RANCANG BANGUN SISTEM MULTI-SENSOR UNTUK PENGUKURAN JARAK SECARA SIMULTAN

SKRIPSI

KEMINATAN TEKNIK KOMPUTER

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Idang Wahyuddin S.
NIM: 135150300111022

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

PENGESAHAN

RANCANG BANGUN SISTEM MULTI-SENSOR UNTUK PENGUKURAN JARAK SECARA SIMULTAN

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Idang Wahyuddin S.
NIM: 135150300111022

Skripsi ini telah diuji dan dinyatakan lulus pada
29 Juni 2018
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Sabriansyah Rizqika Akbar, S.T., M.Eng	Dahnial Syauqy, S.T., M.T., M.Sc.
NIP: 19820809 201212 1 004	NIK: 201607 870423 1 022

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, 7 Mei 2018

Idang Wahyuddin S

NIM: 135150300111022

KATA PENGANTAR

Dengan menyebut nama Allah Subhanahu Wa Ta'alla yang maha pengasih lagi maha penyanyang, puji syukur saya panjatkan atas kehadirat Allah Subhanahu Wa Ta'ala, karena ridha-Nya saya bisa menyelesaikan skripsi ini dan dengan menyelesaikan skripsi ini insya'Allah saya mendapatkan rahmat-Nya. Adapun maksud penyusunan skripsi ini adalah untuk memenuhi salah satu persyaratan dalam memenuhi ujian Sarjana Fakultas Ilmu Komputer dan mengharapkan rahmat Allah Subhanahu Wa Ta'ala. Judul skripsi yang disusun adalah: "Rancang Bangun Sistem Multi-Sensor Untuk Pengukuran Jarak Secara Simultan".

Banyak cobaan dan hambatan yang saya alami dalam menyusun skripsi ini, akan tetapi semua itu teratasi dengan baik berkat ridha Allah Azza Wa Jalla, do'a orang tua yang sama seperti do'a seorang Nabi kepada umatnya dan dukungan dari banyak pihak. Berdasarkan hal tersebut, pada kesempatan ini saya mengucapkan terimaksaih kepada:

1. Allah Subhanahu Wa Ta'ala yang telah meridhai dan memberikan segala kebaikan kepada saya untuk menyelesaikan skripsi ini dan memberikan banyak hikmah dalam proses penyelesaian skripsi.
2. Nabi Muhammad Shalallahu 'Alaihi Wasallam yang telah memberikan sebaik-baik contoh dalam meminta pertolongan kepada Allah Subhanahu Wa Ta'ala dan cara melewati segala kesulitan dengan ketenangan.
3. Kedua orang tua yang telah meridhai saya, dengan ridhanya maka ridha Allah Subhanahu Wa Ta'ala hadir disetiap proses pengerjaan skripsi saya. Dan juga do'a kedua orang tua yang banyak berperan dalam penyelesaian skripsi.
4. Bapak Wayan Firdaus Mahmudy, S.Si, M.T, Ph.D. selaku Dekan Fakultas Ilmu Komputer Universitas Brawijaya Malang.
5. Bapak Heru Nurwasito, Ir., M.Kom. selaku Wakil Ketua I Bidang Akademik Fakultas Ilmu Komputer Universitas Brawijaya Malang.
6. Bapak Tri Astoto Kurniawan, S.T, M.T, Ph.D. selaku Ketua Jurusan Informatika Universitas Brawijaya Malang.
7. Bapak Sabriansyah Rizqika Akbar, S.T, M.Eng. selaku Ketua Program Studi Teknik Komputer Universitas Brawijaya Malang dan selaku dosen pembimbing I yang telah banyak memberikan ide dalam penyelesaian skripsi, membimbing dengan sabar dan pengarahan yang sangat bermanfaat dalam penyelesaian skripsi.
8. Bapak Dahnial Syauqy, S.T., M.T., M.Sc. selaku dosen pembimbing II yang telah banyak memberikan pengarahan dalam menyelesaikan laporan skripsi dan revisi-revisi yang diberikan sangat bermanfaat dalam keilmuan suatu penulisan.
9. Seluruh dosen dan karyawan Fakultas Ilmu Komputer yang telah berperan dalam proses penyelesaian skripsi ini.
10. Teman-teman kontrakan ganteng yang sudah menemani kehidupan sehari-hari dan banyak ilmu yang saya dapat ketika hidup dengan mereka.

11. Teman-teman KASKUB yang sudah menemani diawal perkuliahan. Kita belajar sesuatu yang baru Bersama-sama. Terimakasih atas waktunya.

Saya menyadari bahwa laporan skripsi ini jauh dari sempurna, oleh karena itu untuk segala kritik dan saran yang membangun saya ucapkan terimakasih. Saya mengharapkan semoga laporan skripsi ini dapat berguna.

Malang, 7 Mei 2018

Penulis

Idangws14@gmail.com

ABSTRAK

Sebuah sistem multi-sensor yang berjalan simultan merupakan suatu faktor yang harus diperhatikan ketika membangun sebuah sistem multi-sensor. Agar sistem berjalan secara simultan dibutuhkan sebuah sistem dengan respon yang cepat. Berdasarkan permasalahan tersebut, diperlukan sebuah sistem multi-sensor yang bisa berjalan secara simultan. Dengan menerapkan *multitasking* maka sistem dapat mengeksekusi lebih dari satu program secara bersamaan. RTOS (*Real Time Operating System*) adalah salah satu metode untuk penerapan *multitasking* pada sistem *embedded*. Untuk mengimplementasikan RTOS ke dalam Arduino dibutuhkan *library* FreeRTOS. Mikrokontroler Arduino Mega 2560 dan modul *sensor* HC-SR04 sebanyak 8. Antar muka komunkasi pada sistem ini menggunakan *Parallel Bus Interface* (PBI) untuk mendukung kinerja dalam pengaplikasian sistem-multisensor berjalan simultan. Pada penerapan RTOS sistem ini akan dibagi menjadai 8 *task.* Setiap task akan memproses masing-masing modul sensor HC-SR04. Dari hasil pengujian kinerja modul sensor HC-SR04 memperoleh persentase *error* sebesar 0% dari seluruh sensor. Untuk pengujian stabilitas eksekusi waktu dari masing-masing task terbilang stabil yaitu 1ms pada masing-masing task dalam 10 percobaan. Dari hasil pengujian, sistem yang menerapkan RTOS berjalan dari awal program sampai akhir membutuhkan rata-rata waktu 8,1332ms sedangkan pada sistem yang tidak menerapkan RTOS membutuhkan rata-rata waktu 264ms. Setiap task memiliki waktu eksekusi berkisar antara 1,005-1,583ms. Sistem dengan menggunakan RTOS memerlukan waktu yang lebih sedikit dari pada sistem yang tidak menggunakan RTOS.

Kata kunci: RTOS, FreeRTOS, Multi-sensor, *Multitasking, task,* Simulltan

ABSTRACT

A multi-sensor system that runs simultaneously is a factor to be considered when building a multi-sensor system. In order for the system to run simultaneously required a system with a fast response. Based on these problems, required a multi-sensor system that can run simultaneously. By applying multitasking then the system can execute more than one program simultaneously. RTOS (Real Time Operating System) is one method for multitasking application on embedded system. To implement RTOS into Arduino requires FreeRTOS library. Arduino Mega 2560 Microcontroller and HC-SR04 sensor module as much 8. 8. Communication interface on this system using Parallel Bus Interface (PBI) to support performance in multisensor system running simultaneously. In the application of RTOS this system will be divided into 8 tasks. Each task will process each of the HC-SR04 sensor modules. From the results of performance testing HC-SR04 sensor module gain percentage error of 0% of all sensors. To test the stability of time execution of each task is fairly stable that is 1ms on each task in 10 experiments. From the test results, the system implementing the RTOS running from the beginning of the program to the end requires an average time of 8.1332ms whereas on systems that do not implement RTOS requires an average time of 264ms. Each task has an execution time ranging from 1.005 to 1.583ms. Systems using RTOS require less time than systems that do not use RTOS.

Keywords: RTOS, FreeRTOS, Multitasking, Task, Simultaneously

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB 1 PENDAHULUAN

1.1 Latar belakang

Sebuah sistem multi-sensor yang berjalan simultan merupakan suatu faktor yang harus diperhatikan ketika membangun sebuah sistem multi-sensor. Dengan berjalannya suatu sensor secara bersamaan dapat mengoptimalkan kinerja dari sistem multi-sensor dalam hal respon, ketika suatu sensor berjalan secara bersama maka akan membantu dalam mengambil keputusan. Ketika sensor A, sensor B dan sensor C mendeteksi suatu interaksi secara bersamaan maka ketiga sensor tersebut akan mengirim data ke pusat pengontrol, dan pusat pengontrol akan melakukan eksekusi dari hasil yang didapat dari ketiga sensor tersebut.

Penerapan multi-sensor saat ini sudah sangat luas, seperti penerapannya dalam akan tetapi simultan dari multi-sensor tersebut kurang diperhatikan, meskipun simultan sebuah multi-sensor cukup penting. Pengaplikasian sistem multi-sensor yang paling sesuai untuk merespon perilaku *parallel* proses tersebut adalah *multitasking* (Zouaoui, 2017). Salah satu contoh pengaplikasian multi-sensor ultrasonik adalah pada robot pemadam kebakaran yang digunakan pada *event* kompetitif. Sensor-sensor yang diletakan pada bagian tertentu robot untuk mengambil data yang dibutuhkan berupa jarak disekitar robot antara robot dengan suatu objek yang digunakan untuk menghindari robot berbenturan dengan suatu objek. Dengan berjalannya sensor-sensor tersebut secara simultan maka kinerja dari robot pemadam api bisa berjalan dengan semestinya. Jika tidak, maka robot tersebut akan bekerja tidak semestinya seperti robot akan berbenturan dengan suatu objek dikarenakan kurang tepat waktu dalam memproses data, terlambatnya pengambilan sebuah keputusan yang seharusnya tidak terjadi pada sebuah sistem multi-sensor.

Dalam membangun sebuah sistem multi-sensor yang mampu berjalan secara simultan, dibutuhkan sebuah arsitektur yang dapat menampung sensor dengan jumlah banyak dan program yang dapat menjalankan sensor tersebut secara simultan. *Multitasking* dapat digunakan sebagai solusi untuk masalah tersebut. *multitasking* sendiri bisa mengeksekusi lebih dari satu program secara bersamaan (Putze, 2013). Dengan *multitasking* sensor-sensor tersebut bisa berjalan bersamaan, walaupun pada prosesnya sendiri tidak betul-betul berjalan bersamaan. Karena cepatnya prosesor memproses suatu data dengan kecepatan mikrosekon sampai milisekon sehingga *multitasking* terlihat seperti berjalan bersamaan. Proses tersebut dieksekusi bergantian oleh prosesor.

Task dengan jumlah banyak dengan kondisi mengharuskan mengerjakannya secara bersamaan pada sistem embedded. Situasi semacam itu biasanya diproses oleh sebuah sistem operasi. Hal seperti itu diterapkan pada sebuah sistem real-time. Akan tetapi pada dasarnya prosesor pada sebuah sistem embedded sangat terbatas untuk melakukan sebuah eksekusi suatu program di waktu yang sama. Maka dari itu dibutuhkan sebuah sistem operasi untuk mengatur penjadwalan eksekusi program tersebut. Sistem operasi akan membagi waktu eksekusi dengan

kondisi yang paling tepat untuk kondisi program yang akan dieksekusi. Pada sistem *embedded* dikembangkan sistem operasi *real time,* dikenal dengan *Real Time Operating System* (*RTOS*) (Jatmiko, 2015).

Pada penelitian sebelumnya dalam membangun suatu sistem multi-sensor yang menerapkan RTOS (*Real Time Operating System*) kedalam sistemnya yang berjudul "Sistem Pendeteksi Kebocoran Gas LPG Menggunakan Metode *Fuzzy* yang Diimplementasi dengan *Real Time Operating System* (RTOS)" oleh Lavanna Indanus Ramadhan pada tahun 2017. Pada penelitiannya menggunakan 2 sensor dengan jenis berbeda yaitu sensor suhu LM35 dan sensor gas MQ-6. Pada sistemnya memberikan nilai prioritas yang sama untuk task dalam memproses sensor suhu dan task agar kedua sensor berjalan secara simultan (Ramadhan, 2017). Selanjutnya pada penelitian kedua yang berjudul "Implementasi *System Real Time* untuk Monitoring Pencahayaan Suhu dan Kelembaban pada Tanaman Stroberi" oleh Agung Leona Suparlin pada tahun 2018. Pada sistemnya menerapkan RTOS untuk menjalankan sensornya sejumlah 3 secara simultan dengan memberikan prioritas yang sama dengan tujuan ketiga sensor tersebut berjalan secara simultan. Dan task untuk memproses data yang lain diberikan nilai prioritas yang berbeda (Suparlin, 2018).

Berdasarkan beberapa penjelasan di atas, RTOS dibutuhkan dalam penerapan sistem multi-sensor yang berjalan secara simultan. Multi-sensor yang berjalan secara simultan menerapkan *RTOS* dengan beberapa jumlah sensor ultrasonik yang di pasang mencakup sudut 360^{o} dan tiap sensor akan diberi satu indikator sebagai pemberitahu bahwa sensor tersebut mendeteksi jarak yang sudah di tentukan. Dan pemrosesannya menggunakan mikrokontroller Arduino Mega 2560. Monitoring jarak yang diterima oleh ultrasonik bisa dilakukan di *serial monitor* pada Arduino IDE.

Dengan memanfaatkan alat ini diharapkan perancangan multi-sensor yang berajalan secara simultan dapat dimanfaatkan untuk sebuah sistem yang membutuhkan multi-sensor untuk bekerja seperti sebuah robot pemadam api dan sistem lainnya yang membutuhkan sebuah multi-sensor yang berjalan secara simultan.

1.2 Rumusan masalah

Berdasarkan penjelasan latar belakang diatas, maka dirumuskan beberapa masalah untuk diselesaikan dari penelitian ini antara lain :

1. Bagaimana merancang sebuah rangkaian yang tepat untuk sistem multi-sensor 8 sensor HC-SR04 ?
2. Bagaimana cara mengimplementasikan *RTOS* (*Real Time Operating System*) ke sebuah sistem multi-sensor agar berjalan secara simultan?
3. Seberapa cepat respon sistem multi-sensor ini ketika berjalan secara simultan dengan implementasi *RTOS* ?

1.3 Tujuan

Dari penjabaran rumusan masalah diatas, maka didapat sebuah tujuan dari penelitian ini antara lain :

1. Untuk merancang sebuah rangkaian yang tepat untuk sistem multi-sensor 8 sensor HC-SR04.
2. Untuk membuat kode program implementasi *RTOS* kedalam sistem multi-sensor agar sistem berjalan secara simultan.
3. Untuk mengetahui seberapa cepat respon sistem multi-sensor ini ketika berjalan secara simultan dengan penerapan *RTOS*.

1.4 Manfaat

Diharapkan dari penelitian ini dapat diambil manfaat dari beberapa pihak yang terhubung penelitian ini antar lain :

1. Bagi Penulis
   a. Terpenuhinya salah satu syarat kelulusan untuk mendapat gelar Sarjana Komputer.
   b. Sebagai kesempatan untuk menerapkan ilmu yang telah diperoleh dari bangku perkuliahan maupun di luar bangku perkuliahan.
2. Bagi Perguruan Tinggi

   Perwujudan dari Tridharma Perguruan Tinggi. Dengan dilakukan penelitian ini diharapkan dapat mengembangkan ilmu pengetahuan dibidang teknologi yang nantinya dapat dimanfaatkan ke penelitian selanjutnya maupun diimplementasikan ke sistem baru.
3. Bagi Mahasiswa / Masyarakat
   a. Dapat di terapkan untuk sistem yang berjalan secara otomatis, sehingga dapat mempermudah kegiatan sehari-hari. Atau pada sistem yang bersifat kompetitif seperti sebuah robot.
   b. Dapat digunakan untuk dasar sebagai penelitian selanjutnya yang memiliki keterkaitan dalam bidang teknologi dan metode.

1.5 Batasan masalah

Dari permasalahan yang telah dirumuskan maka diperlukan suatu Batasan agar lebih fokus dalam penelitian ini antara lain :

1. Jumlah sensor yang digunakan berjumlah 8.
2. Jumlah *LED* yang digunakan adalah 8 dan berwarna merah.
3. Sensor yang digunakan adalah HC-SR04 untuk mengukur jarak.

4. Menggunakan *LED* sebagai indikator jarak yang sudah ditentukan.
5. Tegangan yang digunakan sebesar 5V.
6. *LED* menyala jika jarak yang terdeteksi kurang dari sama dengan 15cm.
7. Mikrokontroler yang digunakan adalah Arduino Mega 2560.
8. *Multitasking* pada sistem multi-sensor menggunakan implementasi *RTOS*.
9. Sistem menggunakan satuan ukur jarak cm (*centi meter*) untuk menyesuaikan kemampuan *library* dalam memproses hasil *sensing*.

1.6 Sistematika pembahasan

Agar penjelasan setiap bab dapat dipahami maka sistem pembahasan dari setiap bab akan dijelaskan secara garis besar sebagai berikut :

BAB I : **Pendahuluan**

Menjelaskan latar belakang penelitian, rumusan masalah penelitian, Batasan masalah penelitian, tujuan dari penelitian, manfaat yang didapat dan sistematika pembahasan dari masing-masing bab.

BAB II : **Landasan kepustakaan**

Menguraikan tinjauan pustaka dan dasar teori yang digunakan untuk membantu dalam penelitian ini serta menjelaskan tentang penelitian serupa yang telah dilakukan.

BAB III : **Metodologi**

Menguraikan tentang metode dan langkah kerja yang terdiri dari studi literatur, analisis kebutuhan perangkat keras, analisis kebutuhan perangkat lunak, ambaran umum perancangan sistem, implementasi sistem, pengujian sistem serta pengambilan kesimpulan.

BAB IV : **Rekayasa kebutuhan**

Menguraikan kebutuhan untuk membangun sistem pada penelitian ini yang harus terpenuhi.

BAB V : **Perancangan dan Implementasi**

Menguraikan tentang perancangan sistem yang telah dibahas segala kebutuhannya pada bab sebelumnya dan mengimplementasikan hasil perancangan yang telah dibuat.

BAB VI : **Pengujian**

Menguraikan hasil pengujian dan analisis pada sistem yang telah dibangun berdasarkan perancangan dan implementasi pada bab sebelumnya.

BAB VII : Penutup

Menguraikan kesimpulan yang diperoleh dari sistem yang telah dibangun dan pengujian yang dilakukan pada sistem, serta saran-saran untuk pengembangan selanjutnya.

DAFTAR PUSTAKA

A. Gegana A, Gregorius, 2007. "Teknologi *LED* dalam Pencahayaan Arsitektural Facade Bangunan". Indonesia: Universitas Indonesia.

Arduinolearning team, HC-SR04. Tersedia di : http://arduinolearning.com/wp-content/*upload*s/2014/12/HC-SR04-Ultrasonik-Sensor.jpg [diakses 7 September 2017]

Atmel, *Datasheet* atmega328P. Tersedia di : http://www.atmel.com/Images/Atmel-42735-8-bit-AVR-Microcontroller-ATmega328-328P_Datasheet.pdf [diakses 7 September 2017]

Banzi, M., 2008. *Getting Started with Arduino*. Inggris: O'Reilly.

Barry, R., 2016. *Mastering the FreeRTOS™ Real Time Kernel*. [Online] Tersedia di: http://www.*FreeRTOS*.org, [Diakses 9 November 2017].

Chandane, Mrinal, P. 2016. *Real Time Operating Systems: A Complete Overview. International Journal of Electrical and Electronics Engineers* (IJEEE).

Elec Freaks, *Ultrasonik Ranging Module HC-SR04*. Tersedia di : https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf [diakses 7 September 2017]

ElectronicsTutorial. 2017. *The Light Emitting Diode*. Tersedia di : https://www.electronics-tutorials.ws/diode/diode_8.html, [diakses 2 Mei 2018]

Elfring, Jos. et al., 2016. Multi-sensor *Simultaneous Vehicle Tracking and Shape Estimation*. Sweden: *Intelligent Vehicles Symposium* (IV).

Evans, Brian, W. 2007. *Arduino Programming Notebook*. (Creative Commons Attribution-Noncommercial-Share Alike 3.0) [online] California: Creative Commons (Diterbitkan 2007) Tersedia di: < https://playground.arduino.cc/uploads/Main/arduino_notebook_v1-1.pdf > [Diakses 7 September 2017]

Harris, Tom. et al., 2014. How Light Emitting Diodes Work. Tersedia di : https://electronics.howstuffworks.com/*LED*.htm, [diakses 2 Mei 2018]

Jatmiko, W. et al., 2015. RTOS Teori dan Aplikasi. Depok: Fakultas Ilmu komputer Universitas Indonesia.

Puham, Janez. 2015. *Operating Systems, Embedded Systems, and Real-Time Systems*. Ljubljana: FE Publishing

Putze, Felix. 2013. *Multitasking & Workload*. Jerman: *Karlsruhe Institute of Technology*

Ramadhan, L., Syauqy, D., & Prasetio, B. Sistem Pendeteksi Kebocoran Gas LPG Menggunakan Metode Fuzzy yang Diimplementasikan dengan Real Time Operating System (RTOS). **Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer**, vol. 1, no. 11, p. 1206-1213, juli 2017. ISSN 2548-964X. Tersedia pada: <http://j-ptiik.ub.ac.id/index.php/j-ptiik/article/view/397>. Tanggal Akses: 03 juli 2018

Suparlin, A., Akbar, S., & Syauqi, D. Implementasi System Real Time untuk Monitoring Pencahayaan Suhu dan Kelembaban pada Tanaman Stroberi. **Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer**, vol. 2, no. 9, p. 3278-3286, peb. 2018. ISSN 2548-964X. Tersedia pada: <http://j-ptiik.ub.ac.id/index.php/j-ptiik/article/view/2239>. Tanggal Akses: 03 juli 2018

BAB 7 PENUTUP

Bab ini berisi kesimpulan yang didasarkan dari pengujian dan analisis yang dilakukan selama proses penelitian dan saran yang berisi hal-hal yang diperlukandalam melakukan pengembangan untuk topik skripsi selanjutnya.

7.1 Kesimpulan

Berdasarkan pengujian dan Analisa yang dilakukan terhadap tugas akhir ini maka didapatkan kesimpulan sebagai berikut :

1. Dalam merancang sistem multi-sensor HC-SR04 sebanyak 8 buah menggunakan komunikasi *Parallel Bus Interface* (PBI) untuk membantu system dalam menjalankan sistem secara simultan. Dan pada bagian LED (*Light Emitting Diode*) diberikan resistor sebesar 160Ω yang dihubungkan ke anoda dan katoda LED dihubungkan ke *ground*.
2. Implementasi RTOS ke dalam sistem multi-sensor HC-SR04 menggunakan *library* FreeRTOS yang tersedia untuk mikrokontroller Arduino. Dengan memberikan prioritas setiap *task* bernilai 2 dengan bertujuan agar sistem memproses setiap *task* dengan bersamaan dan sistem *scheduling* diatur sepenuhnya oleh RTOS.
3. Jarak yang terdeteksi oleh sensor mempengaruhi waktu eksekusi tiap *task*. Semakin jauh jarak yang terdeteksi maka semakin jauh pula waktu eksekusi *task*. Dan sebaliknya untuk kondisi pada jarak yang terdeteksi lebih dekat. Hal ini dikarenakan pada saat sensor HC-SR04 menembakkan sinyal yang dilakukan oleh *trigger*, untuk dipantukan oleh suatu objek yang berfungsi untuk mendeteksi seberapa jauh halangan yang ada didepan sensor. Dan nantinya sinyal tersebut akan diterima oleh echo. Semakin jauh halangan yang ada maka semakin lama pula sinyal tersebut akan dipantulkan. Dan sebaliknya untuk kondisi sensor ketika mendeteksi halangan lebih dekat, maka akan semakin cepat sinyal tersebut di pantulkan. Waktu eksekusi setiap *task* dengan jarak yang sama memiliki waktu yang terbilang stabil yaitu 1,0ms sampai 1,5ms dijarak 5cm. Dan jumlah dari proses eksekusi seluruh *task* sama dengan hasil pengujian pada waktu eksekusi seluruh task. Dari segi kecepatan waktu eksekusi dan respon sistem menggunakan RTOS memerlukan waktu eksekusi dan respon lebih sedikit dari pada sistem tanpa RTOS. Waktu eksekusi sistem RTOS untuk menjalankan semua *task* adalah ± 8 ms dan waktu eksekusi sistem tanpa menggunakan RTOS adalah 264ms. Hal ini dikarenakan pada sistem yang menggunakan RTOS pada proses *sensing*, menghitung data hasil *sensing* dan menampilkan *output* perada pada satu fungsi dan satu *task*. Sehingga dalam hal respon dan waktu eksekusi membutuhkan waktu lebih sedikit. Sedangkan sistem tanpa menggunakan RTOS pada proses *sensing*, menghitung hasil *sensing* dan menampilkan *output* terdpat pada fungsi yang berbeda sehingga memerlukan waktu *sensing* atau respon yang lebih lama.

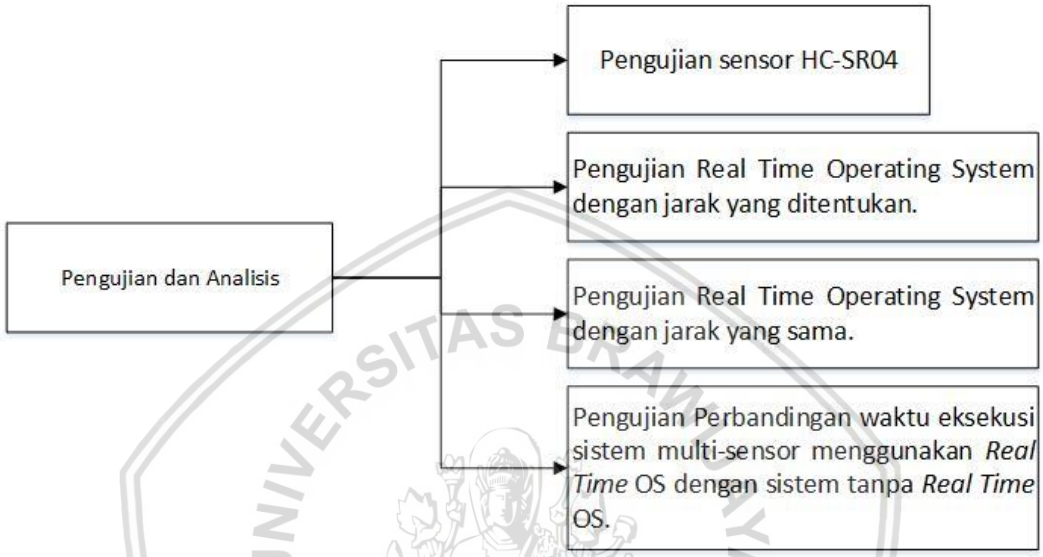
7.2 Saran

Berdasarkan pengujian dan Analisa yang dilakukan terhadap tugas akhir ini maka ada beberapa saran yang dapat digunakan untuk pengembangan sistem ini atau implementasi dengan sistem yang lain sebagai berikut :

1. Untuk penentuan nilai *vTaskDelay* dapat dilakukan lebih dinamis, dapat berubah sesuai dengan batas jarak yang ditentukan.
2. Diharapkan juga pada penelitian berikutnya untuk memberikan aktuator lebih baik dan masih bisa untuk mengikuti respon dari sistem.
3. Untuk penerapan sistem ini agar disesuaikan respon pada sistem yang akan diterapkan dengan sistem multi-sensor ini.

BAB 6 PENGUJIAN DAN ANALISIS

Pada bab ini membahas mengenai proses pengujian sistem multi-sensor dengan mengimplementasikan RTOS. Terdapat 2 pengujian dalam sistem ini, yaitu waktu yang dibutuhkan dalam melakukan eksekusi program tiap *task*, dan pengujian perbedaan sistem yang menggunakan *RTOS* dengan sistem tanpa menggunakan RTOS. Hal tersebut ditunjukkan pada Gambar 6.1.



Gambar 6.1 Grafik pengujian dan analisis

6.1 Pengujian sensor HC-SR04

6.1.1 Tujuan

Untuk mengetahui kinerja setiap sensor HC-SR04 yang digunakan apakah berfungsi dengan baik atau tidak. Pengukuran hasil dari sensor HC-SR04 apakah sama dengan pengukuran menggunakan alat pengukur yang digunakan di kehidupan sehari-hari yaitu penggaris.

6.1.2 Prosedur

Untuk menguji kinerja setiap sensor HC-SR04 yang digunakan maka masing-masing sensor akan digunakan untuk mengukur jarak yang diakukan terpisah dengan sensor HC-SR04 yang lain. Sensor akan di bandingan dengan hasil pengukuran dari penggaris. Pengujian menggunakan program tersendiri yang hanya digunakan untuk mengukur jarak dan menampilkannya pada *Serial monitor*. Prosedur pengujian kinerja sensor HC-SR04 meliputi:

1. Sambungkan prototipe alat dengan komputer atau laptop
2. Buka *source code* yang dibuat untuk mengukur jarak menggunakan sensor HC-SR04 dengan menggunakan *library* NewPing.h. Setelah itu *compile* dan *upload source code* programnya.

3. Amati hasil melalui *serial monitor* dan beri suatu objek padat yang digunakan untuk memantukan sinyal ultrasonik dari sensor.
4. Menggerakkan terus objek padat tersebut tiap satu *centimeter,* sesuaikan perpindahan objek tersebut seperti pengukuran pada penggaris.
5. Kumpulkan data hasil sensing tersebut dan hitung persentase *error* setiap sensor dengan persamaan

$$\%Error = \frac{(Nilai\ terbaca) - (Nilai\ Sebenarnya)}{(Nilai\ Sebenarnya)} \times 100\% \quad \textbf{(6.1)}$$

6. Kesimpulan.

6.1.3 Hasil dan analisis

Hasil dari pengujian akurasi dari setiap sensor yang digunakan dalam sistem *multitasking* ditampilkan pada Tabel 6.1, sedangkan persentase error dari setiap sensor pada Tabel 6.2.

Tabel 6.1 Hasil pengujian sensor

Pengukuran Penggaris (cm)	Sensor Ke- (cm)							
	1	2	3	4	5	6	7	8
3	3	3	3	3	3	3	3	3
4	4	4	4	4	4	4	4	4
5	5	5	5	5	5	5	5	5
6	6	6	6	6	6	6	6	6
7	7	7	7	7	7	7	7	7
8	8	8	8	8	8	8	8	8
9	9	9	9	9	9	9	9	9
10	10	10	10	10	10	10	10	10
11	11	11	11	11	11	11	11	11
12	12	12	12	12	12	12	12	12

Tabel 6.2 Perhitungan Persentase Error Sensor

Pengukuran Penggaris (cm)	Persentase Error Sensor Ke- (cm)							
	1	2	3	4	5	6	7	8
3	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0

Pengukuran Penggaris (cm)	Persentase Error Sensor Ke- (cm)							
	1	2	3	4	5	6	7	8
7	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0

Berdasarkan analisis Tabel 6.2, masing-masing sensor memiliki rata-rata *error* 0% yang di mulai dari jarak 3cm sampai 12cm. Percobaan dilakukan sebanyak 10 kali jadi percobaan hanya bisa sampai pada jarak 12cm. Dengan percobaan sebanyak 10 kali dapat disimpulkan sensor yang dipakai untuk sistem multi-sensor bekerja dengan sangat baik untuk pemrosesan sistem secara simultan. Jika sensor memiliki persentase *error* lebih dari 0% maka cukup mempengaruhi kinerja dari sistem di bagian *output sistem*.

6.2 Pengujian *Real-Time Operating System* setiap *task* dengan jarak yang ditentukan

6.2.1 Tujuan

Untuk mengetahui waktu eksekusi setiap *task* dalam menyelesaikan tugas ketika sensor mendeteksi jarak yang berbeda di setiap pengujian.

6.2.2 Prosedur

Untuk membantu perhitungan waktu eksekusi tiap *task* digunakan fungsi timer pada Arduino yaitu fungsi "*micros()*" untuk menghitug waktu eksekusi dalam satuan *microsecond* dan "*millis()*" untuk menghitung waktu dalam satuan *millisecond*. Fungsi *micros()* yang digunakan dalam pengujian ini dikarenakan mebutuhkan perhitungan *timer* yang lebih detail. Fungsi *micros()* diletakkan pada awal program setelah fungsi *vTaskDelay* untuk mendapatkan waktu eksekusi *task* tersebut dan akhir program dari setiap *task*. Pengujian ini dimulai dari jarak 5cm untuk setiap sensor sampai 14cm. Karena semakin jauh jarak yang terdeteksi maka akan semakin lama pula waktu eksekusi *task* sebab semakin jauh jarak terdeteksi maka semakin lama pula gelombang ultrasonik yang dipantulkan kembali ke sensor, dan sebaliknya untuk kondisi sensor yang mendeteksi jarak lebih dekat. Prosedur pengujian waktu eksekusi *task* meliputi:

1. Sambungkan prototipe alat dengan komputer atau laptop
2. Buka *source code* alat yang dibuat untuk menghitung waktu eksekusi *task* dengan memanfaatkan fungsi *micros()*. Setelah itu compile dan *upload source code* programnya.

3. Berikan suatu objek didepan sensor untuk memantulkan gelombang ultrasonik dengan jarak yang sudah ditentukan.
4. Amati hasil melalui *serial monitor*.
5. Kesimpulan.

6.2.3 Hasil dan analisis

Hasil dari pengujian waktu eksekusi *task* pada sistem ditampilkan pada Tabel 6.3.

Tabel 6.3 Analisis hasil pengujian waktu eksekusi *task*

Jarak (cm)	TaskUS (ms)								Rata-Rata
	1	2	3	4	5	6	7	8	
5	1,084	1,020	1,052	1,068	1,044	1,040	1,072	1,080	1,0575
6	1,096	1,108	1,104	1,072	1,100	1,060	1,150	1,184	1,10925
7	1,152	1,168	1,204	1,188	1,584	1,140	1,168	1,200	1,2255
8	1,212	1,216	1,220	1,224	1,576	1,212	1,236	1,224	1,265
9	1,280	1,280	1,268	1,308	1,284	1,252	1,264	1,256	1,274
10	1,364	1,424	1,372	1,428	1,380	1,616	1,348	1,372	1,413
11	1,416	1,464	1,444	1,424	1,408	1,420	1,420	1,452	1,431
12	1,496	1,520	1,476	1,504	1,496	1,508	1,444	1,476	1,490
13	1,504	1,556	1,540	1,576	1,544	1,544	1,524	1,520	1,528
14	1,612	1,692	1,628	1,616	1,596	1,600	1,624	1,628	1,636

Dari hasil sensing yang didapat dari masing-masing *task* dari jarak 5cm-14cm di Tabel 6.3 dapat dilihat bahwa durasi yang dibutuhkan untuk eksekusi *taks* akan semakin lama ketika jarak yang terdeteksi semakin jauh.

Pada Tabel 6.3 masing-masing jarak yang sudah ditentukan memiliki nilai rata-rata dari 1,0575ms-1,636ms. Meningkatnya nilai eksekusi ketika semakin jauh jarak yang terdeteksi disebabkan sinyal ultrasonik yang ditembakkan oleh *trigger* modul ultrasonik akan semakin lama untuk kembali ke echo jika objek yang digunakan untuk memantulkan sinyal ultrasonik semakin jauh.

6.3 Pengujian *Real-Time Operating System* setiap *task* dengan jarak yang sama

6.3.1 Tujuan

Untuk mengetahui waktu eksekusi setiap *task* dalam menyelesaikan tugas dengan sample pengujian 10 kali.

6.3.2 Prosedur

Untuk membantu perhitungan waktu eksekusi tiap *task* digunakan fungsi timer pada Arduino yaitu fungsi "*micros()*" untuk menghitug waktu eksekusi dalam satuan *microsecond* dan "*millis()*" untuk menghitung waktu dalam satuan *millisecond*. Fungsi *micros()* yang digunakan dalam pengujian ini dikarenakan mebutuhkan perhitungan *timer* yang lebih detail. Fungsi *micros()* diletakkan pada awal program setelah fungsi *vTaskDelay* untuk mendapatkan waktu eksekusi *task* tersebut dan akhir program dari setiap *task*. Pengujian ini dilakukan sebanyak 10 kali. Untuk mengetahui kestabilan nilai waktu eksekusi task. Jarak yang dipilih untuk pengujian adalah 6cm. Prosedur pengujian waktu eksekusi *task* meliputi:

1. Sambungkan prototipe alat dengan komputer atau laptop
2. Buka *source code* alat yang dibuat untuk menghitung waktu eksekusi *task* dengan memanfaatkan fungsi *micros()*. Setelah itu *compile* dan *upload source code* programnya.
3. Berikan suatu objek didepan sensor untuk memantulkan gelombang ultrasonik dengan jarak yang sudah ditentukan.
4. Amati hasil melalui *serial monitor*.
5. Kesimpulan.

6.3.3 Hasil dan analisis

Hasil dari pengujian waktu eksekusi *task* pada sistem ditampilkan pada Tabel 6.5.

Tabel 6.4 Analisis hasil pengujian waktu eksekusi *task* dengan jarak sama

Sample	TaskUS (ms)							
	1	2	3	4	5	6	7	8
1	1,092	1,128	1,116	1,124	1,088	1,096	1,472	1,584
2	1,092	1,092	1,120	1,136	1,076	1,072	1,516	1,580
3	1,092	1,116	1,084	1,116	1,080	1,108	1,504	1,580
4	1,096	1,124	1,116	1,112	1,084	1,064	1,528	1,584
5	1,092	1,124	1,108	1,108	1,088	1,064	1,520	1,584
6	1,092	1,120	1,092	1,064	1,076	1,076	1,588	1,580
7	1,092	1,128	1,052	1,128	1,084	1,076	1,520	1,584
8	1,120	1,124	1,080	1,140	1,088	1,088	1,508	1,584
9	1,116	1,124	1,064	1,160	1,084	1,068	1,524	1,584
10	1,124	1,096	1,100	1,120	1,080	1,064	1,540	1,584
Rata-rata	**1,100**	**1,005**	**1,093**	**1,120**	**1,083**	**1,077**	**1,522**	**1,583**

Berdasarkan analisis Tabel 6.5, masing-masing *task* memiliki rata-rata waktu eksekusi yang hampir sama karena masih bernilai 1ms. Akan tetapi selisih waktu

eksekusi dari masing-masing *task* tidak terlalu jauh. Dikarenakan disetiap *task* memiliki proses yang sama dengan *task* yang lain. Setiap task memproses pebacaan sensor, menampilkan hasil pembacaan sensor dan menyalakan LED ketika jarak yang terdeteksi sensor kurang dari sama dengan 15cm.

6.4 Pengujian perbandingan total waktu eksekusi sistem multi-sensor menggunakan *Real-Time Operating System* (RTOS) dan tanpa RTOS

6.4.1 Tujuan

Untuk membandingkan sistem yang menggunakan RTOS dengan sistem tanpa menggunakan RTOS dari segi kecepatan sistem memproses seluruh *task*.

6.4.2 Prosedur

Dalam pengujian ini sudah dibuat *source code* sederhana yang sama namun tidak menggunakan RTOS dalam penerapannya. *Source code* tersebut menggunakan pengimplementasian *multitasking* yang lain yaitu *millis()*. Pada *source code* tersebut menggunakan 3 fungsi untuk memprosesnya, yaitu fungsi untuk melakukan sensing, fungsi untuk perhitungan sensing dan fungsi untuk melakukan *output* hasil sensing. Pada *source code* tersebut akan menggunakan fungsi *micros()* sebagai penghitung waktu total sistem untuk menyelesaikan sensing, perhitungan nilai hasil sensing, dan menampilkan data hasil sensing dari 8 modul sensor HC-SR04.

Sedangkan sistem yang menggunakan RTOS, maka akan diberikan fungsi *micros()* pada *task* yang memiliki posisi paling atas untuk mengetahui waktu awal *task* tersebut berjalan dan *micros()* pada *task* yang terletak dipaling bawah untuk mengetahui waktu akhir dari *task* tersebut. Nantinya waktu akhir dari *task* akan dikurangi waktu awal dari *task* untuk memperoleh nilai total waktu yang diperoleh sistem tersebut untuk menyelesaikan *task-task* tersebut. Jarak yang digunakan untuk masing-masing metode *multitasking* adalah 6cm. Prosedur pengujian waktu eksekusi *task* meliputi:

1. Sambungkan prototipe alat dengan komputer atau laptop
2. Buka *source code* alat tanpa menggunakan RTOS untuk menghitung waktu eksekusi *task* dengan memanfaatkan fungsi *micros()*. Setelah itu compile dan *upload source code* programnya.
3. Amati hasil melalui *serial monitor*.
4. Selanjutnya *upload source code* alat dengan menggunakan RTOS untuk menghitung waktu eksekusi *task* dengan memanfaatkan fungsi *micros()*. Setelah itu *compile* dan *upload source code* programnya.
5. Amati hasil melalui *serial monitor*.
6. Kesimpulan.

6.4.3 Hasil dan analisa

Hasil dari pengujian waktu eksekusi *task* pada sistem ditampilkan pada Tabel 6.6

Tabel 6.5 Analisis hasil pengujian waktu eksekusi RTOS dan tanpa RTOS

Sample	Waktu eksekusi dengan RTOS (ms)	Waktu eksekusi tanpa RTOS (ms)
1	8,120	264
2	8,140	264
3	7,992	264
4	8,180	264
5	8,148	264
6	8,072	264
7	8,152	264
8	8,196	264
9	8,152	264
10	8,180	264
Rata-rata	**8,1332**	**264**

Berdasarkan analisi Tabel 6.6, sistem yang tidak menggunakan RTOS lebih lama dari pada sistem yang menggunakan RTOS. Dikarenakan sistem yang tidak menggunakan RTOS terdapat nilai interval tiap *task* yaitu sebesar 33ms (*milisecond*) untuk memberikan waktu pemrosesan setiap sensor pada setiap *task*. Sedangkan pada sistem yang menggunakan RTOS pemberian waktu pemrosesan *task* dalam memproses sensor ditentukan oleh FreeRTOS yang menjadikan sistem yang menggunakan RTOS lebih cepat dari pada sistem yang tidak menggunakan RTOS. Di sistem tanpa RTOS terdapat 3 fungsi untuk memproses sistem, yang pertama fungsi untuk melakukan *sensing* 8 sensor yang ada pada sistem, yang kedua fungsi untuk melakukan perhitungan dari hasil *sensing* tersebut agar dapat dimengerti, dan yang terakhir untuk menampilkan *output* dan menyalakan LED.

Jika dibandingkan dengan sistem yang menggunakan RTOS, 3 proses yang ada pada sistem tanpa RTOS sudah menjadi satu pada satu *task*. Yang mana ketika satu sensor melakukan *sensing* maka langsung diproses di dalam *task* dari sensor tersebut. tidak perlu menunggu proses sensing dari 8 sensor, lalu perhitungan nilai dari hasil *sensing* 8 sensor, dan baru ditampilkan. Didalam sistem sistem yang menggunakan RTOS semua itu sudah dikerjakan pada satu *task* untuk satu sensor.

BAB 5 PERANCANGAN DAN IMPLEMENTASI

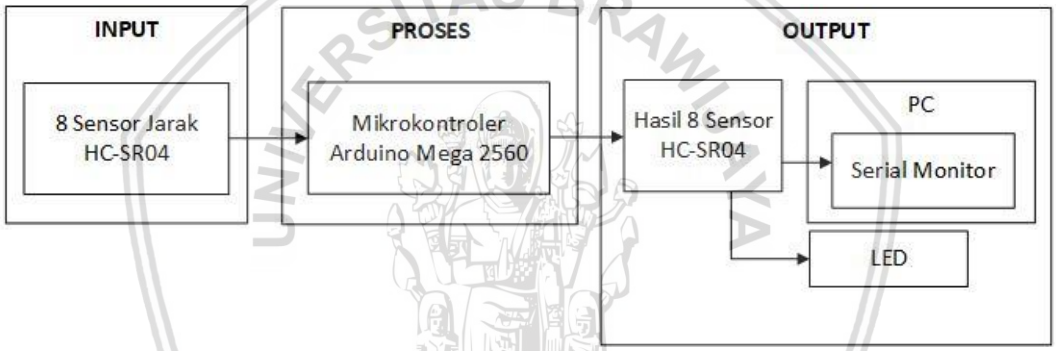
Bab ini akan membahas mengenai perancangan dari sistem yang akan dibuat dan berdasarkan hasil perancangan tersebut akan dilakukan implementasi sistem yang terdiri dari implementasi perangkat lunak dan perangkat keras.

5.1 Perancangan sistem

Pada tahap perancangan ini menjelaskan terkait tahapan perancangan multi-sensor ultrasonik meliputi perancangan perangkat keras dan perancangan perangkat lunak agar sistem tersebut dapat bekerja dengan tepat.

5.1.1 Gambaran umum sistem

Perancangan sistem ini terdapat 3 bagian yaitu *input*, proses dan *output*. Gambar 5.1 merupakan diagram blok dari sistem yang dirancang.



Gambar 5.1 Diagram blok sistem

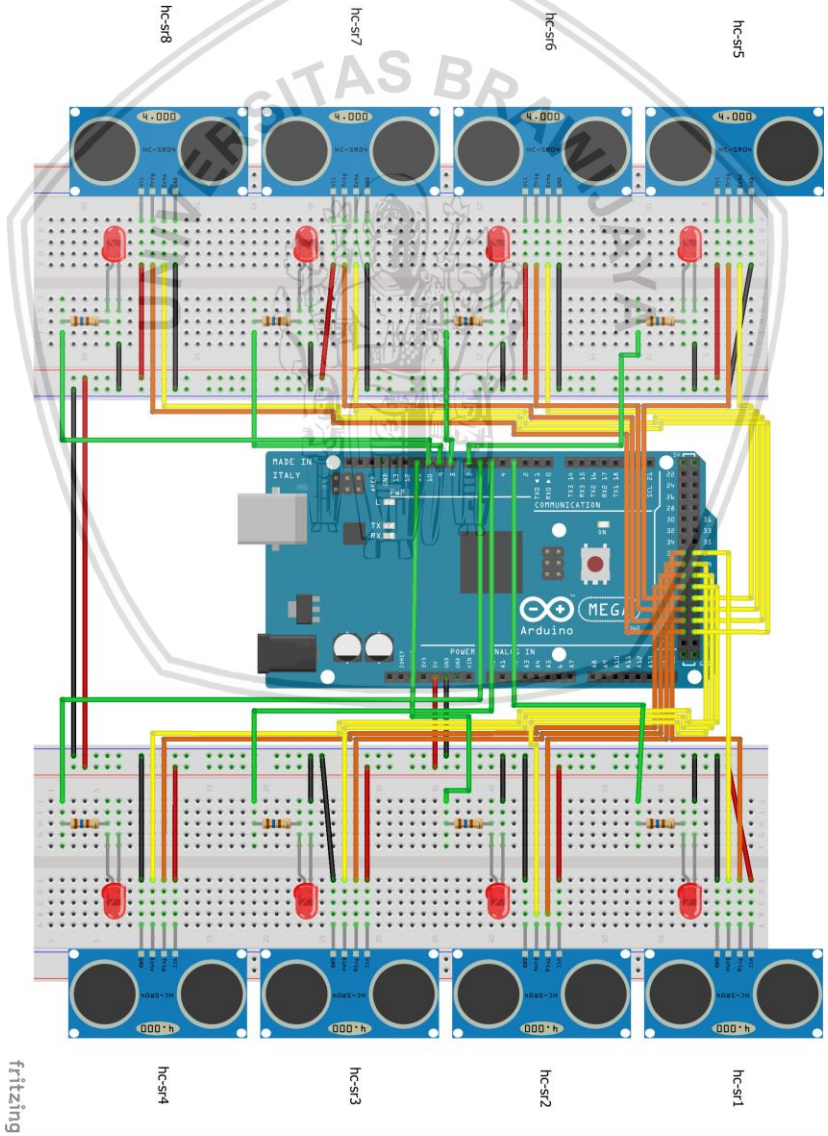
Diaram blok diatas merupakan gambaran umum terkait rancangan sistem yang dibuat, terdapat tiga bagian beruap *input*, proses dan *output*. Pada bagian *input* terdapat 8 sensor jarak HC-SR04 untuk mendeteksi jarak. Setiap hasil sensing dari setiap sensor diteruskan pada mikrokontroler Arduino Mega 2560 untuk diproses dan untuk menentukan jalannya *task* yang digunakan untuk memproses data hasil sensing HC-SR04. *Task* yang digunakan menggunakan implementasi *RTOS* di Arduino dengan memanfaatkan *library FreeRTOS*.

Ketika awal memasuki *task* terdapat proses inisialisasi *baudrate* yang digunakan sebesar 115200 bps. Setelah proses inisialisasi *baudrate* terdapat *vTaskDelay* sebesar 16 milisecond yang digunakan untuk memberi waktu sensor agar siap dalam proses akuisisi data. Jika pada *task* tersebut tidak diberi *vTaskDelay* minimal sebesar 16 milisecond maka akuisisi data akan kacau, pada *serial monitor* Arduino IDE akan memunculkan sebuah karakter atau simbol yang seharusnya tidak ditampilkan dan hasil output *LED* akan berkedip tidak menyala stabil ketika mendeteksi jarak kurang dari sama dengan 15cm. Di dalam *task* tersebut terdapat sebuah kondisi untuk menyalakan atau mematikan *LED*. Ketika sensor mendeteksi jarak lebih dari 15cm maka *LED* dalam kondisi mati, dan sebaliknya jika sensor mendeteksi jarak kurang dari sama dengan 15cm dan tidak

sama dengan 0cm maka *LED* akan dalam kondisi menyala. Dan akhir *task* terdapat *vTaskDelay* sebesar 16 milisecond untuk memberikan waktu pada *RTOS* menyiapkan proses penjadwalan. Ketika di akhir program tidak ada *vTaskDelay* maka *RTOS* tidak bisa berjalan dikarenakan *RTOS* tidak memiliki waktu untuk mengatur proses penjadwalan. *vTaskDelay* untuk *task* sebanyak 8 ini membutuhkan minimal waktu 16 millisecond.

5.1.2 Perancangan perangkat keras

Perancangan sistem multi-sensor terdiri dari beberapa modul yang dirangkai menjadi satu agar sistem berjalan sesuai dengan tujuan. Gambar skematik diagram dari rangkaian sistem multi-sensor ditunjukkan pada Gambar 5.2. Pada rangkaian sistem multi-sensor terdapat beberapa komponen diantranya adalah mikrokontroller Arduino Mega 2560, modul sensor HC-SR04 sebanyak, *LED* merah sebanyak 8 buah, dan resistor 160 Ω disetiap *anoda LED*.



Gambar 5.2 Skematik diagram sistem multi-sensor

5.1.2.1 Perancangan modul sensor HC-SR04

Pada sistem multi-sensor modul sensor ultrasonik merupakan modul yang berfungsi untuk mengakuisisi data jarak dari 2 cm – 400cm. secara garis besar pada modul sensor HC-SR04 terdapat 2 buah ultrasonik, satu sebagai pengirim dan satunya lagi sebagai menerima sinyal *pulse* ultrasonik. Modul sensor ini memiliki empat pin yang terdiri dari pin VCC, GND, ECHO, TRIG. Pada bagian TRIG pada sensor tersebut berfungsi sebagi pengirim sedangkan echo sebagai penerima. Konfigurasi pin pada rangkaian modul sensor HC-SR04 dengan mikrokontroler ditunjukkan pada tabel 5.1.

Tabel 5.1 Keterangan pin modul sensor HC-SR04 ke Arduino Mega 2560

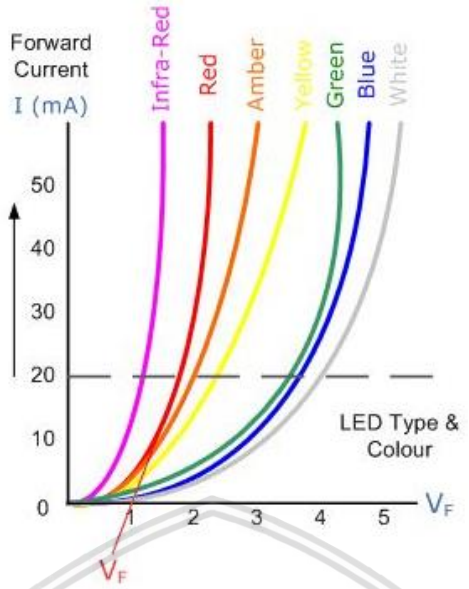
Modul HC-SR04 Ke-	Pin HC-SR04 Ke Arduino Mega 2560			
	VCC	TRIG	ECHO	GND
1	VCC	36	37	GND
2	VCC	38	39	GND
3	VCC	40	41	GND
4	VCC	42	43	GND
5	VCC	44	45	GND
6	VCC	46	47	GND
7	VCC	48	49	GND
8	VCC	50	51	GND

5.1.2.2 Perancangan *LED* (*Light Emitting Diode*)

LED pada sistem multi-sensor ini membutuhkan sebuah resistor sebesar 160 Ω. Karena setiap *LED* memiliki tegangan kerjanya masing-masing, *LED* yang digunakan pada sistem ini berwarna merah yang mana pada tegangan kerjanya sebesar 1.8 V, penjelasan besar tegangan kerja masing-masing warna *LED* dapat dilihat Gambar 5.3. untuk menentukan besar resistor yang dibutuhkan menggunakan perhitungan Persamaan 5.1 sebagai berikut :

$$\boldsymbol{R = \frac{(Vs - Vd)}{I}} \qquad \textbf{(5.1)}$$

Nilai *Vs* adalah besaran tegangan yang didapat nilai tegangan dari mikrokontroler sebesar 5V sedangkan untuk *I* adalah besaran arus dari *LED* itu sendiri. Untuk besaran arus tiap warna *LED* memiliki arus yang sama besar yaitu 20 miliamper atau setara dengan 0.02 ampere. Dari perhitungan menggunakan Persamaan 5.1 tersebut maka hasil yang diperoleh adalah 160Ω. Resistor diletakkan pada bagian anoda dari *LED* untuk mengurangi tegangan yang di berikan oleh mikrokontroler Arduino Mega 2560 sebesar 5V sedangkan *LED* merah hanya membutuhkan 1.8V. Jika *LED* diberikan tegangan melebihi tegangan kerja dari *LED* itu sendiri maka dapat mengakibatkan *LED* tersebut me*LED*ak.



Gambar 5.3 Grafik tegangan kerja tiap warna *LED*

Sumber: (electronics-tutorials.ws, 2017)

Konfigurasi pin pada rangkaian *LED* dengan mikrokontroler ditunjukkan pada table 5.2.

Tabel 5.2 Keterangan pin *LED* ke Arduino Mega 2560

		Pin Arduino Mega 2560
Pin *LED* 1	**Katoda**	GND
	Anoda	3
Pin *LED* 2	**Katoda**	GND
	Anoda	11
Pin *LED* 3	**Katoda**	GND
	Anoda	5
Pin *LED* 4	**Katoda**	GND
	Anoda	6
Pin *LED* 5	**Katoda**	GND
	Anoda	7
Pin *LED* 6	**Katoda**	GND
	Anoda	8
Pin *LED* 7	**Katoda**	GND
	Anoda	9
Pin *LED* 8	**Katoda**	GND
	Anoda	10

5.1.3 Perancangan perangkat lunak

Perangkat lunak terdapat pada sistem ini berupa program pada mikrokontroler Arduino Mega 2560 yang diterapkan *RTOS* sebagai pengatur *multitasking* pada multi-sensor dengan menerapkan sebuah sistem penjadwalan yang bersifat preemptive dari *RTOS*.

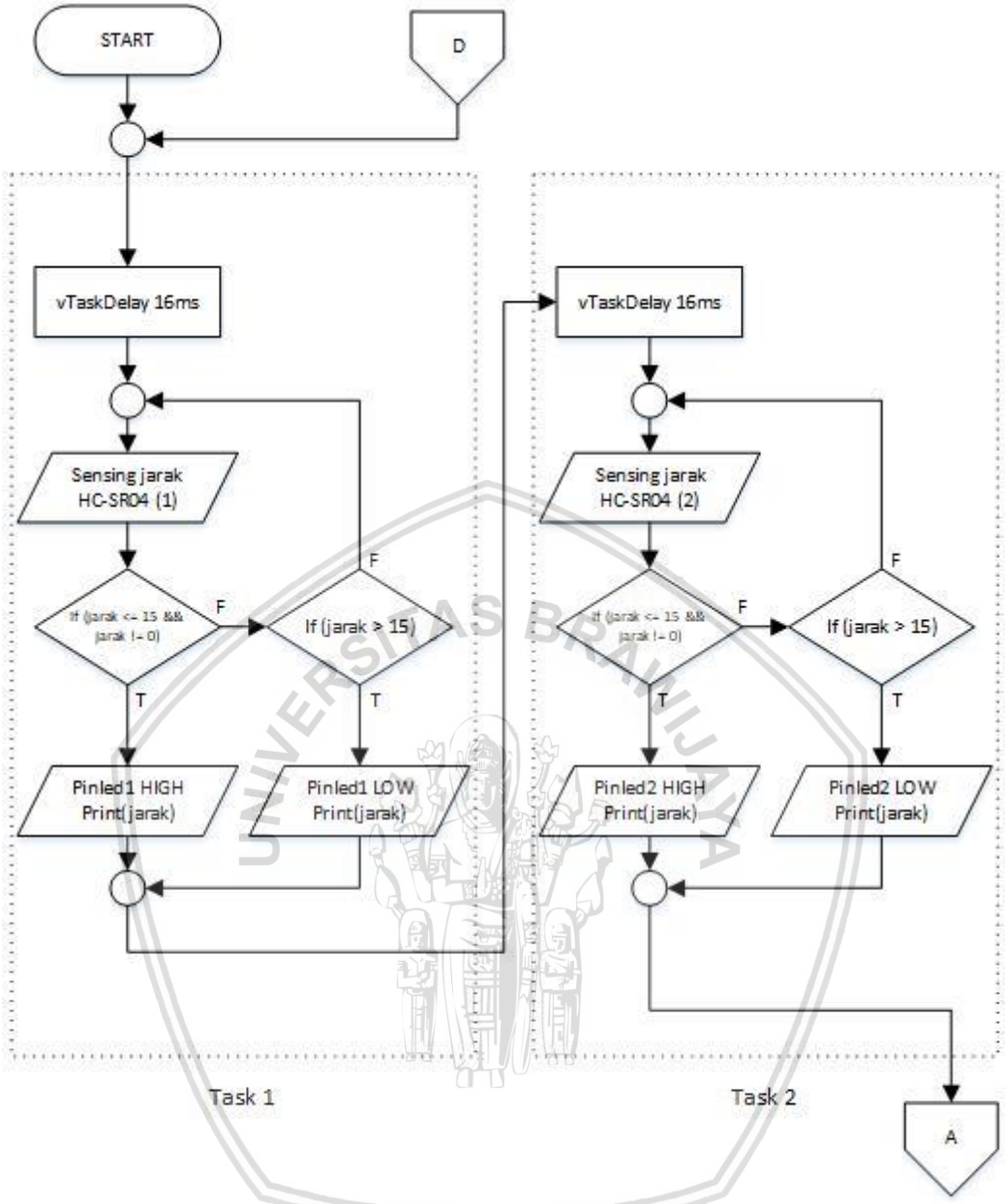
Perancangan perangkat lunak dimulai dari *input* sebuah jarak yang didapat dari sensor HC-SR04 sebanyak 8 buah. Data dari *input* ini akan disimpan pada suatu variabel. Kemudian data tersebut akan diproses oleh setiap *task* yang bertugas sebagai pemroses data hasil *input* dari setiap sensor HC-SR04.

Untuk pengimplementasian *Real-Time Operating System* (*RTOS*), sistem akan dibagi menjadi 8 *task*, dimana dari 8 *task* tersebut memiliki fungsi yaitu memproses hasil sensing dari setiap sensor. *Task* 1 akan memproses hasil sensing dari sensor 1 dan akan mengatur nyala dan mati *LED* 1. Dari *task* 1 sampai *task* 8 memiliki peran seperti itu.

Output dari sistem ini berupa nyala dan mati *LED* dan hasil sensing dari 8 sensor HC-SR04 ke *Serial monitor* Arduino IDE. Kondisi yang terdeteksi pada sistem akan mempengaruhi nyala dan mati pada *LED*, semakin dekat jarak yang terdeteksi maka *LED* akan menyala (jarak yang terdeteksi kurang dari sama dengan 15cm dan tidak sama dengan 0cm) sebaliknya jika jarak yang terdeteksi semakin jauh maka *LED* akan padam (jarak yang terdeteksi lebih dari 15cm).

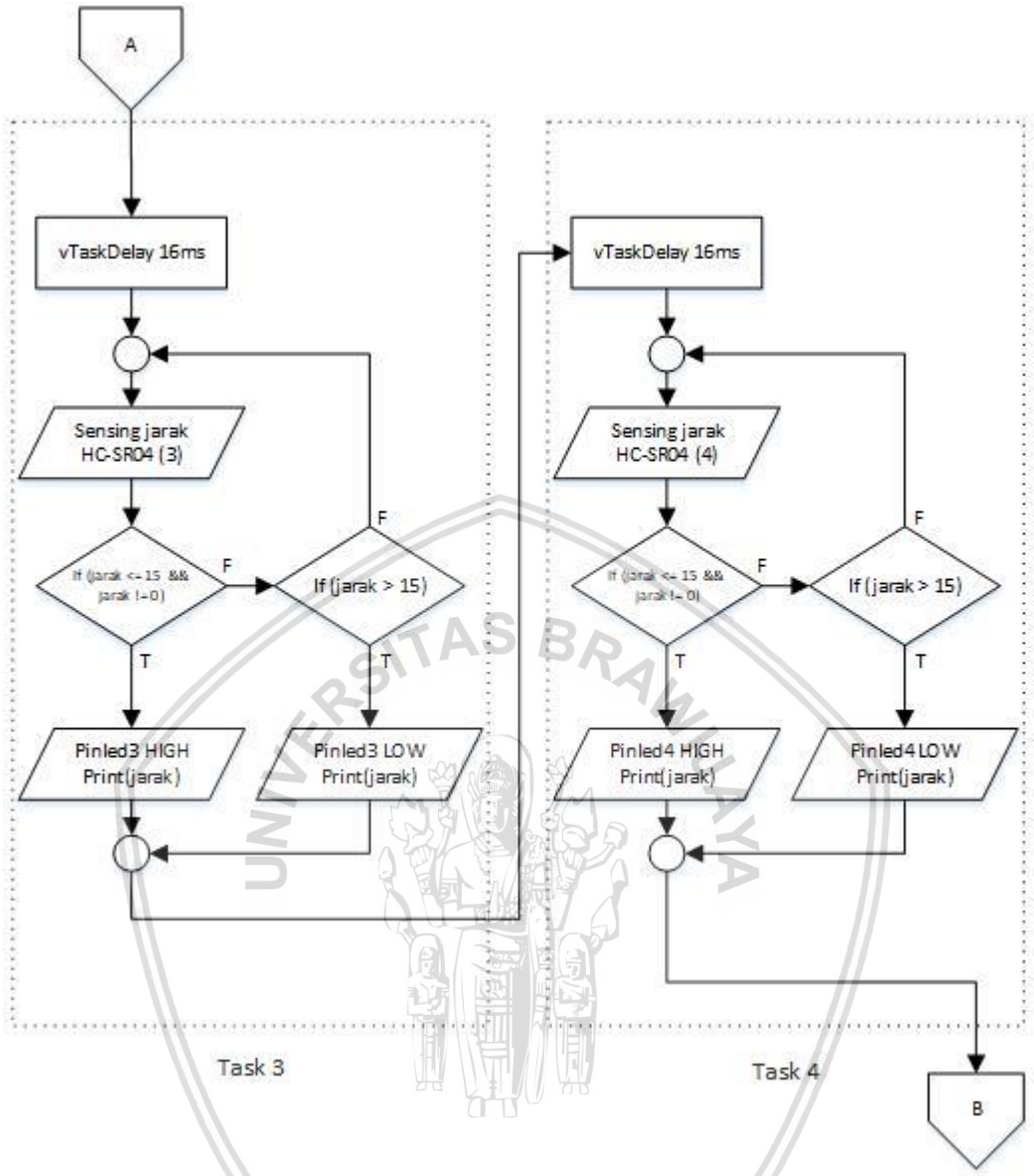
5.1.3.1 Perancangan *Real Time Operating System* (RTOS)

Pada proses perancangan *RTOS* terdiri dari beberapa tahapan yaitu menentukan berapa *task* yang akan digunakan, menentukan tugas tiap *task*, menentukan prioritas tiap *task*, mengatur *stack size* pada task, dan mengatur *deadline* tiap *task*. Pada sistem ini prioritas sistem dijadikan sama semua yaitu bernilai 2. Dengan tujuan agar sistem ini berjalan secara parallel. Sehingga penjadwalan diatur penuh oleh *RTOS*. Faktor lama eksekusi task tergantung dari jarak yang terdeteksi oleh sensor, semakin jauh jarak yang terdeteksi maka akan semakin lama waktu eksekusi nya, sebaliknya jika jarak yang terdeteksi dekat maka waktu eksekusi task akan lebih cepat. Diagram alir perancangan *RTOS* dapat dilihat pada Gambar 5.4, Gambar 5.5, Gambar 5.6 dan Gambar 5.7.



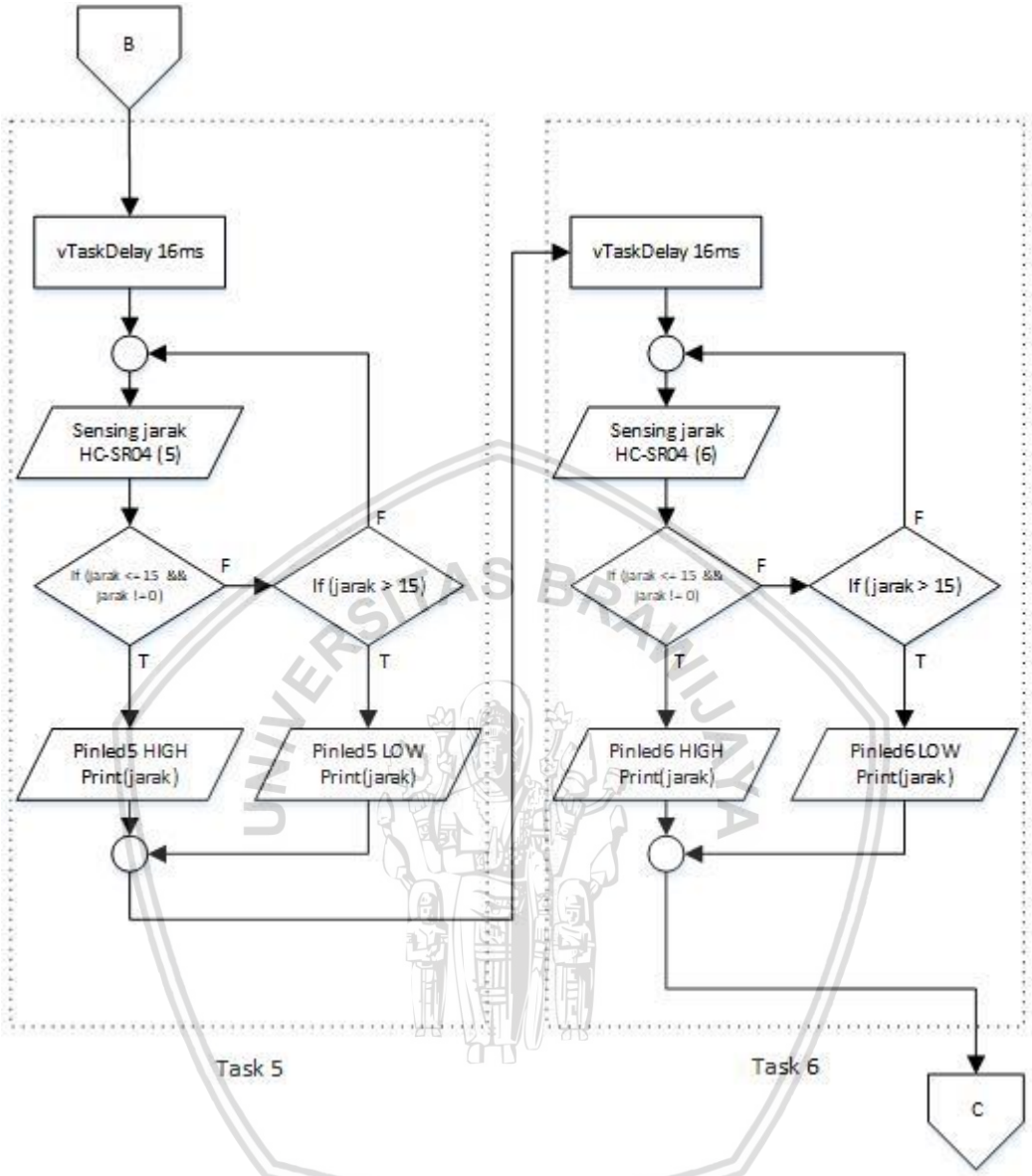
Gambar 5.4 Diagram alur *task* 1 dan *task* 2

Pada Gambar 5.4 terdapat flowchart *task* 1 dan *task* 2. Dari 2 *flowchart* ini terdapat proses pada bagian paling atas setiap *task* terdapat *vTaskDelay* 16ms yang digunakan untuk memberi *deadline* setiap *task.* Pada proses di bawahnya terdapat proses sensing jarak HC-SR04 di masing-masing *task* dan dilanjutkan dengan proses kondisi untuk indikator jarak berupa *LED* yang nantinya akan dinyalakan dan dimatikan tergantung dari jarak yang terdeteksi dan menampilkan nilai berupa angka dengan satuan centimeter. Setelah melakukan semua itu maka akan dilanjutkan dengan memproses *task* selanjutnya.



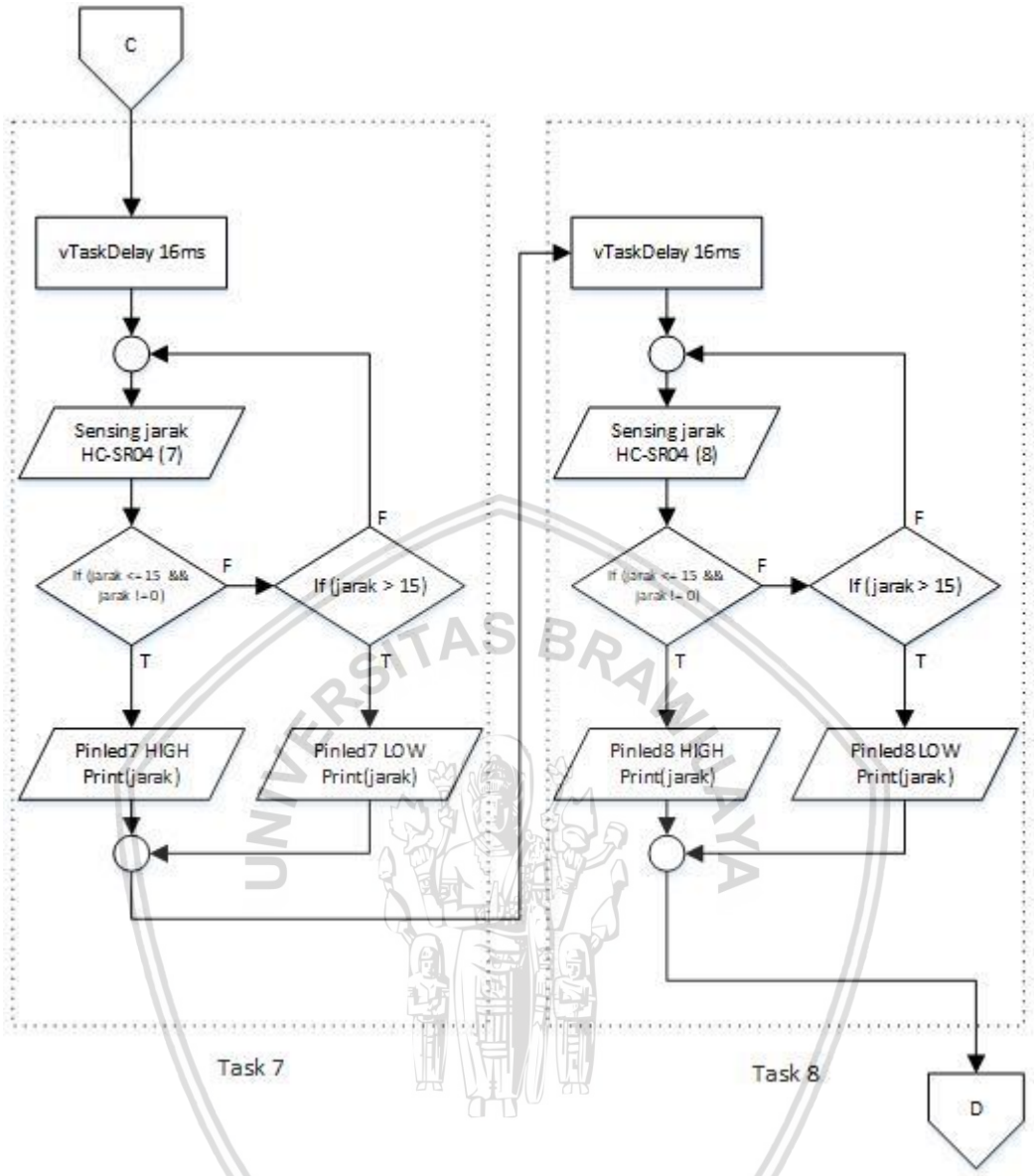
Gambar 5.5 Diagram alur *task* 3 dan *task* 4

Pada Gambar 5.5 terdapat flowchart *task* 3 dan *task* 4. Dari 2 *flowchart* ini terdapat proses pada bagian paling atas setiap *task* terdapat *vTaskDelay* 16ms yang digunakan untuk memberi *deadline* setiap *task.* Pada proses di bawahnya terdapat proses sensing jarak HC-SR04 di masing-masing *task* dan dilanjutkan dengan proses kondisi untuk indikator jarak berupa *LED* yang nantinya akan dinyalakan dan dimatikan tergantung dari jarak yang terdeteksi dan menampilkan nilai berupa angka dengan satuan centimeter. Setelah melakukan semua itu maka akan dilanjutkan dengan memproses *task* selanjutnya.



Gambar 5.6 Diagram alur *task* 5 dan *task* 6

Pada Gambar 5.6 terdapat flowchart *task* 5 dan *task* 6. Dari 2 *flowchart* ini terdapat proses pada bagian paling atas setiap *task* terdapat *vTaskDelay* 16ms yang digunakan untuk memberi *deadline* setiap *task.* Pada proses di bawahnya terdapat proses sensing jarak HC-SR04 di masing-masing *task* dan dilanjutkan dengan proses kondisi untuk indikator jarak berupa *LED* yang nantinya akan dinyalakan dan dimatikan tergantung dari jarak yang terdeteksi dan menampilkan nilai berupa angka dengan satuan centimeter. Setelah melakukan semua itu maka akan dilanjutkan dengan memproses *task* selanjutnya.



Gambar 5.7 Diagram alur *task* 7 dan *task* 8

Pada Gambar 5.7 terdapat flowchart *task* 7 dan *task* 8. Dari 2 *flowchart* ini terdapat proses pada bagian paling atas setiap *task* terdapat *vTaskDelay* 16ms yang digunakan untuk memberi *deadline* setiap *task.* Pada proses di bawahnya terdapat proses sensing jarak HC-SR04 di masing-masing *task* dan dilanjutkan dengan proses kondisi untuk indikator jarak berupa *LED* yang nantinya akan dinyalakan dan dimatikan tergantung dari jarak yang terdeteksi dan menampilkan nilai berupa angka dengan satuan centimeter. Setelah melakukan semua itu maka akan dilanjutkan dengan memproses *task* selanjutnya. Untuk pembagian tugas dan prioritasnya lebih jelasnya dapat dilihat pada Tabel 5.3. Pada tabel tersebut pemberian nilai prioritas pada seluruh *task* sebesar 2. Pemberian prioritas sendiri bertujuan untuk menentukan *task* mana yang akan di eksekusi terlebih dahulu atau *task* yang lebih didahulukan daripada *task* yang lain. Dalam sistem ini prioritas yang diberikan pada masing-masing *task* memiliki nilai yang sama.

Dikarenakan pada sistem ini bertujuan untuk membuat sebuah sistem yang berjalan secara simultan. Maka dari itu dengan pemberian prioritas yang sama setiap *task* bertujuan untuk menjalankan *task* tanpa mendahulukan *task* lain agar sistem bisa berjalan simultan. Proses penjadwalan *task* dari sistem diatur sepenuhnya oleh *RTOS*. Nilai 2 pada prioritas sendiri bertujuan untuk memberikan nilai prioritas *task* yang mendekati dengan nilai prioritas *task* paling bawah yaitu 1 akan tetapi tidak terlalu besar yaitu 3.

Tabel 5.3 Pembagian tugas dan prioritas pada sistem

No.	*Task*	Tugas	Prioritas
1	TaskUS1	Bertugas melakukan akuisisi data sensing pada modul sensor HC-SR04 ke-1, melakukan pengkondisian jarak yang terdeteksi untuk menyalakan atau mematikan *LED* ke-1 dan juga melakukan *output* nilai data hasil sensing pada HC-SR04 ke-1	2
2	TaskUS2	Bertugas melakukan akuisisi data sensing pada modul sensor HC-SR04 ke-2, melakukan pengkondisian jarak yang terdeteksi untuk menyalakan atau mematikan *LED* ke-2 dan juga melakukan *output* nilai data hasil sensing pada HC-SR04 ke-2	2
3	TaskUS3	Bertugas melakukan akuisisi data sensing pada modul sensor HC-SR04 ke-3, melakukan pengkondisian jarak yang terdeteksi untuk menyalakan atau mematikan *LED* ke-3 dan juga melakukan *output* nilai data hasil sensing pada HC-SR04 ke-3	2
4	TaskUS4	Bertugas melakukan akuisisi data sensing pada modul sensor HC-SR04 ke-4, melakukan pengkondisian jarak yang terdeteksi untuk menyalakan atau mematikan *LED* ke-4 dan juga melakukan *output* nilai data hasil sensing pada HC-SR04 ke-4	2
5	TaskUS5	Bertugas melakukan akuisisi data sensing pada modul sensor HC-SR04 ke-5, melakukan pengkondisian jarak yang terdeteksi untuk menyalakan atau mematikan *LED* ke-5 dan juga melakukan *output* nilai data hasil sensing pada HC-SR04 ke-5	2
6	TaskUS6	Bertugas melakukan akuisisi data sensing pada modul sensor HC-SR04 ke-6, melakukan pengkondisian jarak yang terdeteksi untuk menyalakan atau mematikan *LED* ke-6 dan juga melakukan *output* nilai data hasil sensing pada HC-SR04 ke-6	2
7	TaskUS7	Bertugas melakukan akuisisi data sensing pada modul sensor HC-SR04 ke-7, melakukan pengkondisian jarak yang terdeteksi untuk menyalakan atau mematikan *LED* ke-7 dan juga melakukan *output* nilai data hasil sensing pada HC-SR04 ke-7	2
8	TaskUS8	Bertugas melakukan akuisisi data sensing pada modul sensor HC-SR04 ke-8, melakukan pengkondisian jarak yang terdeteksi untuk menyalakan atau mematikan *LED* ke-8 dan juga melakukan *output* nilai data hasil sensing pada HC-SR04 ke-8	2

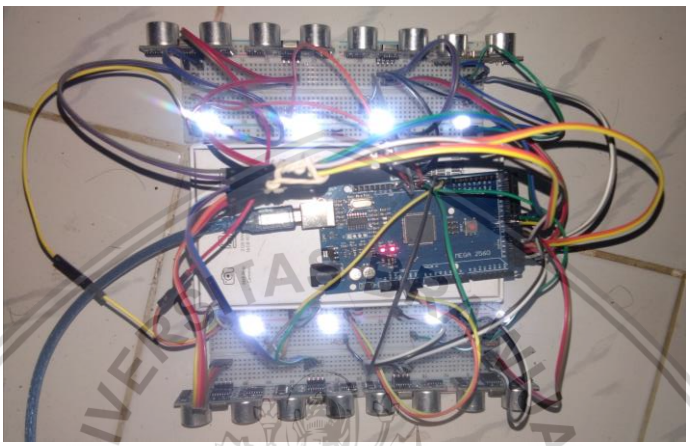
5.2 Implementasi sistem

Pada implementasi sistem akan dibahas implementasi perangkat keras dan perangkat lunak berdasarkan perancangan sistem yang telah dibuat. Penjelasan

ini mencakup spesifikasi sistem, Batasan-batasan implementasi dan implementasi dari *RTOS* yang digunakan.

5.2.1 Implementasi perangkat keras

Pada skripsi ini, implementasi perangkat keras berupa pemasangan komponen Arduino Mega 2560, sensor modul HC-SR04 sebanyak 8 buah dan *LED* 8 buah dalam satu sistem. Pin-pin pada perangkat tersebut dihubungkan sesuai dengan rancangan pada Tabel 5.1 dan Tabel 5.2. Implementasi perangkat keras ditunjukkan pada Gambar 5.8 dibawah.



Gambar 5.8 Implementasi sistem

5.2.2 Implementasi peragkat lunak

Implementasi perangkat lunak pada tugas akhir ini merupakan implementasi sistem dari hasil perancangan yang dibuat sebelumnya. Dalam implementasi perangkat lunak ini direpresentasikan dalam bentuk kode program Bahasa C.

5.2.2.1 Implementasi pembuatan *task* pada *Real Time Operating System*

Dalam penerapan *Real Time Operating System* dalam program, langkah pertama yang harus dilakukan adalah mendefinisikan *task* yang akan digunakan. Sesuai dengan banyak sensor yang digunakan maka didefinisikan 8 *task* yaitu TaskUS1, TaskUS2, TaskUS3, TaskUS4, TaskUS5, TaskUS6, TaskUS7, dan TaskUS8. Deklarasi *task* pada *RTOS* ditampilkan pada Tabel 5.4.

Tabel 5.4 Kode sumber pendefinisi *task*

skripsi*RTOS*.ino

```
. . . . . .

1  void TaskUS1(void *pvParameters);
2  void TaskUS2(void *pvParameters);
3  void TaskUS3(void *pvParameters);
4  void TaskUS4(void *pvParameters);
5  void TaskUS5(void *pvParameters);
6  void TaskUS6(void *pvParameters);
7  void TaskUS7(void *pvParameters);
```

```
8   void TaskUS8(void *pvParameters);

    . . . . . .
```

Pada langkah selanjutnya setelah *task* sudah dideklarasikan maka selanjutnya adalah membuat *task* dan melakukan *setting* pada masing-masing *task*. Pembuatan dan *setting* dari *task* ini meliputi nama, *stack size*, dan *priority*. Untuk nama *task* disesuaikan dengan *task* yang sudah didefinisikan, untuk *stack size* digunakan ukuran default yang diberikan oleh *FreeRTOS*, dan untuk priority disesukian dengan perancangan sistem yang sudah dibuat. Untuk program pembuatan *task* ditampilkan pada Table 5.5.

Tabel 5.5 Kode sumber pembuatan *task*

skripsi*RTOS*.ino

```
    . . . . . .

1   xTaskCreate(
2       TaskUS1
3       ,  (const portCHAR *) "Ultrasonik1"
4       ,  128
5       ,  NULL
6       ,  2
7       ,  NULL );
8     xTaskCreate(
9       TaskUS2
10      ,  (const portCHAR *) "Ultrasonik2"
11      ,  128
12      ,  NULL
13      ,  2
14      ,  NULL );
15    xTaskCreate(
16      TaskUS3
17      ,  (const portCHAR *) "Ultrasonik3"
18      ,  128
19      ,  NULL
20      ,  2
21      ,  NULL );
22    xTaskCreate(
23      TaskUS4
24      ,  (const portCHAR *) "Ultrasonik4"
25      ,  128
26      ,  NULL
27      ,  2
28      ,  NULL );
29    xTaskCreate(
30      TaskUS5
31      ,  (const portCHAR *) "Ultrasonik5"
32      ,  128
33      ,  NULL
34      ,  2
35      ,  NULL );
36    xTaskCreate(
37      TaskUS6
38      ,  (const portCHAR *) "Ultrasonik6"
39      ,  128
```

```
40      ,  NULL
41      ,  2
42      ,  NULL );
43    xTaskCreate(
44      TaskUS7
45      ,  (const portCHAR *) "Ultrasonik7"
46      ,  128
47      ,  NULL
48      ,  2
49      ,  NULL );
50    xTaskCreate(
51      TaskUS8
52      ,  (const portCHAR *) "Ultrasonik8"
53      ,  128
54      ,  NULL
55      ,  2
56      ,  NULL );

      . . . . . .
```

Pada potongan program pada Gambar 5.5 semua *task* memiliki nilai prioritas 2 sama dengan priority dari task yang lain dengan tujuan agar sistem ini berjalan secara paralel. Untuk penjadwalan dengan task yang memiliki priority sama semua diatur penuh oleh *RTOS*. Jalan dari program itu sendiri dimulai dari task yang memiliki posisi paling atas dan memiliki waktu eksekusi paling cepat. Jika task tersebut mendeteksi jarak yang sama semua dan mengakibatkan waktu eksekusi dari tiap task hampir sama, maka task yang akan di eksekusi terlebih dahulu adalah task yang memiliki posisi paling atas atau TaskUS1. Pada bagian *Stack Depth* bernilai 128 yang bertujuan untuk mengalokasikan ukuran *stack* pada task tersebut sebesar 128. Semakin banyak komputasi yang dilakukan task maka membutuhkan semakin besar *stack depth*. Penentuan nilai 128 didapat dari *stack depth* default dari *FreeRTOS*. Karena pada setiap task tidak membutuhkan komputasi yang banyak maka menggunakan *stack depth* sesuai default sudah cukup.

5.2.2.2 Implementasi pembuatan *task* tiap sensor HC-SR04

Setelah *task* dibuat dan diatur langkah selanjutnya adalah mengisi *task* tersebut dengan kode program akuisisi data untuk modul sensor HC-SR04 dan menampilkan nilai hasil akuisisi data sensing modul sensor HC-SR04. Pada implementasi ini jarak yang digunakan pada setiap sensor adalah sama. Implementasi program TaskUS1 ditampilkan pada Table 5.6.

Tabel 5.6 Kode sumber TaskUS1

skripsi*RTOS*.ino

```
      . . . . . .

1     void TaskUS1(void *pvParameters)
2     {
3       (void) pvParameters;
4       Serial.begin(115200 bps);
5       vTaskDelay( 16 / portTICK_PERIOD_MS );
6       for (;;)
7       {
```

```
8       int SRsensor = sonar[0].ping_cm();
9       int pinLED = LED[0];
10      if (SRsensor <= 15 && SRsensor != 0)
11       {
12          Serial.print("Ultra Sonik 1 : ");
13          Serial.println(SRsensor);
14          digitalWrite(pinLED, HIGH);
15       }
16      if (SRsensor > 15)
17       {
18          Serial.print("Ultra Sonik 1 : ");
19          Serial.println(SRsensor);
20          digitalWrite(pinLED, LOW);
21       }
22     }
23  }

        . . . . . .
```

Berdasarkan pada Tabel 5.6, implementasi pada TaskUS1 diawali dengan memberikan *baudrate* sebesar 115200 bps untuk keperluan komunikasi *serial monitor* yang mana pada *library* NewPing.h memerlukan *baudrate* sebesar 115200 bps untuk berkomunikasi dengan *serial monitor*. Lalu pada proses selanjutnya ada *vTaskDelay* sebesar 16 ms untuk memberikan waktu pada task dalam melakukan eksekusi, karena dalam jarak 15cm yang digunakan untuk pengkondisian jarak diperlukan waktu 15 ms untuk memproses task tersebut. jika vTaskDelay lebih kecil dari waktu eksekusi task maka akan mengakibatkan lampu *LED* yang akan nyala mati dikarenakan proses untuk menyalakan *LED* yang melebihi waktu vTaskDelay yang sudah ditentukan akan dipaksa selesai dan berpindah ke task yang lain. Selanjutnya masuk pada perulangan yang didalamnya terdapat deklarasi sebuah variabel SRsensor dengan tipe data *integer*, yang mana variabel tersebut untuk menyimpan data hasil sensing modul sensor HC-SR04 yang ke-1 dan deklarasi variabel selanjutnya adalah variabel pinLED dengan tipedata *integer*, yang berfungsi menyimpan nilai alamat pin dari *LED* yang ke-1. Setelah deklarasi variabel tersebut ada dua buah kondisi dengan menggunakan if, yang mana pada kondisi pertama adalah jika sensor mendeteksi jarak kurang dari sama dengan 15cm dan tidak sama dengan 0cm maka akan masuk pada kondisi pertama yaitu pinLED bernilai HIGH dan menampilkan nilai hasil sensing tersebut ke *serial monitor* pada Arduino IDE. Untuk kondisi yang kedua menggunakan if untuk pengkondisian yang mana jika jarak yang terdeteksi lebih dari 15cm maka pinLED bernilai LOW dan akan menampilkan nilai hasil sensing ke *serial monitor* Arduino IDE. Setelah selesai maka program akan berpindah ke task yang berada tepat dibawah task ini. Implementasi program TaskUS2 ditampilkan pada Tabel 5.7.

Tabel 5.7 Kode sumber TaskUS2

skripsi*RTOS*.ino

```
        . . . . . .

1   void TaskUS2(void *pvParameters)
2   {
3     (void) pvParameters;
```

```
4     Serial.begin(115200 bps);
5     vTaskDelay( 16 / portTICK_PERIOD_MS );
6     for (;;)
7     {
8       int SRsensor = sonar[1].ping_cm();
9       int pinLED = LED[1];
10      if (SRsensor <= 15 && SRsensor != 0)
11       {
12          Serial.print("Ultra Sonik 2 : ");
13          Serial.println(SRsensor);
14          digitalWrite(pinLED, HIGH);
15       }
16      if (SRsensor > 15)
17       {
18          Serial.print("Ultra Sonik 2 : ");
19          Serial.println(SRsensor);
20          digitalWrite(pinLED, LOW);
21       }
22     }
23  }

. . . . . .
```

Berdasarkan pada Tabel 5.7, implementasi pada TaskUS2 diawali dengan memberikan *baudrate* sebesar 115200 bps untuk keperluan komunikasi *serial monitor* yang mana pada *library* NewPing.h memerlukan *baudrate* sebesar 115200 bps untuk berkomunikasi dengan *serial monitor*. Lalu pada proses selanjutnya ada *vTaskDelay* sebesar 16 ms untuk memberikan waktu pada task dalam melakukan eksekusi, karena dalam jarak 15cm yang digunakan untuk pengkondisian jarak diperlukan waktu 15 ms untuk memproses task tersebut. jika vTaskDelay lebih kecil dari waktu eksekusi task maka akan mengakibatkan lampu *LED* yang akan nyala mati dikarenakan proses untuk menyalakan *LED* yang melebihi waktu vTaskDelay yang sudah ditentukan akan dipaksa selesai dan berpindah ke task yang lain.

Selanjutnya masuk pada perulangan yang didalamnya terdapat deklarasi sebuah variabel SRsensor dengan tipe data *integer*, yang mana variabel tersebut untuk menyimpan data hasil sensing modul sensor HC-SR04 yang ke-2 dan deklarasi variabel selanjutnya adalah variabel pinLED dengan tipedata *integer*, yang berfungsi menyimpan nilai alamat pin dari *LED* yang ke-2. Setelah deklarasi variabel tersebut ada dua buah kondisi dengan menggunakan if, yang mana pada kondisi pertama adalah jika sensor mendeteksi jarak kurang dari sama dengan 15cm dan tidak sama dengan 0cm maka akan masuk pada kondisi pertama yaitu pinLED bernilai HIGH dan menampilkan nilai hasil sensing tersebut ke *serial monitor* pada Arduino IDE.

Untuk kondisi yang kedua menggunakan if untuk pengkondisian yang mana jika jarak yang terdeteksi lebih dari 15cm maka pinLED bernilai LOW dan akan menampilkan nilai hasil sensing ke *serial monitor* Arduino IDE. Setelah selesai maka program akan berpindah ke task yang berada tepat dibawah task ini. Implementasi program TaskUS3 ditampilkan pada Tabel 5.8.

Tabel 5.8 Kode sumber TaskUS3

skripsi*RTOS*.ino

```
. . . . . .

void TaskUS3(void *pvParameters)
{
  (void) pvParameters;
  Serial.begin(115200 bps);
  vTaskDelay( 16 / portTICK_PERIOD_MS );
  for (;;)
  {
    int SRsensor = sonar[2].ping_cm();
    int pinLED = LED[2];
    if (SRsensor <= 15 && SRsensor != 0)
     {
        Serial.print("Ultra Sonik 3 : ");
        Serial.println(SRsensor);
        digitalWrite(pinLED, HIGH);
     }
    if (SRsensor > 15)
     {
        Serial.print("Ultra Sonik 3 : ");
        Serial.println(SRsensor);
        digitalWrite(pinLED, LOW);
     }
   }
}

. . . . . .
```

Berdasarkan pada Tabel 5.8, implementasi pada TaskUS3 diawali dengan memberikan *baudrate* sebesar 115200 bps untuk keperluan komunikasi *serial monitor* yang mana pada *library* NewPing.h memerlukan *baudrate* sebesar 115200 bps untuk berkomunikasi dengan *serial monitor*. Lalu pada proses selanjutnya ada *vTaskDelay* sebesar 16 ms untuk memberikan waktu pada task dalam melakukan eksekusi, karena dalam jarak 15cm yang digunakan untuk pengkondisian jarak diperlukan waktu 15 ms untuk memproses task tersebut. jika vTaskDelay lebih kecil dari waktu eksekusi task maka akan mengakibatkan lampu *LED* yang akan nyala mati dikarenakan proses untuk menyalakan *LED* yang melebihi waktu vTaskDelay yang sudah ditentukan akan dipaksa selesai dan berpindah ke task yang lain.

Selanjutnya masuk pada perulangan yang didalamnya terdapat deklarasi sebuah variabel SRsensor dengan tipe data *integer*, yang mana variabel tersebut untuk menyimpan data hasil sensing modul sensor HC-SR04 yang ke-3 dan deklarasi variabel selanjutnya adalah variabel pinLED dengan tipedata *integer*, yang berfungsi menyimpan nilai alamat pin dari *LED* yang ke-3. Setelah deklarasi variabel tersebut ada dua buah kondisi dengan menggunakan if, yang mana pada kondisi pertama adalah jika sensor mendeteksi jarak kurang dari sama dengan

15cm dan tidak sama dengan 0cm maka akan masuk pada kondisi pertama yaitu pinLED bernilai HIGH dan menampilkan nilai hasil sensing tersebut ke *serial monitor* pada Arduino IDE.

Untuk kondisi yang kedua menggunakan if untuk pengkondisian yang mana jika jarak yang terdeteksi lebih dari 15cm maka pinLED bernilai LOW dan akan menampilkan nilai hasil sensing ke *serial monitor* Arduino IDE. Setelah selesai maka program akan berpindah ke task yang berada tepat dibawah task ini. Implementasi program TaskUS4 ditampilkan pada Tabel 5.9.

Tabel 5.9 Kode sumber TaskUS4

skripsi*RTOS*.ino

```
. . . . . .

void TaskUS4(void *pvParameters)
{
  (void) pvParameters;
  Serial.begin(115200 bps);
  vTaskDelay( 16 / portTICK_PERIOD_MS );
  for (;;)
  {
    int SRsensor = sonar[3].ping_cm();
    int pinLED = LED[3];
    if (SRsensor <= 15 && SRsensor != 0)
     {
        Serial.print("Ultra Sonik 4 : ");
        Serial.println(SRsensor);
        digitalWrite(pinLED, HIGH);
     }
    if (SRsensor > 15)
     {
        Serial.print("Ultra Sonik 4 : ");
        Serial.println(SRsensor);
        digitalWrite(pinLED, LOW);
     }
   }
}

. . . . . .
```

Berdasarkan pada Tabel 5.9, implementasi pada TaskUS4 diawali dengan memberikan *baudrate* sebesar 115200 bps untuk keperluan komunikasi *serial monitor* yang mana pada *library* NewPing.h memerlukan *baudrate* sebesar 115200 bps untuk berkomunikasi dengan *serial monitor*. Lalu pada proses selanjutnya ada *vTaskDelay* sebesar 16 ms untuk memberikan waktu pada task dalam melakukan eksekusi, karena dalam jarak 15cm yang digunakan untuk pengkondisian jarak diperlukan waktu 15 ms untuk memproses task tersebut. jika vTaskDelay lebih kecil dari waktu eksekusi task maka akan mengakibatkan lampu *LED* yang akan nyala mati dikarenakan proses untuk menyalakan *LED* yang melebihi waktu vTaskDelay yang sudah ditentukan akan dipaksa selesai dan berpindah ke task yang lain.

Selanjutnya masuk pada perulangan yang didalamnya terdapat deklarasi sebuah variabel SRsensor dengan tipe data *integer*, yang mana variabel tersebut untuk menyimpan data hasil sensing modul sensor HC-SR04 yang ke-4 dan deklarasi variabel selanjutnya adalah variabel pinLED dengan tipedata *integer*, yang berfungsi menyimpan nilai alamat pin dari *LED* yang ke-4. Setelah deklarasi variabel tersebut ada dua buah kondisi dengan menggunakan if, yang mana pada kondisi pertama adalah jika sensor mendeteksi jarak kurang dari sama dengan 15cm dan tidak sama dengan 0cm maka akan masuk pada kondisi pertama yaitu pinLED bernilai HIGH dan menampilkan nilai hasil sensing tersebut ke *serial monitor* pada Arduino IDE.

Untuk kondisi yang kedua menggunakan if untuk pengkondisian yang mana jika jarak yang terdeteksi lebih dari 15cm maka pinLED bernilai LOW dan akan menampilkan nilai hasil sensing ke *serial monitor* Arduino IDE. Setelah selesai maka program akan berpindah ke task yang berada tepat dibawah task ini. Implementasi program TaskUS5 ditampilkan pada Tabel 5.10.

Tabel 5.10 Kode sumber TaskUS5

skripsi*RTOS*.ino

```
. . . . . .

void TaskUS5(void *pvParameters)
{
  (void) pvParameters;
  Serial.begin(115200 bps);
  vTaskDelay( 16 / portTICK_PERIOD_MS );
  for (;;)
  {
    int SRsensor = sonar[4].ping_cm();
    int pinLED = LED[4];
    if (SRsensor <= 15 && SRsensor != 0)
     {
        Serial.print("Ultra Sonik 5 : ");
        Serial.println(SRsensor);
        digitalWrite(pinLED, HIGH);
     }
    if (SRsensor > 15)
     {
        Serial.print("Ultra Sonik 5 : ");
        Serial.println(SRsensor);
        digitalWrite(pinLED, LOW);
     }
   }
}
. . . . . .
```

Berdasarkan pada Tabel 5.10, implementasi pada TaskUS5 diawali dengan memberikan *baudrate* sebesar 115200 bps untuk keperluan komunikasi *serial monitor* yang mana pada *library* NewPing.h memerlukan *baudrate* sebesar 115200 bps untuk berkomunikasi dengan *serial monitor*. Lalu pada proses selanjutnya ada *vTaskDelay* sebesar 16 ms untuk memberikan waktu pada task dalam melakukan eksekusi, karena dalam jarak 15cm yang digunakan untuk pengkondisian jarak diperlukan waktu 15 ms untuk memproses task tersebut. jika vTaskDelay lebih

kecil dari waktu eksekusi task maka akan mengakibatkan lampu *LED* yang akan nyala mati dikarenakan proses untuk menyalakan *LED* yang melebihi waktu vTaskDelay yang sudah ditentukan akan dipaksa selesai dan berpindah ke task yang lain.

Selanjutnya masuk pada perulangan yang didalamnya terdapat deklarasi sebuah variabel SRsensor dengan tipe data *integer*, yang mana variabel tersebut untuk menyimpan data hasil sensing modul sensor HC-SR04 yang ke-5 dan deklarasi variabel selanjutnya adalah variabel pinLED dengan tipedata *integer*, yang berfungsi menyimpan nilai alamat pin dari *LED* yang ke-5. Setelah deklarasi variabel tersebut ada dua buah kondisi dengan menggunakan if, yang mana pada kondisi pertama adalah jika sensor mendeteksi jarak kurang dari sama dengan 15cm dan tidak sama dengan 0cm maka akan masuk pada kondisi pertama yaitu pinLED bernilai HIGH dan menampilkan nilai hasil sensing tersebut ke *serial monitor* pada Arduino IDE. Untuk kondisi yang kedua menggunakan if untuk pengkondisian yang mana jika jarak yang terdeteksi lebih dari 15cm maka pinLED bernilai LOW dan akan menampilkan nilai hasil sensing ke *serial monitor* Arduino IDE. Setelah selesai maka program akan berpindah ke task yang berada tepat dibawah task ini. Implementasi program TaskUS6 ditampilkan pada Table 5.11.

Tabel 5.11 Kode sumber TaskUS6

skripsi*RTOS*.ino

```
. . . . . .
1   void TaskUS6(void *pvParameters)
2   {
3     (void) pvParameters;
4     Serial.begin(115200 bps);
5     vTaskDelay( 16 / portTICK_PERIOD_MS );
6     for (;;)
7     {
8       int SRsensor = sonar[5].ping_cm();
9       int pinLED = LED[5];
10      if (SRsensor <= 15 && SRsensor != 0)
11       {
12          Serial.print("Ultra Sonik 6 : ");
13          Serial.println(SRsensor);
14          digitalWrite(pinLED, HIGH);
15       }
16      if (SRsensor > 15)
17       {
18          Serial.print("Ultra Sonik 6 : ");
19          Serial.println(SRsensor);
20          digitalWrite(pinLED, LOW);
21       }
22    }
23  }

. . . . . .
```

Berdasarkan pada Tabel 5.11, implementasi pada TaskUS6 diawali dengan memberikan *baudrate* sebesar 115200 bps untuk keperluan komunikasi *serial monitor* yang mana pada library NewPing.h memerlukan *baudrate* sebesar 115200

bps untuk berkomunikasi dengan *serial monitor*. Lalu pada proses selanjutnya ada *vTaskDelay* sebesar 16ms untuk memberikan waktu pada *task* dalam melakukan eksekusi, karena dalam jarak 15cm yang digunakan untuk pengkondisian jarak diperlukan waktu 15ms untuk memproses *task* tersebut. Jika *vTaskDelay* lebih kecil dari waktu eksekusi *task* maka akan mengakibatkan lampu LED yang akan nyala mati dikarenakan proses untuk menyalakan LED yang melebihi waktu *vTaskDelay* yang sudah ditentukan akan dipaksa selesai dan berpindah ke *task* yang lain.

Selanjutnya masuk pada perulangan yang didalamnya terdapat deklarasi sebuah variabel SRsensor dengan tipe data *integer*, yang mana variabel tersebut untuk menyimpan data hasil sensing modul sensor HC-SR04 yang ke-6 dan deklarasi variabel selanjutnya adalah variabel pinled dengan tipe data *integer*, yang berfungsi menyimpan nilai alamat pin dari LED yang ke-6. Setelah deklarasi variabel tersebut ada dua buah kondisi dengan menggunakan if, yang mana pada kondisi pertama adalah jika sensor mendeteksi jarak kurang dari sama dengan 15cm dan tidak sama dengan 0cm maka akan masuk pada kondisi pertama yaitu pinled bernilai *HIGH* dan menampilkan nilai hasil *sensing* tersebut ke *serial monitor* pada Arduino IDE.

Untuk kondisi yang kedua menggunakan *if* untuk pengkondisian yang mana jika jarak yang terdeteksi lebih dari 15cm maka pinled bernilai *LOW* dan akan menampilkan nilai hasil *sensing* ke *serial monitor* Arduino IDE. Setelah selesai maka program akan berpindah ke *task* yang berada tepat dibawah *task* ini. Implementasi program TaskUS7 ditampilkan pada Table 5.12.

Tabel 5.12 Kode sumber TaskUS7

skripsi*RTOS*.ino

```
. . . . . .

1  void TaskUS7(void *pvParameters)
2  {
3    (void) pvParameters;
4    Serial.begin(115200 bps);
5    vTaskDelay( 16 / portTICK_PERIOD_MS );
6    for (;;)
7    {
8      int SRsensor = sonar[6].ping_cm();
9      int pinled = led[6];
10     if (SRsensor <= 15 && SRsensor != 0)
11      {
12         Serial.print("Ultra Sonik 7 : ");
13         Serial.println(SRsensor);
14         digitalWrite(pinled, HIGH);
15      }
16     if (SRsensor > 15)
17      {
18         Serial.print("Ultra Sonik 7 : ");
19         Serial.println(SRsensor);
20         digitalWrite(pinled, LOW);
21      }
22    }
23 }
```

```
. . . . . .
```

Berdasarkan pada Tabel 5.12, implementasi pada TaskUS7 diawali dengan memberikan *baudrate* sebesar 115200 bps untuk keperluan komunikasi *serial monitor* yang mana pada library NewPing.h memerlukan *baudrate* sebesar 115200 bps untuk berkomunikasi dengan *serial monitor*. Lalu pada proses selanjutnya ada *vTaskDelay* sebesar 16ms untuk memberikan waktu pada *task* dalam melakukan eksekusi, karena dalam jarak 15cm yang digunakan untuk pengkondisian jarak diperlukan waktu 15ms untuk memproses *task* tersebut. jika *vTaskDelay* lebih kecil dari waktu eksekusi *task* maka akan mengakibatkan lampu LED yang akan nyala mati dikarenakan proses untuk menyalakan LED yang melebihi waktu *vTaskDelay* yang sudah ditentukan akan dipaksa selesai dan berpindah ke *task* yang lain.

Selanjutnya masuk pada perulangan yang didalamnya terdapat deklarasi sebuah variabel SRsensor dengan tipe data *integer*, yang mana variabel tersebut untuk menyimpan data hasil sensing modul sensor HC-SR04 yang ke-7 dan deklarasi variabel selanjutnya adalah variabel pinled dengan tipe data *integer*, yang berfungsi menyimpan nilai alamat pin dari LED yang ke-7. Setelah deklarasi variabel tersebut ada dua buah kondisi dengan menggunakan *if*, yang mana pada kondisi pertama adalah jika sensor mendeteksi jarak kurang dari sama dengan 15cm dan tidak sama dengan 0cm maka akan masuk pada kondisi pertama yaitu pinled bernilai *HIGH* dan menampilkan nilai hasil *sensing* tersebut ke *serial monitor* pada Arduino IDE.

Untuk kondisi yang kedua menggunakan *if* untuk pengkondisian yang mana jika jarak yang terdeteksi lebih dari 15cm maka pinled bernilai *LOW* dan akan menampilkan nilai hasil *sensing* ke *serial monitor* Arduino IDE. Setelah selesai maka program akan berpindah ke *task* yang berada tepat dibawah task ini. Implementasi program TaskUS8 ditampilkan pada Tabel 5.13.

Tabel 5.13 Kode sumber TaskUS8

skripsi*RTOS*.ino

```
. . . . . .

1   void TaskUS8(void *pvParameters)
2   {
3     (void) pvParameters;
4     Serial.begin(115200 bps);
5     vTaskDelay( 16 / portTICK_PERIOD_MS );
6     for (;;)
7     {
8       int SRsensor = sonar[7].ping_cm();
9       int pinled = led[7];
10      if (SRsensor <= 15 && SRsensor != 0)
11       {
12          Serial.print("Ultra Sonik 8 : ");
13          Serial.println(SRsensor);
14          digitalWrite(pinled, HIGH);
15       }
16      if (SRsensor > 15)
```

```
17        {
18           Serial.print("Ultra Sonik 8 : ");
19           Serial.println(SRsensor);
20           digitalWrite(pinled, LOW);
21        }
22     }
23  }

    . . . . . .
```

Berdasarkan pada Tabel 5.13, implementasi pada TaskUS8 diawali dengan memberikan *baudrate* sebesar 115200 bps untuk keperluan komunikasi *serial monitor* yang mana pada library NewPing.h memerlukan *baudrate* sebesar 115200 bps untuk berkomunikasi dengan *serial monitor*. Lalu pada proses selanjutnya ada *vTaskDelay* sebesar 16ms untuk memberikan waktu pada *task* dalam melakukan eksekusi, karena dalam jarak 15cm yang digunakan untuk pengkondisian jarak diperlukan waktu 15ms untuk memproses *task* tersebut. jika *vTaskDelay* lebih kecil dari waktu eksekusi *task* maka akan mengakibatkan lampu LED yang akan nyala mati dikarenakan proses untuk menyalakan LED yang melebihi waktu *vTaskDelay* yang sudah ditentukan akan dipaksa selesai dan berpindah ke *task* yang lain.

Selanjutnya masuk pada perulangan yang didalamnya terdapat deklarasi sebuah variabel SRsensor dengan tipe data *integer*, yang mana variabel tersebut untuk menyimpan data hasil *sensing* modul sensor HC-SR04 yang ke-8 dan deklarasi variabel selanjutnya adalah variabel pinled dengan tipe data *integer*, yang berfungsi menyimpan nilai alamat pin dari LED yang ke-8. Setelah deklarasi variabel tersebut ada dua buah kondisi dengan menggunakan *if*, yang mana pada kondisi pertama adalah jika sensor mendeteksi jarak kurang dari sama dengan 15cm dan tidak sama dengan 0cm maka akan masuk pada kondisi pertama yaitu pinled bernilai *HIGH* dan menampilkan nilai hasil *sensing* tersebut ke *serial monitor* pada Arduino IDE.

Untuk kondisi yang kedua menggunakan *if* untuk pengkondisian yang mana jika jarak yang terdeteksi lebih dari 15cm maka pinled bernilai *LOW* dan akan menampilkan nilai hasil *sensing* ke *serial monitor* Arduino IDE. Setelah selesai maka program akan berpindah ke *task* yang berada tepat dibawah *task* ini.

BAB 4 REKAYASA KEBUTUHAN

Dalam bab ini menjelaskan tentang segala kebutuhan dalam membangun sistem. Dengan merekayasa kebutuhan untuk sistem penelititan bisa lebih terarah dan sistem bisa berjalan dengan baik sesuai fungsi.

4.1 Deskripsi umum

Pada bab ini akan dijelaskan secara detail mengenai segala sesuatu yang dibutuhkan untuk membangun sistem yang harus terpenuhi. Dengan melakukan rekayasa kebutuhan diharapkan sistem bekreja dengan baik dan sesuai fungsi.

4.1.1 Perspektif sistem

Sistem multi-sensor yang berjalan secara simultan menggunkan Real Time Operating Sistem (*RTOS*) untuk melakukan *multitasking* pada sistem multi-sensor tersebut. sistem ini akan menampilkan jarak yang dibaca oleh 8 sensor secara simultan. Akan tetapi simultan yang dimaksud pada sistem ini tidak benar-benar melakukan banyak pekerjaan pada satu waktu yang bersamaan. Karena prosesor sendiri hanya bisa mengerjakan satu intruksi pada satu prosesor. Untuk *RTOS* yang digunakan untuk sistem ini adalah *FreeRTOS*. Karena *FreeRTOS* lebih mudah pengaplikasiannya dari pada *RTOS* yang lain dan juga memori yang digunakan lebih sedikit. Pada bagian output akan menggunakan 2 jenis output, yaitu menampilkan data hasil pengukuran berupa angka dengan nilai ukur centimeter di dalam *serial monitor* dan menampilkan indikator jarak yang terdeteksi dengan *LED* (*Light Emiting Diode*) yang diletakkan di dekat sensor.

4.1.2 Karakteristik pengguna

Karakteristik pengguna dari sistem ini bersifat pasif, yang mana pengguna hanya bisa memonitor data hasil sensing melalui *serial monitor* dan *LED*. Sedangkan pengaturan jarak untuk menyalakan sebuah *LED* hanya bergantung pada data hasil sensing, apabila ingin merubah hanya bisa dilakukan dengan cara *upload* ulang sketch atau program yang telah di modifikasi melalui aplikasi Arduino IDE.

4.1.3 Batasan sistem

Beberapa Batasan yang ada pada sistem ini antara lain :

1. Jumlah sensor yang digunakan berjumlah 8.
2. Sensor yang digunakan adalah sensor HC-SR04.
3. Menggunakan *LED* sebagai indikator jarak yang sudah ditentukan dan kerja *multitasking*.
4. *LED* yang digunakan berwarna merah.
5. Tegangan yang digunakan sebesar 5 V.

6. *LED* akan menyala jika jarak yang terdeteksi kurang dari sama dengan 15cm.
7. *vTaskDelay* tiap *task* sebesar 16 milisecond di awal program.
8. *Multitasking* pada sistem multi-sensor menggunakan implementasi *RTOS*.
9. Sistem menggunakan satuan ukur jarak cm (*centi meter*) untuk menyesuaikan kemampuan *library* dalam memproses hasil *sensing*.

4.1.4 Asumsi dan Ketergantungan

Beberapa asumsi dan ketergantungan yang ada pada sistem ini antara lain :

1. Menggunakan hambatan sebesar 160 Ohm tiap *LED* warna merah.
2. Sistem ini membutuhkan tegangan 5 V untuk menjalankan data sensor.
3. Untuk melihat jarak sensor yang terdeteksi hanya dapat dilakukan di *serial monitor* Arduino IDE
4. Jarak yang terdeteksi sensor kurang dari 2 cm akan dianggap 0 pada sistem dan jarak yang melebihi 200 cm akan dianggap 0 pada sistem.

4.2 Rekayasa kebutuhan

Pada bagian ini menjelaskan tentang Kebutuhan antar muka pengguna, Kebutuhan perangkat keras, Kebutuhan perangkat lunak, Kebutuhan komunikasi, dan Kebutuhan fungsional.

4.2.1 Kebutuhan antarmuka pengguna

User dapat berinteraksi dengan sistem dengan cara memonitor hasil data sensing yang didapatkan dari masing masing sensor untuk ditampilkan melalui *serial monitor* dan indikator nyala LED ketika mendeteksi jarak minimal menyalakan LED yang telah diberikan.

4.2.2 Kebutuhan perangkat keras

Kebutuhan perangkat keras terdiri dari Arduino Mega 2560, modul sensor HC-SR04 8 buah. Akan di jabarkan setiap perangkat keras yang dibutuhkan beserta penjelasannya di bawah ini.

1. Arduino Mega 2560 berfungsi sebagai pusat control sistem yang mengolah data hasil sensor dan menentukan jalan tiap *task* untuk mengontrol jalannya sensor dan menentukan kondisi untuk menyalakan *LED*. Dengan jumlah pin Arduino Mega 2560 yang banyak maka akan menguntungkan dalam membangun sistem ini ketika yang menggunakan antarmuka komunikasi *PBI (Parallel Bus Interface)* untuk mempercepat proses sistem.

2. Modul sensor HC-SR04 merupakan sebuah sensor jarak yang menggunakan ultrasonik untuk mendeteksi jarak dengan cara menembakan sinyal *pulse* dan menerima sinyal *pulse* yang tadi ditembakkan untuk mengetahui jarak dari objek yang memantulkan sinyal *pulse* dari sensor.
3. *LED (Light Emiting Diode)* merupakan sebuah komponen elektronik yang dapat memancarkan cahaya ketika diberikan tegangan. Pada system ini *LED* berfungsi sebagai indikator jarak, ketika sensor mendeteksi pada jarak minimal yang sudah ditentukan di system maka *LED* akan menyala.

4.2.3 Kebutuhan perangkat lunak

Kebutuhan perangkat lunak terdiri dari Windows 10, Arduino IDE dan beberapa *library* pendukung. Akan diuraikan setiap perangkat lunak yang dibutuhkan beserta penjelasannya di bawah ini.

1. Arduino IDE merupakan sebuah program untuk menulis dan *upload* program / sketch pada mikrokontroller agar sistem dapat berjalan sesuai yang diharapkan.
2. Arduino *FreeRTOS library* merupakan *library* yang digunakan untuk menangani pengimplementasian *RTOS* pada Arduino. Diantara fungsi dari Arduino *FreeRTOS* yang digunakan adalah:
   a. x*Task*Create() fungsi ini digunakan untuk membuat sebuah tugas baru dan menambahkan ke daftar tugas yang siap dijalankan.
   b. *vTaskDelay*() fungsi ini digunakan untuk memberikan sebuah *delay* pada *task* dengan nilai satuan tick.
3. *NewPing library* merupakan *library* yang digunakan untuk menangani perhitungan *pulse* yang di peroleh modul sensor HC-SR04. Diantara fungsi dari NewPing yang digunakan adalah:
   a. NewPing() digunakan untuk setup pin dari sensor ultrasonik yang digunakan dan jarak maksimal untuk sensor. Ketika sensor mendeteksi jarak lebih dari nilai maksimal range maka sensor akan memberikan nilai 0.
   b. VariabelName.Ping_cm() digunakan untuk mendapatkan nilai jarak yang terdeteksi dari sensor dalam nilai ukur centimeter.

4.2.4 Kebutuhan fungsional

Kebutuhan fungsional merupakan kebutuhan yang harus terpenuhi agar sistem dapat dijalankan sesuai dengan tujuan, penjelasan terkait kebutuhan fungsional pada sistem ini adalah sebagai berikut :

4.2.4.1 Fungsi pembacaan data sensor HC-SR04

Pada fungsi ini sensor HC-SR04 harus diimplementasi pada sistem untuk keperluan pengambilan data jarak, yang mana sensor mengirimkan sinyal *pulse* yang nantinya akan di terima kebali untuk mengatahui pada jarak berapa sinyal *pulse* tadi di kembalikan atau dipantulkan. Setelah *input* nilai dari sensor HC-SR04 diterima oleh mikrokontroller, data tersebut kemudian disimpan dan diolah kembali oleh mikrokontroller.

4.2.4.2 Fungsi mekanisme *multitasking* pada *RTOS*

Pada fungsi ini sistem mengolah penjadwalan untuk tiap *task* yang diperuntukkan untuk tiap sensor ultrasonik yang digunakan. Satu sensor memiliki satu *task* untuk mengatur proses penjadwalan kapan sensor tersebut diaktifkan dan juga memproses data yang diberikan oleh sensor sehingga bisa ditampilkan ke *serial monitor* Arduino IDE berupa angka nilai jarak yang terdeteksi oleh sensor dengan satuan centimeter. Di dalam *task* tersebut terdapat sebuah proses pengkondisian yang mengatur nyala tiap *LED* yang terdapat di belakang sensor.

4.2.4.3 Fungsi pengolahan data sensing

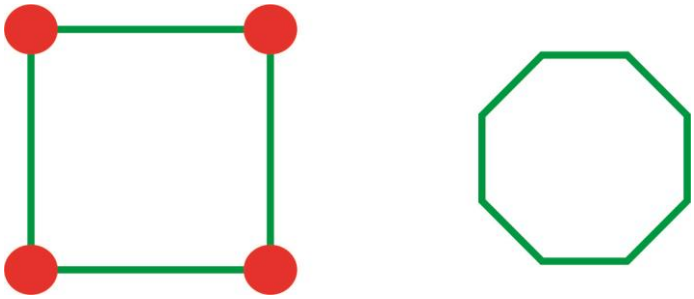
Pada fungsi ini mengharuskan mikrokontroller dapat mengolah data hasil sensing dari tiap sensor untuk ditampilkan ke *serial monitor*. Dengan menggunakan fungsi namaVariable.ping_cm(). Semua perhitungan hasil sensing dari sensor HC-SR04 akan di proses oleh fungsi tersebut. yang mana fungsi tersebut adalah fungsi dari *library* NewPing.h.

4.2.4.4 Fungsi pengontrol nyala *LED*

Pada fungsi ini mengharuskan mikrokontroler untuk mengatur nyala *LED* ketika mendeteksi jarak minimal yang sudah diatur pada kode program atau *sketch* sebesar 15cm. Ketika sensor mendeteksi jarak kurang dari sama dengan 15cm dan tidak sama dengan 0cm maka *LED* akan menyala. Sebaliknya jika sensor mendeteksi jarak lebih dari 15cm maka *LED* akan mati.

4.2.5 Kebutuhan performansi sistem

Sistem dapat bekerja secara optimal apabila kebutuhan dapat terpenuhi beserta beberapa faktor pendukung seperti kesesuaian menyebarnya gelombang sinyal *pulse* yang ditembakan oleh modul sensor HC-R04. Faktor lainnya penataan dari sensor sendiri, sensor akan di letakkan bersebalahan dengan sensor jarak yang lain akan tetapi dibentuk seperti persegi 8 sehingga sudut pandang dari sensor itu sendiri bisa mencakup sudut 360^{o}. Alasan kenapa menggunakan sensor HC-SR04 dengan jumlah 8 buah akan di ilustrasikan pada Gambar 4.1.



Gambar 4.1 Ilustrasi jumlah sensor

Jika jumlah sensor sebanyak 4 dan memerlukan sudut pandang 360^{o} maka akan terlihat seperti persegi pada penataan sensornya. Dan pada bagian sudut dari persegi tersebut tidak akan dapat mendeteksi halangan ketika terdapat halangan menyerong di bagian pojok kanan atas, pojok kiri atas, pojok kanan bawah, dan pojok kiri bawah yang di gambarkan pada Gambar 4.1. Pada Gambar 4.1 di gambar persegi di bagian sudut-sudut terdapat lingkaran kecil berwarna merah, pada bagian tersebut adalah bagian yang tidak bisa melakukan sensing ketika terdapat halangan yang menyerong. Maka dari itu diperlukan sensor pada bagian sudut-sudut persegi tersebut. Penataan dari sensor tersebut jika ditambah sensor-sensor di bagian sudut pesegi tersebut maka akan terlihat seperti persegi delapan yang digambarkan pada Gambar 4.1 di persegi 8. Jika penataan sensor berbentuk seperti persegi 8 maka sistem tersebut bisa mendeteksi halangan yang berbentuk menyerong. Maka dari itu dalam sistem ini memerlukan 8 sensor untuk membentuk penataan sensor berbentuk persegi 8. Jarak maksimal yang ditentukan dalam sistem sebesar 200cm agar sistem tidak terlalu lama dalam melakukan eksekusi. Sehingga waktu ekseskusi pada sistem hanya selama waktu sistem mendeteksi jarak sejauh 200cm. Dan penerapan sistem ini kedalam sebuah robot berkaki pemadam kebakaran yang diletakkan didalam sebuah maze yang membutuhkan jarak tidak lebih dari 200cm.

4.2.6 Spesifikasi perangkat keras

Dalam perancangan sistem diperlukan perngkat keras yang digunakan, diantara perangkat keras yang dibutuhkan dimulai dari tahapan perancangan hinga pengujian sistem dijelaskan pada bagian ini.

4.2.6.1 *Personal Computer*

Personal Computer (PC) yang digunakan mulai dari tahapan perancangan hingga pengujian sistem ini adalah Laptop HP Pavilion Notebook – 14-v203tx dengan prosesor Intel(R) Core(TM) i5-5200U 2.20GHz (4 CPU) dilengkapi dengan NVIDIA GeForce 840M sebagai video grafiknya. Pada Gambar 4.2 merupakan tampilan dari Personal Computer yang digunakan, sedangkan untuk spesifikasinya pada Tabel 4.1.

Gambar 4.2 Laptop HP Pavilion Notebook – 14-v203tx

Sumber: (HP Customer Support, 2017)

Tabel 4.1 Spesifikasi Laptop HP Pavilion Notebook – 14-v203tx

Tipe Model	Notebook
CPU	Intel(R) Core(TM) i5-5200U 2.20GHz
Mode GPU	NVIDIA GeForce 840M
RAM	8 GB DDR3L SDRAM 1600 MHz
Slot Memori	2 DIMM
HDD	WDC WD7500BPVX-60JC3T0 - 698.64 GB
Format HDD	NTFS
Sistem Operasi	Windows 10 64-bit

4.2.6.2 Mikrokontroler

Mikrokontroler yang digunaan pada sistem ini adalah mikrokontroler Arduino Mega yang meggunakan IC ATmega 2560. Terkait spesifikasinya ada di Tabel 4.2.

Tabel 4.2 Spesifikasi Arduino Mega 2560

Mikrokontroler	ATmega2560
Tegangan Operasi	5V
Tegangan Masukan (*recommended*)	7V – 12V
Tegangan Masukan (*limit*)	6-20V
Pin I/O Digital	54 (14 pin mendukung untuk PWM)
Pin Masukan Analog	16
Arus DC Per Pin I/O	20 mA
Arus DC untuk Pin 3.3V	50mA
Flash Memory	256 KB (8 KB digunakan bootloader)
SRAM	8 KB
EEPROM	4 KB
Kecepatan Clock	16 MHz

LED_BUILTIN	13
Panjang	101.52 mm
Lebar	53.3 mm
Berat	37 g

4.2.6.3 Modul sensor HC-SR04

HC-SR04 adalah modul sensor jarak ultrasonik dengan jarak pengukuran 2 cm – 400cm. akurasi pengukuran jarak modul ini mencapai 3 mm. Spesifikasi HC-SR04 dijelaskan pada Tabel 4.3.

Tabel 4.3 Spesifikasi sensor HC-SR04

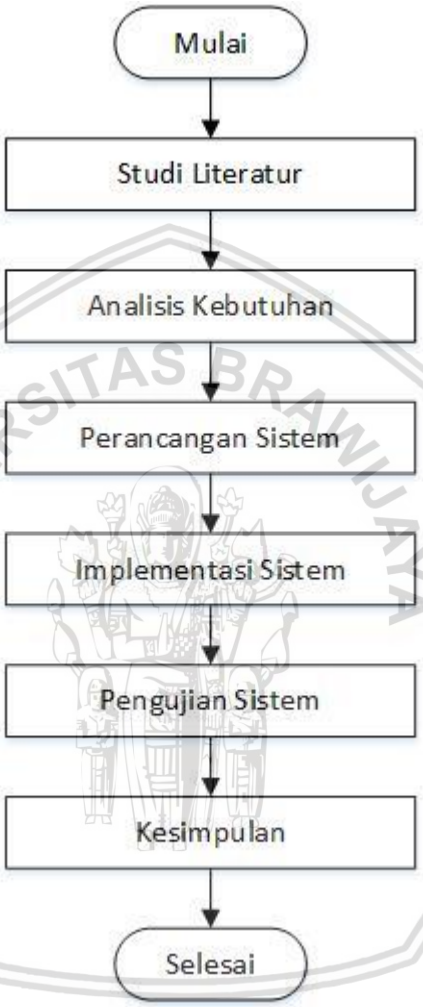
Tegangan Kerja	5V DC
Arus Kerja	15 mA
Frekuensi Kerja	40Hz
Jarak Maksimal	4m
Jarak Minimal	2cm
Pengukuran sudut	15^{o}
Trigger Input Signal	10uS TTL *Pulse*
Echo Output Signal	Menyesuaikan TTL sinyal *input*
Dimesion	45 x 20 x 15 mm

4.2.6.4 Light Emiting Diode (*LED*)

Light Emitting Diode (*LED*) adalah sebuah komponen elektronik yang dapat memancarkan cahaya ketika diberi tegangan. *LED* merupakan keluraga diode yang terbuat dari bahan semikonduktro. Warna-warna cahaya yang dipancarkan tergantung pada jenis bahan semikonduktor yang digunakan. Sehingga tiap warna *LED* memiliki tegangan kerja berbeda-beda. Pada sistem ini menggunakan *LED* berwarna merah yang memiliki tegangan kerja sebesar 1,8 V – 2,1 V. Pada sistem ini menggunakan *LED* yang hanya satu warna bukan *LED RGB* yang memiliki 4 kaki pin sehingga kaki *LED* yang digunakan berjumlah dua yaitu katoda dan anoda. Pada anoda akan dihubungkan ke sumber tegangan pada sistem karena bersifat positif sedangkan pada katoda akan dihubungkan pada ground sistem karena bersifat negative.

BAB 3 METODOLOGI

Pada bab ini menjelaskan tentang metode yang digunakan dalam membangun sistem pada penelitian ini yang berjudul Rancang Bangun Sistem Multi-Sensor untuk Pengukuran jarak Secara Simultasn. Alur proses metode penelitian yang digunakan dapat dilihat pada Gambar 3.1.



Gambar 3.1 Proses metode penelitian

3.1 Studi literatur

Studi literatur dilakukan dengan tujuan untuk mengumpulkan bahan berupa literatur yang digunakan untuk membantu membangun sistem ini. Literatur yang dipakai adalah jurnal, makalah, forum yang membahas Arduino dan FreeRTOS, e-book, buku dll.

3.2 Analisis kebutuhan

Analisa kebutuhan bertujuan untuk mengenalisis kebutuhan yang digunakan dalam membentuk sistem, menguji dan mengenalisis hasil. Analisis ini dimulai dari

kebutuhan perangkat keras seperti komponen apa saja yang diperlukan dalam membangun sistem ini. Kebutuhan perangkat lunak yang akan ditanam dalam sistem. Dengan adanya analisis ini memudahkan dalam mebangun sistem.

3.2.1 Kebutuhan perangkat keras

Kebutuhan perangkat keras yang digunakan pada penelitian ini akan dijelaskan pada table 3.1.

Tabel 3.1 Perangkat keras

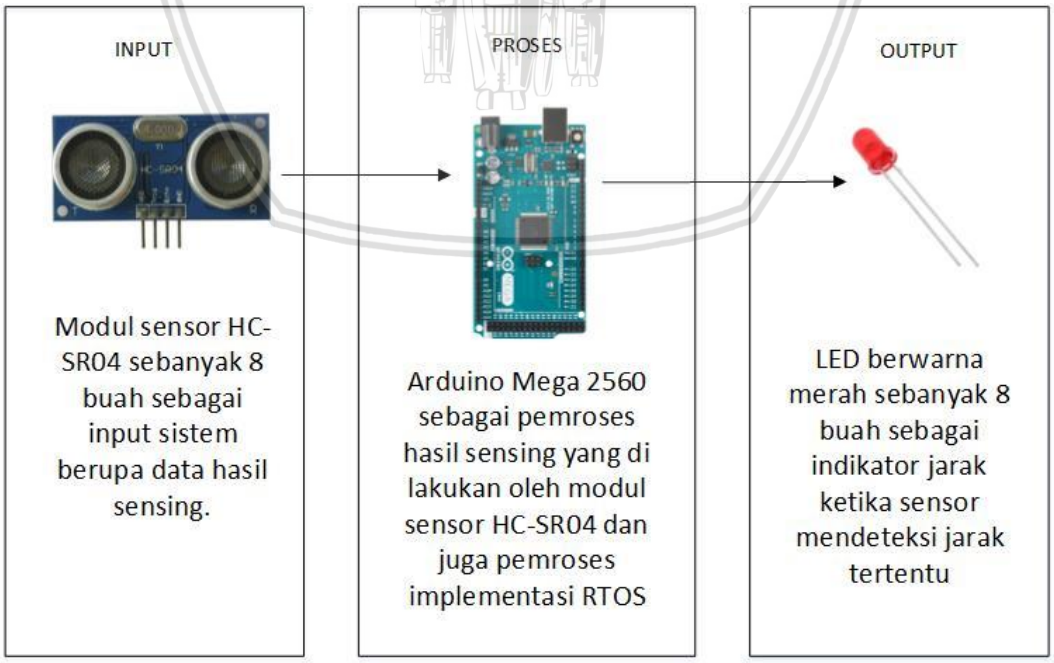
Nama Perangkat	Kegunaan
Arduino Mega 2560	Sebagai mikrokontroler untuk mengontrol sistem.
HC-SR04	Sebagai sensor ultrasonik untuk mengukur jarak.
LED (Light Emiting Diode)	Sebagai indikator jarak yang terdeteksi oleh sensor.

3.2.2 Kebutuhan perangkat lunak

Kebutuhan perangkat lunak yang dibutuhkan IDE arduino dan beberapa *library* Arduino yaitu *Arduino_FreeRTOS.h* untuk menggunakan *FreeRTOS* pada Arduino dan *NewPing.h* untuk perhitungan pengukuran yang dilakukan sensor HC-SR04.

3.3 Gambaran umum perancangan

Pada tahap perancangan menjelaskan tentang perancangan sistem multi-sensor. Menjelaskan tentang *input output* pada sistemnya. Pada gambaran perancangan ini tidak menggambarkan sistem yang telah jadi, hanya menggambarkan sistem sederhananya yang di tunjukan pada Gambar 3.2.



Gambar 3.2 Perencanaan perancangan

Pada Gambar 3.2 menggambarkan tentang perencanaan perancangan sistem yang mana pada gambar tersebut terdapat HC-SR04, Arduino Mega 2560, *LED*. Pada gambaran tersebut Arduino Mega 2560 sebagai pusat pemrosesan data yang diterima dari sensor-sensor HC-SR04. Dan jarak yang terdeteksi oleh sensor HC-SR04 akan ditampilkan melalui *serial monitor* dengan *baudrate* 115200 bps. Sedangkan *LED* sebagai indikator jarak yang terdeteksi oleh sensor. Ketika jarak yang terdeteksi kurang dari sama dengan 15cm dan tidak sama dengan 0cm maka *LED* akan menyala, sebaliknya jika sensor mendeteksi jarak lebih dari 15cm maka *LED* akan mati.

3.4 Tahapan implementasi sistem

Implementasi sistem ini dilakukan berdasarkan pada perancangan yang dibuat sebelumnya. Pada sistem terdapat 8 sensor ultrasonik HC-SR04 yang nantinya akan berjalan secara simultan dengan menerapkan *multitasking* pada sistem tersebut dengan memanfaatkan *FreeRTOS* pada Arduino yang digunakan untuk keperluan *multitasking*. Dan nantinya jarak yang terdeteksi oleh sensor dapat di lihat pada *serial monitor*.

3.5 Langkah pengujian sistem

Pada tahapan pengujian sistem ii digunakan untuk memberikan parameter pada pengujian sistem dan menganalisis sistem yang dibangun berdasarkan dari permasalahan yang dirumuskan dan memeberikan dasil yang dapat membantu penelitian selanjutnya dalam membangun sistem.

3.6 Penarikan kesimpulan

Pengambilan kesimpulan diambil untuk menjawab pertanyaan dari rumusan masalah yang telah dirumuskan. Muali dari cara membangun arsitektur sistem yang tepat untuk sebuah multi-sensor, cara implementasi RTOS kedalam sistem, dan mengukur seberapa efektif sistem yang telah dibangun.

BAB 2 LANDASAN KEPUSTAKAAN

Bab ini berisi penjelasan tentang tinjauan pustaka yang digunakan untuk membantu sebagai acuan penelitian yang dilakukan dan pebanding dengan penelitian yang dilakukan. Dasar teori berisi tentang penjelasan teori yng digunakan untuk membangun penelitian ini.

2.1 Tinjauan pustaka

Tinjauan pustaka berupa penelitian-penelitian yang telah ada sebelumnya dan memiliki keterkaitan dengan pembuatan penelitian ini. Penelitian dengan judul *"*Sistem Pendeteksi Kebocoran Gas LPG Menggunakan Metode *Fuzzy* yang Diimplementasi dengan *Real Time Operating System* (RTOS)*"* oleh Lavanna Indanus Ramadhan pada tahun 2017. Pada penelitian tersebut membangun sebuah sistem multi-sensor yang digunakan untuk mendeteksi kebocoran gas LPG. Pada sistem tersebut menggunakan 2 sensor yaitu sensor suhu LM35 dan sensor gas MQ-6. Dalam pemrosesan data hasil sensing menggunakan algoritma *fuzzy* dalam mengambil suatu keputusan. Pada implementasi RTOS dalam sistem tersebut, bagian sensor diberikan nilai prioritas yang sama yaitu 3 dengan tujuan sensor bekerja secara bersamaan. Dan task dengan prioritas dibawahnya yaitu task inferensi bernilai 2 dan task defuzzyfikasi bernilai 1 (Ramadhan, 2017).

Penelitian kedua adalah milik Agung Leona Suparlin pada tahun 2018 dengan judul "Implementasi *System Real Time* untuk Monitoring Pencahayaan Suhu dan Kelembaban pada Tanaman Stroberi". Penelitian ini membahas tentang penerapan RTOS untuk menjalankan sebuah sistem multi-sensor sebanyak 3 jenis yang digunakan yaitu sensor suhu LM35, suhu cahaya LDR dan sensor kelembapan SEN0114. Pada sistem tersebut prioritas dari 3 sensor diberikan nilai yang sama dengan tujuan agar sistem berjalan secara simultan (Suparlin, 2018).

Tabel 2.1 Tinjauan pustaka

No.	Nama Penulis, Tahun, dan Judul	Persamaan	Perbedaan	
			Penelitian terdahulu	Rencana Penelitian
1.	Lavanna Indanus Ramadhan, 2017, Sistem Pendeteksi Kebocoran Gas LPG Menggunakan Metode *Fuzzy* yang Diimplementasikan dengan Real Time Operating System (RTOS)	Membangun sistem multi-sensori dengan menerapkan RTOS	Jumlah sensor yang digunakan sebanyak 2	Jumlah sensor yang digunakan sebanyak 8
2	Agung Leona Suparlin, 2018, Implementasi *System Real Time* untuk Monitoring Pencahayaan Suhu dan	Membangun sistem multi-sensori dengan menerapkan RTOS	Sensor yang digunakan sebanyak 3	Jumlah sensor yang digunakan sebanyak 8

No.	Nama Penulis, Tahun, dan Judul	Persamaan	Perbedaan	
			Penelitian terdahulu	Rencana Penelitian
	Kelembaban pda Tanaman Stroberi			

2.2 Dasar teori

2.2.1 HC-SR04

HC-SR04 adalah modul sensor jarak ultrasonik dengan jarak pengukuran 2cm – 400cm. Akurasi pengukuran jarak modul ini mencapai 3mm. Pada modul sensor ini terdapat ultrasonik sebagai transmitter, ultrasonik sebagai receiver dan control sirkuit. Cara kerja modul ini antara lain, menggunakan IO untuk memicu setidaknya 10µs high level sinyal, modul akan secara otomatis mengirimkan delapan 40kHz dan mendeteksi apakah ada sinyal *pulse* yang kembali, jika sinyal kembali melalui high level waktu durasi high *output* IO adalah waktu dari mengirim ultrasonik sampai kembali,

$$test\ jarak = \frac{high\ level\ time\ \times velocity\ of\ sound(340\ m/s)}{2} \quad \textbf{(2.1)}$$

(Elecfreaks Team, 2015). 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 2.1. Untuk elektrik parameter akan dijelaskan pada Tabel 2.2.

Tabel 2.2 Parameter HC-SR04

Working Voltage	DC 5 V
Working Current	15mA
Working Frequency	40Hz
Max Range	4m
Min Range	2cm
Measuring Angle	*15 degree*
Trigger Input Signal	*10uS TTL pulse*
Echo Output Signal	*Input TTL lever signal and range in proportion*
Dimension	45*20*15mm

Sumber: (Elecfreaks Team, 2015)

Gambar 2.1 Modul HC-SR04

Sumber : Arduinolearning

2.2.2 *Real-Time Operating System (RTOS)*

Real Time Operating System (RTOS) adalah sebuah sistem operasi yang dikembangkan untuk pengaplikasian sistem embedded yang menerapkan *real-time* pada sistemnya. Sistem ini membolehkan untuk merubah prioritas proses dirubah secara instan dan data yang diproses cukup cepat sehingga sisa waktu dapat digunakan untuk merspon proses lainnya yang terjadi di waktu yang sama. RTOS memiliki kemampuan untuk segera merespon dengan cara yang telah ditentukan dan memprediksi kejadian-kejadian eksternal yang akan terjadi (Chandane, 2016).

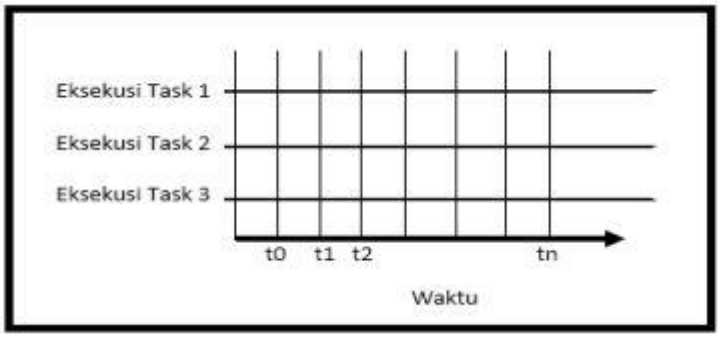
Real Time Operating System (RTOS) adalah OS dengan fitur-fitur khusus yang membuatnya cocok untuk membangun aplikasi komputasi realtime yang juga disebut sebagai Real-Time Systems (RTS). RTS adalah sistem (komputasi) di mana kebenaran komputasi tidak hanya bergantung pada kebenaran hasil logis dari perhitungan, tetapi juga pada waktu hasil pengiriman. RTS diharapkan merespon secara tepat waktu dan dapat diprediksi terhadap stimulus eksternal yang sulit diprediksi (Puham, 2015).

2.2.3 Konsep dasar *Real-Time Operating System (RTOS)*

RTOS memiliki beberapa konsep dasar diantaranya *multitasking*, *scheduling* dan *context switching*. Berikut adalah penjelasan lebih lanjut konsep-konsep dasar *RTOS*.

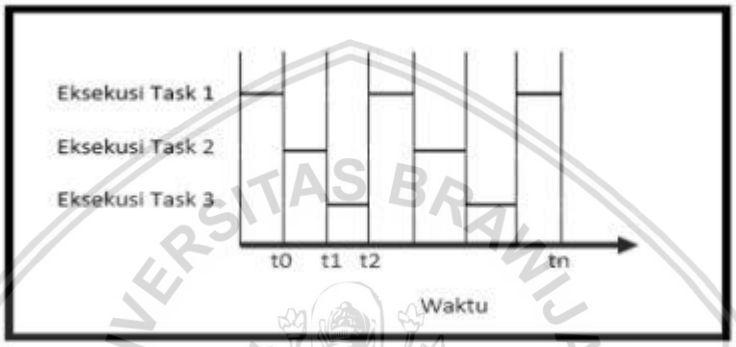
2.2.3.1 *Multitasking*

Sistem operasi memiliki komponen utama untuk mengatur adanya *multitasking*. *Multitasking* adalah kemampuan sistem operasi dalam menjalankan banyak *task* atau thread dalam rentan waktu tertentu. Konsep *multitasking* dan konkuren dapat dilihat pada Gambar 2.2 dan Gambar 2.3.



Gambar 2.2 Konsep *multitasking*

Sumber : (Jatmiko, et al., 2015)



Gambar 2.3 Konsep konkuren

Sumber : (Jatmiko, et al., 2015)

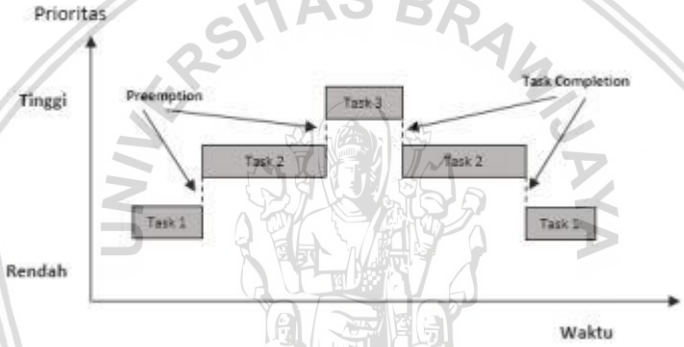
Pada Gambar 2.2 menunjukkan bagaimana konsep *multitasking* bekerja. Tiga *task* akan berjalan dalam waktu yang sama dan waktu eksekusi yang sama. Akan tetapi pada memori konvensional, konsep *multitasking* belum dapat dijalankan. Keterbatasan sumber daya prosesor dan memori membatasi *multitasking*. Namun *task-task* yang ada secara bergantian akan dijalankan. Hal tersebut dinamakan dengan konkurensi. Konsep konkurensi digambarkan pada Gambar 2.3. Sistem operasi sudah mengetahui *task* mana saja yang akan dijalankan dalam rentan waktu tertentu. Selanjutnya sistem operasi membagi rentan waktu yang ada untuk mengeksekusi beberapa *task*. Sistem operasi mengatur kapan waktu eksekusi setiap *task* dan menyimpan state/posisi dari setiap *task* pada memori, sehingga ketika prosesor menjalankan suatu *task*, prosesor akan mengambil state terakhir dari *task* tersebut kemudian melanjutkan eksekusi *task* tersebut. Proses perpindahan antara eksekusi *task* dilakukan dengan cepat dan konkuren sehingga seolah - olah terlihat procesor melakukan beberapa *task* dalam rentan waktu bersamaan. Embedded sistem dapat menggunakan *RTOS* untuk menjalankan beberapa *task* secara konkuren. Pada akhirnya sistem tertanam seolah-olah dapat bekerja secara *multitasking*. (Jatmiko, et al., 2015)

2.2.3.2 *Scheduling*

Scheduling merupakan konsep pembagian waktu eksekusi *task-task* yang ada. Untuk melakukan *scheduling* dibutuhkan adanya sebuah scheduler. Scheduler berfungsi untuk menentukan waktu eksekusi suatu *task*, lama eksekusi *task*,

waktu suatu *task* ditunda (*suspend*), dan waktu *task* dijalankan kembali (*resume*). Pada penetuan waktu eksekusi *task* diperlukan suatu algoritma *scheduling* (Jatmiko, et al., 2015).

RTOS menggunakan algoritma *Preemptive Priority-Based Scheduling* sebagai algoritma default untuk *scheduling*. Algoritma ini mengatur pada setiap waktu *task* dengan prioritas paling tinggi akan dieksekusi. *Task* tersebut akan didahulukan dari *task* maupun yang juga sudah siap untuk dieksekusi. Gambar 2.4 menunjukkan bagaimana algoritma *Preemptive Priority-Based Scheduling* bekerja. *Task* dengan prioritas tinggi akan dieksekusi, sedangkan *task* lain akan ditunda dulu hingga *task* dengan prioritas lebih tinggi selesai dieksekusi. Pada *RTOS*, penentuan prioritas pada setiap *task* dilakukan pada saat pertama kali dibuat. Perubahan prioritas juga perlu memperhatikan pembagian prioritas yang sudah ada sehingga tidak menimbulkan *priority inversion*, *deadlock*, maupun kegagalan sistem (Jatmiko, et al., 2015).



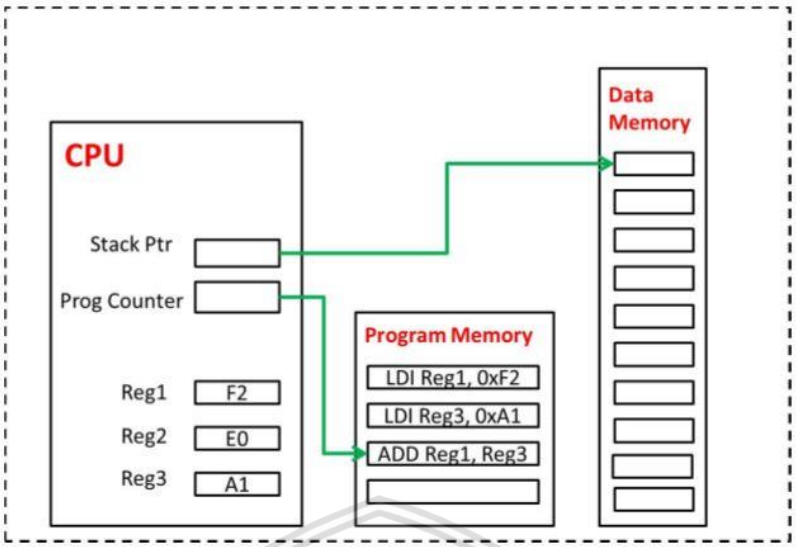
Gambar 2.4 Algoritma *preemptive priority-based scheduling*

Sumber : (Jatmiko, et al., 2015)

2.2.3.3 *Context switching*

Eksekusi suatu task membutuhkan sumber daya di komputer ataupun embedded system. Sebuah task membutuhkan akses ke register, Random Access Memory (RAM), Read Only Memory (ROM). Sumber daya tersebut digunakan secara bersama-sama oleh beberapa task. Pembagian sumber daya bersama dikenal dengan istilah context (Jatmiko, et al., 2015).

Task tidak tahu kapan dirinya ditunda (suspend) atau dieksekusi kembali (resume). Hal tersebut dapat mengakibatkan kesalahan pada instruksi suatu task yang melibatkan register dan memory. Contoh kasus yang mungkin terjadi, terdapat dua buah task, task 1 dan task 2. Ketika task 1 sedang melakukan instruksi perhitungan dimana data diambil dari nilai yang disimpan dalam beberapa register. Hal tersebut diilustrasikan Gambar 2.5.



Gambar 2.5 Context switching

Sumber : (Jatmiko, et al., 2015)

Task 1 mengambil nilai 2 buah register dan melakukan instruksi ADD. Task 1 secara tiba-tiba ditunda (suspend) (sebelum instruksi ADD dijalankan) diganti dengan task 2, kemudian register yang bersangkutan juga digunakan oleh task 2. Setelah task 2 selesai dijalankan kemudian task 1 dieksekusi kembali (resume). Task 1 tidak mengetahui bahwa nilai di register telah diubah. Akibatnya instruksi ADD yang dilakukan task 1 akan menghasilkan nilai yang tidak tepat. (Jatmiko, et al., 2015)

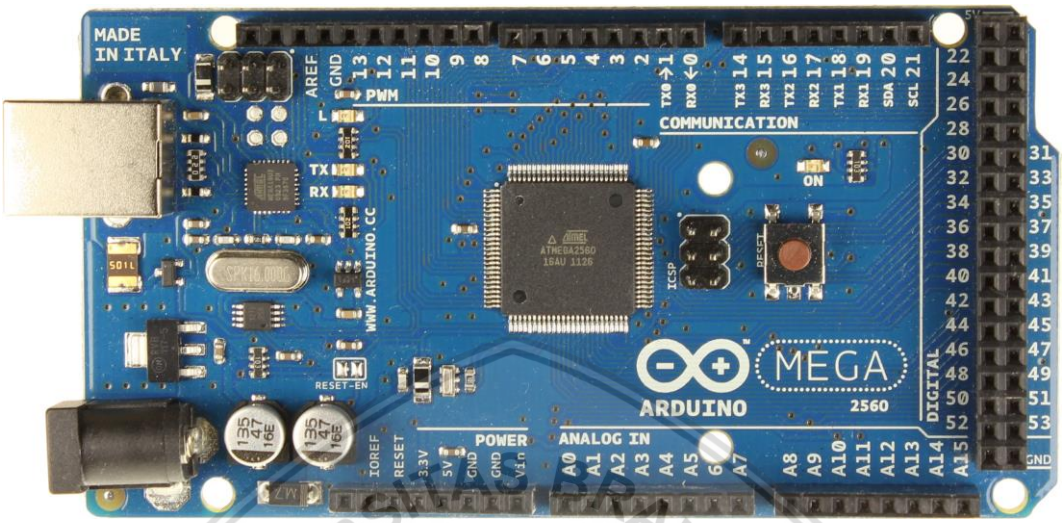
2.2.4 Free Real Time Operating Sistem (*FreeRTOS*)

FreeRTOS adalah merupakan sebuah mini kernel portable yang merupakan *core* (jantung) dari sebuah sistem operasii dan mengatur bagaimana *user* (pengguna) mengakses komputer dan menjalankan program pada komputer. Ukurannya hanya berkisar antara 4kB – 9kB. Karena ukurannya yang kecil, FreeRTOS mampu memberikan prosesor kemampuan prosesing yang lebih mudah dan waktu prosesing yang yang lebih cepat. FreeRTOS bersifat *open source* dan royalty free, artinya *source code* kernel dapat dikembangkan dan didistribusikan oleh semua rang secara bebas, FreeRTOS didesain untuk *small embedded systems* (sistem komputer berskala kecil). FreeRTOS sendiri ditulis dengan menggunakan Bahasa pemrograman C (Barry, 2016).

2.2.5 Arduino Mega 2560

Arduino Mega 2560 adalah sebuah *board* mikrokontroler berdasarkan ATmega2560. Arduino Mega 2560 memiliki 54 pin *input* / *output* digital (14 pin digunakan sebagai *output* PWM), 16 *input* analog, 4 UART (*hardware serial ports*), oscillator kristal 16 MHz, koneksi USB, power jack, header ISCP, dan tombol reset. Semua yang dibutuhkan untuk mendukung sebuah mikrokontroler ada pada spesifikasi Arduino Mega 2560. Cukup dengan menghubungkannya dengan

komputer melalui kabel USB atau menyalakan kekomputer dengan adaptor AC to -DC atau sebuah baterai. Untuk bentuk fisik dari Arduino Mega 2560 dijelaskan pada Gambar 2.6



Gambar 2.6 Arduino Mega 2560

Sumber : (Arduino.cc, 2017)

2.2.6 *LED (Light Emitting Diode)*

LED (Light Emitting Diodes) adalah suatu semikonduktor yang memancarkan cahaya monokromatik yang tidak koheren ketika diberi tegangan maju atau searah. Atau secara bahasa bisa diartikan sebagai dioda yang memancarkan cahaya bila dialirkan arus listrik. Semikonduktor adalah material yang dapat bertindak sebagai konduktor (penghantar arus listrik) dan isolator (penahan arus listrik). Sedangkan dioda adalah bahan semikonduktor yang terdiri dari N-*type* material dan P-*type* material yang saling terhubung dan di kedua ujungnya terdapat elektroda (katoda/ N-type & anoda/P-type).

2.2.6.1 Cara kerja *LED*

LED mengubah sebagian besar energi listrik menjadi cahaya. Cahaya adalah suatu bentuk energi yang dilepaskan oleh sebuah atom. Cahaya dihasilkan dari banyak partikel-partikel kecil yang mempunyai energi dan momentum yang disebut *photons* yang merupakan unit utama dari suatu cahaya. *Photons* merupakan hasil dari pergerakan electron (Harris, T.et al., 2014). *Photons* pada suatu diode dapat kita lihat jika diode tersusun dari material tertentu. Pada diode normal, yang biasanya terbuat dari silikon atau *germanium,* memancarkan cahaya berupa gelombanginframerah sehinga tidak dapat dilihat mata manusia. *LED* memancarkan cahaya semata-mata oleh pergerakan elektron pada material. Dan *LED* terdiri dari bahan semikonduktor yang memancarkan gelombang cahaya yang dapat dilihat oleh mata manusia dan memancarkannya dalam jumlah besar.

Bahan semikonduktor dibungkus dalam plastik sehingga mengkonsentrasikan cahaya yang dihasilkan pada arah tertentu. Bahan plastik penutup dapat juga

diberi warna, namun hal ini hanya untuk estetika dan memperkuat tampilan warna yang dihasilkan. Pewarnaan plastik tidak berpengaruh pada gelombang warna yang dihasilkan bergantung pada bahan semikonduktor yang dipakai (A. Gregorius A. Gegana, 2007).