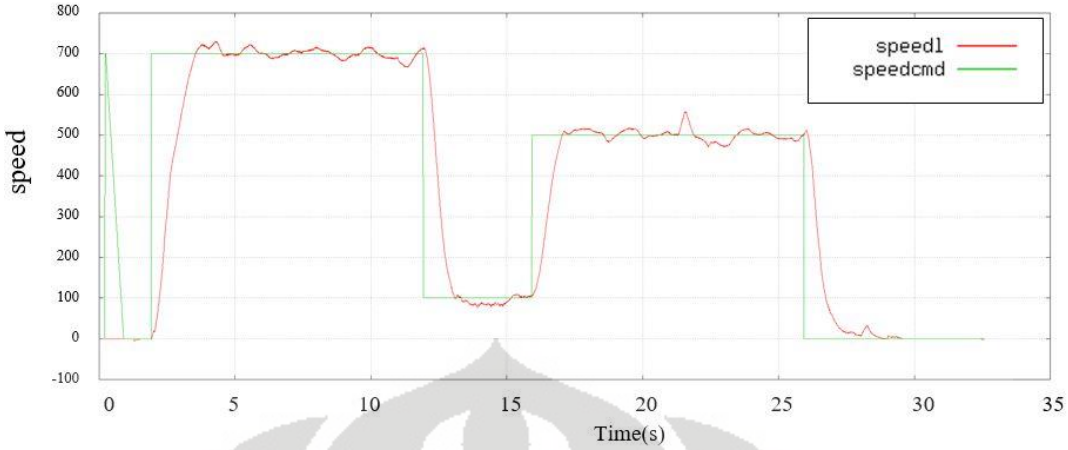
Setelah diuji denan menggunakan waktu cuplik yang sesuai dengan waktu cuplik yang akan digunakan dalam sistem, yaitu 20 ms, didapatkan bahwa dengan menggunakan gain low pass filter yang sesuai dan digunakan pada sistem ini, yaitu 75 didapatkan bahwa nilai frekuensi cut-off sebenarnya adalah 238.73 Hz.

Kemudian pada subbab ini juga akan diamati mengenai pengaruh gain *low pass filter* terhadap resepons sistem secara keseluruhan. Pengamatan dilakukan melalui hasil pengujian pemberian variasi gain low pass filter dengan nilai konstanta pengendali yang statis/tetap. Pada pengujian ini digunakan nilai pengendali yang tetap agar dapat diamati pengaruh dari low pass filter tanpa pengaruh lainnya sehingga dapat disimpulkan nilai low pass filter yang cocok untuk digunakan untuk aplikasi mobile robot komunikatif.

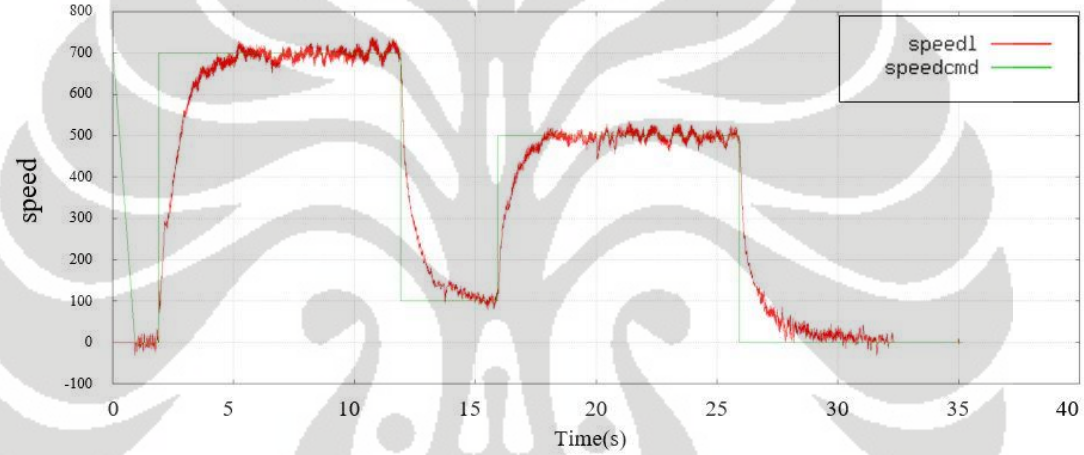
Berikut merupakan hasil pengujian pengaruh gain *low pass filter* pada lingkar pengendalian kecepatan motor dengan berbagai variasi gain yaitu 1, 5, 50, 75 dan 100 dengan menggunakan konstanta Kp pengendali kecepatan bernilai 6 dan juga Kv pengendali kecepatan bernilai 4.9 untuk masing-masing gainnya.



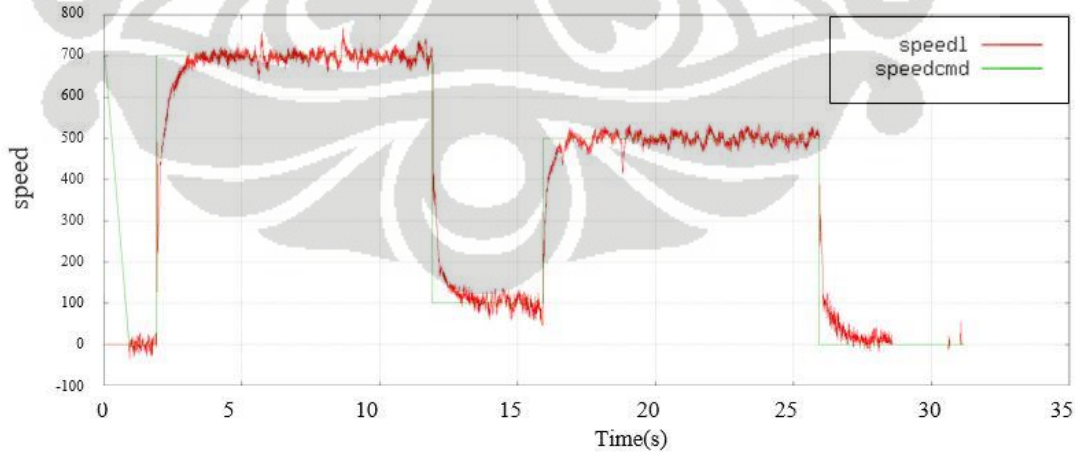
(a)



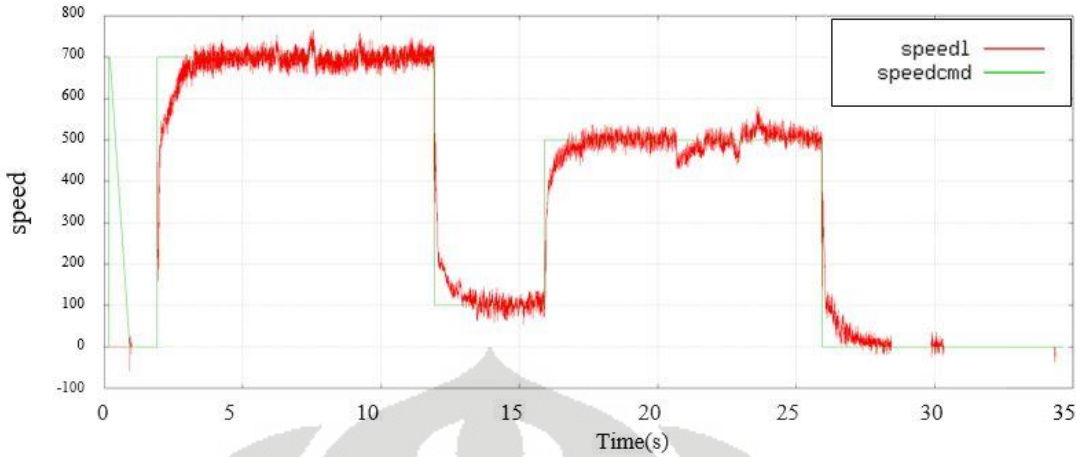
(b)



(c)



(d)



(e)

Gambar 4.2 Hasil step response untuk berbagai gain LPF : (a) Gain =1; (b) Gain=5; (c) Gain=50; (d) Gain =75; (e) Gain=100 dengan Pengendali Tetap

Berdasarkan kelima grafik hasil *step response* untuk berbagai gain di atas, dapat diamati bahwa beberapa parameter yang berganti/ berubah seiring dengan penggantian gain *low pass filter* antara lain adalah ripple/ tingkat kehalusan grafik yang diperoleh, data yang dilewatkan oleh filter, nilai *overshoot* sistem yang didapatkan, *settling time* dan juga *rise time* dari sistem.

Setelah membandingkan keempat grafik tersebut, dapat dilihat bahwa semakin besar gain *low pass filter* maka grafik akan semakin halus, nilai *overshoot* mengecil dan juga *settling time* beserta *rise time* sistem semakin cepat. Akan tetapi, dapat dilihat berdasarkan fungsi alih pada Gambar 3.4, dapat diamati bahwa semakin besar gain yang diberikan maka frekuensi cutoff yang dimiliki sistem semakin besar sehingga melewatkan lebih banyak data ataupun *noise* sehingga mempengaruhi tingkat kehalusan grafik.

Secara keseluruhan pengaruh pemberian nilai gain *low pass filter* dapat dilihat melalui tabel di bawah ini.

Tabel 4.2 Tabel Pengaruh Gain Low Pass Filter terhadap Respon Sistem

No	Parameter	Gain	
		Diperbesar	Diperkecil
1	Ripple	bertambah banyak	bertambah sedikit
2	Data yang terbuang	lebih sedikit	lebih banyak
3	Overshoot	lebih kecil	lebih besar
4	Settling Time	lebih cepat	lebih lambat
5	Rise Time	lebih cepat	lebih lambat

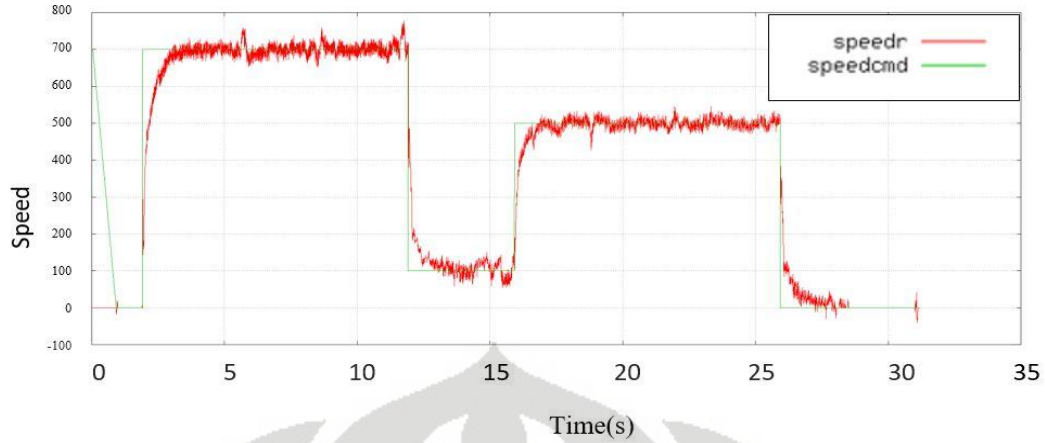
Setelah mengamati pengaruh perubahan gain low pass filter terhadap sistem maka selanjutnya akan masuk ke pembahasan pengendalian unit mobile robot yang terdiri dari pengendali motor dan pengendali posisi. Pembahasan akan dimulai dengan pengendali motor setiap unit terlebih dahulu.

4.2.2 Pengujian Nilai Pengendali Motor Setiap Unit

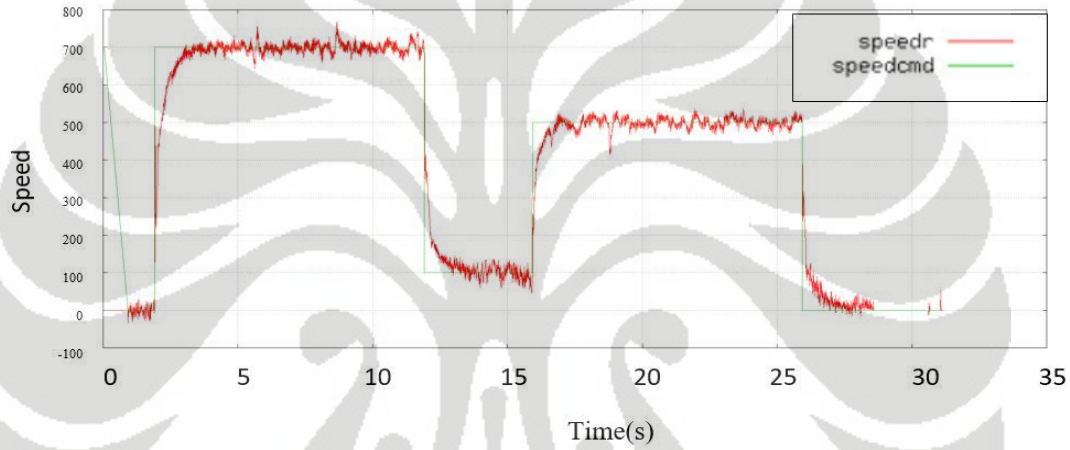
Kemudian, akan dibahas mengenai pengujian nilai konstanta pengendali PD untuk lingkar pengendalian kecepatan motor. Pemberian nilai konstanta Kp dan Kv yang berbeda akan memberikan respon sistem yang berbeda pula. Kp atau konstanta *proporsional* akan mempengaruhi besarnya sinyal pengendali ketika error membesar/mengecil sedangkan Kv atau konstanta *derivative* akan mempengaruhi osilasi dari respons. Berikut merupakan tabel pengaruh pemberian nilai konstanta Kp dan Kv yang berbeda untuk sistem secara general berdasarkan percobaan pengaruh masing-masing konstanta yang pernah dilakukan.

Oleh karena itu, perlu dicari nilai kombinasi konstanta pengendali Kp dan Kv serta gain derivative untuk pengendali *derivative* yang sesuai agar sistem dapat dikompensasi/ dikoreksi menjadi respons yang diinginkan/ sesuai spesifikasi yang diperlukan. Setelah melakukan pengujian nilai konstanta pengendali yang akan dipakai untuk sistem, didapatkan kombinasi yang paling efektif adalah pengendali dengan nilai Kp=6, Kv=4.9, dan dengan gain *low pass filter* bernilai 75.

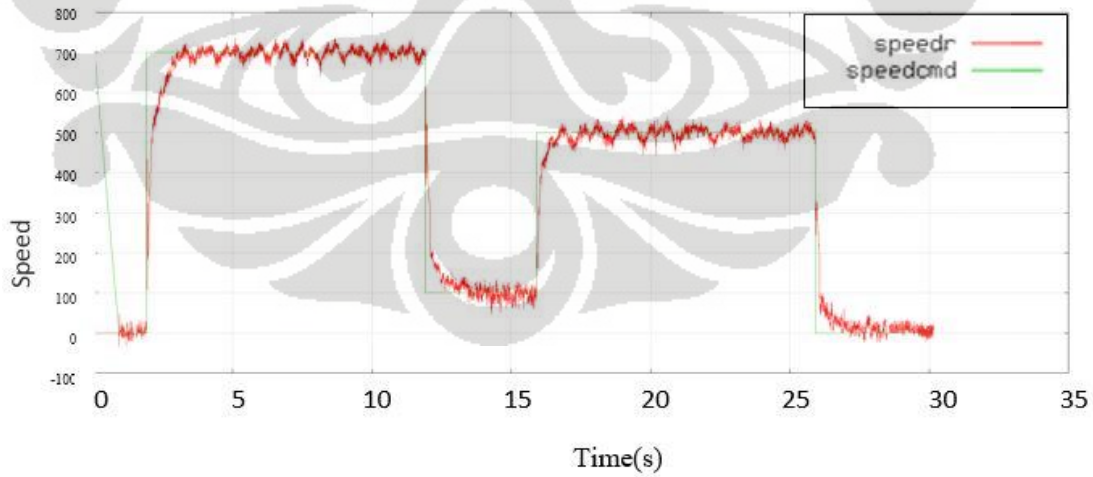
Berikut merupakan hasil respons sistem roda kanan dan roda kiri untuk nilai kombinasi tersebut.



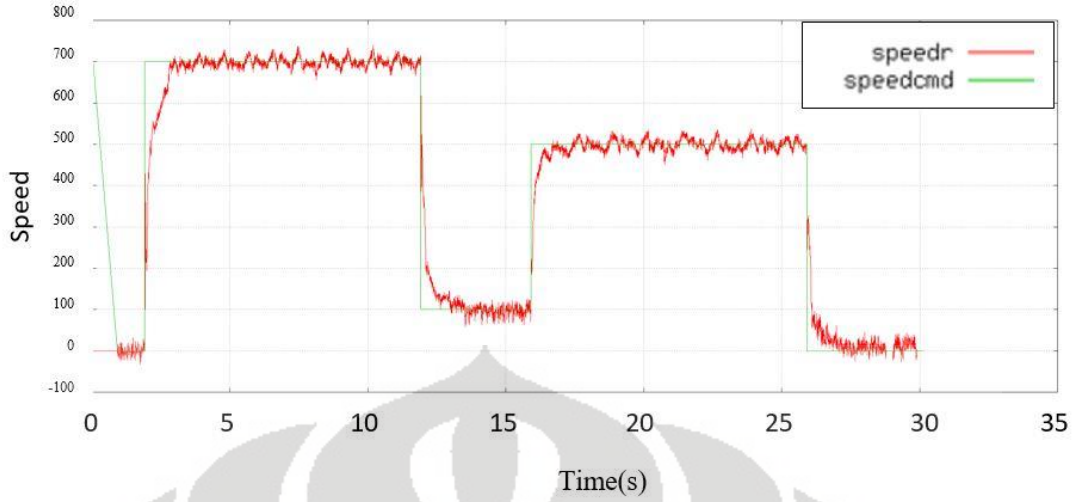
(a)



(b)



(c)



(d)

Gambar 4.3 Grafik Respon Motor DC denan Kp=6, Kv=4.9, dan Gain LPF=75 : (a) Motor Kiri Pengirim; (b) Motor Kanan Pengirim; (c) Motor Kiri Penerima; dan (d) Motor Kanan Penerima

Berdasarkan grafik dapat dilihat bahwa sistem sudah dapat mengikuti set point yang diberikan dengan waktu yang relatif cepat dengan *overshoot* yang sangat kecil dengan osilasi dan *ripple* yang tidak terlalu banyak. Hal tersebut merupakan hal yang ingin dicapai sehingga nilai kombinasi ini dapat dikatakan cukup efektif. Akan tetapi, untuk lebih jelasnya dapat dilihat melalui tabel respon transient motor kedua robot yang digunakan dibawah ini.

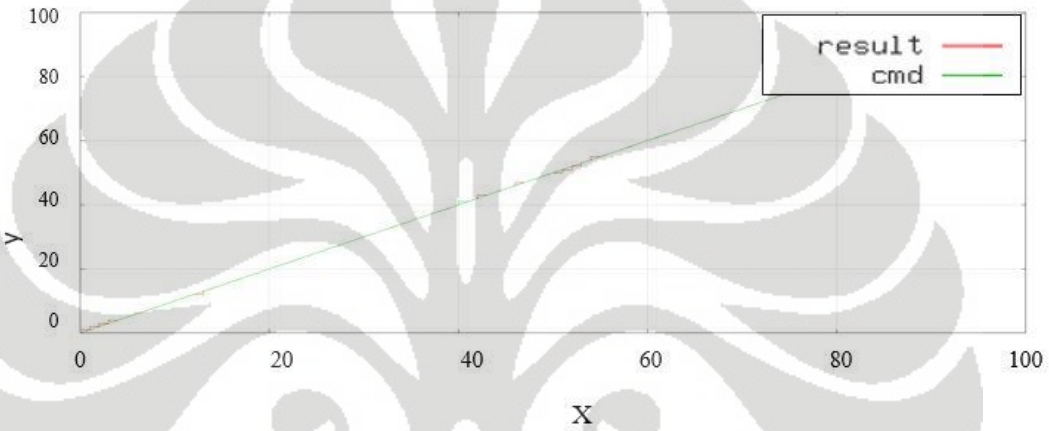
Tabel 4.3 Tabel Respons Transient Motor Kedua Robot yang Digunakan

No	Parameter	Robot Pengirim		Robot Penerima	
		Kiri	Kanan	Kiri	Kanan
1	Overshoot	7.57%	6.70%	5.28%	5.86%
2	Rise Time	0.593 s	0.552 s	0.586 s	0.582 s
3	Settling Time	1.179 s	1.163 s	1.426 s	1.405 s

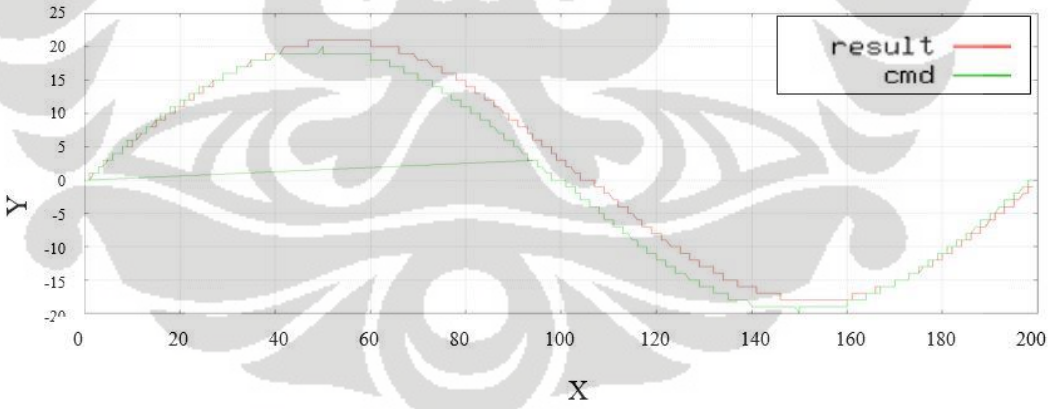
Berdasarkan tabel yang didapatkan dapat dilihat bahwa parameter respon transient yang didapatkan masih dalam batas normal, dimana overshoot yang kurang dari 10% dan juga parameter waktu yang dinilai sudah cukup cepat dengan *rise time* rata-rata bernilai 0.58 s dan *settling time* rata-rata bernilai 1.3 s. Selanjutnya akan dibahas mengenai hasil pengujian pengendali mobile robot dan trayektori yang digunakan.

4.2.2 Nilai Pengendali Posisi Mobile Robot

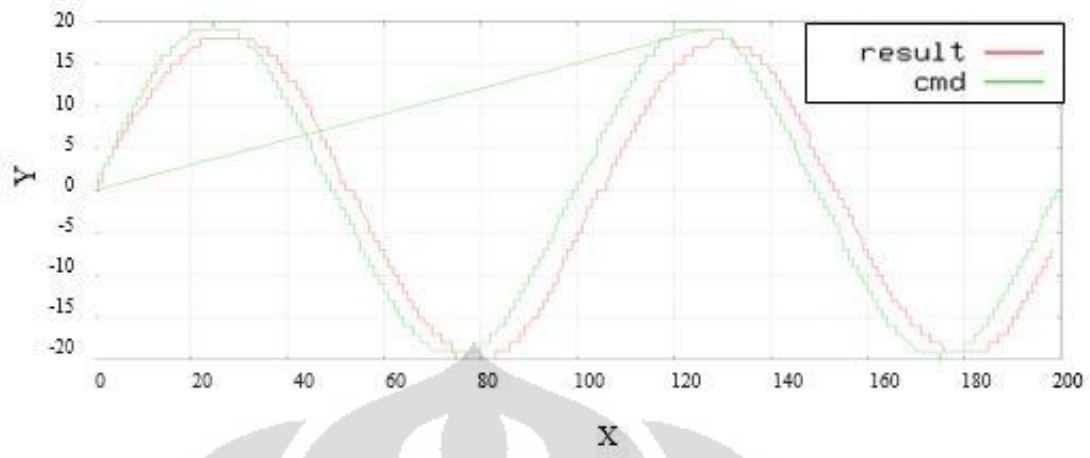
Pada subab ini akan dibahas mengenai pencarian nilai parameter konstanta pengendali posisi dan orientasi *mobile robot*, dimana dalam konteks ini posisi *mobile robot* adalah koordinat kartesian sumbu x dan sumbu y. Berikut merupakan hasil grafik yang diperoleh dari percobaan pengujian konstanta pengendali posisi *mobile robot* dengan nilai pengendali posisi yaitu Kp=1 dan Kv=2 dan pengendali orientasi dengan Kpϕ=25 dan Kvϕ= 10 untuk trayektori linier dengan gradien dan sinusoidal robot pengirim.



(a)



(b)

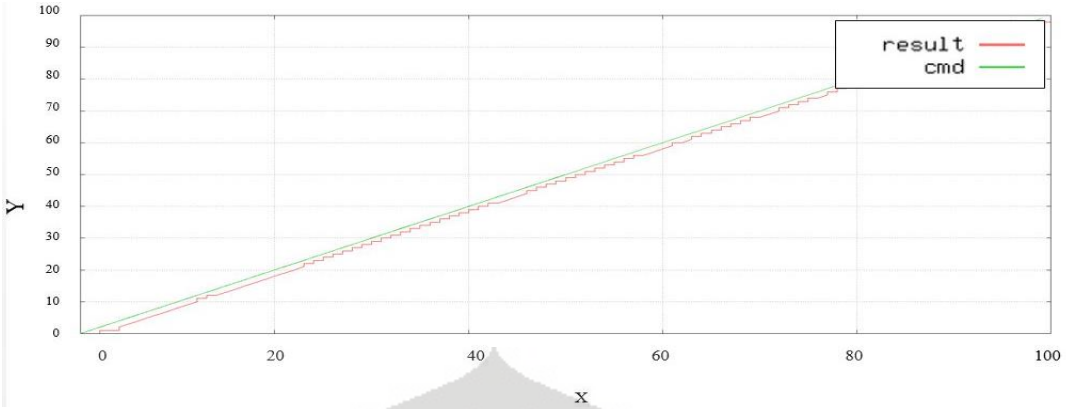


(c)

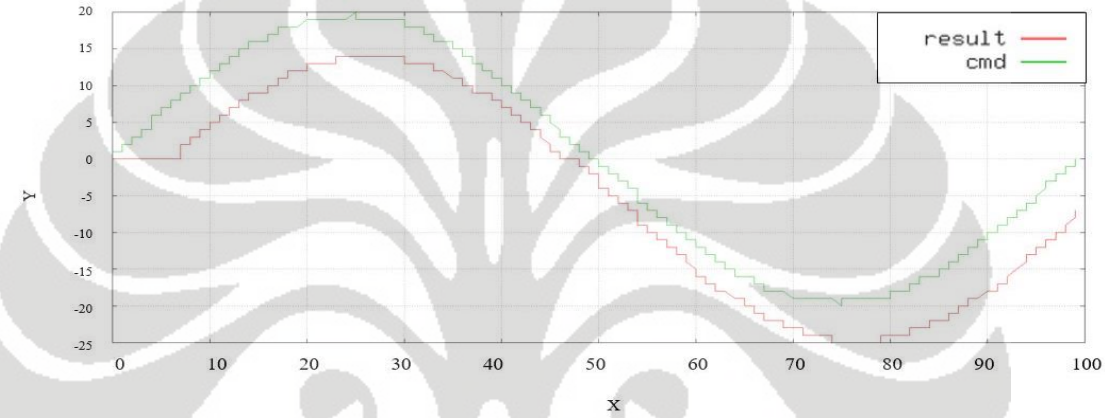
Gambar 4.4 Hasil Pengujian Pengendali Mobile Robot Pengirim dengan Kp=1, Kv=2 Kpϕ=25, dan Kvϕ=10 untuk trajectory (a) linear dengan gradien; (b) sinusoidal dengan ω=2Π; (c) sinusoidal dengan ω=4Π

Berdasarkan grafik di atas dapat dilihat bahwa dengan nilai pengendali tersebut *mobile robot* sudah dapat mengikuti trayektori dengan perubahan orientasi baik untuk trayektori linier dengan gradient dan juga sinusoidal. Pada trayektori linier dapat dilihat bahwa hasil yang didapatkan sudah cukup dekat dengan nilai referensi yang diberikan. Kemudian untuk sinusoidal, untuk kedua frekuensi walaupun tidak dapat seakurat trayektori linier dapat dilihat bahwa *mobile robot* sudah dapat mengikuti trayektori yang diberikan.

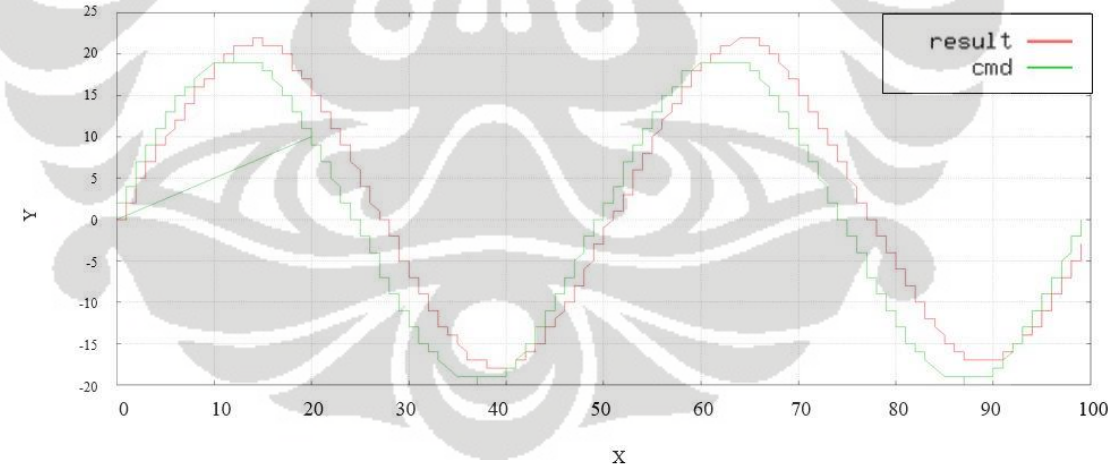
Sebelumnya, sudah dibahas hasil pengujian untuk unit mobile robot pengirim. Kemudian, akan dibahas mengenai pengujian nilai pengendali orientasi mobile robot penerima dengan parameter yang sama. Berikut merupakan hasil pengujian mobile robot penerima untuk trayektori linier dan sinusoidal dengan nilai Kp=2, Kv=1, Kpϕ=25, dan Kvϕ= 10.



(a)



(b)



(c)

Gambar 4.5 Hasil Pengujian Pengendali Mobile Robot Penerima dengan Kp=1, Kv=2 Kpϕ=25, dan Kvϕ=10 untuk trajectory (a) linear dengan gradien; (b) sinusoidal dengan ω=2Л; (c) sinusoidal dengan ω=4Л

Berdasarkan grafik yang diperoleh dapat dilihat bahwa pengendali posisi dengan Kp=2 dan Kv=1 dan pengendali orientasi Kpϕ=25, dan Kvϕ= 10 sudah

dapat memberikan respon yang mendekati set poin yang diberikan untuk grafik linier sepanjang sumbu x sehinga dapat dianalisa bahwa nilai pengendali yang digunakan sudah sesuai, akan tetapi respon yang didapatkan dari mobile robot penerima tidak sebaik respon yang diperoleh dari robot pengirim.

4.3 Implementasi Komunikasi Antar-Unit Mobile Robot

Setelah dilakukan pengujian untuk setiap unitnya, perlu dilakukan pengujian dan implementasi dari komunikasi *internet socket* untuk kedua unit *mobile robot* bertukar informasi. Seperti yang telah dijelaskan sebelumnya pada bab perancangan, komunikasi dilakukan dengan menggunakan model *client-server* dimana *client* menerima informasi dari *server*.

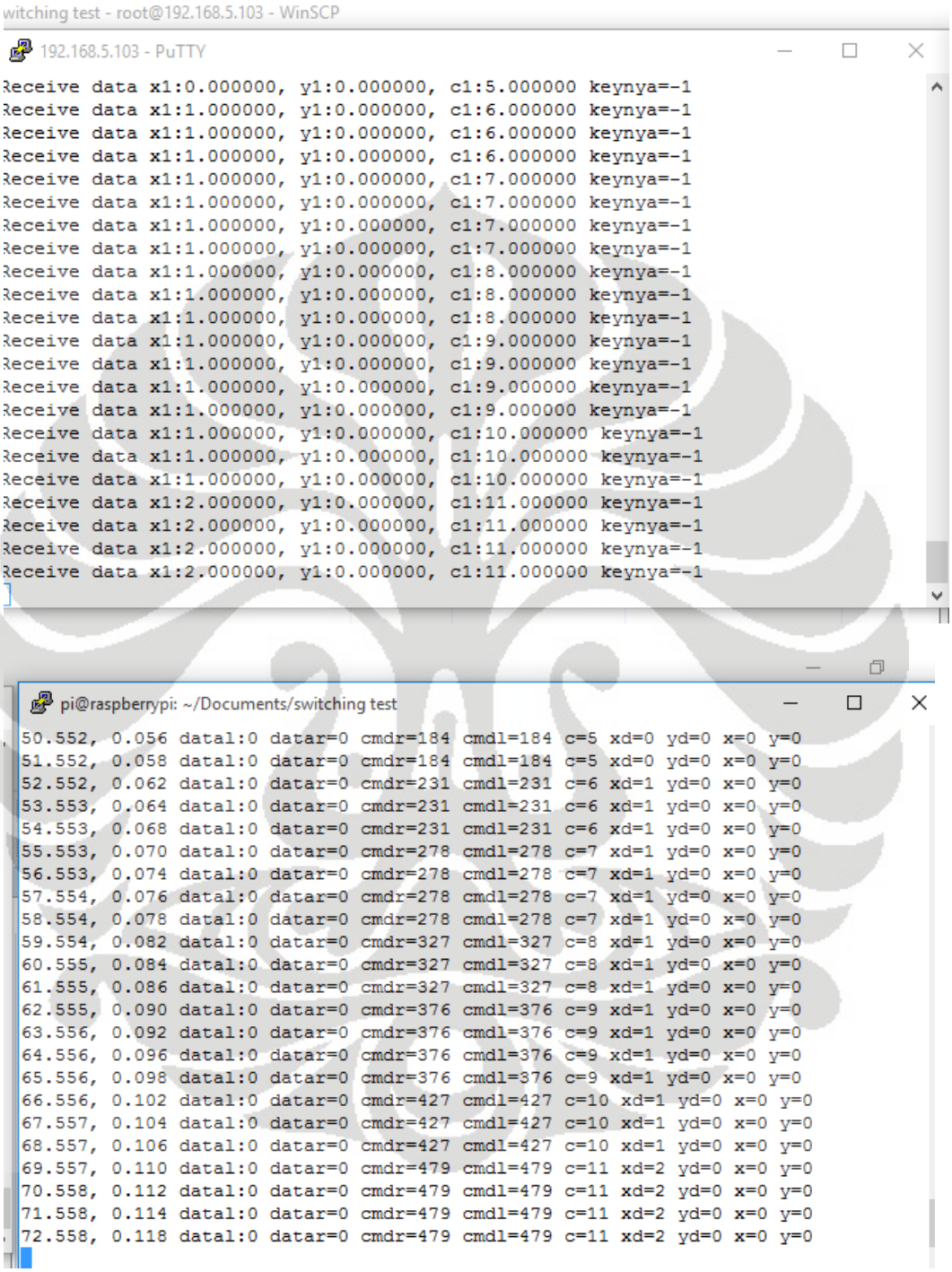
Pada implementasi ini, dilakukan pengiriman data dari robot pengirim ke robot penerima menggunakan *internet socket* dengan data yang dikirim berupa data beberapa data *byte* dengan berbagai tipe data yang digabungkan dengan *struct* untuk dilakukan pengiriman secara bersamaan. Data yang dikirim ditambahkan 1 *byte header* pada awalnya dan dibatasi dengan 1 *byte error checking* di akhir datanya.

Header digunakan untuk memastikan data yang diterima merupakan data yang dikirim dan merupakan penanda dari awal data yang seharusya ditangkap oleh sisi penerima. *Header* pada implementasi ini digunakan *byte* dengan nilai yang jarang digunakan. Sedangkan *error checking* yang ditempelkan pada akhir data digunakan untuk melakukan pengecekan apakah terdapat data yang hilang atau salah ketika pengiriman.

Pada skripsi ini, *error checking* yang digunakan untuk memastikan kebenaran data yang diterima adalah dengan menggunakan *exclusive or (*XOR). Hal ini dapat dilakukan dengan melakukan operasi XOR kepada variable *error checking* untuk semua data yang dikirim pada sisi pengirim dan semua data yang diterima pada sisi penerima.

Kemudian data yang dikirimkan melalui *struct* adalah data posisi dan juga data perintah *keyboard* yang berjumlah 16 *byte* termasuk *header* dan *error*

checking. Berikut merupakan gambar implementasi komunikasi *datagram socket* yang telah dilakukan.



Gambar 4.6 Implementasi Komunikasi Internet Socket Antar Unit Robot

Komunikasi yang sudah dirancang diimplementasikan dalam bentuk Bahasa pemrograman C dan dimasukkan ke dalam mikrokontroler yang digunakan

sehingga kedua robot dapat berkomunikasi. Pengamatan implementasi dapat dilakukan dengan menggunakan applikasi SSH, yaitu PuTTy dengan melakukan koneksi SSH untuk kedua alamat IP dari mobile robot yang digunakan.

Melalui terminal PuTTy, mikrokontroller robot yang digunakan dapat diakses oleh komputer pengguna sehingga hasil yang didapatkan bisa diamati dan dapat segera dihentikan apabila terjadi kesalahan. Data yang dikirim dan yang telah diterima kemudian disimpan setiap pengirimannya ke dalam sebuah file CSV pada setiap robot.

Melalui data yang telah terekam akan dilakukan pengolahan untuk meninjau kemampuan robot berkomunikasi menggunakan *socket*. Kemudian berikut akan dibahas mengenai hasil pengujian dan analisa dari komunikasi antar-unit mobile robot yang telah dilakukan.

4.4 Hasil Pengujian dan Analisa Komunikasi Antar-Unit Mobile Robot

Berdasarkan implementasi komunikasi antar-unit mobile robot yang telah dilakukan didapatkan bahwa komunikasi socket untuk mengirim informasi dari pengirim ke penerima sudah dapat dilakukan. Akan tetapi, pengiriman data melalui *datagram socket* yang dilakukan perlu disesuaikan dengan waktu pengiriman/ penerimaan data melalui serial agar tidak menyebabkan layar menjadi berhenti dikarenakan waktu pengiriman yang tidak sesuai antara data serial, data *socket*, dan data yang akan dicetak di layar.

.Melalui data yang telah direkam, dapat diolah sehingga diketahui parameter pengiriman data sampai tiba ke sisi robot penerima. Selain itu, juga dapat dilakukan validasi antara data yang telah dikirim oleh robot pengirim terhadap data yang diterima oleh robot penerima. Berikut merupakan tabel ringkasan hasil parameter berdasarkan data yang telah diolah.

Tabel 4.4 Tabel Hasil Pengujian Parameter Pengiriman Data

No	Parameter	Hasil Pengujian
1	Total paket yang dikirim	258
2	Total paket yang diterima	328
3	Jumlah paket yang hilang	0
4	Jumlah paket terduplikasi	70
5	Jumlah data salah	0

Berdasarkan data yang telah diperoleh dapat dilihat bahwa permasalahan dalam pengiriman melalui protokol UDP adalah ketidakandalan data yang diterima, sehingga data yang didapatkan dapat hilang ataupun terduplikasi. Pengujian mendapatkan hasil tidak ada paket yang hilang tetapi banyak paket yang terduplikasi.

Hal ini disebabkan karena *switch* akan mengirim paket ke semua interface ketika menggunakan *broadcast* seperti yang digunakan pada aplikasi skripsi ini. Duplikasi terjadi akibat terdapat pengulangan diantara *switch* sehingga data yang diterima menjadi terduplikasi. Akan tetapi, pada aplikasi ini duplikasi tidak terlalu berpengaruh langsung terhadap perintah yang akan diterima oleh robot penerima.

Berdasarkan kebutuhan dalam aplikasi *mobile robot* komunikatif, dibutuhkan waktu pengiriman yang cepat dengan paket yang terkirim semua. Oleh sebab itu, parameter sesuai dan berhasilnya komunikasi antar robot melalui *socket datagram* akan lebih dititik beratkan terhadap error dan jumlah paket yang hilang beserta waktu pengiriman rata-rata.

Berdasarkan data pada tabel 4.4 dapat dilihat bahwa tidak ada paket yang hilang pada saat pengiriman dan juga nilai paket yang diterima tidak mempunyai kesalahan terhadap nilai yang sebenarnya dikirim. Akan tetapi, untuk parameter waktu pengiriman yang didapatkan masih dikatakan bahwa nilai yang didapatkan ketika pengujian ini, yaitu 0.542s yang didapatkan masih tergolong tidak valid dikarenakan tidak sinkronnya waktu pemulaian program pada kedua robot yang menyebabkan ketidaksinkronan terhadap waktu yang didapatkan dalam eksekusi program.

Oleh karena itu, perlu dilakukan pengukuran ulang dengan waktu yang telah disinkronisasi antar kedua mikrokontroler. Sedangkan, untuk sinkronisasi waktu dapat dilakukan dengan menggunakan GPS atau dengan menghubungkan kedua mikrokontroler ke internet dengan menggunakan server FTP tunggal. Kedua metode sinkronisasi tersebut dinilai tidak dapat dilakukan dikarenakan keterbatasan sumber. Akan tetapi, secara keseluruhan komunikasi dengan menggunakan *internet socket* dapat berjalan dengan baik.

BAB 5

KESIMPULAN

Berdasarkan pengujian yang dilakukan pada *Mobile Robot* Komunikatif, dapat diambil kesimpulan, yaitu:

- Hasil pengendalian masing-masing unit *mobile robot* menunjukkan bahwa *mobile robot* mampu mengikuti trayektori linier dan sinusoidal
- Parameter pengendali motor yang digunakan untuk kedua unit, yaitu Kp_{motor}= 6 , Kv_{motor}=4.9, dengan *Gain Low-Pass Filter* =75, dinilai sudah mampu mengendalikan motor sehingga mencapai respon yang diinginkan, yaitu cepat dengan *overshoot* yang kecil dengan data yang valid.
- Parameter pengendali *mobile robot* yang digunakan untuk kedua unit, yaitu Kp_{pos}=2, Kv_{pos}=1, $Kp\phi$=25, dan $Kv\phi$= 10 dengan *Gain Low-Pass Filter*=75, sudah cukup baik untuk aplikasi trayektori linier dan sinusoidal
- Komunikasi menggunakan *internet socket* antar kedua robot sudah dapat dilakukan dengan presentase kesalahan 0% tetapi masih mempunyai waktu pengiriman yang ambigu terkait permasalahan sinkronisasi kedua mikrokontroler yang digunakan.

DAFTAR ACUAN

[1] Yu. N. Zolotukhin, K. Yu Kotov, A. S, Maltsev, A.A Nesterov, M.A Sobolev, M. N Filippov. 2015. A Relative Measurement based Leader-follower Formation Control of Mobile Robots.Nobosibirsk, Russia.

[2] Muis, Abdul. 2016. *Motion Control Slides*. Universitas Indonesia. Depok, Indonesia.

[3] Dixon, Jonathan. 1997. *Mobile Robot Navigation*. Imperial College London. London, United Kingdom.

[4] E. Stella, F.P. Lovergine, L. Caponetti, and A. Distante, "*Mobile robot navigation using vision and odometry*," di Proceedings of the Intelligent Vehicles '94 Symposium, October 1994, pp. 417–422

[5] Research Gate. *Control Loop Hardware*. Tersedia pada : https://www.researchgate.net/file.PostFileLoader.html?id=5895ce7f217e20e93b3b3406&assetKey=AS%3A458023563993092%401486212735716. [diakses tanggal 19 Mei 2017]

[6] Sommerville, Ian. 2011. *Software Engineering : 9th Edition*. Pearson Education, Inc. Boston. Massachussets.

[7] Arduino. "*Timer1: attachedInterrupt(function,period)"*.Arduino playground. Tersedia pada: https://playground.arduino.cc/Code/Timer1 [diakses tanggal 19 Mei 2017]

[8] National Instrument. 2017. *Encoder Measurement: How to Guide*. Tersedia pada: http://www.ni.com/tutorial/7109/en/ [diakses tanggal 19 Mei 2017]

[9] Agillent Technologies. "Quadrature Decoder / Counter Interface ICs" HCTL-2032 Datasheet. 2007.

[10] Mitton, Natalie. 2014. Wireless Sensor and Robot Networks : From Topology Control to Communication Aspects. World Scientific. Perancis

[11] Cao, Y., Fukunaga, A. and Kahng, A. (1997). Cooperative mobile robotics: Antecedents and directions, Autonomous robots 4(1), pp. 7{27.

[12] R. E. Bryant dan D. R. O'Hallaron, *Computer Systems: A Programmer's Perspective*, 2 edition. Boston: Pearson, 2010.

[13] B. Hall, *Beej's Guide to Network Programming*. Jorgensen Publishing.

[14] I. Fette, A. Melnikov. 2011. *The Websocket Protocol*. Internet Engineering Task Force (IETF). Tersedia pada : https://tools.ietf.org/html/rfc6455. [diakses tanggal 2 Juni 2017]