IMPLEMENTASI *REAL TIME* PADA PERGERAKAN ROBOT *QUADRUPED* MENGGGUNAKAN MULTISENSOR DAN RTOS

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Teknik

Disusun oleh:
Hendriawan Dwi Saputro
NIM: 145150300111110

PROGRAM STUDI TEKNIK KOMPUTER
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2018

PENGESAHAN

IMPLEMENTASI *REAL TIME* PADA PERGERAKAN ROBOT
QUADRUPED MENGGGUNAKAN MULTISENSOR DAN RTOS

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Teknik

Disusun Oleh :
Hendriawan Dwi Saputro
NIM: 145150300111110

Skripsi ini telah diuji dan dinyatakan lulus pada
01 Agustus 2018
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Rizal Maulana , S.T., M.T., M.Sc.	Mochammad Hannats Hanafi Ichsan, S.ST, M.T
NIK: 201607 891009 1 001	NIK: 201405 881229 1 001

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T, M.T, Ph.D
NIP: 19710518 200312 1 001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 01 Agustus 2018

Hendriawan Dwi Saputro

NIM: 145150300111110

KATA PENGANTAR

Alhamdulillahirabbil'alamin, Puji syukur kehadirat Allah SWT, karena atas kehendak dan rahmat-Nya lah penulis dapat menyelesaikan skripsinya dengan judul "**IMPLEMENTASI *REAL TIME* PADA PERGERAKAN ROBOT *QUADRUPED* MENGGUNAKAN MULTISENSOR DAN RTOS**". Penulis menyadari bahwa pembuatan skripsi ini tidak lepas dari bantuan baik motivasi, doa dan jasa dari berbagai pihak. Maka sebab itu, penulis menyampaikan rasa hormat dan terimakasih kepada:

1. Kedua orang tua penulis, Bapak Suparman dan Ibu Siswi Fajarini yang selalu memberikan doa, motivasi, fasilitas dan semangat sehingga penulis dapat menyelesaikan skripsinya.
2. Bapak Rizal Maulana, S.T, M.T., M.Sc selaku dosen pembimbing pertama yang telah memberikan dukungan dan bimbingannya kepada penulis untuk segera menyelesaikan skripsi ini.
3. Bapak Mochammad Hannats Hanafi Ichsan, S.ST, M.T selaku dosen pembing kedua yang telah memberikan dukungan dalam menyelesaikan laporan skripsi ini.
4. Bapak Tri Astoto Kurniawan, S.T, M.T, Ph.D selaku Ketua Jurusan Teknik Informatika.
5. Bapak Sabriansyah Rizqika S.T., M.Eng selaku ketua Program Studi Teknik Komputer yang telah memberikan dukungan dan motivasi penulis.
6. Bapak Ibu Dosen Program Studi Teknik Komputer yang telah memberikan ilmu selama berkuliah.
7. Saudari penulis Diesna Anggraeni Partiwi dan keluarga yang telah membantu penulis dalam perkuliahan.
8. Teman-teman RoboTIIK Imam Ghozali, Agastya Bramanta, Yusuf Hidayat, Tezza Rangga, Hafizhudin, Yoga Sukma, Haqqi Rizqi, Andrika Wahyu, Dienastya Galih serta semua teman-teman RoboTIIK angkatan 2011 hingga angkatan 2017 yang tidak dapat disebutkan satu persatu.
9. Teman-teman sedulur TEKKOM E dan semua mahasiswa program studi Teknik Komputer angkatan 2014 yang selalu memberikan semangat.
10. Teman-teman belajar bersama Oggy Setiawan dan Rafi Dharmawan yang telah memberikan dukungan dan bantuan.
11. Soni Wicaksono, Ayodya Damas, teman-teman kos Halimah Castille, HalloLaptop_mlg, dan *Farmstation* yang telah memberi ilmu dan motivasi untuk penulis.
12. Seluruh pihak yang selalu mendukung dan mendoakan penulis yang tidak dapat disebutkan satu persatu.

Penulis menyadari bahwa dalam penyelesaian skripsi ini masih banyak kekurangan, oleh sebab itu penulis berharap adanya pengembangan lebih lanjut oleh pihak-pihak terkait. Semoga laporan skripsi ini dapat memberikan manfaat dan referensi untuk melakukan penelitian sebagai langkah untuk penyempurnaan sistem.

Malang, 01 Agustus 2018

Penulis

Hendriawan.saputro@gmail.com

ABSTRAK

Perkembangan teknologi dalam dunia robotika terus mengalami perkembangan pesat, salah satunya pengembangan sistem kontrol robot seperti peningkatan kecepatan eksekusi program maupun performa prosesor yang semakin baik memungkinkan pemakaian *Real Time Operating System* (RTOS) pada *embedded system*. Peningkatan performa dibutuhkan oleh robot agar robot dapat memenuhi target yaitu ketepatan dalam bergerak dan kecepatan waktu eksekusi. Dengan perkembangan penerapan RTOS dalam penelitian ini akan mengimplementasikan RTOS pada *embedded system* yaitu pergerakan robot *quadruped*. RTOS yang digunakan dalam penelitian ini ialah *FreeRTOS* karena sistem tersebut sudah tersedia dalam mikrokontroler Arduino dan untuk tipe RTOS ialah *soft realtime operating system* yang memberi toleransi ketika robot tidak sesuai waktu eksekusi. RTOS dalam sistem akan diterapkan pada pengambilan data sensor dan proses pengolahan data untuk mengambil keputusan. Robot dalam penelitian ini memiliki alat gerak berjumlah empat kaki atau bisa disebut robot *quadruped* yang tersusun dari komponen HC-SR04, Arduino Mega 2560, OpenCM9.04 dan OpenCM485 dan servo AX-12. Dari hasil penelitian robot memiliki ketepatan dalam bergerak yang cukup bagus ketika menerapkan *Real Time Operating System*.

Kata kunci: robot *quadruped*, *real time operating system*, *free*RTOS, embedded *system*

ABSTRACT

Technological developments in robotics experience rapid development continuously, one of them the development of robot control systems such as increased program execution speed and better processor performance allows the use of Real Time Operating System (RTOS) in embedded systems. Improved performance is required by the robot so that the robot can reach the target of precision in moving and speed time execution. By the development of RTOS application in this research, it will apply RTOS in embedded system that is movement of quadruped robot. The RTOS used in this study is FreeRTOS because the system is already available in the Arduino microcontroller. For the RTOS type, it is a soft realtime operating system which tolerates when the robot does not match the execution time. RTOS in the system will be applied to the data retrieval of sensors and data processing for decision making. The robot in this study has a four-foot motion tool or can be called a quadruped robot composed of components HC-SR04, Arduino Mega 2560, OpenCM9.04 and OpenCM485 and servo AX-12. From the results of the study, robots have pretty good precision in moving when applying Real Time Operating System.

Keywords: *quadruped robot, real time operating system, freeRTOS,* embedded *system*

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB 1 PENDAHULUAN

1.1 Latar belakang

Perkembangan teknologi tidak dapat dihindari lagi dalam kehidupan ini. Perkembangan teknologi yang paling pesat salah satunya ialah teknologi dalam bidang robotika. Robot bukan hanya sebuah mainan atau karya yang canggih, namun robot adalah sebuah penerapan dari konsep ilmu pengetahuan (Kusuma, 2011). Banyak jenis robot yang ada diantaranya yang sering kita temui ialah robot manipulator dan *mobile* robot. Robot manipulator merupakan robot yang terdiri dari sendi (*joint*) dan terhubung dengan lengan (*link*), robot manipulator banyak digunakan pada industri. Sedangkan *mobile* robot adalah robot yang dapat bergerak dengan leluasa dan dapat berpindah posisi.

Mobile robot merupakan robot yang memiliki ciri khas mempunyai aktuator berupa roda atau kaki untuk menggerakkan keseluruhan badan robot, sehingga robot tersebut dapat melakukan gerak perpindahan posisi dari satu titik ke titik yang lain (Ermadi, 2009). Robot berkaki adalah robot yang bermanuver dengan kaki-kaki buatan, baik dengan 2 kaki yang sering disebut dengan robot *humanoid*, berkaki tiga (*tripod*), berkaki empat (*quadruped*), robot berkaki enam (*hexapod*) dan robot berkaki banyak lainnya. Robot beroda sering diaplikasikan karena memiliki kecepatan tinggi dalam melintasi bidang yang rata, namun dalam dunia nyata bidang atau daerah yang akan sering dilintasi oleh robot ialah bidang yang tidak rata. Dengan permasalahan tersebut maka robot berkaki memiliki kelebihan dibanding dengan robot beroda dalam hal kemampuanya untuk melewati daerah tidak rata (Wicaksono, et al., 2008). Robot berkaki juga memiliki target untuk kalibrasi salah satunya ialah ketepatan (Conrad & Shiakolas, 2000), Ketepatan sendiri menurut *National Institute of Institute Standard and Technology* adalah ukuran perbedaan antara posisi/jalan yang benar-benar dicapai dari robot dari nilai sebenarnya, jika robot memiliki nilai ketepaan yang rendah maka robot tidak berjalan dengan baik bahkan dapat menimbulkan masalah, maka untuk mendukung target tersebut dan meningkatkan hasil ketepatan robot dalam penelitian ini akan dicoba menggunakan sistem operasi yaitu *Real Time Operating System* (RTOS).

Sistem Operasi adalah kumpulan program untuk mengontrol perangkat keras pada komputer (Atmadja , et al., 2014), contoh sistem operasi seperti linux, *windows*, dan mac os merupakan sistem operasi GPOS. *General Purpose Operating System* (GPOS) sering digunakan untuk aplikasi sehari-hari yang tidak memerlukan kebutuhan khusus karena pada GPOS tidak ada kemampuan untuk memperoleh perilaku deterministik atau waktu kristis, sedangkan RTOS merupakan sistem operasi yang dapat memproses berbagai tugas dengan tingkat prioritas yang dapat disesuaikan, menghasilkan hasil yang dapat diprediksi, dan tepat saat menjalankan tugas kritis. RTOS memiliki respons sistem yang lebih baik daripada GPOS, respons sistem ini penting untuk *mobile* robot, seperti *mobile* robot harus berhenti sesegera mungkin saat sensor mendeteksi dinding (Atmadja , et al., 2014).

Real time sering digunakan dalam simulasi, istilah *real time* sering dimengerti sebagai cepat, namun sebenarnya yang dimaksud adalah sistem yang bisa menyamai proses sebenarnya di dunia nyata (Kurniawan, 2011). Sistem waktu nyata atau lebih dikenal dengan *real time system* adalah sistem yang harus menghasilkan respons yang tepat dalam batas waktu yang telah ditentukan. Terdapat dua model sistem *real time*, yaitu *hard real time system* dan *soft real time system* (Kurniawan, 2011). Sistem RTOS dalam perkembangan sekarang terdapat 128 RTOS dengan target penggunaan, dan implementasi perangkat yang berbeda diantaranya *Abassi*, *ChibiOS*, *Free*RTOS, *Micrium*. Implementasi dalam sistem ini akan digunakan dalam penelitian ini adalah *Free*RTOS dengan model *soft real time system* karena bersifat *open source* dan dapat diimplementasikan pada *embedded system*. *Free*RTOS adalah sistem preemptif atau kooperatif, yang menyediakan penjadwalan tugas yang dinamis (prioritas tertinggi pertama, untuk tugas dengan prioritas yang berbeda dan *round robin* di antara tugas dengan prioritas yang sama) (Cristina, 2014).

Penelitian ini bertujuan untuk mengimplementasikan sistem RTOS pada mikrokontroler ATMEGA 328 atau *board* arduino MEGA 2560 untuk respons gerak robot berkaki empat (*quadruped)* terhadap input merupakan nilai jarak yang dihasilkan oleh sensor *ultrasonic.* Robot *quadruped* dalam penelitian ini merupakan *mobile* robot dengan pergerakan menggunakan kaki yang berjumlah empat dan disetiap kaki terdiri dari 3 *Degree of Freedom* (DOF). Sensor *ultrasonic* dalam penelitian ini menggunakan sensor HC-SR04, dengan hasil jarak yang dihasilkan oleh sensor *ultrasonic* dan sistem *multi*-sensor yang berjalan secara simultan nanti akan menghasilkan nilai yang menjadi *input* pada proses penentuan langkah berikutnya dengan teknik enkoding digital, diharapkan penelitian ini dapat memberikan solusi untuk membuat respons gerak robot berkaki lebih cepat dan tepat.

1.2 Rumusan masalah

Berdasarkan latar belakang di atas, maka rumusan masalah pada penelitian ini antara lain:

1. Bagaimana merancang *Real Time Operating System* (RTOS) pada pergerakan robot empat kaki *(quadruped)*?
2. Bagaimana mengimplementasikan *Real Time Operating System* (RTOS) pada pergerakan robot empat kaki *(quadruped)* ?
3. Bagaimana hasil yang didapatkan untuk penerapan *Real Time Operating System* (RTOS) pada pergerakan robot *quadruped* ?

1.3 Tujuan

Berikut merupakan tujuan penelitian, antara lain :

1. Merancang *Real Time Operating System* (RTOS) pada pergerakan robot empat kaki *(quadruped)*.

2. Mengimplementasikan *Real Time Operating System* (RTOS) pada pergerakan robot empat kaki *(quadruped)*.
3. Mengetahui hasil dari penerapan *Real Time Operating System* (RTOS) pada pergerakan robot *quadruped.*

1.4 Manfaat

Manfaat yang ingin dicapai dalam penelitian ini ialah memberikan gambaran implementasi *Real Time Operating System* (RTOS) pada Arduino MEGA 2560 sehingga dapat memenuhi target robot *quadruped* yaitu sebagai *mobile robot* yang memiliki respons atau reaksi cepat terhadap *input* dan *output*.

1.5 Batasan masalah

Untuk membuat pembahasan dalam penelitian lebih terarah maka dalam penelitian implementasi perancangan pergerakan robot empat kaki *(quadruped)* secara *real time* terdapat batasan masalah. Batasan-batasan permasalahan lain, antara lain :

4. Robot memiliki 8 gerakan (maju, mundur, kanan, kiri, serong kanan depan, serong kiri depan, serong kanan belakang, serong kiri belakang).
5. Impementasi *Real Time Operating System* (RTOS) diterapkan pada mikrokontroler arduino.
6. Robot menggunakan empat kaki dimana setiap kaki terdiri dari 3 *Degree of Freedom* (DOF).

1.6 Sistematika pembahasan

Uraian singkat mengenai metodologi penelitian ini dibagi atas beberapa bab :

BAB I : **Pendahuluan**

Menjelaskan tentang latar belakang, rumusan masalah, tujuan, manfaat, batasan masalah dan sistematika pembahasan dari "Implementasi *Real Time* pada Pergerakan Robot *Quadruped* Menggunakan Multisensor dan RTOS".

BAB II : **Landasan Kepustakaan**

Pada bab ini akan memuat landasan teori dan pustaka yang terkait dengan penelitian yang dilakukan. Teori dan pustaka diambil dari buku literatur dan teori yang dibahas tentang penelitian *Real Time Operating System* (RTOS).

BAB III : **Metode Penelitian**

Membahas tentang langkah kerja yang dilakukan dalam penulisan diantaranya studi literatur, analisis kebutuhan sistem, desain sistem, dan implementasi *Real Time Operating System* (RTOS) dalam pergerakan kaki robot *quadruped* dengan *Free*RTOS.

BAB IV : **Perancangan**

Bab ini menjelaskan perancangan sistem penelitian yaitu implementasi *Real Time Operating System* (RTOS). Akhir dari perancangan ini akan mendapat hasil sesuai dengan yang diharapkan.

BAB V : Implementasi

Pada bab ini membahas tentang implementasi *Real Time Operating System* (RTOS) pada pergerakan kaki robot *quadruped*. Implementasi ini dilakukan sesuai dengan hasil analisa dan perancangan pada bab sebelumnya.

BAB VI : Pengujian dan Analisis

Membahas pengujian dan analisis berdasarkan perancangan dan implementasi. Pengujian ini dilakukan dengan menggunakan parameter keberhasilan respons pada gerakan robot.

BAB VII : Kesimpulan dan Saran

Membuat kesimpulan yang diperoleh dari hasil pengujian dan analisis berdasarkan perancangan dan implementasi serta saran-saran untuk pengembangan lebih lanjut mengenai *robot quadruped*.

BAB 2 LANDASAN KEPUSTAKAAN

2.1 Tinjauan Pustaka

Tinjauan pustaka membahas mengenai peneltian sebelumnya dengan beberapa objek dan metode yang berbeda, objek serta metode akan menjadi acuan dalam penelitian yang akan dilakukan. Pada penelitian ini pembahasan terpusat pada *Real Time Operating System* (RTOS).

Penelitian "Rancang Bangun Sistem Multi-Sensor Untuk Pengukuran Jarak Secara Simultan" yang dilakukan oleh Wahyudin (2018) merupakan penelitian sebagai kajian utama dalam penelitian ini. Di dalam penelitian tersebut membahas pembacaan sensor *ultrasonic* secara simultan dengan metode RTOS, untuk hasil dari penelitian tersebut sangatlah baik karena sensor *ultrasonic* dapat melakukan pembacaannya sangatlah cepat dan sesuai dengan target penggunaan RTOS. Akan tetapi dalam penelitian tersebut belumlah memiliki *output* yang sangat terlihat jelas, *output* penelitian tersebut menggunakan LED sebagai indikator dan pembacaan waktu melalui serial monitor Arduino pada laptop. Sehingga dalam penelitian ini akan menambah sistem yaitu indikator akan diganti dengan *actuator* yaitu kaki robot *quadruped*.

Penelitian oleh Atmadja et al. (2014) dengan judul "*Real Time Operating System on Embedded Linux with Ultrasonic Sensor for Mobile Robot*" dalam penelitian tersebut membahas penggunaan *Real Time Operating System* dengan menggunakan sensor *ultrasonic* dan membandingkan fungsi *Real Time Operating System* (RTOS) dengan *General Operating System* (GPOS). Dalam penelitian tersebut mengahasilkan bahawa *Real Time Operating System* (RTOS) sangatlah bermanfaat ketika di dalam sistem terdapat kondisi khusus. Di dalam penelitian ini juga sistem masih belum menerapkan pada *actuator* atau masih belum mengimplentasikan secara penuh pada objek. Maka pada penelitian "Implementasi *Real Time* pada Pergerakan Robot *Quadruped* Menggunakan Multisensor dan RTOS" akan mengimplementasikan secara penuh *Real Time Operating System* (RTOS) dengan menerapkannya pada robot berkaki *quadruped*.

Setiawan dan Utomo (2011) melakukan penelitian mengenai penerapan *inverse kinematic* menggunakan RTOS pada robot *quadruped*, dalam penelitian tersebut memiliki perbedaan pada implementasi RTOS yang diterapkan untuk pergeakkan inverse kinematic, dan memilik persamaan yaitu susunan kaki robot *quadruped* Beberapa penelitian yang terkait dapat dilihat pada tabel 2.1.

Tabel 2. 1 Kajian Pustaka

No.	Judul	Persamaan	Perbedaan	
			Penelitian terdahulu	**Rencana Penelitian**
1.	Rancang Bangun Sistem Multi-Sensor	Membangun sistem multi	*Output* yang dikeluarkan	*Output* dapat dilihat lebih

No.	Judul	Persamaan	Perbedaan	
			Penelitian terdahulu	**Rencana Penelitian**
	Untuk Pengukuran Jarak Secara Simultan (Wahyudin, 2018)	sensor secara simultan	dalam penelitian masih berupa indikator	jelas karena diterapkan pada robot yang memiliki aktuator
2.	*Multitasking* Pada Mikrokontroler Atmega16 Menggunakan *Real Time Operating System* (RTOS) Jenis *Cooperative* (Kurniawan, 2011)	Membangun sistem dengan menggunakan RTOS	Menggunakan CocoOS sebagai sistem RTOS	Menggunakan *Free*RTOS sebagai sistem RTOS
3.	*Real Time Operating System on Embedded Linux with Ultrasonic Sensor for Mobile Robot* (Atmadja , et al., 2014)	Membangun sistem pada robot dengan menggunakan RTOS	Robot yang digunakan ialah robot beroda	Robot yang digunakan ialah robot berkaki
4.	*Inverse Kinematic* dengan *Real Time Operating System* pada *robot Quadruped* (Setiawan & Utomo, 2011)	Membangun robot berkaki 4 *(quadruped)* dan menggunakan RTOS	RTOS digunakan untuk *inverse kinematic gerakkan robot*	RTOS digunakan untuk pengambilan data sensor dan proses keputusan selanjutnya
5.	*Hard Real-Time Execution Environment Extension for FreeRTOS* (Cristina, 2014)	Membangun sistem *real time* dengan tipe *hard real time*	Sistem menggunakan *hard real time*	Sistem menggunakan *soft real time*

2.2 Dasar Teori

2.2.1 Real Time Operating System

Real Time Operating System (RTOS) adalah sistem operasi yang memproses berbagai tugas dengan tingkat prioritas yang dapat disesuaikan dan menghasilkan hasil yang dapat diprediksi serta tepat saat menjalankan tugas penting (Atmadja , et al., 2014). *Real time Operating System* menjalankan atau mengeksekusi

program-program dalam sebuah pola yang teratur. RTOS bukan merupakan sistem operasi seperti umumnya yang berkerja pada komputer. Sistem operasi pada komputer akan bekerja sesaat setelah *power* masuk ke komputer, kemudian program komputer akan berjalan. Sedangkan RTOS dijalankan oleh sebuah program dengan cara otomatis, program tersebut merupakan sebuah kernel (Jatmiko, et al., 2015).

RTOS menurut Reddy (2006) diklasifikasikan dalam 3 tipe yaitu *Hard Real Time* RTOS, *Firm Real Time* RTOS, dan *Soft Real Time* RTOS. Berikut ialah penjelasan masing-masing dari tipe RTOS:

a. *Hard Real Time*: kehilangan tenggat waktu suatu task sangat kecil, idealnya adalah nol, karena ketidak sesuaian waktu yang kecil saja dapat menyebabkan bencana atau kerusakan parah pada sistem. Sebagai contoh sistem otomatis kendali pada mobil, sistem otomatisasi rudal, dan roket.
b. *Firm Real Time*: Kehilangan tenggat waktu menyebabkan berkurangnya kualitas yang tidak dapat diterima. sebagai contoh suara putus-putus pada video dan mp3 player.
c. *Soft Real Time*: tenggat waktu yang hilang dapat di kembalikan dan dapat diterima. Terlambatnya memiliki cukup level prioritas yang dibutuhkan, terutama saat penggunaan prioritas scheduling. Sebagai contoh sistem penjualan otomatis.

Dalam sisitem yang akan dibuat ini menggunakan *soft real time* karena sistem robot membutuhakan gerak secara terus-menurus dan waktu eksekusi setiap *task* dapat mentoleransi kelebihan waktu eksekusi.

2.2.2 Konsep Dasar *Real Time Operating System* (RTOS)

Real Time Operating System (RTOS) memiliki beberapa konsep dasar diantaranya ialah *multitasking* dan *real time application*. Berikut merupakan penjelasan dari konsep-konsep dasar RTOS.

2.2.2.1 *Multitasking*

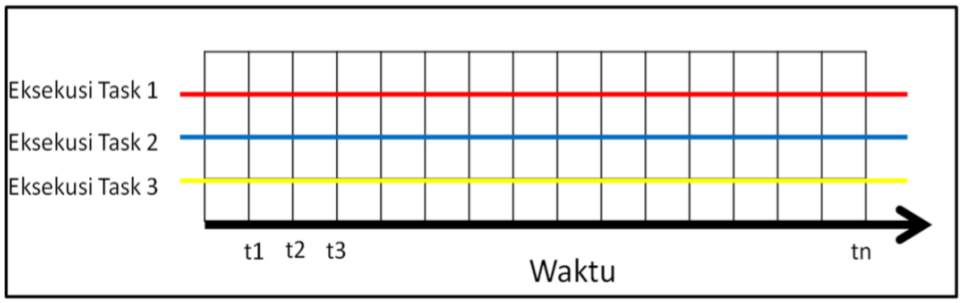
Sistem operasi memiliki komponen utama untuk mengatur adanya *multitasking. Multitasking* sendiri adalah kemampuan yang dimiliki oleh sistem operasi unutk menjalankan banyak *task* atau *thread* dalam rentang waktu tertentu. Setiap *task* yang ada dibentuk dari setiap proses yang harus dijalankan oleh sebuah komputer ataupun *embedded system.*

Konsep *multitasking* dari sistem operasi akan memudahkan dalam desain sebuah aplikasi atau sistem yang kompleks. Terdapat beberapa keuntungan dari penerapan *multitasking* sebagai berikut:

a. Memungkinkan sebuah aplikasi atau sistem yang kompleks dibagi menjadi bagian lebih kecil, sederhana, dan mudah diatur.
b. Bagian-bagian kecil dari sebuah aplikasi dapat dengan mudah diuji coba, diatur ulang, dan ditelusuri untuk kebutuhan lain.

c. Detail alur dari aplikasi atau sistem serta pengaturan waktu eksekusi yang kompleks dapat dihilangkan dari kode aplikasi. Hal tersebut menjadi tanggung jawab dari sistem operasi itu sendiri.

Gambar 2.1 menunjukkan bagaimana konsep dari *multitasking* bekerja. Tiga *task* akan berjalan delam rentang waktu sama dan dengan waktu eksekusi sama. Akan tetapi pada prosesor konvensional, konsep *multitasking* belum dapat diterapkan. Keterbatasan sumber daya prosesor dan memori membatasi penerapan *multitasking.* Prosesor belum memungkinkan mengeksekusi lebih dari satu *task* dalam waktu bersamaan.

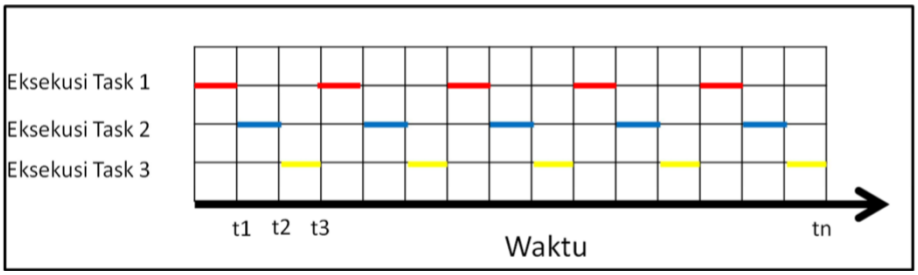


Gambar 2. 1 Konsep *Multitasking*

Sumber : (Jatmiko, et al., 2015)

Namun *task-task* yang ada pada aplikasi secara bergantian dapat dijalankan. Hal tersebut dinamakan dengan konsep konkurensi. Konsep konkurensi digambarkan pada Gambar 2.2. Sistem operasi sudah mengetahui *task* apa saja yang akan dijalankan dalam rentang waktu tertentu. Selanjutnya sistem operasi akan membagi rentang waktu yang ada untuk mengeksekusi beberapa *task*. Pada Gambar 2.2 hanya task 1 yang akan dieksekusi terlebih dahulu pada rentang waktu t0 hingga t1, kemudian t selanjutnya hanya task 2 yang dieksekusi. Pada satu waktu hanya satu *task* yang dieksekusi. Setiap *task* yang ada akan dieksekusi dalam rentang waktu yang kecil.

Sistem operasi mengatur kapan ketika waktu eksekusi setiap *task* dan menyimpan *state*/posisi dari setiap *task* pada memori. Sehingga ketika prosessor menjalankan suatu *task*, prosessor akan mengambil *state* bagian terakhir dari *task* tersebut kemudian melanjutkan eksekusi *task* tersebut. Proses perpindahan antar eksekusi *task* dilakukan dengan cepat dan secara konkuren sehingga seolah-olah terlihat prosessor melakukan beberapa *task* dalam rentang waktu secara bersamaan. *Embedded system* dapat menggunakan *Real Time Operating System* (RTOS) untuk menjalankan beberapa task secara konkuren. Pada akhirnya sistem tertanam seolah-olah dapat bekerja secara *multitasking*.



Gambar 2. 2 Konsep Konkurensi

Sumber: (Jatmiko, et al., 2015)

2.2.2.2 *Real Time Application*

Sistem tertanam berbasis *Real Time Operating System* (RTOS) memberi respons tepat waktu terhadap *event* yang muncul. Ada tenggat waktu antara munculnya *event* dengan respons dari sistem tersebut. Respons harus segera terpenuhi dalam tenggat waktu tersebut. Tenggat waktu yang ada harus dimanfaatkan seefisien mungkin karena kemungkinan dalam waktu tersebut terdapat banyak *task* sekaligus. Oleh karena itu, perlu terdapat prioritas pada setiap *task* yang ada. *Task* dengan nilai prioritas tinggi akan dieksekusi terlebih dahulu.

RTOS dapat bekerja dengan maksimal apabila telah dapat memenuhi beberapa kriteria dasar seperti di bawah ini:

a. Setiap proses atau *task* memiliki waktu eksekusi yang maksimum tanpa tergantung pada proses lain. Hal tersebut dapat tercapai apabila operasi-operasi yang ada pada RTOS berjalan dengan efisien. Operasi-operasi tersebut diantaranya pengenalan *task* dengan prioritasnya masing-masing serta menjalankan *context switching*.
b. Penundaan eksekusi suatu *task* didefinisikan dengan cukup jelas atau *delay* ditentukan.
c. Pada beberapa kasus *Real Time Operating System* RTOS mampu membagi sumber daya prosesor dengan sistem operasi lain yang juga berjalan pada komputer ataupun pada *embedded system*.

2.2.3 *Free*RTOS

*Free*RTOS yang disediakan oleh *Real Time Enginers Ltd.,* adalah sistem operasi *real-time* yang bersifat *open source*, terukur, dan terdokumentasi dengan baik, yang telah didukung lebih dari 30 arsitektur perangkat keras berbeda. Semua fitur membuat *Free*RTOS menjadi sistem operasi *real time* yang populer. Selain itu, sumber-sumbernya sebagian besar ditulis dalam Bahasa pemrogaman C dengan sangat sedikit rutinitas perakitan ekstra, yang membuat sistem portabel dan mudah dipahami.

Fitur lain yang terlihat adalah ukuran kernelnya yang kecil: membutuhkan sekitar 5 hingga 10 kB ROM, melihat pada kompilator, arsitektur dan konfigurasi kernel. Sisi lain dibutuhkan dari hingga beberapa kB RAM untuk semua RAM yang

tersedia, tergantung pada jumlah tugas, jumlah antrian yang dibuat dan skema alokasi memori. Setiap tugas memiliki tumpukan yang dialokasikan sendiri (ukuran tumpukan *default* terkecil menjadi 140 kata untuk implementasi ARM *Cortex* M3. *Free*RTOS juga menyediakan 3 mode untuk mengimplementasikan *heap*. Mode yang paling sederhana mengimplementasikan alokasi memori, tetapi tidak ada de-alokasi memori atau penggunaan kembali memori sementara mode yang paling kompleks mengimplementasikan baik alokasi dan pengalokasian.

2.2.4 Teknik Enkoding Digital

Teknik enkoding digital merupakan rangkaian kombinasional yang nantinya menghasilkan keluaran berupa kode spesifik seperti bilangan biner atau huruf ABCD sebagai respons terhadap masukan. Teknik enkoding memiliki 2 tipe yaitu enkoding biner dan prioritas *encoder*. *Output* dari enkoding biner memiliki data lebih sedikit jumlahnya dibandingkan dengan nilai *input,* dengan hal tersebut enkoding biner biasanya berguna untuk mengecilkan ukuran data *memory* dan dapat dibangun dari susunan gerbang *AND* dan *OR* sederhana.

Encoder prioritas merupakan pengembangan enkoding biner karena tipe enkoding biner memiliki kelemhan ketika nilai *input* yang banyak masuk dalam satu waktu. *Encoder* prioritas melihat nilai *input* yang diprioritaskan, untuk penerapanya *encoder* prioritas sebagai pengendali interupsi pada mikroprosesor dimana akan menganalisa *input* prioritas tertinggi terlebih dahulu (Electronics-Tutorials, 2017). Berikut contoh mekanisme enkoding dengan 4 data *input* dan menghasilkan 2 data *output* pada Tabel 2.2.

Tabel 2. 2 Contoh Enkoding

I_0	I_1	I_2	I_3	A	B
1	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1

Sesuai dengan Tabel2.2 kebenaran di atas maka didapat dua persamaan sebagai keluaran yaitu :

$$A = I_0' \, I_1' \, I_2 \, I_3' + I_0' \, I_1' \, I_2' \, I_3$$
$$B = I_0' \, I_1 \, I_2' \, I_3' + I_0' \, I_1' \, I_2' \, I_3$$

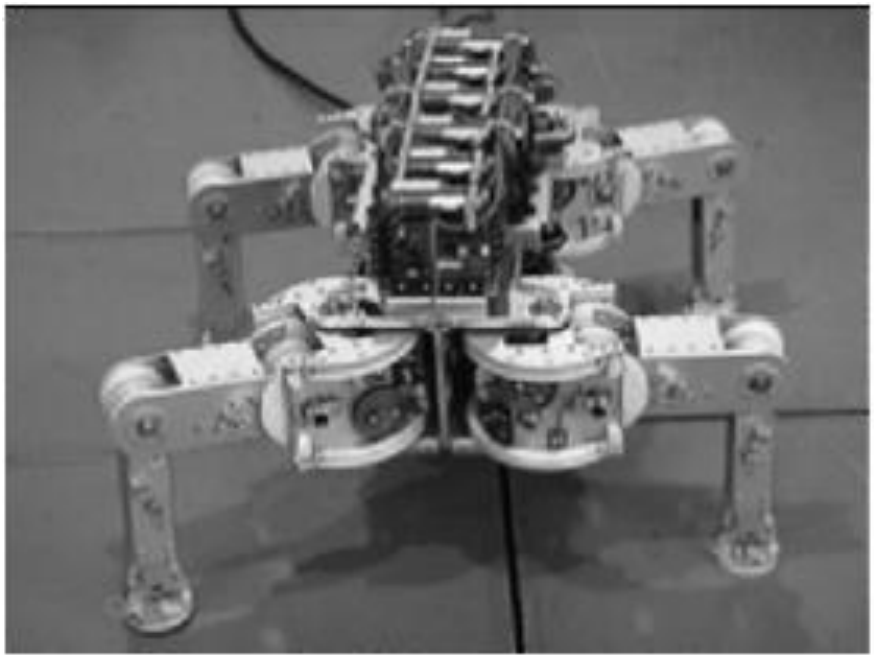
2.2.5 Robot *Quadruped*

Robot adalah sebuah manipulator yang dapat di program ulang untuk memindahkan *tool*, material, atau peralatan tertentu dengan berbagai program pergerakan untuk berbagai tugas dan juga mengendalikan serta mensinkronkan peralatan dengan pekerjaannya (Robot Institue of America, 1979). Sedangkan

Mobile robot merupakan robot dengan ciri khas mempunyai aktuator berupa roda atau kaki untuk menggerakkan keseluruhan badan robot, sehingga robot tersebut dapat melakukan perpindahan posisi dari satu titik ke titik yang lain (Ermadi, 2009).

Robot berkaki adalah robot yang bermanuver dengan kaki-kaki buatan, baik dengan 2 kaki yang sering disebut dengan robot *humanoid*, berkaki tiga (*tripod*), berkaki empat (*quadruped*), robot berkaki enam (*hexapod*) dan robot berkaki banyak lainnya. Robot beroda sering diaplikasikan karena memiliki kecepatan tinggi dalam melintasi bidang yang rata, namun dalam dunia nyata bidang atau daerah yang akan dilintasi oleh robot ialah bidang yang tidak rata. Dengan permasalahan tersebut maka robot berkaki 4 *(quadruped)* memiliki kelebihan dibanding dengan robot beroda dalam hal kemampuanya melewati daerah tidak rata (Wicaksono, et al., 2008).

Robot berkaki merupakan robot yang cukup kompleks dalam perancangan dan pembuatannya, karena perlu memperhatikan beberapa faktor diantaranya keseimbangan dan kecepatan (Sandy, 2007). Kecepatan pada robot berkaki 4 *(quadruped)* tidak hanya kecepatan perpindahan posisi, robot berkaki juga perlu memiliki respons atau reaksi cepat dalam menjalankan tugasnya. Perpindahan kaki robot berkaki diatur oleh setiap derajat posisi setiap motor yang digunakan. Pergerakan setiap kaki dalam robot dilakukan secara bergantian agar menjaga kestabilan dari robot dan robot dapat berjalan dengan baik, untuk mecapai hal tersebut maka terdapat perancangan *motion* gerak robot yang mengatur perpindahan setiap motor dengan derajat sehingga membentuk gerakan robot. Bentuk robot *quadruped* dapat dilihat pada Gambar 2.3.

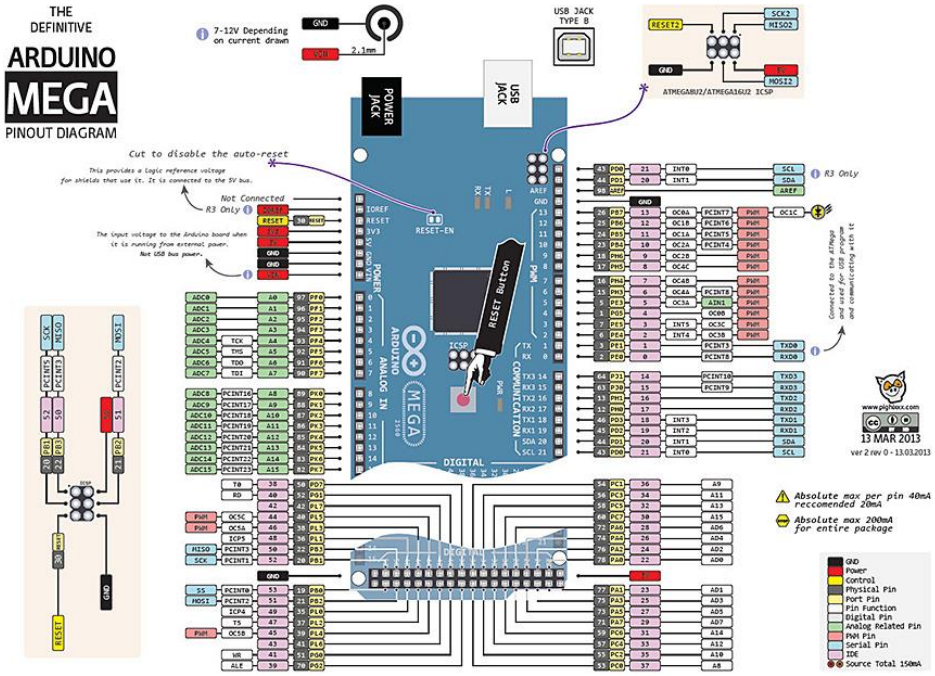


Gambar 2. 3 Mobile Robot

Sumber: (Siegwart & Nourbakhsh, 2004)

2.2.6 Arduino MEGA 2560

Arduino MEGA 2560 merupakan *board* pengembangan mikrokontroler yang berbasis arduino dengan menggunakan chip ATmega 2560. Arduino Mega ini memiliki pin *input/output* yang cukup banyak, dengan jumlah 54 pin digital *input/output* dengan spesifikasi pin (15 pin diantaranya adalah *Pulse Width Modulation*), 16 pin analog *input*, 4 pin UART (*serial port hardware*). Dalam *board* Arduino ini secara langsung sudah lengkap, terdapat sebuah 16 Mhz, sebuah *port USB*, *power jack* DC, ICSP *header*, dan tombol *reset*. *Board* Arduino ini juga dapat digunakan dengan cukup sederhana dan sudah memiliki segala sesuatu yang dibutuhkan sebuah mikrokontroler dengan harga yang cukup terjangkau juga. Untuk spesifikasi lebih lengkap *board* Arduino MEGA 2560 dapat dilihat pada Gambar 2.4 dan Tabel 2.3.



Gambar 2. 4 Pinout Arduino Mega 2560

Sumber: (Arduino, 2017)

Tabel 2. 3 Spesifikasi Arduino 2560

Deskripsi	Spesifikasi
Mikrokontroler	*ATmega2560*
Tegangan	*5V*
Tegangan i*nput*	*7-12V*
I/O Pin digital	*54* (dengan 15 *output* PWM)
Pin i*nput* Analog	*20*
Arus DC per I/O Pin	*20 mA*
Arus DC untuk 3.3V Pin	*50 mA*
Memori *flash*	*256 KB*(8 KB digunakan untuk *bootloader)*

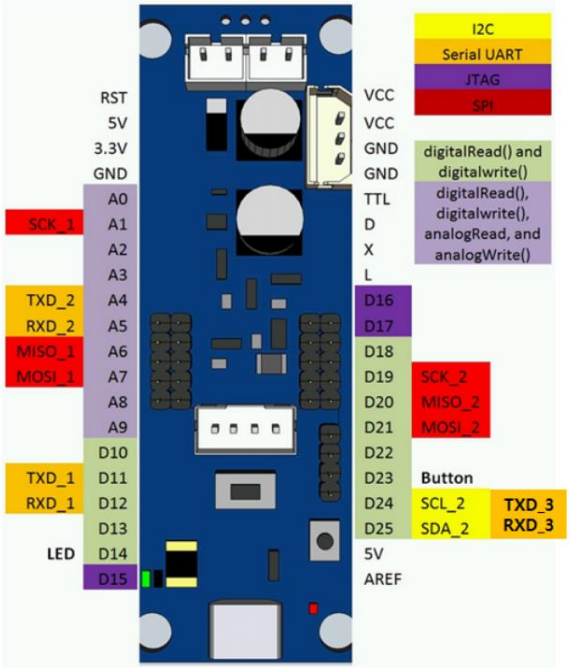
Deskripsi	Spesifikasi
SRAM	*8 KB*
EEPROM	*4 KB*
Kecepatan *clock*	*16 MHz*
LED pin	*13*
Panjang	*101.52 mm*
Lebar	*53.3 mm*
Berat	37 gr

Sumber : (Arduino, 2017)

2.2.7 *Open*CM9.04

*Open*CM9.04 adalah *board* mikrokontroler yang berbasis pada 32bit ARM Cortex-M3. Skema *Open*CM9.04 dan kode sumber bersifat *open-source* atau tersedia secara gratis. Tersedia dalam 3 tipe dari *Open*CM9.04 yaitu tipe A, tipe B dan tipe C. Perbedaan antara tipe A, tipe B dan tipe C adalah pada ketersediaan konektor.

*Open*CM9.04 memempunyai kelebihan diantaranya sudah terdapat pin analog, digital, dan komunikasi pengiriman data di dalamnya, untuk dari *Open*CM9.04 dapat dilihat pada Gambar 2.5 dan Tabel 2.4. Dengan mikrokontroler tersebut juga dapat langsung digunakan untuk pergerakan mengontrol servo *dynamixel* AX-12 (ROBOTIS, 2010).



Gambar 2. 5 Pin GPIO OpenCM9.04

Sumber: (ROBOTIS, 2010)

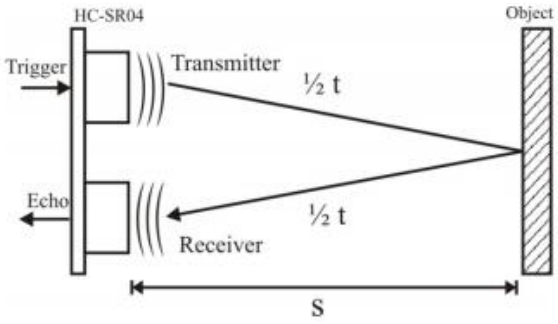
Tabel 2. 4 Spesifikasi OpenCM 485

Deskripsi	Spesifikasi
Size	*68 mm X 66.5 mm x 16 mm*
Weight	*32 g*
Input Voltage	*5~30V*
Power	*SMPS, LIPO, DXL PRO 24V*
Power Switch	*1*
DYNAMIXEL Port	*4Pin x 5, 3Pin x 5*
Buttons	*2*
LED	*5*

Sumber: (ROBOTIS, 2017)

2.2.8 HCSR-04

HC-SR04 adalah modul sensor jarak *ultrasonic* dengan jarak pengukuran 2 cm – 400 cm. akurasi pengukuran jarak modul ini mencapai 3 mm. HC-SR04 memiliki 2 komponen utama sebagai penyusun yaitu *ultrasonic transmitter* dan *ultrasonic receiver*. *Ultrasonic transmitter* memiliki fungsi memancarkan gelombang *ultrasonic* dengan *frekuensi* 40 KHz kemudian *ultrasonic receiver* berfungsi untuk menangkap hasil pantulan gelombang *ultrasonic* yang mengenai suatu objek. Waktu tempuh gelombang *ultrasonic* dari pemancar hingga sampai kembali sebanding dengan 2 kali jarak antara sensor dan bidang pantul seperti yang diperlihatkan pada Gambar 2.6.



Gambar 2. 6 Gambaran Proses Sensor HC-SR04

Sumber: (Nugraha, 2016)

Perhitungan jarak antara sensor dengan halangan dapat dihitung menggunakan persamaan (2.1).

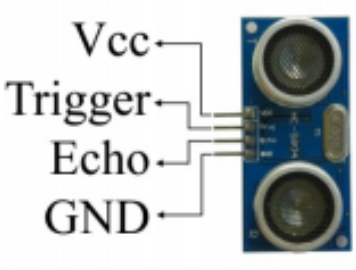
$$S = 340 \times \frac{t}{2} \quad (2.1)$$

Keterangan:

S = Jarak

t = Selisih waktu dipancarkan dan waktu diterima gelombang

Pin yang terdapat pada modul ini ada 4 pin, 5V supply, Trigger pulse *input*, echo pulse *output*, 0V *Ground* yang akan di tampilkan pada Gambar 2.7 dan spesifikasi dapat dilihat pada Tabel 2.5.



Gambar 2. 7 HC-SR04

Sumber: (ITead Studio, 2010)

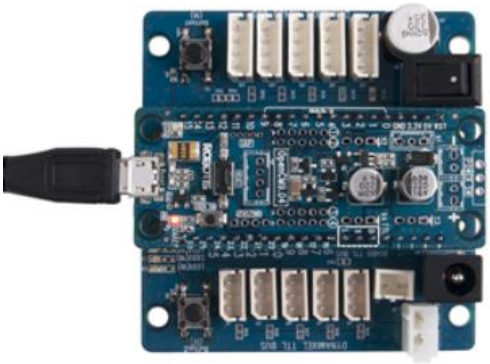
Tabel 2. 5 Spesifikasi HC-SR04

Deskripsi	Spesifikasi
Tegangan	5 Volt DC
Arus	15mA
Frekuensi	40Hz
Maksimal Jarak	4 meter
Minimal Jarak	2 cm
Derajat jangkauan	15 derajat
Komunikasi	UART
Dimensi	45 * 20 * 15 mm

Sumber: (ITead Studio, 2010)

2.2.9 Board Ekpansi OpenCM 485

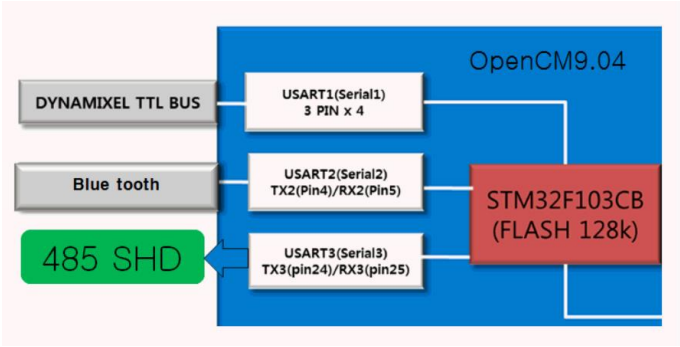
Ekspansi *Open*CM 485 merupakan board ekspansi untuk mengontrol servo *dynamixel*. Ekspansi *Open*CM 485 terhubung ke pengendali *Open*CM9.04. Papan pengontrol *Open*CM9.04 diperlukan untuk menggunakan *Open*CM 485 *Expansion Board* untuk layout dari ekpansi dapat dilihat pada gambar 2.8.



Gambar 2. 8 Layout OpenCM 485 dengan OpenCM9.04

Sumber: (ROBOTIS, 2017)

Kontroler *Open*CM9.04 mendukung *dynamixels* menggunakan RS-485 dan TTL yang dijelaskan pada gambar 2.9 dan Tabel 2.6. Komunikasi yang digunakan ialah komunikasi serial *Universal Synchronus Asychronus Receiver Transmitter* (USART), salah satu metode komunikasi data dengan hanya mengirimkan satu bit pada waktu tertentu.



Gambar 2. 9 Jalur komunikasi OpenCM9.04 dengan OpenCM 485

Sumber: (ROBOTIS, 2017)

Tabel 2.6 Spesifikasi OpenCM9.04

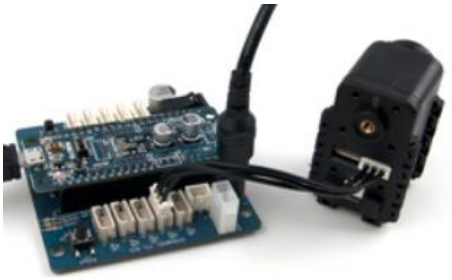
Deskripsi	Spesifikasi
CPU	*STM32F103CB (ARM Cortex-M3)*
Tegangan	5V-16V
I/O	GPIO 26
Timer	*4 (16bit)*
Memory Flash	*128 Kbytes*
SRAM	*20 Kbytes*
Clock	72Mhz
USB	*1 (2.0 Full-Speed) Micro B Type*
USART	3
SPI	2
I2C (TIWI)	2
Debug	*JTAG & SWD*
Dynamixel TT: Bs 3 pin	4
Dimensi	27mm x 66.5mm

Sumber: (ROBOTIS, 2010)

2.2.10 Servo AX-12A

Aktuator servo AX-12A dari Robotis adalah aktuator paling maju yang ada di pasaran saat ini dan telah menjadi standar defakto untuk generasi berikutnya robotika. Servo AX-12A memiliki kemampuan untuk melacak kecepatan, suhu, posisi poros, tegangan, dan beban. Seolah-olah ini tidak cukup, algoritme kontrol yang digunakan untuk mempertahankan posisi poros pada aktuator AX-12 dapat disesuaikan secara individual untuk masing-masing servo, yang memungkinkan mengendalikan kecepatan dan kekuatan respons motor. Penggunaan servo dengan *Open*CM9.04 dapat dilihat pada gambar 2.10. Semua manajemen sensor

dan kontrol posisi ditangani oleh mikrokontroler servo *built-in*. Pendekatan terdistribusi ini membuat pengendali utama bebas melakukan fungsi lainnya (ROBOTIS, 2006). Spesifikasi dari servo AX-12A dapat dilihat pada Tabel 2.7.



Gambar 2. 10 Layout komunikasi Servo AX-12 dengan OpenCM

(ROBOTIS, 2017)

Tabel 2. 7 Tabel Spesifikasi Servo *Dynamixel* AX-12

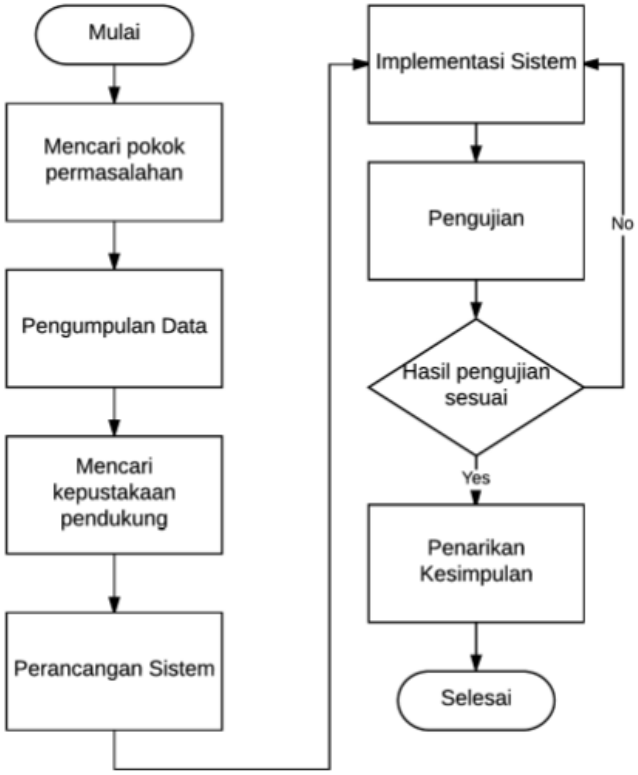
Deskripsi	Spesifikasi
Berat	54.6g
Dimensi	32mm x 50mm x 40mm
Perbandingan gigi	254 : 1
Voltase	12
Stall Torque (N.m)	1.5 (12V)
Stall Current (A)	1.5
Tanpa Kecepatan	59 (12V)
Motor	Cored Motor
Minimum kontrol sudut	0.29 *degrees* x 1,024
Jangkauan Operasi	Mode Aktuator : 300 derajat
Tegangan kerja	9~12V (Rekomendasi : 11.1V)
Arus Maksimal	900mA
Arus siaga	50mA
Temperatur	-5°C ~ 70°C
Sinyal Perintah	Paket Digital
Protokol	Half duplex Asynchronous Serial Komunikasi (8bit, 1*stop*, *No Parity*)
Link (fisik)	TTL *Level Multi Drop*
ID	254 ID (0~253)
Baud Rate	7843bps ~ 1 Mbps
Fungsi balik	Posisi, Suhu, Beban, Tegangan masuk.
Bahan	Plastik
Posisi Sensor	Potentiometer
ID awal	ID #1 (1 Mbps)

Sumber: (ROBOTIS, 2006)

BAB 3 METODOLOGI

3.1 Metode Penelitan

Metode Penelitian pada tugas akhir ini merupakan langkah-langkah pengerjaan dan cara penyelesaian tugas akhir, berikut ini merupakan alur metode penelitian yang dilakukan :



Gambar 3. 1 Alur Metode Penelitian

Berdasarkan pada Gambar 3.1 Metodologi Penelitian yang merupakan gambar alur metodologi penelitian, semua proses dilakukan secara berurut dari studi dan pengkajian literatur sampai dengan kesimpulan dan saran. Terdapat tahap memiliki syarat untuk dapat dilanjutkan ke tahap berikutnya. Jika implementasi sesuai dengan semua yang sudah dirancang sebelumnya pada tahap perancangan sistem, maka sistem dinyatakan siap untuk ke tahap pengujian dan analisis. Implementasi dapat dikatakan sesuai atau tidak berdasarkan hasil ketika sistem dijalankan dalam tahap pengujian. Ketika pengujian dan analisis sesuai dengan hipotesis awal, maka dapat ditarik kesimpulan dari keseluruhan tahapan. Jika pada tahap pengujian dan analisi tidak sesuai, maka akan kembali ke tahap implementasi sistem.

3.2 Studi Literatur

Pada Penelitian tugas akhir ini, studi literatur dilakukan untuk pemahaman terhadap tinjauan pustaka dan dasar teori yang mendukung dalam perancangan dan proses realisasi alat, berikut ini merupakan studi literatur yang dilakukan:

1. *Real Time Operating System* yang diterapkan pada mikrokontroler untuk pengaksesan sensor.
2. Konstruksi robot berkaki 4 *(quadruped)* dengan setiap kaki terdiri 3 *Degree of Freedom* (DOF).
3. Pengkajian pembacaan multi sensor jarak HC-SR04 secara simultan menggunakan mikrokontroler Arduino MEGA 2560.
4. Kontrol aktuator *dynamixel* Ax-12 sejumlah 12 buah pada *Open*CM 9.04 dan ekspansi *Open*CM 485.

3.3 Analisis kebutuhan

Proses analisis kebutuhan sistem akan menjelaskan kebutuhan fungsional dan kebutuhan *non* fungsional dari sistem. Kebutuhan fungsional menjelaskan kebutuhan utama dari sistem yang menjadi kriteria utama penelitian. Sedangkan kebutuhan *non* fungsional ialah gambaran kebutuhan diluar sistem untuk menjalankan sistem yang dibangun.

3.3.1 Analisis Kebutuhan Fungsional

Analisis terhadap kebutuhan fungsional pada penelitian ini meliputi pergerakan robot *quadruped,* pembacaan sensor jarak HC-SR04, mekanisme *multitasking* dengan *Real Time Operating System* (RTOS)*,* Pengolahan hasil data dari *input* sensor HC-SR04, dan pengontrolan gerak kaki robot *quadruped* dengan melihat hasil pengolahan data input sensor HC-SR04*.*

3.3.2 Analisis Kebutuhan *Non* Fungsional

Merupakan penggambaran kebutuhan bagian luar sistem yang diperlukan untuk menjalankan sistem yang akan dibangun. Sistem ini membutuhakan kebutuhan *non* fungsional yaitu tegangan yang diberikan harus sesuai dengan kebutuhan setiap komponen, sensor HC-SR04 dapat membaca jarak antara robot dengan halangan, servo dapat dikontrol menggunakan *Open*CM9.04 yang *input* datanya dari Arduino.

3.3.3 Analisis Kebutuhan Perangkat Keras

Analisis kebutuhan yang diperlukan dibagi menjadi dua yaitu kebutuhan perangkat keras untuk *software* dan kebutuhan perangkat keras dalam pembangunan robot *quadruped.* Berikut merupakan spesifikasi perangkat keras minimum untuk *software* yang direkomendasikan:

1. *Processor* Dual Core 2.1 Ghz
2. *Memory* RAM 512 MB

Spesifikasi perangkat keras yang diperlukan untuk membangun robot *quadruped* pada penelitian ini adalah sebagai berikut:

a. Mikrokontroler Arduino MEGA 2560
b. Motor servo *dynamixel* AX-12
c. Sensor jarak HC-SR04
d. Servo kontroller *Open*CM0.4 dan Ekspansi *Open*CM 485

3.3.4 Analisis Kebutuhan Perangkat Lunak

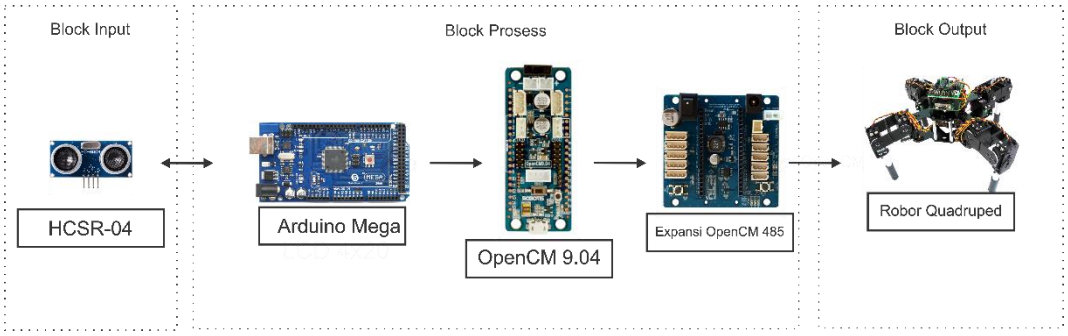
Kebutuhan perangkat lunak yang diharapkan dapat memenuhi kebutuhan dalam menjalankan robot *quadruped* ini adalah sebagai berikut:

Tabel 3. 1 Spesifikasi Perangkat Lunak yang dibutuhkan

No.	Jenis
1	Sistem Operasi Windows 10
2	Arduino IDE
3	Arduino_FreeRTOS.h
4	Newping.h

3.4 Perancangan Sistem

Tahap perancangan sistem ini dilakukan dengan tujuan agar penelitian yang dilakukan menjadi terstruktur. Dimulai dengan langkah-langkah perancangan sistem dengan membuat diagram blok sistem. Diagram blok akan memberi gambaran mengenai kinerja sistem, dimulai dari *input,* proses, dan hingga *output* yang nanti dihasilkan, berikut Gambar 3.3 diagram blok sistem.



Gambar 3. 2 Diagram Blok Sistem

Pada Gambar 3.2 dan Gambar 3.3 dapat dilihat bahwa perancangan sistem teridir dari tiga proses utama, yaitu:

1. *Input*

   Input adalah masukkan nilai data sensor yang berupa hasil baca jarak sensor *ultrasonic* HC-SR04 antara robot dengan penghalang.

2. Proses

   Proses yang dilakukan pertama ialah memberi inisialisai nilai variabel yang dimiliki setiap *task* atau setiap sensor, lalu nilai variabel tersebut akan diproses dengan menggunakan teknik enkoding digital.

3. *Output*

   Output adalah keluaran dari sistem yang berupa gerakkan robot dengan penerapan RTOS.

3.5 Implementasi Sistem

Implementasi merupakan proses setelah perancangan sistem penelitian telah selesai dan dirasa sudah sesuai dengan tujuan penelitian. Adapun tahapan implementasi dari penelitian "Implementasi *Real Time* pada Pergerakan Robot *Quadruped* Menggunakan Multisensor dan RTOS", yaitu:

1. Implementasi pembuatan robot berkaki 4 *(quadruped).*
2. Implementasi *multitasking* dengan *Real Time Operating System* (RTOS) untuk pembacaan sensor jarak HC-SR04.
3. Implementasi pengolahan hasil data dari input sensor HC-SR04.
4. Implementasi pengontrolan gerak kaki robot *quadruped* dengan melihat hasil pengolahan data input sensor HC-SR04.

3.6 Pengujian dan Analisis

Pada tahap pengujian sistem ini menggunakan parameter yang disesuaikan pada perancangan sistem. Tahap ini akan menjelaskan bagaimana penelitian dilakukan dari perancangan dan implementasi lalu dilakukan pengujian dan analisa yang akan dijelaskan pada bab 6. Dalam pengujian sistem ini dilakukan 3 jenis pengujian. Pertama pengujian ini dilakukan pada akurasi dan nilai jarak minimum dan maksimum pada sensor HC-SR04. Kedua akan dilakukan pengamatan waktu eksekusi gerakkan kaki robot apakah sudah sama dengan waktu eksekusi yang telah diberikan dan mengamati hasil dari penerapan sistem *real time* apakah memberikan dampak. Ketiga ialah pengujian membandingkan robot ketika menggunakan sistem *real time* dengan tidak menggunakan sitem *real time* pada pembacaan sensor.

3.7 Kesimpulan

Pada bab kesimpulan ini akan menjelaskan bagaimana hasil dari penelitian yang telah dilakukan dan saran yang nantinya akan membantu penelitian selanjutnya. Kesimpulan akan didapat dengan melihat hasil pengujian yaitu mengetahui akurasi dan nilai jarak sensor HC-SR04, waktu eksekusi dan hasil penerapan sistem real time terhadap robot berkaki *quadruped,* dan perbandingan penggunaan sistem yang diterapkan pada robot *quadruped.*

BAB 4 REKAYASA KEBUTUHAN

4.1 Dekripsi Umum Sistem

Sistem *real time* merupakan sistem yang berguna untuk menentukan waktu eksekusi dari sebuah sistem. Sistem *real time* sangat memungkinkan jika diterapkan dalam pengaksesan sensor dan dilanjutkan dengan pergerakan robot, karena dalam pergerakan robot dibutuhkan ketepatan waktu dalam eksekusi dimulai dari pengaksesan sensor hingga proses pengolahan data.

4.2 Kebutuhan Sistem

Pada sub bab ini membahas kebutuhan fungsional, kebutuhan non fungsional yang dianalisis sesuai dengan kebutuhan yang dibutuhkan oleh sistem sehingga diharapkan dapat mempermudah dalam melakukan tahap mendesain sistem dan implementasi sistem.

4.2.1 Kebutuhan Fungsional

Sistem mempunyai kebutuhan fungsional yang harus terpenuhi, yaitu:

1. Mikrokontroler Arduino dapat membaca data dari sensor HC-SR04

   Sensor *ultrasonic* HC-SR04 yang telah tertanam pada robot *quadruped* mampu membaca jarak antara sensor dengan halangan sebagai nilai *input* dan output dari sensor tersebut akan diolah kembali dalam mikrokontroler Aduino untuk menentukan arah gerak robot.

2. Mekanisme *multitasking* dengan *Real Time Operating System* (RTOS)

   Dalam sistem ini menggunakan Arduino MEGA 2560 yang termasuk mikrokontroler konvensional, sehingga konsep *multitasking* belum dapat diterapkan. Oleh sebab itu dalam sistem ini menggunakan konsep konkurensi yang hampir sama dengan konsep *multitasking.*

3. Pengolahan hasil data dari *input* sensor HC-SR04

   Sistem ini melihat *input* yaitu jarak antara sensor *ultrasonic* dengan halangan, dari jarak tersebut nanti akan memberi nilai *variabel* yang pada setiap sensor untuk dilanjutkan dalam proses pengolahan data.

4. Pengontrolan gerak kaki robot *quadruped* dengan melihat hasil pengolahan data input sensor HC-SR04

   Robot *quadruped* akan bergerak sesuai hasil pengolahan data dan robot *quadruped* ini memiliki 8 pergerakan yaitu depan, belakang, kiri, kanan, serong kanan depan, serong kiri depan, serong kanan belakang, dan serong kiri belakang.

4.2.2 Kebutuhan Non-fungsional

Kebutuhan non-fungsional menjelaskan kebutuhan bagian luar sistem yang diperlukan untuk menjalankan sistem. Adapun kebutuhan non-fungsional dijelaskan dibawah ini:

1. Tegangan yang diberikan harus sesuai dengan kebutuhan setiap komponen.

   Tegangan yang dibutuhkan dari komponen ada yang berbeda. Pada komponen Arduino, *Open*CM9.04, ekspansi *Open*CM 485, dan servo AX-12 membutuhkan daya sebesar 12volt untuk dapat bekerja optimal sedangkan sensor HC-SR04 membutuhkan daya sebesar 5volt.

2. Sensor HC-SR04 dapat membaca jarak antara robot dengan halangan

   Sistem ini menggunakan 8 sensor HC-SR04 dan dapat mendeteksi halangan di sekeliling robot. Peletakkan sensor juga dapat mempengaruhi pendeteksian halangan dan berpengaruh pada gerakan robot.

3. Servo dapat dikontrol menggunakan *Open*CM9.04 yang *input* datanya dari Arduino

   Servo dalam sistem ini untuk pergerakanya dikontrol oleh *Open*CM9.04 yang data untuk mengontrol servo didapat dari Arduino untuk itu harus ada komunikasi antara *Open*CM9.04 dengan Arduino dan mekanisme pembacaan data.

4.2.3 Kebutuhan Perangkat Keras

Pada penelitian ini dibutuhkan perangkat keras untuk menunjang serta bagian dalam penelitian ini, antara lain:

1. *Personal Computer* atau Laptop

   Peronal Computer atau laptop dibutuhkan untuk mengakses perangkat lunak *(software)* yang dibutuhkan dalam sistem. Dalam sistem ini spesifikasi yang diperlukan untuk *Peronal Computer* atau laptop memiliki *processor* dual core 2.1 Ghz dan *memory* RAM 512 MB.

2. Mikrokontroler Arduino MEGA 2560

   Mikrokontroler Arduino MEGA 2560 dalam sistem ini digunakan sebagai kontroler utama yang dapat menerima *input* sensor, melakukan proses pengambilan dan pengolahan data dengan penerepan *Free*RTOS, dan pengiriman hasil pengolahan data berupa perintah robot bagaimana harus bergerak.

3. Motor Servo *Dynamixel* AX-12

   Aktuator atau penggerak dalam sistem ini ialah sero *dynamixel* AX-12, dengan aktuator ini merupakan salah satu pemebeda dengan penelitan sebelumnya. Motor servo ini memiliki banyak kelebihan salah satunya

mampu menahan beban yang cukup berat. Sistem ini menggunkan 12 servo AX-12 dengan kebutuhan setiap kaki robot *quadruped* menggunakan 3 servo *dynamixel*.

4. Sensor HC-SR04

   Sensor HC-SR04 merupakan sensor *ultrasonic* yang berfungsi untuk mendeteksi jarak antara robot *quadruped* dengan halangan. Nilai yang dihasilkan berupa jarak dengan satuan *centimeter*.

5. *Open*CM9.04 dan papan Ekspansi *Open*CM 485

   *Open*CM9.04 dan papan ekspansi *Open*CM 485 yang berfungsi sebagai untuk mengatur pergerakan *actuator* berupa servo *dynamixel* AX-12. Dan untuk mengontrol *actuator* AX-12 ini *Open*CM9.04 memiliki code tersendiri.

4.2.4 Kebutuhan Perangkat Lunak

Pada penelitian ini dibutuhkan beberpa perangkat lunak untuk menunjang penelitian dan menjalankan robot *quadruped*, anatara lain:

1. Sistem Operasi *Windows* 10

   Perangkat lunak yang digunakan sebagai sistem operasi oleh *personal computer*/laptop. *Windows* 10 juga sudah.

2. Arduino IDE

   Perangkat lunak yang berfungsi untuk menuliskan program sekaligus *upload* code program yang telah dibuat ke mikrokontroler Arduino MEGA 2560.

3. Arduino_FreeRTOS.h

   Perangkat lunak yang merupakan *library* dengan fungsi untuk menangani pengimplementasian RTOS pada mikrokontroler Arduino.

4. Newping.h

   Perangkat lunak yang merupakan *library* dengan fungsi untuk menangani pembacaan sensor *ultrasonic* HC-SR04.

4.3 Batasan Desain Sistem

Agar sistem ini dapat diterapkan sesuai dengan harapan yang telah disusun, maka perlu diterapkan batasan-batasan implementasi desain sistem, anatara lain:

1. Sistem menggunakan sensor *ultrasonic* HC-SR04 sejumlah 8 buah.
2. Sensor HC-SR04 hanya memiliki jangkauan 4meter dilihat dari *datasheet*.
3. Robot memiliki lebar 25cm dan tinggi maksimal 27cm dan memiliki 4 kaki.

4. Robot hanya dapat melakukan 8 pergerakan yaitu pergerakan maju, mundur, kanan, kiri, serong depan kanan, serong depan kiri, serong belakang kanan, dan serong belakang kiri.
5. Pengambilan keputusan pergerakan robot menggunakan teknik enkoding digital dan tidak dijelaskan secara rinci pada implementasi
6. Pengiriman data serial UART dari mikrokontroler Arduino ke *Open*CM 9.04 dan tidak dijelaskan secara rinci pada implementasi.
7. Robot bergerak ketika data dari Arduino telah dikirimkan ke *Open*CM9.04 untuk menggerakkan aktuator.

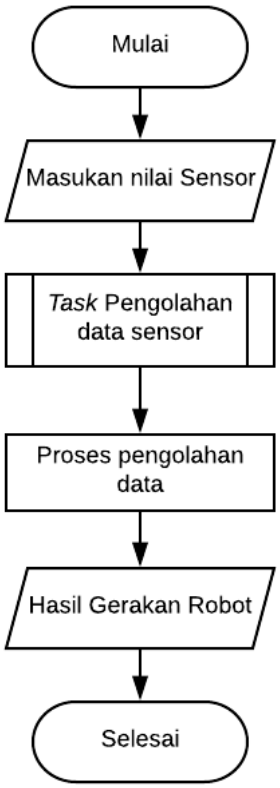
BAB 5 PERANCANGAN DAN IMPLEMENTASI

5.1 Perancangan Sistem

Pada tahap ini menjelaskan terkait perancangan penyusunan sistem *real time* pada pergerakan kaki robot *quadruped* meliputi perancangan perangkat keras dan perancangan perangkat lunak agar sistem yang akan di implementasikan dapat bekerja seperti yang diinginkan.

5.1.1 Gambaran Umum Sistem

Gambaran umum sistem yang akan dibuat, terdapat 3 bagian penyusun berupa *input*, proses, dan *output*, seperti pada Gambar 5.1 *flowchart* sistem.



Gambar 5. 1 *Flowchart* Sistem

Pada bagian *input* atau dalam *flowchart* bagian masukan nilai sensor terdiri dari 8 sensor *ultrasonic* HC-SR04 yang memiliki fungsi untuk mendeteksi jarak robot dengan halangan. Bagian proses terdapat 3 komponen *hardware* yaitu Arduino MEGA 2560, OpenCM *9.04*, dan ekspansi OpenCM 456. Arduino berfungsi untuk memproses data yang dihasilkan oleh sensor *ultrasonic* HCSR-04 menggunakan RTOS. Sedangkan *openCM 9.04* digunakan sebagai servo *controller*. Sedangkan bagian *output* ialah gerakkan kaki pada robot *quadruped,* robot akan bergerak sesuai hasil sensing yang dihasilkan oleh mikrokontroler Arduino.

Jarak yang dihasilkan sensor *ultrasonic* merupakan *input* utama dari sistem ini. Setiap hasil sensing sensor akan diteruskan pada mikrokontroler Arduino untuk

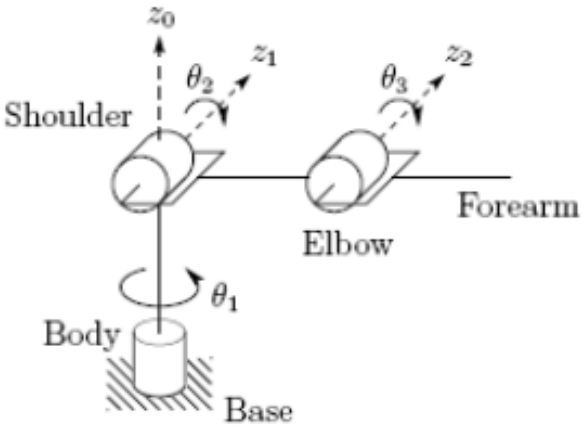
diproses dan menentukan jalannya *task* yang digunakan menggunakan implementasi RTOS pada Arduino dengan menggunakan *library Free*RTOS. Pada proses Arduino data akan masuk pada *task* dengan menggunakan *baudrate* sebesar 115200bps, *baudrate* merupakan istilah yang digunakan untuk kecepatan aliran data. Proses selanjutnya di dalam *task* terdapat sebuahh kondisi dimana berfungsi untuk memberikan nilai pada *variable* setiap sensor yang digunakan untuk teknik enkoding digital. Ketika sensor mendeteksi jarak lebih dari yang ditentukan maka *variable* sensor tersebut bernilai 0, dan sebaliknya jika sensor mendeteksi jarak kurang dari yang ditentukan dan tidak sama dengan 0 cm maka *variable* sensor tersebut bernilai 1. Setelah semua variable sensor memiliki nilai maka akan diproses dengan teknik enkoding digital untuk menentukan pergerakan arah robot.

5.1.2 Perancangan Perangkat Keras

Perancangan perangkat keras dalam sistem ini terbagi menjadi 2 yaitu perancangan robot *quadruped* dan perancangan pembacaan sensor jarak HC-SR04.

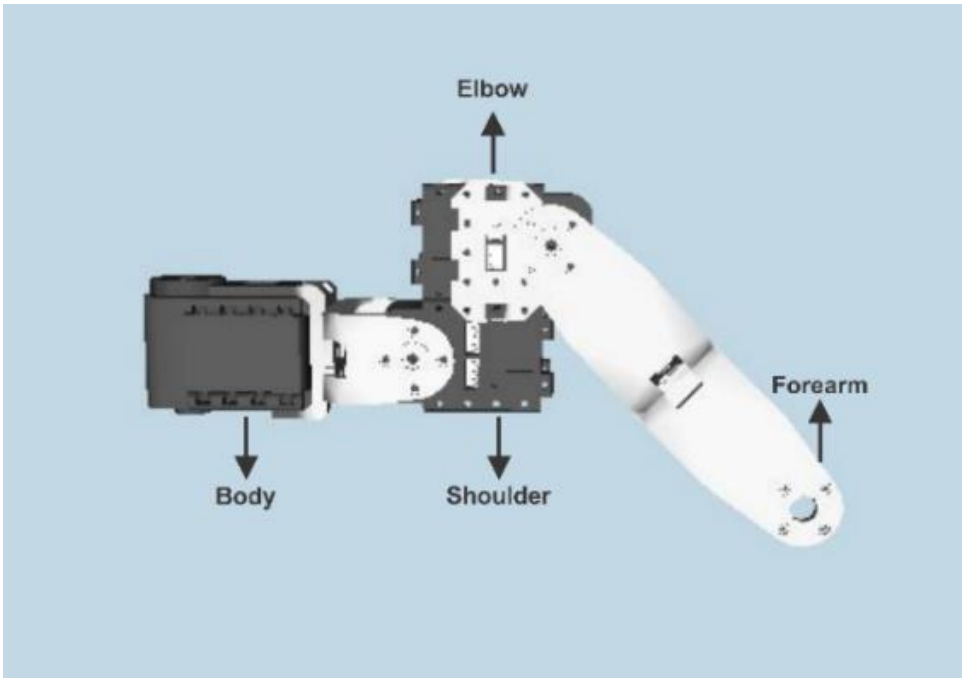
5.1.2.1 Perancangan Robot Quadruped

Robot *quadruped* sesuai dengan namanya memiliki 4 kaki, dengan komposisi penyusun 1 kaki robot terdiri dari 3 *Degree Of Freedom* (DOF) silindrikal. *Joint* yang digunakan dari tiga motor tersebut ialah *revolute joint* dengan susunan masing-masik penggerak ialah *body*, *shoulder*, dan *elbow* seperti pada Gambar 5.2.



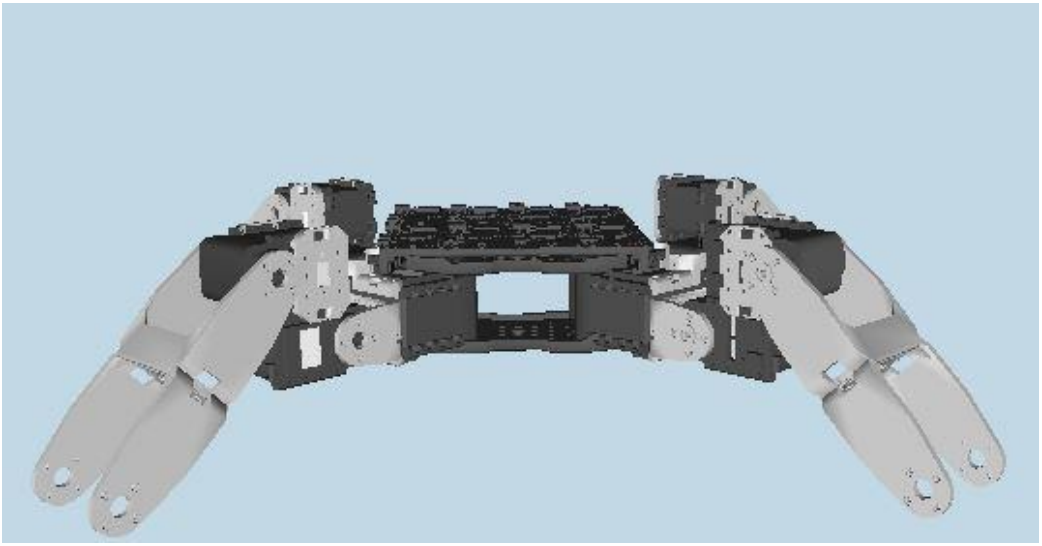
Gambar 5. 2 *Joint* Yang Digunakan

Body merupakan bagian yang bertugas sebagai penggerak agar robot dapat bergerak kedepan atau belakang. *Shoulder* bagian dimana servo bergerak keatas dan kebawah difungsikan ketika robot akan melakukan perpindahan. *Elbow* bagian untuk menahan posisi robot agar berdiri sesuai dengan posisi yang telah dirancang. Desain kaki robot *quadruped* dalam penelitian ini juga merujuk pada penelitian (Setiawan & Utomo, 2011), berikut juga disertakan bentuk kaki robot 3 dimensi pada Gambar 5.3.



Gambar 5. 3 Desain 1 Kaki Robot

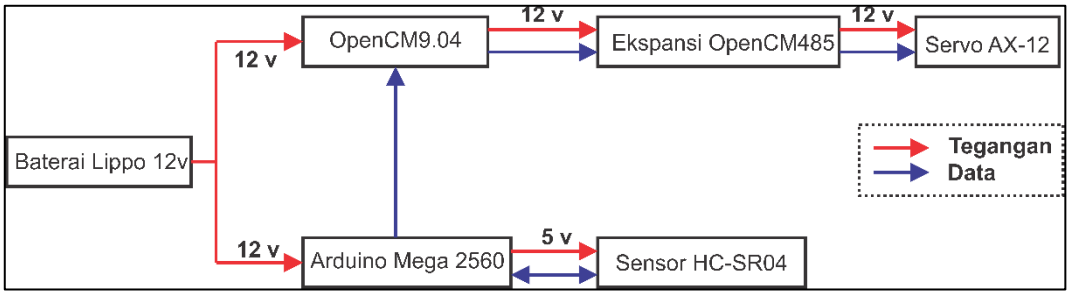
Dengan jumlah total kaki robot yang digunakan ialah 4 maka dalam desain 1 kaki robot pada Gambar 5.2 dan Gambar 5.3 akan dirangkai untuk menjadi desain kaki robot secara penuh seperti pada Gambar 5.4.



Gambar 5. 4 Desain Kaki Robot Quadruped

Setelah perancangan penggerak robot maka dilanjutkan dengan perancangan jalur tegangan dan data antar perangkat keras. Tegangan dalam sistem ini bersumber dari baterai lippo 12v, dengan tegangan tersebut akan mengaliri Arduino Mega 2560, *Open*CM9.04, ekspansi *Open*CM485 dan servo AX-12. Tegangan 5v juga terdapat dalam sistem ini dengan sumber keluaran dari Arduino untuk mengaliri tegangan sensor HC.SR04. Perancangan komunikasi untuk pengiriman data dalam sistem ini menggunakan komunikasi serial UART.

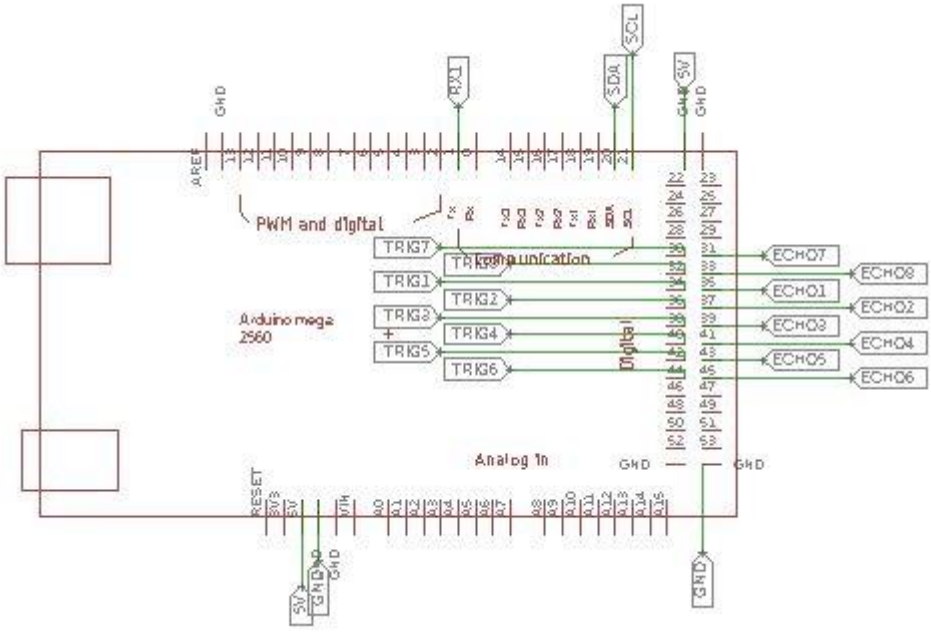
Komunikasi UART digunakan pada pengiriman data antara Arduino dengan *Open*CM9.04. perancangan dalam diagram blok dapat dilihat pada Gambar 5.5.



Gambar 5. 5 Diagram Blok Jalur Tegangan dan Data Sistem

5.1.2.2 Perancangan Pembacaan Sensor Jarak HC-SR04

Perancangan diawali dengan membuat skematik rangkaian antara mikrokontroler Arduino dengan sensor jarak HC-SR04. Sistem ini menggunakan 8 sensor, karena melihat mobilitas pergerakan minimum *mobile* robot yaitu maju, mundur, kanan, kiri, dan berjalan serong. Setiap sensor juga memiliki fungsi atas arah gerak dari robot pada sistem ini. Sensor jarak HC-SR04 terdiri dari 4 pin yaitu *trigger*, *echo, vcc*, dan *ground*. Pin *trigger* berfungsi sebagai pembangkit sinyal *ultrasonic*. Pin *echo* berfungsi untuk mendeteksi sinyal pantulan dari *ultrasonic*, dalam hal ini pin *trigger* dan *echo* memiliki tugas masing-masing sensor sehingga untuk terhubung ke Arduino harus mempersiapkan 16 pin digital. Pin *vcc* dan *ground* untuk 8 sensor terhubung pada satu sumber pada Arduino dan untuk hubungan sumber tegangan antar sensor menggunakan hubungan seri karena sensor *ultrasonic* akan selalu aktif ketika istem telah menyala. Berikut gambar rangkaian skematik untuk sensor HC-SR04 pada Gambar 5.6.



Gambar 5. 6 Rangkaian Skematik untuk Sensor HC-SR04

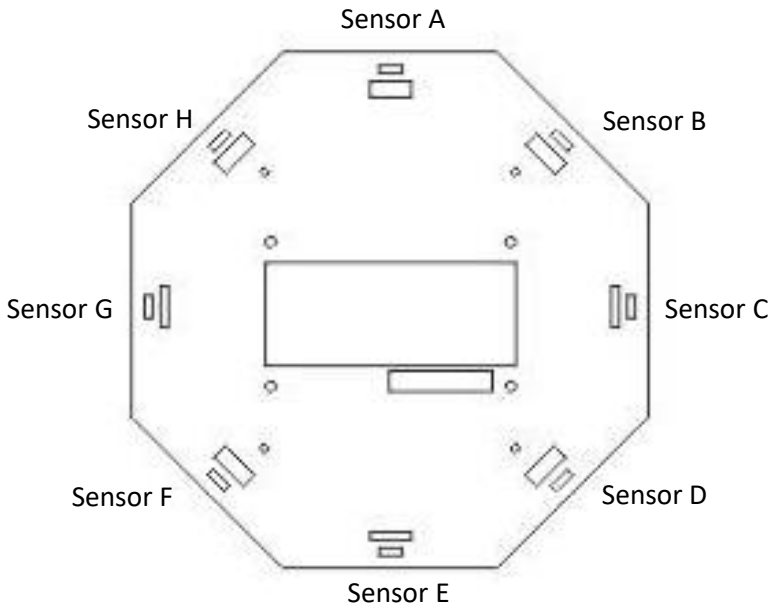
Berdasarkan pada Gambar 5.6 yaitu peletakkan pin antara Arduino dengan HC-SR04 untuk mempermudah pembacaan rangkaian skematik dibuatlah Tabel 5.1 di bawah ini.

Tabel 5. 1 Tabel Pin Skematik

Modul	Pin modul Ke Arduino Mega 2560			
	VCC	TRIG	ECHO	GND
HC-SR04 ke-1	VCC	34	35	GND
HC-SR04 ke-2	VCC	36	37	GND
HC-SR04 ke-3	VCC	38	39	GND
HC-SR04 ke-4	VCC	40	41	GND
HC-SR04 ke-5	VCC	42	43	GND
HC-SR04 ke-6	VCC	44	45	GND
HC-SR04 ke-7	VCC	30	31	GND
HC-SR04 ke-8	VCC	32	33	GND

Tabel 5.1 menunjukkan tabel pin dari setiap modul yang terhubung pada mikrokontroler Arduino berdasarkan rangkaian skematik yang telah dibuat. Tabel tersebut terlihat jumlah pin yang dimiliki dari setiap modul dan untuk *input* tegangan yang masuk pada setiap modul diambil dari output tegangan mikrokontroler Arduino sebesar 5v.

Perancangan setelah jalur sensor dengan mikrokontroler ialah peletakkan sensor dengan melihat acuan gerakkan yang akan dilakukan oleh robot dan jumlah dari sensor yang digunakan, berikut penempatan sensor dapat dilihat pada Gambar 5.7.



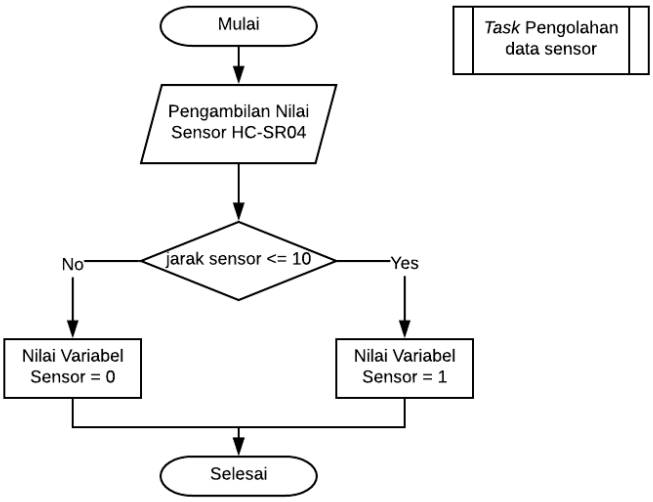
Gambar 5. 7 Peletakkan Sensor

5.1.3 Perancangan Perangkat Lunak

Perancangan perangkat lunak dalam sistem ini terbagi menjadi 3 yaitu perancangan mekanisme *multitasking* dengan RTOS, Pengolahan hasil data dari input sensor HC-SR04, dan pengontrolan gerak kaki robot *quadruped* dengan melihat hasil pengolahan data input sensor HC-SR04.

5.1.3.1 Perancangan *Multitasking* Menggunakan RTOS

RTOS dalam penelitian ini digunakan untuk *multitasking* pembacaan sensor dan proses penentuan pergerakan robot. Perancangan ini dimulai dari menentukan jumlah *task* yang diperlukan, menentukan tugas tiap *task*, menentukan prioritas pada setiap *task* dan batas waktu *task* untuk dieksekusi. *Task* yang diperlukan dalam sistem ini ialah 9 karena pada sisitem ini menggunaka 8 sensor HC-SR04 sebagai input data dan 1 proses, untuk prioritas pada *task* pembacaan sensor dibuat sama dan untuk *task* proses memiliki prioritas yang lebih tinggi. Penjadualan untuk *task* yang memiliki prioritas sama menggunakan algoritme *round-robin scheduling*. *Round robin scheduling* menggunakan *time slicing* untuk membagi mendapatkan alokasi waktu yang sama pada setiap *task*. Setiap *task* dalam sistem ini juga memiliki *delay*, dalam *free*RTOS fungsi *delay* dapat digunakan dengan memberi nilai pada fungsi *xtaskDelay*. *Task* akan tidak dijalankan selama nilai *delay* berjalan dan akan dibuka setelah selesai, dengan fungsi itu maka penjadualan dapat berjalan dengan yang diharapkan yaitu membentuk sistem multisensor. Pembacaan sensor dalam sistem ini menggunakan *library newping.h*, dengan menggunakan *library* ini akan mempermudah dalam mengatur sensor HC-SR04 seperti inisialisasi pin dan pengaturan jarak maksimum sensor.



Gambar 5. 8 *Flowchart* Untuk *Task* Sensor

Gambar 5.8 merupakan *flowchart* dalam pembuatan *task* sensor dan berlaku pada semua *task* sensor. Pada *task* untuk pembacaan sensor terdapat kondisi untuk memberi nilai pada variabel yang dimiliki setiap sensor. Nilai pada

variabel ialah nilai angka 1 atau 0 yang nantinya akan menjadi *input* pada teknik enkoding digital sebagai pegambilan keputusan gerak robot. Kondisi pemberian nilai pada variabel berparameter pada hasil pengukuran jarak dari HC-SR04. Parameter dalam sistem ini dapat disesuaikan dengan kondisi area robot namun memiliki batasan yaitu 2cm - 400cm karena batas kemampuan deteksi sensor HC-SR04.

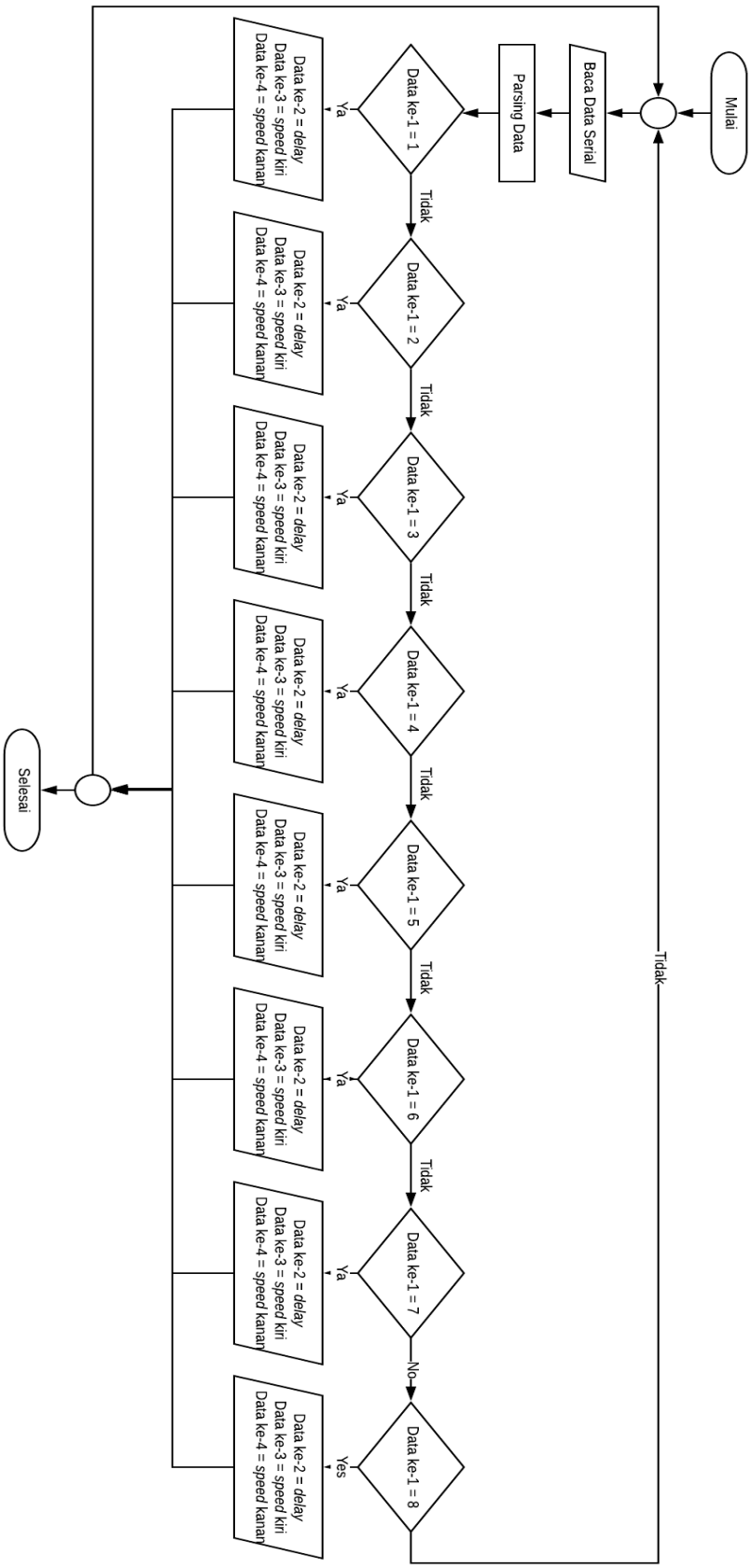
5.1.3.2 Perancangan Pengolahan Hasil Data Dari Input Sensor HC-SR04

Data dari setiap *task* sensor yang berjumlah 8 akan langsung diolah dengan menggunakan teknik enkoding digital. Perancangan pengolahan data menggunakan teknik enkoding digital dimulai dari membuat tabel kebenaran dari jumlah kemungkinan yaitu 256 karena menggunakan 8 nilai input. Hasil dari tabel kebenaran akan diolah menjadi rumus untuk masing-masing gerakan dan tahap selanjutnya rumus tersebut disederhanakan menggunakan K-MAP. Teknik enkoding dalam perancangan ini merujuk pada penelitian (Setiawan, 2018) yang membahas implementasi teknik enkoding digital pembacaan sensor *ultrasonic*.

Teknik pengolahan hasil data nanti akan diproses pada *task* yang berbeda dari *task* sensor karena kemungkinan membutuhkan waktu yang lebih lama dan prioritasnya harus lebih tinggi agar proses pengolahan data diutamakan ketika waktu sistem berjalan.

5.1.3.3 Perancangan Pengontrolan Gerak Kaki Robot *Quadruped*

Pergerakan robot akan dimulai pada saat *Open*CM9.04 menerima data dari Arduino Mega 2560, data yang dikirimkan nanti akan diparsing sehingga di dapat data pada 4 variabel. Data pertama memiliki fungsi untuk menentukkan arah gerak robot, data kedua memiliki fungsi untuk memberi waktu *delay*, data ketiga memberikan nilai kecepatan pada pergerakan kaki kiri, dan data yang terakhir memberikan nilai kecepatan pada pergerakan kaki kanan. Gerak robot dilakukan dengan memberi nilai derajat setiap motor dan setiap kaki robot melakukan gerakan secara bergantian.Berikut gambar *flowchart* gerak kaki robot pada Gambar 5.9.



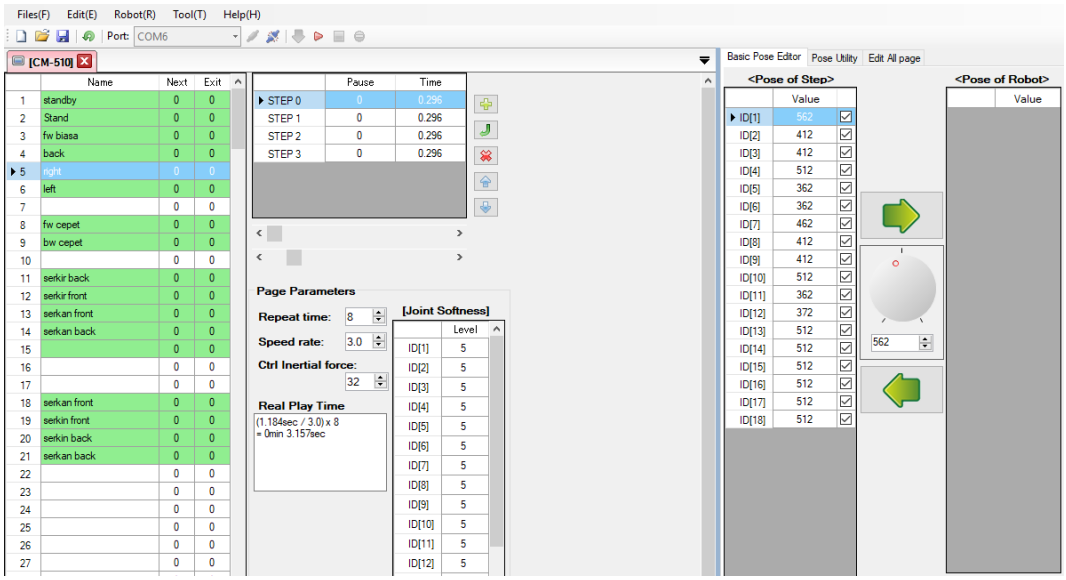
Gambar 5. 9 *Flowchart* Gerak Kaki Robot

Terdapat 4 data yang dikirim untuk data ke 2,3,4 setiap arah dapat bernilai sama dan data pertama untuk menentukkan arah dari pergerakan robot. Terdapat 8 arah gerakkan robot, untuk perancangan arah gerak robot dapat dilihat pada Tabel 5.2.

Tabel 5. 2 Perancangan Arah Gerak Robot

Nialai Data 1	Arah Gerak Robot
1	Depan
2	Belakang
3	Kanan
4	Kiri
5	Serong Kanan Depan
6	Serong Kanan Belakang
7	Serong Kiri Belakang
8	Serong Kiri Depan

Gerakan robot dirancang dengan melakukan desain *motion*, desain tersebut akan memberi nilai derajat pada setiap motor hingga membentuk suatu gerakan dan robot bergerak dengan menggerakan kakinya secara bergantian.Desain *motion* akan dibuat menggunakan aplikasi roboplus seperti pada Gambar 5.10.



Gambar 5. 10 Aplikasi Roboplus untuk Desain *Motion*

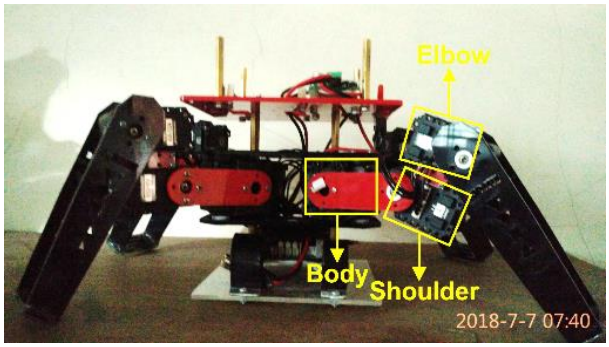
5.2 Implementasi

Pada subab ini akan melanjutkan dari subab perancangan, dimana perancangan perangkat keras dan perangkat lunak akan di implementasikan secara penuh.

5.2.1 Implementasi Perangkat Keras

5.2.1.1 Implementasi Kaki Robot

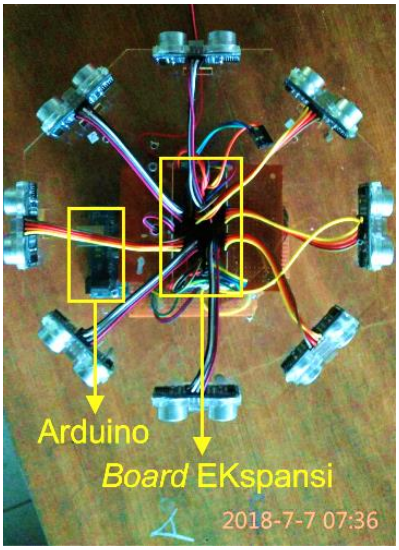
Pembuatan desain robot dilanjutkan dengan membuat rancangan bagian bagian dari robot secara digital dengan ukuran yang sebenarnya. Hasil dari desain kemudian dilanjutkan dengan pemotongan desain pada bahan *acrylic*. Setelah desain *acrylic* sudah di implementasikan menjadi potongan-potongan yang sesuai dengan rancangan maka dilanjutkan dengan merangkai aktuator yaitu servo AX-12 sesuai lokasi dan menempatkan seluruh komponen pada *body* robot. Gambar 5.11 merupakan implementasi *body*.



Gambar 5. 11 Implementasi robot

5.2.1.2 Implementasi Peletakan Sensor HC-SR04

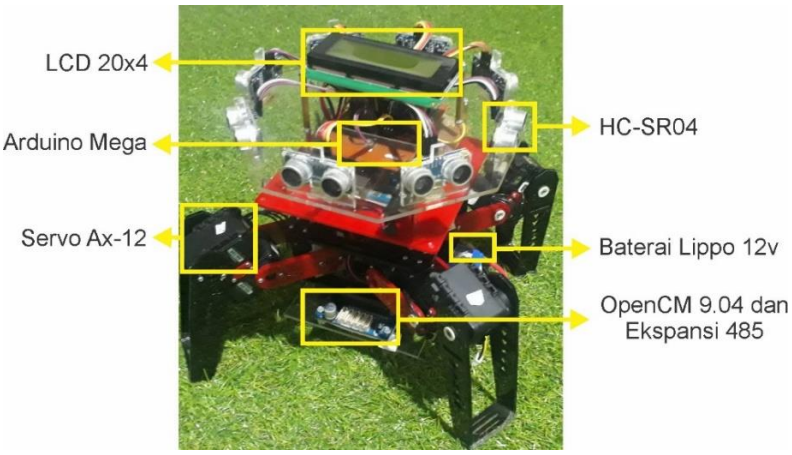
Sub bab ini akan mengimplementasikan perancangan peletakkan sensor sesuai pada Gambar 5.7. Sensor akan terhubung pada sebuah *board* untuk memudahkan pemasangan pin antara Arduino dengan semua sensor HC-SR04. Keunggualan menggunakan *board* tambahan ialah untuk menghindari terputusnya kabel saat sistem bekerja. Berikut Gambar 5.12 implementasi peletakkan sensor HC-SR04.



Gambar 5. 12 Implementasi Peletakkan Sensor HC-SR04

5.2.1.3 Implementasi Keseluruhan Robot *Quadruped*

Setelah implementasi kaki robot dan peletekan sensor HC-SR04 selesai maka akan dilanjutkan dengan menggabungkannya. Penggabungan komponen memperhitungkan dimensi setiap komponen dan jalur kabel untuk tegangan maupun kabel untuk *transfer* data. Pada Gambar 5.13 merupakan implementasi terakhir sesuai dengan perancangan. Robot telah siap digunakan dan bisa dilanjutkan pada tahap implementasi perangkat lunak.



Gambar 5. 13 Implementasi Robot Quadruped

5.2.2 Implementasi Perangkat Lunak

5.2.2.1 Implementasi Pembuatan *Task* Pada RTOS

Tahapan pertama implementasi perangkat lunak ialah membuat *task-task* yang akan digunakan dengan sistem RTOS. Dalam sistem ini terdiri dari 8 sensor dan 1 proses pengolahan data, maka task yang digunakan berjumlah 9 task yaitu *Task*US1, *Task*US2, *Task*US3, *Task*US4, *Task*US5, *Task*US6, *Task*US7, *Task*US8, dan *Taskus*9. Deklarasi *task* pada program sistem ditampilkkan pada Tabel 5.3.

Tabel 5. 3 Kode Program Pembuatan *Task*

Baris	*Source Code*
1	`xTaskCreate(`
2	`  TaskUS1`
3	`  , (const portCHAR *) "Ultrasonic1"`
4	`  , 128 // Stack size`
5	`  , NULL // pvParameters`
6	`  , 2 // Priority`
7	`  , NULL );`

Baris kedua dalam Tabel 5.3 adalah *pvTaskCode Pointer* yang berfungsi untuk penamaan fungsi yang akan mengimplementasikan tugas *task*. *usStackDepth* merupakan bagian pada baris keempat dengan fungsi mengalokasikan lebar tumpukan tugas. Baris kelima ialah *pvParameters* dengan fungsi meneruskan nilai ke *task* berikutnya sebagai parameter, pada sistem ini

tidak ada nilai yang diteruskan sebagai parameter maka ditulis *NULL*. *uxPriority* dengan fungsi memberi nilai prioritas eksekusi *task* terletak pada baris keenam. Baris terakhir *pxCreatedTask* yang digunakan untuk meneruskan sebuah *handle* ke *task* yang dibuat dari fungsi *xTaskCreate (). pxCreatedTask* bersifat opsional dan dapat diberi nilai *NULL*. Sistem ini untuk pengisian nilai dalam program pembuatan *task* hampir semua bernilai sama terkecuali pada *task* 9 yang berisi proses diberi nilai lebih tinggi atau prioritasnya diutamakan.

5.2.2.2 Implementasi Pembuatan Isi Dari *Task*

Setelah *task* telah dibuat dan diatur parameternya maka langkah selanjutnya ialah mengisi *task* dengan kode program sesuai pada Gambar 5.8. Kode program dari 9 *task* berisi hampir sama terkeculai pada *task* ke-9 karena berisi pengolahan data dari *task* sensor ke-1 sampai *task* ke-8. Berikut kode program *task* sensor pada Tabel 5.4.

Tabel 5. 4 Kode Program Isi *Task* Sensor

Baris	Kode Program
1	`void TaskUS1(void *pvParameters)  // This is a`
2	`task.`
3	`{`
4	`  (void) pvParameters;`
5	
6	`  // initialize serial communication at 115200`
7	`bits per second`
8	`  Serial.begin(115200);`
9	`   for (;;)`
10	`  {`
11	`    int SRsensor = sonar[0].ping_cm();`
12	`    if (SRsensor <= 15  && SRsensor != 0)`
13	`     {`
14	`          IA = 1;`
15	`     }`
16	`    else`
17	`     {`
18	`          IA = 0;`
19	`     }`
20	`     vTaskDelay( 100 / portTICK_PERIOD_MS );`
21	`   }`
22	`                        }`

Implementasi kode *task* sensor diawali dengan memberikan niali *baudrate* sebesar 115200 bps untuk keperluan pengiriman data ke laptop dan *Open*CM9.04. Proses selanjutanya ialah bagian perulangan untuk pengambilan nilai sensor dan dilanjutkan dengan kondisi penentuan jarak sensor apakah masuk dalam jangkauan jarak yang ditentukan atau tidak. Kondisi dalam program ini akan menjadi *input* pada *task* 9. Terakhir pada program *task* sensor terdapat *vTaskDelay* sebesar 100 ms dengan fungsi memberi waktu pada sensor agar bersiap-siap melakukan proses selanjutnya dan memberikan waktu pada RTOS dalam melakukan penjadwalan. Nilai 100ms di dapat dari hasil pengujian ketika robot dijalankan, karena jika nilai dibawah nilai tersebut robot tidak akan berjalan sedangkan jika diatas nilai tersebut robot akan memiliki respon yang sangat lama.

5.2.2.3 Implementasi Pengolahan Data Dan Pengontrolan Gerak

Pengolahan data akan dilakukan pada *Task* 9 dengan penerapan teknik enkoding digital*. Input* merupakan nilai variabel pada *task* sensor. Pada *task 9* juga dilakukan proses pengiriman data ke *Open*CM9.04. Penerapan teknik enkoding digital seperti dijelaskan pada sebelumnya merujuk pada penelitian (Setiawan, 2018) yang menerapkan implementasi teknik enkoding digital untuk pengambilan keputusan robot dengan 8 sensor *ultrasonic*. Berikut implementasi dari pengolahan data dengan teknik enkoding digital pada Tabel 5.9.

Tabel 5. 5 Implementasi Pengolahan Data Dengan Teknik Enkoding

Baris	Kode Program
1	A = (!IA && !IB && ID && !IE && IG && !IH) ||
2	(!IA && !IB && IC && !IE && IG && !IH) ||
3	(!IA && !IB && IE && IF && !IG && !IH) ||
4	(!IA && !IB && ID && IF && !IG && !IH) ||
5	(!IA && !IB && IC && IF && !IG && !IH) ||
6	(!IA && !IB && ID && IE && IG && !IH) ||
7	(!IA && !IB && IC && IE && IG && !IH) ||
8	(!IA && !IB && !IC && IE &&!IF && !IG && !IH);
9	B = (!IA && !IB && !IC && ID && !IE && IG) ||
10	(!IA && !IB && !IC && ID && !IE && !IF && IH) ||
11	(!IA && !IB && !IC && ID && !IE && IF && IH) ||
12	(!IA && !IB && !IC && !IE && IF && IG && !IH) ||
13	(!IA && !IB && !IC && IE && IG) ||
14	(!IA && !IB && !IC && IE && !IF && !IG && IH) ||
15	(!IA && !IB && !IC && IE && IF && !IG && IH) ||
16	(!IA && !IB && !IC && !ID && IF && !IG && !IH);
17	C = (IA && !IB && !IC && !ID && IF) ||
18	(IA && !IB && !IC && !ID && IE && !IF) ||
19	(!IA && !IB && !IC && !ID && !IE && IG) ||
20	(!IA && !IB && !IC && !ID && !IE && IF && IH) ||
21	(!IA && !IB && !IC && !ID && IE && IF && IH) ||
22	(!IA && !IB && !IC && !ID && IE && !IF && IH);
23	D = (IB && !IC && !ID && !IE && IG) ||
24	(IB && !IC && !ID && !IE && IF) ||
25	(!IB && !IC && !ID && !IE && !IF && IH) ||
26	(IA && !IB && !IC && !ID && !IE && IF) ||
27	(IA && !IB && !IC && !ID && !IE && !IF && IG && !IH);
28	E = (IA && !IB && !IC && !ID && !IE && !IF && !IG) ||
29	(IA && IC && !ID && !IE && !IF && IG) ||
30	(IA && IC && !ID && !IE && !IF && !IG && IH) ||
31	(IB && !IC && !ID && !IE && !IF && IG) ||
32	(IB && !IC && !ID && !IE && !IF && !IG && IH) ||
33	(IA && IB && !IC && !ID && !IE && !IF && !IG) ||
34	(!IA && IC && !ID && !IE && !IF && IG) ||
35	(!IA && IC && !ID && !IE && !IF && !IG && IH);
36	F = (!IA && IC && !IE && !IF && !IG && IH) ||
37	(!IA && !IB && ID && !IE && !IF && !IG && IH) ||
38	(!IA && IB && ID && !IE && !IF && !IG && IH) ||
39	(!IA && IB && IC && !ID && !IE && !IF && !IG) ||
40	(IA && IC && !IE && !IF && !IG) ||
41	(IA && !IB && !IC && ID && !IE && !IF && !IG) ||
42	(IA && IB && !IC && ID && !IE && !IF && !IG) ||
43	(IB && !IC && !ID && !IE && !IF && !IG && !IH);
44	G = (!IA && IC && !IE && !IF && !IG && !IH) ||
45	(!IA && IB && !IC && ID && !IE && !IF && !IG && !IH) ||
46	(!IA && IB && IE && !IF && !IG && !IH) ||
47	(IA && IE && !IF && !IG && !IH) ||
48	(IA && IB && ID && !IE && !IF && !IG && !IH) ||
49	(IA && !IB && ID && !IE && !IF && !IG && !IH);

Baris	Kode Program
50	`  H = (!IA && IB && IE && !IF && !IG && !IH) ||`
51	`      (!IA && !IB && IC && !ID && IE && !IF && !IG && !IH) ||`
52	`      (!IA && IB && IF && !IG && !IH) ||`
53	`      (!IA && IC && IF && !IG && !IH) ||`
54	`      (!IA && !IB && ID && !IF && !IG && !IH);`
55	`  if (KondisiSebelumnya == "A") {`
56	`    //Serial.println("masuk A");`
57	`    lcd.setCursor(0,2);`
58	`    lcd.print ("               ");`
59	`    lcd.setCursor(0,2);`
60	`    lcd.print ("masuk A");`
61	`    if (A == 1) {`
62	`     //Serial.println("Gerak A");`
63	`     lcd.setCursor(0,0);`
64	`     lcd.print ("Gerak A");`
65	`     function = 3;`
66	`     Serial.print("*");`
67	`     Serial.print(function);`
68	`     Serial.println(",120,1000,1000#");`
69	`     KondisiSebelumnya = "A";`
70	`    }`
71	`    else if (B == 1) {`
72	`     // Serial.println("Gerak B");`
73	`     lcd.setCursor(0,0);`
74	`      lcd.print ("               ");`
75	`      lcd.setCursor(0,0);`
76	`      lcd.print ("Gerak B");`
77	`      function = 6;`
78	`      Serial.print("*");`
79	`      Serial.print(function);`
80	`      Serial.println(",120,1000,1000#");`
81	`      KondisiSebelumnya = "B";`
82	`    }`
83	`    else if (H == 1) {`
84	`      //Serial.println("Gerak H");`
85	`      lcd.setCursor(0,0);`
86	`      lcd.print ("               ");`
87	`      lcd.setCursor(0,0);`
88	`      lcd.print ("Gerak B");`
89	`      function = 6;`
90	`      Serial.print("*");`
91	`      Serial.print(function);`
92	`      Serial.println(",120,1000,1000#");`
93	`      KondisiSebelumnya = "H";`
94	`    }`
95	`    else if (C == 1) {`
96	`      //Serial.println("Gerak C");`
97	`      lcd.setCursor(0,0);`
98	`      lcd.print ("               ");`
99	`      lcd.setCursor(0,0);`
100	`      lcd.print ("Gerak C");`
101	`      function = 1;`
102	`      Serial.print("*");`
103	`      Serial.print(function);`
104	`      Serial.println(",120,1000,1000#");`
105	`      KondisiSebelumnya = "C";`
106	`    }`
	`    else if (G == 1) {`
	`      //Serial.println("Gerak G");`
	`      lcd.setCursor(0,0);`
	`      lcd.print ("               ");`
	`      lcd.setCursor(0,0);`
	`      lcd.print ("Gerak G");`
	`      function = 2;`

Baris	Kode Program
107	`      Serial.print("*");`
108	`      Serial.print(function);`
109	`      Serial.println(",120,1000,1000#");`
110	`      KondisiSebelumnya = "G";`
111	`    }`
112	`    else if (D == 1) {`
113	`      //Serial.println("Gerak D");`
114	`      lcd.setCursor(0,0);`
115	`      lcd.print ("               ");`
116	`      lcd.setCursor(0,0);`
117	`      lcd.print ("Gerak D");`
118	`      function = 8;`
119	`      Serial.print("*");`
120	`      Serial.print(function);`
121	`      Serial.println(",120,1000,1000#");`
122	`      KondisiSebelumnya = "D";`
123	`    }`
124	`    else if (F == 1) {`
125	`      //Serial.println("Gerak F");`
126	`      lcd.setCursor(0,0);`
127	`      lcd.print ("               ");`
128	`      lcd.setCursor(0,0);`
129	`      lcd.print ("Gerak F");`
130	`      function = 7;`
131	`      Serial.print("*");`
132	`      Serial.print(function);`
133	`      Serial.println(",120,1000,1000#");`
134	`      KondisiSebelumnya = "F";`
135	`    }`
136	`    else if (E == 1) {`
137	`      //Serial.println("Gerak E");`
138	`      lcd.setCursor(0,0);`
139	`      lcd.print ("               ");`
140	`      lcd.setCursor(0,0);`
141	`      lcd.print ("Gerak E");`
142	`      function = 4;`
143	`      Serial.print("*");`
144	`      Serial.print(function);`
145	`      Serial.println(",120,1000,1000#");`
146	`      KondisiSebelumnya = "E";`
147	`    }`
148	`    else {`
149	`     //Serial.println("DIAM");`
150	`     lcd.setCursor(0,0);`
151	`      lcd.print ("               ");`
152	`      lcd.setCursor(0,0);`
153	`      function = 3;`
154	`      Serial.print("*");`
	`      Serial.print(function);`
	`      Serial.println(",120,1000,1000#");`
	`      lcd.print ("DIAM");`
	`    }`
	`  }`

Dari Tabel 5.5 merupakan implementasi teknik enkoding digital yang dihasilkan dari perumusan 8 nilai *input*. Baris 1 sampai 54 merupakan implementasi teknik enkoding digital dan baris 55-154 penentuan pergerakan robot yang melihat kondisi pergerakan arah gerak robot sebelumnya beserta pengiriman data ke OpenCM9.04.

5.2.2.4 Implementasi Pengontrolan Gerak Kaki Robot Quadruped

Gerak kaki robot *quadruped* membutuhkan beberapa tahapan agar robot dapat bergerak dengan baik. Data pergerakan robot dapat dilihat pada Tabel 5.5, selanjutnya data akan diterima *Open*CM9.04 dan diparsing untuk menentukan arah dan kecepatan dari robot. Kode program untuk pengontrolan gerak kaki robot dapat dilihat pada Tabel 5.6, selain itu terdapat kode program untuk gerakan manuver robot yaitu gerakan maju dapat dilihat pada Tabel 5.7, pergerakan mundur pada tabel 5.8, pergerakan kanan pada Tabel 5.9, pergerakan kiri pada Tabel 5.10, pergerakan serong kanan depan pada Tabel 5.11, pergerakan serong kiri depan pada Tabel 5.12, pergerakan serong belakang kanan pada Tabel 5.13, pergerakan serong kiri belakang pada Tabel 5.14. Pergerakan setiap arah terjadi karena dalam program akan memberi nilai derajat pada setiap motor.

Tabel 5. 6 Kode Program Kontrol Gerak Kaki

Baris	Kode Program
1	`Dxl.begin(3);`
2	`  Dxl.jointMode(ID_NUM1);`
3	`  Dxl.jointMode(ID_NUM2);`
4	`  Dxl.jointMode(ID_NUM3);`
5	`  Dxl.jointMode(ID_NUM4);`
6	`  Dxl.jointMode(ID_NUM5);`
7	`  Dxl.jointMode(ID_NUM6);`
8	`  Dxl.jointMode(ID_NUM7);`
9	`  Dxl.jointMode(ID_NUM8);`
10	`  Dxl.jointMode(ID_NUM9);`
11	`  Dxl.jointMode(ID_NUM10);`
12	`  Dxl.jointMode(ID_NUM11);`
13	`  Dxl.jointMode(ID_NUM12);`
14	`  Stand();`
15	`  //  delay(3000);`
16	`  SerialUSB.begin(115200);`
17	`  Serial1.begin(115200);`
18	`  delay(2000);`
19	`//  SerialUSB.println("Robot Ready);`
20	`}`
21	`void loop() {`
22	`//SerialUSB.println("Robot Ready");`
23	`  if (Serial1.available() > 0) {`
24	`    char inChar = (char)Serial1.read();`
25	`    dataIn += inChar;`
26	`    //    SerialUSB.println(dataIn);`
27	`    if (inChar == '\n') {`
28	`      parsing = true;`
29	`    }`
30	`  }`
31	`  if (parsing) {`
32	`    parsingData();`
33	`    parsing = false;`
34	`    dataIn = "";`
35	`    int function = dt[0].toInt();`
36	`    int dely = dt[1].toInt();`
37	`    int speedL = dt[2].toInt();`
38	`    int speedR = dt[3].toInt();`
39	`    Serial.print("Method = ");`
40	`    Serial.println(function);`
41	`    Serial.print("Delai = ");`
42	`    Serial.println(dely);`
43	`    Serial.print("Speed L = ");`

```
44        Serial.println(speedL);
45        Serial.print("Speed R = ");
46        Serial.println(speedR);
47        if (function == 1) {
48          forward(dely, speedL, speedR);
49        } else if (function == 2) {
50          backward(dely, speedL, speedR);
51        } else if (function == 3) {
52          right(dely, speedL, speedR);
53        } else if (function == 4) {
54          left(dely, speedL, speedR);
55        } else if (function == 5) {
56          serkaback(dely, speedL, speedR);
57        } else if (function == 6) {
58          serkafront(dely, speedL, speedR);
59        } else if (function == 7) {
60          serkiback(dely, speedL, speedR);
61        } else if (function == 8) {
62          serkifront(dely, speedL, speedR);
63        }else if (function == 9) {
64          Stand();
65        }else{
66          Serial.println("nothing");
67        }
68        SerialUSB.print("\n");
69      }
70    }
71    void parsingData() {
72      int j = 0;
73      //kirim data yang telah diterima sebelumnya
74      SerialUSB.print("data masuk : ");
75      SerialUSB.print(dataIn);
76    //  SerialUSB.print("\n");
77      dt[j] = "";
78      for (i = 1; i < dataIn.length(); i++) {
79        if ((dataIn[i] == '#') || (dataIn[i] == ',')) {
80          j++;
81          dt[j] = "";
82        } else {
          dt[j] = dt[j] + dataIn[i];
        }
                                  }
```

Dalam kode program kontrol gerak kaki pada Tabel 5.6 dapat dilihat proses pertama baris 1-17 merupakan deklarasi servo yang digunakan dan nilai *baudrate* yang digunakan. Sebelum masuk pada *class loop* yang memiliki fungsi sebagai proses utama pemilihan gerak kaki robot dalam program terdapat *class parsing*Data pada baris 69 sampai 82 yang memiliki fungsi untuk memisah data yang telah diterima oleh *Open*CM9.04. Pada proses utama setelah data diparsing akan disimpan pada variabel baru yang nanti akan dikirim pada program gerakan kaki robot.

Kode program gerakan kaki terdapat 8 gerakan sesuai dengan perancangan dan setiap gerakan memiliki kode program masing-masing karena servo dikontrol menggunakan nilai derajat. Kode program pertama pada Tabel 5.7 untuk gerakan robot arah ke depan. Program berisi untuk memberi nilai derajat pada masing-masing servo setiap melakukan pergerakan, pada pergerakan arah depan tedapat 4 langkah untuk membuat robot bergerak ke arah depan.

Tabel 5. 7 Gerakan Robot Arah Depan

Baris	Kode Program
1	`void forward(int delai,int spdl, int spdr) {`
2	`  step_f1(spdl, spdr);`
3	`  delay(delai);`
4	`  step_f2(spdl, spdr);`
5	`  delay(delai);`
6	`  step_f3(spdl, spdr);`
7	`  delay(delai);`
8	`  step_f4(spdl, spdr);`
9	`  delay(delai);`
10	`}`
11	`void step_f1(int spdl, int spdr) {`
12	`  Dxl.setPosition(ID_NUM1, 512, spdr);`
13	`  Dxl.setPosition(ID_NUM2, 362, spdr);`
14	`  Dxl.setPosition(ID_NUM3, 372, spdr);`
15	`  Dxl.setPosition(ID_NUM4, 562, spdl);`
16	`  Dxl.setPosition(ID_NUM5, 412, spdl);`
17	`  Dxl.setPosition(ID_NUM6, 412, spdl);`
18	`  Dxl.setPosition(ID_NUM7, 512, spdr);`
19	`  Dxl.setPosition(ID_NUM8, 362, spdr);`
20	`  Dxl.setPosition(ID_NUM9, 362, spdr);`
21	`  Dxl.setPosition(ID_NUM10, 462, spdl);`
22	`  Dxl.setPosition(ID_NUM11, 412, spdl);`
23	`  Dxl.setPosition(ID_NUM12, 412, spdl);`
24	`}`
25	`void step_f2(int spdl, int spdr) {`
26	`  Dxl.setPosition(ID_NUM1, 512, spdr);`
27	`  Dxl.setPosition(ID_NUM2, 362, spdr);`
28	`  Dxl.setPosition(ID_NUM3, 372, spdr);`
29	`  Dxl.setPosition(ID_NUM4, 562, spdl);`
30	`  Dxl.setPosition(ID_NUM5, 362, spdl);`
31	`  Dxl.setPosition(ID_NUM6, 362, spdl);`
32	`  Dxl.setPosition(ID_NUM7, 512, spdr);`
33	`  Dxl.setPosition(ID_NUM8, 362, spdr);`
34	`  Dxl.setPosition(ID_NUM9, 362, spdr);`
35	`  Dxl.setPosition(ID_NUM10, 462, spdl);`
36	`  Dxl.setPosition(ID_NUM11, 362, spdl);`
37	`  Dxl.setPosition(ID_NUM12, 362, spdl);`
38	`}`
39	`void step_f3(int spdl, int spdr) {`
40	`  Dxl.setPosition(ID_NUM1, 562, spdr);`
41	`  Dxl.setPosition(ID_NUM2, 412, spdr);`
42	`  Dxl.setPosition(ID_NUM3, 412, spdr);`
43	`  Dxl.setPosition(ID_NUM4, 512, spdl);`
44	`  Dxl.setPosition(ID_NUM5, 362, spdl);`
45	`  Dxl.setPosition(ID_NUM6, 362, spdl);`
46	`  Dxl.setPosition(ID_NUM7, 462, spdr);`
47	`  Dxl.setPosition(ID_NUM8, 412, spdr);`
48	`  Dxl.setPosition(ID_NUM9, 412, spdr);`
49	`  Dxl.setPosition(ID_NUM10, 512, spdl);`
50	`  Dxl.setPosition(ID_NUM11, 362, spdl);`
51	`  Dxl.setPosition(ID_NUM12, 372, spdl);`
52	`}`
53	`void step_f4(int spdl, int spdr) {`
54	`  Dxl.setPosition(ID_NUM1, 562, spdr);`
55	`  Dxl.setPosition(ID_NUM2, 362, spdr);`
56	`  Dxl.setPosition(ID_NUM3, 362, spdr);`
57	`  Dxl.setPosition(ID_NUM4, 512, spdl);`
58	`  Dxl.setPosition(ID_NUM5, 362, spdl);`
59	`  Dxl.setPosition(ID_NUM6, 362, spdl);`
60	`  Dxl.setPosition(ID_NUM7, 462, spdr);`
61	`  Dxl.setPosition(ID_NUM8, 362, spdr);`
62	`  Dxl.setPosition(ID_NUM9, 362, spdr);`

Baris	Kode Program
62	Dxl.setPosition(ID_NUM10, 512, spdl);
63	Dxl.setPosition(ID_NUM11, 362, spdl);
64	Dxl.setPosition(ID_NUM12, 372, spdl);
65	}

Kode program pada Tabel 5.8 untuk robot melakukan gerakan ke arah belakang. Program berisi untuk memberi nilai derajat pada masing-masing servo setiap melakukan pergerakan, pada pergerakan arah belakang tedapat 4 langkah untuk membuat robot bergerak ke arah belakang.

Tabel 5. 8 Gerakan Robot Arah Belakang

Baris	Kode Program
1	void backward(int delai,int spdl, int spdr) {
2	step_b1(spdl, spdr);
3	delay(delai);
4	step_b2(spdl, spdr);
5	delay(delai);
6	step_b3(spdl, spdr);
7	delay(delai);
8	step_b4(spdl, spdr);
9	delay(delai);
10	}
11	void step_b1(int spdl, int spdr) {
12	Dxl.setPosition(ID_NUM1, 512, spdl);
13	Dxl.setPosition(ID_NUM2, 362, spdl);
14	Dxl.setPosition(ID_NUM3, 362, spdl);
15	Dxl.setPosition(ID_NUM4, 462, spdr);
16	Dxl.setPosition(ID_NUM5, 412, spdr);
17	Dxl.setPosition(ID_NUM6, 412, spdr);
18	Dxl.setPosition(ID_NUM7, 512, spdl);
19	Dxl.setPosition(ID_NUM8, 362, spdl);
20	Dxl.setPosition(ID_NUM9, 362, spdl);
21	Dxl.setPosition(ID_NUM10, 562, spdr);
22	Dxl.setPosition(ID_NUM11, 412, spdr);
23	Dxl.setPosition(ID_NUM12, 412, spdr);
24	}
25	void step_b2(int spdl, int spdr) {
26	Dxl.setPosition(ID_NUM1, 512, spdl);
27	Dxl.setPosition(ID_NUM2, 362, spdl);
28	Dxl.setPosition(ID_NUM3, 362, spdl);
29	Dxl.setPosition(ID_NUM4, 462, spdr);
30	Dxl.setPosition(ID_NUM5, 362, spdr);
31	Dxl.setPosition(ID_NUM6, 372, spdr);
32	Dxl.setPosition(ID_NUM7, 512, spdl);
33	Dxl.setPosition(ID_NUM8, 362, spdl);
34	Dxl.setPosition(ID_NUM9, 372, spdl);
35	Dxl.setPosition(ID_NUM10, 562, spdr);
36	Dxl.setPosition(ID_NUM11, 362, spdr);
37	Dxl.setPosition(ID_NUM12, 362, spdr);
38	}
39	void step_b3(int spdl, int spdr) {
40	Dxl.setPosition(ID_NUM1, 462, spdl);
41	Dxl.setPosition(ID_NUM2, 412, spdl);
42	Dxl.setPosition(ID_NUM3, 412, spdl);
43	Dxl.setPosition(ID_NUM4, 512, spdr);
44	Dxl.setPosition(ID_NUM5, 362, spdr);
45	Dxl.setPosition(ID_NUM6, 372, spdr);
46	Dxl.setPosition(ID_NUM7, 562, spdl);
47	Dxl.setPosition(ID_NUM8, 412, spdl);

Baris	Kode Program
48	Dxl.setPosition(ID_NUM9, 412, spdl);
49	Dxl.setPosition(ID_NUM10, 512, spdr);
50	Dxl.setPosition(ID_NUM11, 362, spdr);
51	Dxl.setPosition(ID_NUM12, 362, spdr);
52	}
53	void step_b4(int spdl, int spdr) {
54	Dxl.setPosition(ID_NUM1, 462, spdl);
55	Dxl.setPosition(ID_NUM2, 362, spdl);
56	Dxl.setPosition(ID_NUM3, 372, spdl);
57	Dxl.setPosition(ID_NUM4, 512, spdr);
58	Dxl.setPosition(ID_NUM5, 362, spdr);
59	Dxl.setPosition(ID_NUM6, 372, spdr);
60	Dxl.setPosition(ID_NUM7, 562, spdl);
61	Dxl.setPosition(ID_NUM8, 362, spdl);
62	Dxl.setPosition(ID_NUM9, 372, spdl);
63	Dxl.setPosition(ID_NUM10, 512, spdr);
64	Dxl.setPosition(ID_NUM11, 362, spdr);
65	Dxl.setPosition(ID_NUM12, 362, spdr);
	}

Kode program pada Tabel 5.9 untuk robot melakukan gerakan ke arah Kanan. Program berisi untuk memberi nilai derajat pada masing-masing servo setiap melakukan pergerakan, pada pergerakan arah kanan tedapat 4 langkah untuk membuat robot bergerak ke arah kanan.

Tabel 5. 9 Gerakan Robot Arah Kanan

Baris	Kode Program
1	void right(int delai,int spdl, int spdr) {
2	step_r1(spdl, spdr);
3	delay(delai);
4	step_r2(spdl, spdr);
5	delay(delai);
6	step_r3(spdl, spdr);
7	delay(delai);
8	step_r4(spdl, spdr);
9	delay(delai);
10	}
11	void step_r1(int spdl, int spdr) {
12	Dxl.setPosition(ID_NUM1, 562, spdl);
13	Dxl.setPosition(ID_NUM2, 412, spdl);
14	Dxl.setPosition(ID_NUM3, 412, spdl);
15	Dxl.setPosition(ID_NUM4, 512, spdl);
16	Dxl.setPosition(ID_NUM5, 362, spdl);
17	Dxl.setPosition(ID_NUM6, 362, spdl);
18	Dxl.setPosition(ID_NUM7, 462, spdr);
19	Dxl.setPosition(ID_NUM8, 412, spdr);
20	Dxl.setPosition(ID_NUM9, 412, spdr);
21	Dxl.setPosition(ID_NUM10, 512, spdr);
22	Dxl.setPosition(ID_NUM11, 362, spdr);
23	Dxl.setPosition(ID_NUM12, 372, spdr);
24	}
25	void step_r2(int spdl, int spdr) {
26	Dxl.setPosition(ID_NUM1, 562, spdl);
27	Dxl.setPosition(ID_NUM2, 362, spdl);
28	Dxl.setPosition(ID_NUM3, 362, spdl);
29	Dxl.setPosition(ID_NUM4, 512, spdl);
30	Dxl.setPosition(ID_NUM5, 362, spdl);
31	Dxl.setPosition(ID_NUM6, 362, spdl);
32	Dxl.setPosition(ID_NUM7, 462, spdr);

Baris	Kode Program
33	Dxl.setPosition(ID_NUM8, 362, spdr);
34	Dxl.setPosition(ID_NUM9, 372, spdr);
35	Dxl.setPosition(ID_NUM10, 512, spdr);
36	Dxl.setPosition(ID_NUM11, 362, spdr);
37	Dxl.setPosition(ID_NUM12, 372, spdr);
38	}
39	void step_r3(int spdl, int spdr) {
40	Dxl.setPosition(ID_NUM1, 512, spdl);
41	Dxl.setPosition(ID_NUM2, 362, spdl);
42	Dxl.setPosition(ID_NUM3, 362, spdl);
43	Dxl.setPosition(ID_NUM4, 462, spdl);
44	Dxl.setPosition(ID_NUM5, 412, spdl);
45	Dxl.setPosition(ID_NUM6, 412, spdl);
46	Dxl.setPosition(ID_NUM7, 512, spdr);
47	Dxl.setPosition(ID_NUM8, 362, spdr);
48	Dxl.setPosition(ID_NUM9, 372, spdr);
49	Dxl.setPosition(ID_NUM10, 562, spdr);
50	Dxl.setPosition(ID_NUM11, 412, spdr);
51	Dxl.setPosition(ID_NUM12, 412, spdr);
52	}
53	void step_r4(int spdl, int spdr) {
54	Dxl.setPosition(ID_NUM1, 512, spdl);
55	Dxl.setPosition(ID_NUM2, 362, spdl);
56	Dxl.setPosition(ID_NUM3, 362, spdl);
57	Dxl.setPosition(ID_NUM4, 462, spdl);
58	Dxl.setPosition(ID_NUM5, 362, spdl);
59	Dxl.setPosition(ID_NUM6, 362, spdl);
60	Dxl.setPosition(ID_NUM7, 512, spdr);
61	Dxl.setPosition(ID_NUM8, 362, spdr);
62	Dxl.setPosition(ID_NUM9, 372, spdr);
63	Dxl.setPosition(ID_NUM10, 562, spdr);
64	Dxl.setPosition(ID_NUM11, 362, spdr);
65	Dxl.setPosition(ID_NUM12, 372, spdr);
	}

Kode program pada Tabel 5.10 untuk robot melakukan gerakan ke arah Kiri. Program berisi untuk memberi nilai derajat pada masing-masing servo setiap melakukan pergerakan, pada pergerakan arah kiri tedapat 4 langkah untuk membuat robot bergerak ke arah kiri.

Tabel 5. 10 Gerakan Robot Arah Kiri

Baris	Kode Program
1	void left(int delai,int spdl, int spdr) {
2	step_l1(spdl, spdr);
3	delay(delai);
4	step_l2(spdl, spdr);
5	delay(delai);
6	step_l3(spdl, spdr);
7	delay(delai);
8	step_l4(spdl, spdr);
9	delay(delai);
10	}
11	void step_l1(int spdl, int spdr) {
12	Dxl.setPosition(ID_NUM1, 462, spdr);
13	Dxl.setPosition(ID_NUM2, 412, spdr);
14	Dxl.setPosition(ID_NUM3, 412, spdr);
15	Dxl.setPosition(ID_NUM4, 512, spdr);
16	Dxl.setPosition(ID_NUM5, 362, spdr);
17	Dxl.setPosition(ID_NUM6, 372, spdr);

Baris	Kode Program
18	Dxl.setPosition(ID_NUM7, 562, spdl);
19	Dxl.setPosition(ID_NUM8, 412, spdl);
20	Dxl.setPosition(ID_NUM9, 412, spdl);
21	Dxl.setPosition(ID_NUM10, 512, spdl);
22	Dxl.setPosition(ID_NUM11, 362, spdl);
23	Dxl.setPosition(ID_NUM12, 362, spdl);
24	}
25	void step_l2(int spdl, int spdr) {
26	Dxl.setPosition(ID_NUM1, 462, spdr);
27	Dxl.setPosition(ID_NUM2, 362, spdr);
28	Dxl.setPosition(ID_NUM3, 372, spdr);
29	Dxl.setPosition(ID_NUM4, 512, spdr);
30	Dxl.setPosition(ID_NUM5, 362, spdr);
31	Dxl.setPosition(ID_NUM6, 372, spdr);
32	Dxl.setPosition(ID_NUM7, 562, spdl);
33	Dxl.setPosition(ID_NUM8, 362, spdl);
34	Dxl.setPosition(ID_NUM9, 362, spdl);
35	Dxl.setPosition(ID_NUM10, 512, spdl);
36	Dxl.setPosition(ID_NUM11, 362, spdl);
37	Dxl.setPosition(ID_NUM12, 362, spdl);
38	}
39	void step_l3(int spdl, int spdr) {
40	Dxl.setPosition(ID_NUM1, 512, spdr);
41	Dxl.setPosition(ID_NUM2, 362, spdr);
42	Dxl.setPosition(ID_NUM3, 372, spdr);
43	Dxl.setPosition(ID_NUM4, 562, spdr);
44	Dxl.setPosition(ID_NUM5, 412, spdr);
45	Dxl.setPosition(ID_NUM6, 412, spdr);
46	Dxl.setPosition(ID_NUM7, 512, spdl);
47	Dxl.setPosition(ID_NUM8, 362, spdl);
48	Dxl.setPosition(ID_NUM9, 362, spdl);
49	Dxl.setPosition(ID_NUM10, 462, spdl);
50	Dxl.setPosition(ID_NUM11, 412, spdl);
51	Dxl.setPosition(ID_NUM12, 412, spdl);
52	}
53	void step_l4(int spdl, int spdr) {
54	Dxl.setPosition(ID_NUM1, 512, spdr);
55	Dxl.setPosition(ID_NUM2, 362, spdr);
56	Dxl.setPosition(ID_NUM3, 372, spdr);
57	Dxl.setPosition(ID_NUM4, 562, spdr);
58	Dxl.setPosition(ID_NUM5, 362, spdr);
59	Dxl.setPosition(ID_NUM6, 372, spdr);
60	Dxl.setPosition(ID_NUM7, 512, spdl);
61	Dxl.setPosition(ID_NUM8, 362, spdl);
62	Dxl.setPosition(ID_NUM9, 362, spdl);
63	Dxl.setPosition(ID_NUM10, 462, spdl);
64	Dxl.setPosition(ID_NUM11, 362, spdl);
65	Dxl.setPosition(ID_NUM12, 362, spdl);
	}

Kode program pada Tabel 5.11 untuk robot melakukan gerakan ke arah serong kanan depan. Program berisi untuk memberi nilai derajat pada masing-masing servo setiap melakukan pergerakan, pada pergerakan arah serong kanan depan tedapat 5 langkah untuk membuat robot bergerak ke arah serong kanan depan.

Tabel 5. 11 Gerakan Robot Arah Serong Kanan Depan

Baris	Kode Program
1	`void serkafront(int delai,int spdl, int spdr) {`
2	`  step_srf1(spdl, spdr);`
3	`  delay(delai);`
4	`  step_srf2(spdl, spdr);`
5	`  delay(delai);`
6	`  step_srf3(spdl, spdr);`
7	`  delay(delai);`
8	`  step_srf4(spdl, spdr);`
9	`  delay(delai);`
10	`  step_srf5(spdl, spdr);`
11	`  delay(delai);`
12	`}`
13	`void step_srf1(int spdl, int spdr) {`
14	`  Dxl.setPosition(ID_NUM1, 512, spdl);`
15	`  Dxl.setPosition(ID_NUM2, 412, spdl);`
16	`  Dxl.setPosition(ID_NUM3, 362, spdl);`
17	`  Dxl.setPosition(ID_NUM4, 512, spdl);`
18	`  Dxl.setPosition(ID_NUM5, 362, spdl);`
19	`  Dxl.setPosition(ID_NUM6, 362, spdl);`
20	`  Dxl.setPosition(ID_NUM7, 512, spdr);`
21	`  Dxl.setPosition(ID_NUM8, 412, spdr);`
22	`  Dxl.setPosition(ID_NUM9, 362, spdr);`
23	`  Dxl.setPosition(ID_NUM10, 512, spdr);`
24	`  Dxl.setPosition(ID_NUM11, 362, spdr);`
25	`  Dxl.setPosition(ID_NUM12, 362, spdr);`
26	`}`
27	`void step_srf2(int spdl, int spdr) {`
28	`  Dxl.setPosition(ID_NUM1, 612, spdl);`
29	`  Dxl.setPosition(ID_NUM2, 412, spdl);`
30	`  Dxl.setPosition(ID_NUM3, 362, spdl);`
31	`  Dxl.setPosition(ID_NUM4, 512, spdl);`
32	`  Dxl.setPosition(ID_NUM5, 362, spdl);`
33	`  Dxl.setPosition(ID_NUM6, 362, spdl);`
34	`  Dxl.setPosition(ID_NUM7, 412, spdr);`
35	`  Dxl.setPosition(ID_NUM8, 412, spdr);`
36	`  Dxl.setPosition(ID_NUM9, 362, spdr);`
37	`  Dxl.setPosition(ID_NUM10, 512, spdr);`
38	`  Dxl.setPosition(ID_NUM11, 362, spdr);`
39	`  Dxl.setPosition(ID_NUM12, 362, spdr);`
40	`}`
41	`void step_srf3(int spdl, int spdr) {`
42	`  Dxl.setPosition(ID_NUM1, 612, spdl);`
43	`  Dxl.setPosition(ID_NUM2, 362, spdl);`
44	`  Dxl.setPosition(ID_NUM3, 362, spdl);`
45	`  Dxl.setPosition(ID_NUM4, 512, spdl);`
46	`  Dxl.setPosition(ID_NUM5, 362, spdl);`
47	`  Dxl.setPosition(ID_NUM6, 362, spdl);`
48	`  Dxl.setPosition(ID_NUM7, 412, spdr);`
49	`  Dxl.setPosition(ID_NUM8, 362, spdr);`
50	`  Dxl.setPosition(ID_NUM9, 362, spdr);`
51	`  Dxl.setPosition(ID_NUM10, 512, spdr);`
52	`  Dxl.setPosition(ID_NUM11, 362, spdr);`
53	`  Dxl.setPosition(ID_NUM12, 362, spdr);`
54	`}`
55	`void step_srf4(int spdl, int spdr) {`
56	`  Dxl.setPosition(ID_NUM1, 512, spdl);`
57	`  Dxl.setPosition(ID_NUM2, 362, spdl);`
58	`  Dxl.setPosition(ID_NUM3, 362, spdl);`
59	`  Dxl.setPosition(ID_NUM4, 512, spdl);`
60	`  Dxl.setPosition(ID_NUM5, 412, spdl);`
61	`  Dxl.setPosition(ID_NUM6, 362, spdl);`
	`  Dxl.setPosition(ID_NUM7, 512, spdr);`

Baris	Kode Program
62	Dxl.setPosition(ID_NUM8, 362, spdr);
63	Dxl.setPosition(ID_NUM9, 362, spdr);
64	Dxl.setPosition(ID_NUM10, 512, spdr);
65	Dxl.setPosition(ID_NUM11, 412, spdr);
66	Dxl.setPosition(ID_NUM12, 362, spdr);
67	}
68	void step_srf5(int spdl, int spdr) {
69	Dxl.setPosition(ID_NUM1, 512, spdl);
70	Dxl.setPosition(ID_NUM2, 362, spdl);
71	Dxl.setPosition(ID_NUM3, 362, spdl);
72	Dxl.setPosition(ID_NUM4, 512, spdl);
73	Dxl.setPosition(ID_NUM5, 362, spdl);
74	Dxl.setPosition(ID_NUM6, 412, spdl);
75	Dxl.setPosition(ID_NUM7, 512, spdr);
76	Dxl.setPosition(ID_NUM8, 362, spdr);
77	Dxl.setPosition(ID_NUM9, 362, spdr);
78	Dxl.setPosition(ID_NUM10, 512, spdr);
79	Dxl.setPosition(ID_NUM11, 362, spdr);
	Dxl.setPosition(ID_NUM12, 312, spdr);
	}

Kode program pada Tabel 5.12 untuk robot melakukan gerakan ke arah serong kiri depan. Program berisi untuk memberi nilai derajat pada masing-masing servo setiap melakukan pergerakan, pada pergerakan arah serong kiri depan tedapat 5 langkah untuk membuat robot bergerak ke arah serong kiri depan.

Tabel 5. 12 Gerakan Robot Arah Serong Kiri Depan

Baris	Kode Program
1	void serkifront(int delai,int spdl, int spdr) {
2	step_srfl1(spdl, spdr);
3	delay(delai);
4	step_srfl2(spdl, spdr);
5	delay(delai);
6	step_srfl3(spdl, spdr);
7	delay(delai);
8	step_srfl4(spdl, spdr);
9	delay(delai);
10	step_srfl5(spdl, spdr);
11	delay(delai);
12	}
13	void step_srfl1(int spdl, int spdr) {
14	Dxl.setPosition(ID_NUM1, 512, spdl);
15	Dxl.setPosition(ID_NUM2, 362, spdl);
16	Dxl.setPosition(ID_NUM3, 362, spdl);
17	Dxl.setPosition(ID_NUM4, 512, spdl);
18	Dxl.setPosition(ID_NUM5, 412, spdl);
19	Dxl.setPosition(ID_NUM6, 362, spdl);
20	Dxl.setPosition(ID_NUM7, 512, spdr);
21	Dxl.setPosition(ID_NUM8, 362, spdr);
22	Dxl.setPosition(ID_NUM9, 362, spdr);
23	Dxl.setPosition(ID_NUM10, 512, spdr);
24	Dxl.setPosition(ID_NUM11, 412, spdr);
25	Dxl.setPosition(ID_NUM12, 362, spdr);
26	}
27	void step_srfl2(int spdl, int spdr) {
28	Dxl.setPosition(ID_NUM1, 512, spdl);
29	Dxl.setPosition(ID_NUM2, 362, spdl);
30	Dxl.setPosition(ID_NUM3, 362, spdl);
31	Dxl.setPosition(ID_NUM4, 612, spdl);
32	Dxl.setPosition(ID_NUM5, 412, spdl);

Baris	Kode Program
33	Dxl.setPosition(ID_NUM6, 362, spdl);
34	Dxl.setPosition(ID_NUM7, 512, spdr);
35	Dxl.setPosition(ID_NUM8, 362, spdr);
36	Dxl.setPosition(ID_NUM9, 362, spdr);
37	Dxl.setPosition(ID_NUM10, 412, spdr);
38	Dxl.setPosition(ID_NUM11, 412, spdr);
39	Dxl.setPosition(ID_NUM12, 362, spdr);
40	}
41	void step_srfl3(int spdl, int spdr) {
42	Dxl.setPosition(ID_NUM1, 512, spdl);
43	Dxl.setPosition(ID_NUM2, 362, spdl);
44	Dxl.setPosition(ID_NUM3, 362, spdl);
45	Dxl.setPosition(ID_NUM4, 612, spdl);
46	Dxl.setPosition(ID_NUM5, 362, spdl);
47	Dxl.setPosition(ID_NUM6, 362, spdl);
48	Dxl.setPosition(ID_NUM7, 512, spdr);
49	Dxl.setPosition(ID_NUM8, 362, spdr);
50	Dxl.setPosition(ID_NUM9, 362, spdr);
51	Dxl.setPosition(ID_NUM10, 412, spdr);
52	Dxl.setPosition(ID_NUM11, 362, spdr);
53	Dxl.setPosition(ID_NUM12, 362, spdr);
54	}
55	void step_srfl4(int spdl, int spdr) {
56	Dxl.setPosition(ID_NUM1, 512, spdl);
57	Dxl.setPosition(ID_NUM2, 412, spdl);
58	Dxl.setPosition(ID_NUM3, 362, spdl);
59	Dxl.setPosition(ID_NUM4, 512, spdl);
60	Dxl.setPosition(ID_NUM5, 362, spdl);
61	Dxl.setPosition(ID_NUM6, 362, spdl);
62	Dxl.setPosition(ID_NUM7, 512, spdr);
63	Dxl.setPosition(ID_NUM8, 412, spdr);
64	Dxl.setPosition(ID_NUM9, 362, spdr);
65	Dxl.setPosition(ID_NUM10, 512, spdr);
66	Dxl.setPosition(ID_NUM11, 362, spdr);
67	Dxl.setPosition(ID_NUM12, 362, spdr);
68	}
69	void step_srfl5(int spdl, int spdr) {
70	Dxl.setPosition(ID_NUM1, 512, spdl);
71	Dxl.setPosition(ID_NUM2, 412, spdl);
72	Dxl.setPosition(ID_NUM3, 362, spdl);
73	Dxl.setPosition(ID_NUM4, 512, spdl);
74	Dxl.setPosition(ID_NUM5, 362, spdl);
75	Dxl.setPosition(ID_NUM6, 362, spdl);
76	Dxl.setPosition(ID_NUM7, 512, spdr);
77	Dxl.setPosition(ID_NUM8, 412, spdr);
78	Dxl.setPosition(ID_NUM9, 412, spdr);
79	Dxl.setPosition(ID_NUM10, 512, spdr);
	Dxl.setPosition(ID_NUM11, 362, spdr);
	Dxl.setPosition(ID_NUM12, 362, spdr);
	}

Kode program pada Tabel 5.13 untuk robot melakukan gerakan ke arah serong kanan belakang. Program berisi untuk memberi nilai derajat pada masing-masing servo setiap melakukan pergerakan, pada pergerakan serong kanan belakang depan tedapat 5 langkah untuk membuat robot bergerak ke arah serong kanan belakang.

Tabel 5. 13 Gerakan Robot Arah Serong Kanan Belakang

Baris	Kode Program
1	`void serkaback(int delai,int spdl, int spdr) {`
2	`  step_srbr1(spdl, spdr);`
3	`  delay(delai);`
4	`  step_srbr2(spdl, spdr);`
5	`  delay(delai);`
6	`  step_srbr3(spdl, spdr);`
7	`  delay(delai);`
8	`  step_srbr4(spdl, spdr);`
9	`  delay(delai);`
10	`  step_srbr5(spdl, spdr);`
11	`  delay(delai);`
12	`}`
13	`void step_srbr1(int spdl, int spdr) {`
14	`  Dxl.setPosition(ID_NUM1, 512, spdl);`
15	`  Dxl.setPosition(ID_NUM2, 362, spdl);`
16	`  Dxl.setPosition(ID_NUM3, 362, spdl);`
17	`  Dxl.setPosition(ID_NUM4, 512, spdl);`
18	`  Dxl.setPosition(ID_NUM5, 412, spdl);`
19	`  Dxl.setPosition(ID_NUM6, 362, spdl);`
20	`  Dxl.setPosition(ID_NUM7, 512, spdr);`
21	`  Dxl.setPosition(ID_NUM8, 362, spdr);`
22	`  Dxl.setPosition(ID_NUM9, 362, spdr);`
23	`  Dxl.setPosition(ID_NUM10, 512, spdr);`
24	`  Dxl.setPosition(ID_NUM11, 412, spdr);`
25	`  Dxl.setPosition(ID_NUM12, 362, spdr);`
26	`}`
27	`void step_srbr2(int spdl, int spdr) {`
28	`  Dxl.setPosition(ID_NUM1, 512, spdl);`
29	`  Dxl.setPosition(ID_NUM2, 362, spdl);`
30	`  Dxl.setPosition(ID_NUM3, 362, spdl);`
31	`  Dxl.setPosition(ID_NUM4, 412, spdl);`
32	`  Dxl.setPosition(ID_NUM5, 412, spdl);`
33	`  Dxl.setPosition(ID_NUM6, 362, spdl);`
34	`  Dxl.setPosition(ID_NUM7, 512, spdr);`
35	`  Dxl.setPosition(ID_NUM8, 362, spdr);`
36	`  Dxl.setPosition(ID_NUM9, 362, spdr);`
37	`  Dxl.setPosition(ID_NUM10, 612, spdr);`
38	`  Dxl.setPosition(ID_NUM11, 412, spdr);`
39	`  Dxl.setPosition(ID_NUM12, 362, spdr);`
40	`}`
41	`void step_srbr3(int spdl, int spdr) {`
42	`  Dxl.setPosition(ID_NUM1, 512, spdl);`
43	`  Dxl.setPosition(ID_NUM2, 362, spdl);`
44	`  Dxl.setPosition(ID_NUM3, 362, spdl);`
45	`  Dxl.setPosition(ID_NUM4, 412, spdl);`
46	`  Dxl.setPosition(ID_NUM5, 362, spdl);`
47	`  Dxl.setPosition(ID_NUM6, 362, spdl);`
48	`  Dxl.setPosition(ID_NUM7, 512, spdr);`
49	`  Dxl.setPosition(ID_NUM8, 362, spdr);`
50	`  Dxl.setPosition(ID_NUM9, 362, spdr);`
51	`  Dxl.setPosition(ID_NUM10, 612, spdr);`
52	`  Dxl.setPosition(ID_NUM11, 362, spdr);`
53	`  Dxl.setPosition(ID_NUM12, 362, spdr);`
54	`}`
55	`void step_srbr4(int spdl, int spdr) {`
56	`  Dxl.setPosition(ID_NUM1, 512, spdl);`
57	`  Dxl.setPosition(ID_NUM2, 412, spdl);`
58	`  Dxl.setPosition(ID_NUM3, 362, spdl);`
59	`  Dxl.setPosition(ID_NUM4, 512, spdl);`
60	`  Dxl.setPosition(ID_NUM5, 362, spdl);`
61	`  Dxl.setPosition(ID_NUM6, 362, spdl);`
	`  Dxl.setPosition(ID_NUM7, 512, spdr);`

Baris	Kode Program
62	Dxl.setPosition(ID_NUM8, 412, spdr);
63	Dxl.setPosition(ID_NUM9, 362, spdr);
64	Dxl.setPosition(ID_NUM10, 512, spdr);
65	Dxl.setPosition(ID_NUM11, 362, spdr);
66	Dxl.setPosition(ID_NUM12, 362, spdr);
67	}
68	void step_srbr5(int spdl, int spdr) {
69	Dxl.setPosition(ID_NUM1, 512, spdl);
70	Dxl.setPosition(ID_NUM2, 412, spdl);
71	Dxl.setPosition(ID_NUM3, 462, spdl);
72	Dxl.setPosition(ID_NUM4, 512, spdl);
73	Dxl.setPosition(ID_NUM5, 362, spdl);
74	Dxl.setPosition(ID_NUM6, 362, spdl);
75	Dxl.setPosition(ID_NUM7, 512, spdr);
76	Dxl.setPosition(ID_NUM8, 412, spdr);
77	Dxl.setPosition(ID_NUM9, 312, spdr);
78	Dxl.setPosition(ID_NUM10, 512, spdr);
79	Dxl.setPosition(ID_NUM11, 362, spdr);
	Dxl.setPosition(ID_NUM12, 362, spdr);
	}

Kode program pada Tabel 5.14 untuk robot melakukan gerakan ke arah serong kiri belakang. Program berisi untuk memberi nilai derajat pada masing-masing servo setiap melakukan pergerakan, pada pergerakan serong kiri belakang depan tedapat 5 langkah untuk membuat robot bergerak ke arah serong kiri belakang.

Tabel 5. 14 Gerakan Robot Arah Serong Kiri Belakang

Baris	Kode Program
1	void serkiback(int delai,int spdl, int spdr) {
2	step_srb1(spdl, spdr);
3	delay(delai);
4	step_srb2(spdl, spdr);
5	delay(delai);
6	step_srb3(spdl, spdr);
7	delay(delai);
8	step_srb4(spdl, spdr);
9	delay(delai);
10	step_srb5(spdl, spdr);
11	delay(delai);
12	}
13	void step_srb1(int spdl, int spdr) {
14	Dxl.setPosition(ID_NUM1, 512, spdl);
15	Dxl.setPosition(ID_NUM2, 412, spdl);
16	Dxl.setPosition(ID_NUM3, 362, spdl);
17	Dxl.setPosition(ID_NUM4, 512, spdl);
18	Dxl.setPosition(ID_NUM5, 362, spdl);
19	Dxl.setPosition(ID_NUM6, 362, spdl);
20	Dxl.setPosition(ID_NUM7, 512, spdr);
21	Dxl.setPosition(ID_NUM8, 412, spdr);
22	Dxl.setPosition(ID_NUM9, 362, spdr);
23	Dxl.setPosition(ID_NUM10, 512, spdr);
24	Dxl.setPosition(ID_NUM11, 362, spdr);
25	Dxl.setPosition(ID_NUM12, 362, spdr);
26	}
27	void step_srb2(int spdl, int spdr) {
28	Dxl.setPosition(ID_NUM1, 412, spdl);
29	Dxl.setPosition(ID_NUM2, 412, spdl);
30	Dxl.setPosition(ID_NUM3, 362, spdl);

Baris	Kode Program
31	Dxl.setPosition(ID_NUM4, 512, spdl);
32	Dxl.setPosition(ID_NUM5, 362, spdl);
33	Dxl.setPosition(ID_NUM6, 362, spdl);
34	Dxl.setPosition(ID_NUM7, 612, spdr);
35	Dxl.setPosition(ID_NUM8, 412, spdr);
36	Dxl.setPosition(ID_NUM9, 362, spdr);
37	Dxl.setPosition(ID_NUM10, 512, spdr);
38	Dxl.setPosition(ID_NUM11, 362, spdr);
39	Dxl.setPosition(ID_NUM12, 362, spdr);
40	}
41	void step_srb3(int spdl, int spdr) {
42	Dxl.setPosition(ID_NUM1, 412, spdl);
43	Dxl.setPosition(ID_NUM2, 362, spdl);
44	Dxl.setPosition(ID_NUM3, 362, spdl);
45	Dxl.setPosition(ID_NUM4, 512, spdl);
46	Dxl.setPosition(ID_NUM5, 362, spdl);
47	Dxl.setPosition(ID_NUM6, 362, spdl);
48	Dxl.setPosition(ID_NUM7, 612, spdr);
49	Dxl.setPosition(ID_NUM8, 362, spdr);
50	Dxl.setPosition(ID_NUM9, 362, spdr);
51	Dxl.setPosition(ID_NUM10, 512, spdr);
52	Dxl.setPosition(ID_NUM11, 362, spdr);
53	Dxl.setPosition(ID_NUM12, 362, spdr);
54	}
55	void step_srb4(int spdl, int spdr) {
56	Dxl.setPosition(ID_NUM1, 512, spdl);
57	Dxl.setPosition(ID_NUM2, 362, spdl);
58	Dxl.setPosition(ID_NUM3, 362, spdl);
59	Dxl.setPosition(ID_NUM4, 512, spdl);
60	Dxl.setPosition(ID_NUM5, 412, spdl);
61	Dxl.setPosition(ID_NUM6, 362, spdl);
62	Dxl.setPosition(ID_NUM7, 512, spdr);
63	Dxl.setPosition(ID_NUM8, 362, spdr);
64	Dxl.setPosition(ID_NUM9, 362, spdr);
65	Dxl.setPosition(ID_NUM10, 512, spdr);
66	Dxl.setPosition(ID_NUM11, 412, spdr);
67	Dxl.setPosition(ID_NUM12, 362, spdr);
68	}
69	void step_srb5(int spdl, int spdr) {
70	Dxl.setPosition(ID_NUM1, 512, spdl);
71	Dxl.setPosition(ID_NUM2, 362, spdl);
72	Dxl.setPosition(ID_NUM3, 362, spdl);
73	Dxl.setPosition(ID_NUM4, 512, spdl);
74	Dxl.setPosition(ID_NUM5, 362, spdl);
75	Dxl.setPosition(ID_NUM6, 312, spdl);
76	Dxl.setPosition(ID_NUM7, 512, spdr);
77	Dxl.setPosition(ID_NUM8, 362, spdr);
78	Dxl.setPosition(ID_NUM9, 362, spdr);
79	Dxl.setPosition(ID_NUM10, 512, spdr);
	Dxl.setPosition(ID_NUM11, 362, spdr);
	Dxl.setPosition(ID_NUM12, 412, spdr);
	}

BAB 6 PENGUJIAN DAN ANALISIS

6.1 Pengujian Sensor HC-SR04

Pengujian pertama ialah melakukan pengujian terhadap akurasi pembacaan sensor *ultrasonic* HC-SR04. Pengujian akan dilakukan dengan melakukan pembacaan sensor HC-SR04 dan membandingkannya dengan kondisi sebenarnya.

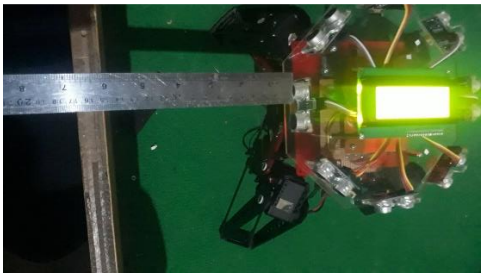
6.1.1 Tujuan Pengujian

Tujuan dari pengujian sensor HC-SR04 adalah untuk mengetahui tingkat akurasi dari pembacaan sensor HC-SR04 pada Arduino Mega2560 terhadap jarak sesungguhnya. Dengan melihat tingkat akurasi sensor maka dapat memperhitungkan kerja sensor yang berpengaruh pada sistem yang telah dibuat.

6.1.2 Prosedur Pengujian

Pengujian akurasi sensor HC-SR04 dilakukan dalam beberapa langkah sebagai berikut:

1) Menghubungkan mikrokontroler Arduino Mega2560 yang telah terhubung sensor HC-SR04 dengan laptop.
2) Jalankan kode program sensor HC-SR04 yang telah dibuat dan *upload* ke mikrokontroler Arduino Mega 2560.
3) Posisikan sensor yang diuji berhadapan dengan bidang datar dan ukur menggunakan alat ukur jarak sensor terhadap bidang datar tersebut, seperti pada Gambar 6.1.



Gambar 6. 1 Pengujian Sensor

4) Pengukuran dilakukan pada jarak terukur pada alat ukur adalah jarak sensor dengan bidang datar sejauh 5cm, 10cm, 15cm, 20cm, 25cm, 30cm.
5) Mengamati nilai jarak dari sensor ultrasonik HC-SR04 pada serial monitor dan membandingkan nilai yang terbaca dengan nilai jarak dari alat ukur.

6) Menentukan besarnya *error* pembacaan sensor dengan cara mengkonversi nilai jarak dari alat ukur dan nilai jarak dari sensor *ultrasonic* HC-SR04.

Terdapat cara untuk mengukur persentase *error* pembacaan sensor yaitu dengan menggunakan persamaan 6.1:

$$\text{Persentase } error = \frac{\text{Selisih nilai pembacaan}}{\text{Pembacaan alat ukur}} \times 100\% \quad (6.1)$$

Sedangkan menghitung selisih nilai pembacaan sensor dengan menggunakan Persamaan 6.2

$$\text{Selisih nilai pembacaan} = \text{alat ukur} - \text{pembacaan sensor} \quad (6.2)$$

6.1.3 Hasil Pengujian

Tabel 6. 1 Hasil Pengujian Sensor A

No	Jarak Pada Alat ukur	Jarak Terbaca Pada Serial Monitor	Selisih	*Error* (%)
1	40	40	0	0,0
2	30	30	0	0,0
3	25	25	0	0,0
4	20	20	0	0,0
5	15	15	0	0,0
6	10	10	0	0,0
7	5	5	0	0,0
Rata-rata				0,0

Tabel 6. 2 Hasil Pengujian Sensor B

No	Jarak Pada Alat ukur	Jarak Terbaca Pada Serial Monitor	Selisih	*Error* (%)
1	40	40	0	0,0
2	30	30	0	0,0
3	25	25	0	0,0
4	20	20	0	0,0
5	15	15	0	0,0
6	10	10	0	0,0
7	5	5	0	0,0
Rata-rata				0,0

Tabel 6. 3 Hasil Pengujian Sensor C

No	Jarak Pada Alat ukur	Jarak Terbaca Pada Serial Monitor	Selisih	*Error* (%)
1	40	40	0	0,0
2	30	30	0	0,0
3	25	25	0	0,0

No	Jarak Pada Alat ukur	Jarak Terbaca Pada Serial Monitor	Selisih	*Error* (%)
4	20	20	0	0,0
5	15	15	0	0,0
6	10	10	0	0,0
7	5	5	0	0,0
Rata-rata				0,0

Tabel 6. 4 Hasil Pengujian Sensor D

No	Jarak Pada Alat ukur	Jarak Terbaca Pada Serial Monitor	Selisih	*Error* (%)
1	40	40	0	0,0
2	30	30	0	0,0
3	25	25	0	0,0
4	20	20	0	0,0
5	15	15	0	0,0
6	10	10	0	0,0
7	5	5	0	0,0
Rata-rata				0,0

Tabel 6. 5 Hasil Pengujan Sensor E

No	Jarak Pada Alat ukur	Jarak Terbaca Pada Serial Monitor	Selisih	*Error* (%)
1	40	40	0	0,0
2	30	30	0	0,0
3	25	25	0	0,0
4	20	20	0	0,0
5	15	15	0	0,0
6	10	10	0	0,0
7	5	5	0	0,0
Rata-rata				0,0

Tabel 6. 6 Hasil Pengujian Sensor F

No	Jarak Pada Alat ukur	Jarak Terbaca Pada Serial Monitor	Selisih	*Error* (%)
1	40	40	0	0,0
2	30	30	0	0,0
3	25	25	0	0,0
4	20	20	0	0,0
5	15	15	0	0,0
6	10	10	0	0,0
7	5	5	0	0,0
Rata-rata				0,0

Tabel 6. 7 Hasil Pengujian Sensor G

No	Jarak Pada Alat ukur	Jarak Terbaca Pada Serial Monitor	Selisih	*Error* (%)
1	40	40	0	0,0
2	30	30	0	0,0
3	25	25	0	0,0
4	20	20	0	0,0
5	15	15	0	0,0
6	10	10	0	0,0
7	5	5	0	0,0
Rata-rata				0,0

Tabel 6. 8 Hasil Pengujian Sensor H

No	Jarak Pada Alat ukur	Jarak Terbaca Pada Serial Monitor	Selisih	*Error* (%)
1	40	40	0	0,0
2	30	30	0	0,0
3	25	25	0	0,0
4	20	20	0	0,0
5	15	15	0	0,0
6	10	10	0	0,0
7	5	5	0	0,0
Rata-rata				0,0

Tabel 6. 9 Hasil Pengujian Semua Sensor

Sensor Ultrasonik Ke-	Rata-Rata persentase *Error* (%)
1	0,0
2	0,0
3	0,0
4	0,0
5	0,0
6	0,0
7	0,0
8	0,0
Rata-rata	0,0

Berdasarkan pada persamaan (6.1) dan (6.2) untuk menentukan persentase *error* pembacaan sensor, data pengujian di tunjukan pada Tabel 6.1, Tabel 6.2, Tabel 6.3, Tabel 6.4, Tabel 6.5, Tabel 6.6, Tabel 6.7, dan Tabel 6.8. Pada Tabel 6.9 menunjukkan rata-rata *error* yang dimiliki sensor ialah 0%. Contoh perhitungan persentase *error* pada pengujian sensor :

Jarak pada alat ukur = 25cm
Jarak terbaca pada serial monitor = 25cm

$$\text{Selisih nilai pembacaan} = \text{pembacaan alat ukur} - \text{pembacaan sensor}$$
$$\text{Selisih nilai pembacaan} = 25\text{cm} - 25\text{cm}$$
$$\text{Selisih nilai pembacaan} = 0$$

$$\text{Persentase } error = \frac{\text{Selisih nilai pembacaan}}{\text{pembacaan alat ukur}} x100\%$$
$$\text{Persentase } error = \frac{0}{25} x100\%$$
$$\text{Persentase } error = 0\%$$

Untuk menghitung nilai rata-rata *error* keseluruhan pengujian adalah sebagai berikut:

$$\text{Rata} - \text{rata } error = \frac{\text{Jumlah persentase error}}{\text{Jumlah pengujian}}$$
$$\text{Rata} - \text{rata } error = \frac{0\% + 0\% + 0\% + 0\% + 0\% + 0\% + 0\% + 0\%}{8}$$
$$\text{Rata} - \text{rata } error = 0{,}0\%$$

Rata-rata *error* pada sensor *ultrasonic* HC-SR04 sebesar 0%, sehingga akurasi dari pembacaan sensor ialah sangatlah baik.

6.1.4 Analisis Hasil Pengujian

Dari data yang diperoleh pada subab hasil pengujian, didapatkan bahwa tingkat *error* pembacaan sensor *ultrasonic* yang dipakai 1 sampai dengan 8 sensor adalah 0% atau dapat dikatakan tingkat akurasi sensor sebesar 100%. Tingkat akurasi sensor *ultrasonic* sangatlah baik, namun sensor ini memiliki jarak jangkauan maksimal 400cm dan minimal 3cm.

6.2 Pengujian Waktu Eksekusi *Task* Dan Prioritas

Dalam sistem ini memiliki 9 *task* dengan susunan 8 *task* dengan prioritas sama yaitu *task* pembacaan sensor dan 1 *task* proses pengolahan data yang memiliki prioritas lebih tinggi.

6.2.1 Tujuan Pengujian

Pengujian kedua ini melakukan pengujian terhadap sistem *Free*RTOS yang diterapkan dengan melihat waktu eksekusi *task* dan eksekusi prioritas setiap *task*. Pengujian ini dilakukan untuk mengetahui performansi dari sistem yang telah dibuat.

6.2.2 Prosedur Pengujian

Prosedur pengujian waktu eksekusi *task* dan prioritas dilakukan dengan langsung menjalankan robot namun tetap terhubung antara laptop dengan mikrokontroler Arduino agar dapat melihat waktu eksekusi melalui serial monitor.

Kode program sebelum dijalankan akan ditambah fungsi *millis()* pada setiap awal *task* dan akhir *task.* Fungsi *millis()* berguna untuk menghitung waktu eksekusi dalam satuan *millisecond*. Prosedur pengujian waktu *task* dan prioritas meliputi:

1. Jalankan robot namun antara Arduino dengan laptop masih terhubung.
2. Buka kode program yang telah dibuat untuk menghitung waktu eksekusi *task* dengan memanfaatkan fungsi *millis()*, setelah itu *upload* kode programnya.
3. Amati hasil melalui serial monitor pada laptop.

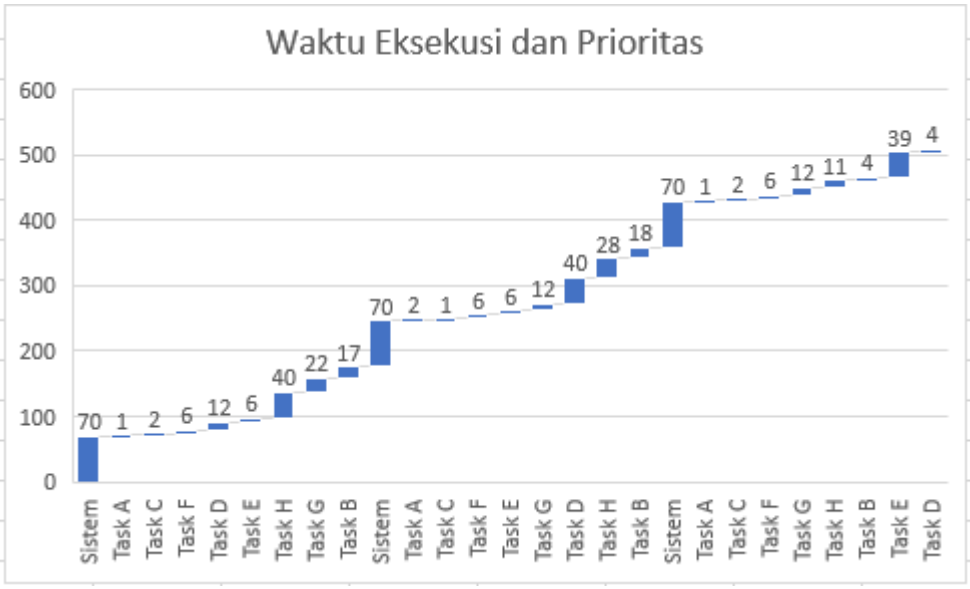
6.2.3 Hasil Pengujian

Berikut hasil pengujian yang telah dilakukan dengan mengambil 10 kali proses sistem pada Tabel 6.10 dan grafik dengan 3 kali proses sistem.

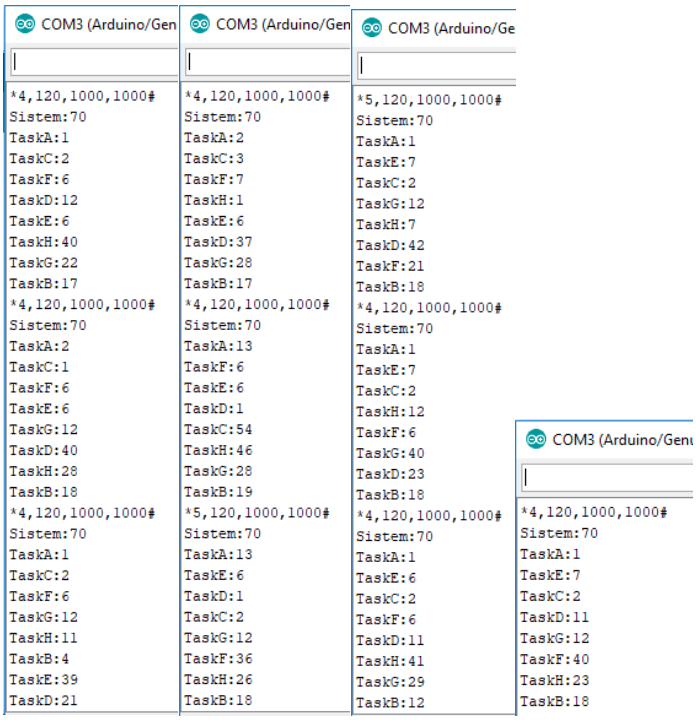
Tabel 6. 10 Hasil Pengujian *Task* Dan Prioritas

Proses Ke-	Urutan Prioritas									Waktu Eksekusi (ms)								
	Task									*Task*								
	A	B	C	D	E	F	G	H	S	A	B	C	D	E	F	G	H	S
1	2	9	3	5	6	4	8	7	1	1	17	2	12	6	6	22	40	70
2	2	9	3	7	5	4	6	8	1	2	18	1	40	6	6	12	28	70
3	2	7	3	9	8	4	5	6	1	1	4	2	21	39	6	12	11	70
4	2	9	3	7	6	4	8	5	1	2	17	3	37	6	7	28	1	70
5	2	9	6	5	4	3	8	7	1	13	19	54	1	6	6	28	46	70
6	2	9	5	4	3	7	6	8	1	13	18	2	1	6	36	12	26	70
7	2	9	4	7	3	8	5	6	1	1	18	2	42	7	21	12	7	70
8	2	9	4	8	3	6	7	5	1	1	18	2	23	7	6	40	12	70
9	2	9	4	6	3	5	8	7	1	1	12	2	11	6	6	29	41	70
10	2	9	4	5	3	7	6	8	1	1	18	2	11	7	40	12	23	70

Dari Tabel 6.10 untuk *task* A sampai H merupakan *task* sensor, sedangkan *task* s merupakan *task* proses. Satuan yang dipakai untuk menghitung waktu eksekusi ialah *millisecond*. Dari data pada Tabel6.10 tersebut dapat dilihat dalam bentuk grafik dengan 3 proses seperti pada Gambar 6.2 dan ditampilkan juga *screenshot* dari data pada serial monitor pada Gambar 6.3.



Gambar 6. 2 Grafik Hasil Pengujian *Task*



Gambar 6. 3 *Screenshot* Data Pada Serial Monitor

6.2.4 Analisis Pengujian

Melihat hasil data yang diperoleh pada pengujian dapat diketahui bahwa prioritas yang dieksekusi sistem sesuai dengan perancangan, dengan selalu memprioritaskan *task* sistem dan untuk *task* sensor dibuat sama yang penjadwalannya diatur oleh *Free*RTOS. Waktu eksekusi dalam pengujian juga sesuai dengan yang diharapkan yaitu tidak melebihi 100ms, dengan hal tersebut dapat diketahui perancangan sesuai dengan implementasi.

6.3 Pengujian Ketepatan Robot Bergerak

Sistem ini dibuat karena ingin mengetahui bagaimana RTOS jika diimplementasikan pada robot *quadruped* yang memiliki 8 sensor. Ketepatan robot bergerak merupakan hal yang penting dalam pergerakan robot, oleh karena itu akan dibuat perbandingan ketepatan gerak antara robot implementasi RTOS dengan robot yang tidak menggunakan RTOS pada programnya.

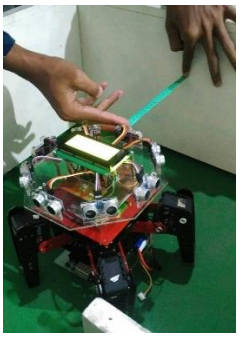
6.3.1 Tujuan Pengujian

Tujuan pengujian ini ialah dapat melihat perbandingan ketepatan gerak robot yang menerapkan RTOS dengan yang tidak menerapkan RTOS. Ketepatan gerak robot merupakan salah satu latar belakang dari penelitian ini karena dalam tugasnya robot nanti harus berjalan tepat dalam segi waktu dan data yang masuk. Ketepatan yang diuji dalam hal ini ialah ketepatan robot ketika menghindar atau mendeteksi pada halangan di depan sensor.

6.3.2 Prosedur Pengujian

Dalam pengujian ini sudah disiapkan satu kode program sama dalam proses meliputi pengambilan data sensor dan proses pengolahan data sensor namun tidak menggunakan RTOS. Perhitungan ketepatan gerak akan melihat pada jarak sensor dengan halangan. Jarak sensor terlebih dahulu diatur dan terdapat 1 sensor yang jika berjarak sesuai akan mematikan robot, dari hal tersebut maka dapat dihitung apakah robot bergerak dengan tepat sesuai jarak sensor dengan halangan. Prosedur pengujian ketepatan gerak meliputi:

1. Jalankan robot namun antara Arduino dengan laptop masih terhubung.
2. Buka kode program yang tidak menggunakan RTOS dan atur jarak sensor terhadap halangan, setelah itu *upload* kode programnya.
3. Amati hasil jarak sensor dengan halangan ketika berhenti dan ukur secara manual menggunakan penggaris, seperti pada Gambar 6.4.



Gambar 6. 4 Pengukuran Pengujian Ketepatan

4. Upload kode program yang menggunakan RTOS dan sudah diatur jarak sensor terhadap halangan.
5. Amati hasil jarak sensor dengan halangan ketika berhenti dan ukur secara manual menggunakan penggarais, seperti pada Gambar 6.4.

6.3.3 Hasil Pengujian

Hasil pengujian ketepatan gerak robot menggunakan RTOS dan tidak menggunakan RTOS dapat dilihat pada Tabel 6.11. Pada tabel tersebut terdapat presentasi *error* yang diukur menggunakan persamaan 6.1.

Tabel 6. 11 Hasil Pengujian Ketepatan

Jarak	Menggunakan RTOS (cm)	*Error (%)*	Tidak Menggunakan RTOS (cm)	*Error (%)*
10	9.8	2	8.3	17
15	15.1	0.7	12.5	16.7
20	20.2	1	17	15
25	24.8	0.8	21.9	12.4
30	29.9	0.3	26.7	11
35	36	2.9	32.6	6.9
40	38	5	36.9	7.75
Rata-rata		5.3		12.3

6.3.4 Analisis Pengujian

Dari hasil pengujian perbandingan ketepatan robot bergerak antara sistem menggunakan RTOS dengan sistem tidak menggunakan RTOS yang dilakukan pada 7 jarak yang berbeda, mendapatkan hasil bahwa sistem menggunakan RTOS memiliki persentase *error* sebesar 5.3 % dan sistem yang tidak menggunakan RTOS memiliki persentase sebesar 12.3% dengan itu dapat diambil kesimpulan bahwa sistem menggunakan RTOS memiliki persentase yang sedikit dibandingkan dengan sistem yang tidak menggunakan RTOS. Hasil pengujian ini juga membuktikan bahwa sistem menggunakan RTOS baik untuk ketepatan pergerkan robot *quadruped*.

6.4 Pengujian Perbandingan Waktu Kerja Sistem menggunakan RTOS dan Tanpa RTOS

Selain dari segi ketepatan robot bergerak sistem ini juga dilihat waktu kerja sistem, oleh karena itu akan dibuat perbandingan antara robot implementasi RTOS dengan robot yang tidak menggunakan RTOS pada programnya.

6.4.1 Tujuan Pengujian

Dapat melihat perbandingan robot yang menerapkan RTOS dengan yang tidak menerapkan RTOS. Selain ketepatan gerak robot menghindar terhadap halangan akan dilihat kecepatan waktu kerja robot ketika bergerak, dengan hal

tersebut maka nanti dapat ditarik kesimpulan apakah RTOS cocok untuk diimplementasikan pada pergerakan robot *quadruped.*

6.4.2 Prosedur Pengujian

Dalam pengujian ini sudah disiapkan satu kode program sama dalam proses meliputi pengambilan data sensor dan proses pengolahan data sensor namun tidak menggunakan RTOS. Perhitungan waktu dilakukan pada tahap awal dari kedua program pada bagian pengambilan data sensor dan tahap akhir dari proses pengolahan data dari sensor. Prosedur pengujian waktu kerja sistem meliputi:

1. Jalankan robot namun antara Arduino dengan laptop masih terhubung.
2. Buka kode program yang tidak menggunakan RTOS untuk menghitung waktu kerja dengan memanfaatkan fungsi *millis()*, setelah itu *upload* kode programnya.
3. Amati hasil melalui serial monitor pada laptop.
4. Upload kode program yang menggunakan RTOS untuk menghitung waktu kerja sama dengan sebelumnya memanfaatkan fungsi *millis()*.
5. Amati hasil melalui serial monitor pada laptop.

6.4.3 Hasil Pengujian

Hasil pengujian waktu kerja sistem menggunakan RTOS dan tidak menggunakan RTOS dapat dilihat pada Tabel 6.12.

Tabel 6. 12 Hasil Pengujian Waktu Kerja Sistem

Sampel	Menggunakan RTOS (ms)	Tidak Menggunakan RTOS (ms)
1	130	131
2	135	132
3	123	131
4	136	125
5	135	126
6	134	126
7	134	113
8	111	132
9	117	131
10	129	121
Rata-rata	128.4	126.8

6.4.4 Analisis Pengujian

Berdasarkan hasil pengujian pada Tabel 6.12, dapat dilakukan analisis atas perbedaan waktu kerja sistem. Hasil dari pengujian ialah sistem dengan RTOS memiliki waktu eksekusi yang lebih lama 1.6ms dikarenakan pada sistem menggunakan *vTaskDelay* sebesar 100ms agar sistem dapat berjalan tidak terlalu cepat dan akan menimbulkan kerusakan sistem. Sistem menggunakan RTOS meskipun memiliki waktu lebih lama sistem tersebut terlihat lebih tepat waktu dengan ketepatan gerak robot yang sesuai, sedangkan sistem tidak menggunakan RTOS memiliki waktu cepat namun ketika diterapkan pada robot untuk ketepatan tidak sesuai atau robot sering menabrak halangan.

BAB 7 PENUTUP

7.1 Kesimpulan

Berdasarkan rumusan masalah yang diajukan di awal penelitian ini serta berdasarkan hasil analisis pengujian yang dilakukan maka didapatkan beberapa kesimpulan sebagai berikut :

1. Sensor ultrasonik HC-SR04 yang difungsikan sebagai pendeteksi halangan robot *quadruped* dapat membaca halangan berukuran panjang 4cm dan tinggi 20cm dengan rata-rata *error* sebesar 0,0% atau dapat dikatakan sensor berkerja dengan baik dan sensor dapat membaca halangan dengan jarak 3cm sampai 400cm. Penelitian ini juga melihat prioritas sistem RTOS, penentuan prioritas pada *task* RTOS berpengaruh tehadap penjadwalan eksekusi *task.* Pada *task* dengan prioritas bernilai sama akan digunakan algoritme *round robin* sebagai penjadwalannya dan untuk *task* prioritas tinggi akan di eksekusi terlebih dahulu. Proses waktu eksekusi *task* juga dipengaruhi data yang dikelola setiap *task*.

2. *Real Time Operating System* (RTOS) berhasil diimplementasikan pada pergerakan robot *quadruped*. Pergerakan robot menggunakan RTOS menghasilkan ketepatan gerak menghindar terhadap halangan yang sangat baik dibandingkan dengan pergerakan robot tidak menggunakan RTOS. Jarak halangan yang semakin jauh juga berpengaruh karena akan menambah waktu eksekusi dari *task* pembacaan sensor.

3. Segi kecepatan waktu kerja sistem menggunkan RTOS memerlukan waktu lebih lama dibandingkan dengan sistem tidak menggunakan RTOS. Sistem menggunakan RTOS memiliki waktu rata-rata 128.4 sedangkan sistem tidak menggunakan RTOS memiliki waktu rata-rata 126.8, selisih waktu kerja dari hasil pengujian ialah 1.6ms. Waktu kerja sistem menggunakan RTOS lebih lama karena pada sistem menggunakan fungsi *vTaskDelay* sebesar 100ms agar sistem berjalan tidak terlalu cepat dan tidak merusak sistem.

7.2 Saran

Setelah melakukan penelitian ini, terdapat beberapa saran untuk pengembangan peneletian lebih lanjut, antara lain:

1. Menggunakan desain kaki yang lebih baik dengan menerapkan *inverse knimatic* dalam pergerakan kaki robot *quadruped* agar robot dapat berjalan dengan seimbang.
2. Lebih memanfaatkan fitur-fitur yang tersedia pada RTOS agar sistem berjalan lebih baik.
3. Pada penelitian berikutnya diharapkan menerapkan kondisi posisi robot yang lebih banyak setelah proses enkoding selesai.

DAFTAR PUSTAKA

Arduino, 2017. *Arduino Mega 2560 Rev3 Pinout, Atmega2560 Pin Mapping, Eagle Files, Schematics.* [Online] Available at: www.element14.com [Accessed 9 April 2018].

Arduinolearning, 2017. *Sensor Ultrasonik HC-SR04.* [Online] Available at: http://arduinolearning.com/code/hc-sr04-ultrasonic-sensor-example.php

Atmadja , W., Christian , B. & Kristofel, L., 2014. *Real Time Operating System on Embedded Linux with Ultrasonic Sensor for Mobile Robot ,* Bali: IAICT.

Conrad, K. L. & Shiakolas, P. S., 2000. *ROBOTIC CALIBRATION ISSUES: ACCURACY, REPEATABILITY AND CALIBRATION ,* Arlington: University of Texas at Arlington.

Cristina, 2014. *Hard Real-Time Execution Environment Extension for FreeRTOS,* Roma: Politehnica University of Timisaora.

Electronics-Tutorials, 2017. *Priority Encoder.* [Online] Available at: https://www.electronics-tutorials.ws/combination/comb_4.html

Ermadi, A., 2009. *Perancangan dan Implementasi Robot Mobil Pendeteksi dan Pemadam Api Menggunakan Sensor Ultrasonik dan Sensor Ultraviolet Berbasis Mikrokontroler Renesas R8C/13,* Padang: Universitas Andalas.

ITead Studio, 2010. *Ultrasonic ranging module : HC-SR04,* s.l.: info@iteadstudio.com.

Jatmiko, W. et al., 2015. *Real Time Operating System,* Depok: Universitas Indonesia.

Kurniawan, F. R., 2011. *MULTITASKING PADA MIKROKONTROLER ATMEGA16 MENGGUNAKAN REAL TIME OPERATING SYSTEM (RTOS) JENIS COOPERATIVE,* Semarang: Universitas Diponegoro.

Kusuma, J. W., 2011. *PENERAPAN INVERS KINEMATIK TERHADAP PERGERAKAN KAKI PADA ROBOT HEXAPOD ,* Palembang: STMIK GI MDP .

Nugraha, F., 2016. *Sensor Ultrasonik HC-SR04,* Makassar: Universitas Hasanuddin.

Reddy, S. R., 2006. *Selection of RTOS for an Efficient Design of Embedded Systems ,* Delhi: Indraprasta University.

Robot Institue of America, 1979. *Robot Definition.* [Online] Available at: www.robotiiksistem.com [Accessed 3 maret 2018].

Robotics, T., 2018. *Dynamixel AX-12A Robot Actuator.* [Online] Available at: http://www.trossenrobotics.com/dynamixel-ax-12-robot-actuator.aspx

ROBOTIS, 2010. *OpenCM9.04.* [Online] [Accessed 9 20 2017].

Robotis, 2016. *User Manual Dynamixel AX-12,* s.l.: Robotis.

ROBOTIS, 2017. *OPENCM 485 EXPANSION BOARD.* [Online] Available at: http://www.robotis.us/opencm-485-expansion-boar/

Rouse, M., 2016. *Robot.* [Online].

Sandy, H., 2007. *Merancang Mobile Robot Pembawa Objek Menggunakan OOPic-R,* Jakarta: PT Elex Media Komputindo.

Setiawan, L. & Utomo, D., 2011. *Inverse Kinematic dengan Real Time Operating System pada robot Quadruped,* Salatiga: Universitas Kristen Satya Wacana.

Setiawan, O., 2018. *IMPLEMENTASI TEKNIK ENKODING DIGITAL PEMBACAAN SENSOR ULTRASONIK UNTUK MEMETAKAN KEPUTUSAN AKSI ROBOT QUADRUPED,* Malang: Universitas Brawijaya.

Siegwart, R. & Nourbakhsh, I. R., 2004. *Introduction to Autonomous Mobile Robots,* London: Massachusetts Institute of Technology.

Team, E., 2015. *Parameter HC-SR04,* s.l.: s.n.

Utomo, A. D., Setiawan, I. & Andromeda, r., 2011. *SISTEM KONTROL NAVIGASI PADA MOBILE ROBOT BERBASIS PCBC (PIECEWISE CUBIC BEZIER CURVE),* Semarang: Universitas Diponegoro.

Wahyudin, I., 2018. *RANCANG BANGUN SISTEM MULTI-SENSOR UNTUK PENGUKURAN JARAK SECARA SIMULTAN,* Malang: s.n.

Wicaksono, H. et al., 2008. *Perancangan Sistem Navigasi Otonom pada Behavior Based Hexapod Robot,* Surabaya: Petra.