DESAIN SISTEM NAVIGASI ROBOT DENGAN ISYARAT MATA MENGGUNAKAN METODE *CANNY* DAN *HOUGH TRANSFORM*

SKRIPSI

REDA ANGGRA DISTIRA
NIM : 071910201079

PROGRAM STUDI STRATA-1 TEKNIK ELEKTRO
JURUSAN TEKNIK ELEKTRO
FAKULTAS TEKNIK
UNIVERSITAS JEMBER
2012

DESAIN SISTEM NAVIGASI ROBOT DENGAN ISYARAT MATA MENGGUNAKAN METODE *CANNY* DAN *HOUGH TRANSFORM*

SKRIPSI

diajukan guna melengkapi skripsi dan memenuhi syarat-syarat
untuk menyelesaikan Program Studi Teknik Elektro (S1)
dan guna mencapai gelar Sarjana Teknik

REDA ANGGRA DISTIRA
NIM : 071910201079

PROGRAM STUDI STRATA-1 TEKNIK ELEKTRO
JURUSAN TEKNIK ELEKTRO
FAKULTAS TEKNIK
UNIVERSITAS JEMBER
2012

بسم الله الرحمن الرحيم

PERSEMBAHAN

Syukur Alhamdulillah penulis panjatkan kehadirat Allah SWT atas segala rahmat dan karunia-Nya, sehingga penulis dapat menyelesaikan skripsi yang berjudul **"Desain Sistem Navigasi Robot Dengan Isyarat Mata Menggunakan Metode *Canny* Dan *Hough Transform*"** Skripsi ini disusun untuk memenuhi salah satu syarat dalam menyelesaikan pendidikan strata satu (S1) pada Jurusan Teknik Elektro, Fakultas Teknik, Universitas Jember.

Penulisan skripsi ini tidak lepas dari bantuan berbagai pihak, oleh karena itu penulis ingin menyampaikan ucapan terima kasih yang tiada terhingga kepada:

1. Allah SWT.
2. Rasulullah Muhammad SAW.
3. Bapakku Hadi Budiman dan Ibuku Laikah yang selalu memberikan doa dan dukungan dari segi apapun, serta kasih sayang yang tidak pernah putus. Aku menyayangi kalian.
4. Kakakku Dhike Respaty dan Adekku Yora Erlangga terima kasih doa dan bantuan, Terus semangat dalam menempuh pendidikan semoga diberi kemudahan.
5. Aulia Hardani Hartono yang dengan tulus memberikan doa, semangat dan semuanya beserta keluarga.
6. Semua Dosen Jurusan Teknik Elektro Fakultas Teknik Universitas Jember yang telah membimbing dan memberikan ilmu. Terutama Bapak. Ir, Widyono Hadi, M.T selaku DPU, Bapak. Sumardi, S.T., M.T selaku DPA yang telah meluangkan waktu dan pikiran serta perhatiannya guna memberikan bimbingan dan pengarahan demi terselesaikannya skripsi ini, Ibu Ike Fibriani., S.T.,M.T. Dosen Penguji I Bapak. Dr. Azmi Saleh, S.T., M.T., dan Dosen Penguji II Bapak. Dr. Triwahju Hardianto, S.T., MT.

7. Seluruh Guru-guruku dari TK, SD, SLTP, SMA dan Guru mengaji yang telah membimbing dengan sabar dan memberikan ilmu.
8. Teman-teman satu Laboratorium yang telah membantu dan menemani dalam susah senang mengerjakan skripsi ini, Sukses buat kalian "perjuangan ini tidak berhenti sampai disini".
9. Keluarga besar TETRO 07: Alfredo Satria, S.T., Bambang Nurdiansyah S.T, Ditaria Panjaitan S.T, Rengga Elga S.T, Berty Restanty S.T., Raga Raditya S.T, Andik Hikmawan, S.T., Danu Fami, S.T., Arif Setyawan S.T, Haqqi Prananda, M. Maskhur H, Yustinus S.T, Riska Ayu, dan Semua keluarga **TETRO 07** yang belum disebutkan, "**sebuah persahabatan yang tidak pernah berakhir… TEKNIK JOSSS!!!".**
10. Teman Elektro Diploma 3 angkatan 07 **"kebersamaan selama ini sangat berharga"**.
11. Keluarga besar UKM ROBOTIKA UNEJ **"karena kalian ilmu ini menjadi bermanfaat TETAP BERJUANG DEMI ALMAMETER TERCINTA"**
12. Keluarga besar SR/3 no 10 terutama Anggi Hernandia, Andi Fajar K, Tintus Ardi, Toni , Rio Mahadi, Suryadani Imam (atas printernya) tak lupa Mami dan Mbak Heni yang selalu memberi pengarahan,
13. Keluarga KKT desa Tanjungrejos, Vina Yudhyani ,Yosep Ari W (Mas bro), Anggi Dwi P (a.k.a Titun), Eni (si Geje), Jessica (si sekretaris).
14. Semua pihak yang telah membantu dalam kelancaran penulisan skripsi ini yang tidak dapat disebutkan satu per satu.

MOTTO

وَمَا جَعَلَهُ ٱللَّهُ إِلَّا بُشۡرَىٰ لَكُمۡ وَلِتَطۡمَئِنَّ قُلُوبُكُم بِهِۦۚ وَمَا ٱلنَّصۡرُ إِلَّا مِنۡ عِندِ ٱللَّهِ ٱلۡعَزِيزِ ٱلۡحَكِيمِ ﴿١٢٦﴾

"*Dan Allah tidak menjadikan pemberian bala bantuan itu melainkan sebagai khabar gembira bagi (kemenangan)mu, dan agar tenteram hatimu karenanya. Dan kemenanganmu itu hanyalah dari Allah Yang Maha Perkasa lagi Maha Bijaksana.*"

(Terjemahan Q.S Ali Imron : 126)

"Saya datang, saya bimbingan, saya ujian, saya revisi dan saya menang!"

(Skripsi)

"Jika kita menginginkan sesuatu yang belum pernah kita miliki,kita harus bersedia melakukan sesuatu yang belum pernah kita lakukan"

(My Handle Life)

"May be i'm not the best, but i'm not easy to lose and give all my strungle for the best"

(Reda Anggra Distira)

PERNYATAAN

Saya yang bertandatangan di bawah ini :

Nama : Reda Anggra Distira

NIM : 071910201079

Menyatakan dengan sesungguhnya bahwa karya tulis yang berjudul:

DESAIN SISTEM NAVIGASI ROBOT DENGAN ISYARAT MATA MENGGUNAKAN METODE *CANNY* DAN *HOUGH TRANSFORM*

adalah benar-benar hasil karya sendiri, kecuali jika dalam pengutipan substansi disebutkan sumbernya, dan belum pernah diajukan pada institusi manapun, serta bukan karya jiplakan. Saya bertanggungjawab atas keabsahan dan kebenaran isinya sesuai dengan sikap ilmiah yang harus dijunjung tinggi.

Demikian pernyataan ini saya buat dengan sebenarnya, tanpa adanya tekanan dan paksaan dari pihak manapun serta bersedia mendapat sanksi akademik jika di kemudian hari pernyataan ini tidak benar.

Jember, 9 Februari 2012

Yang menyatakan,

Reda Anggra Distira

NIM 071910201079

SKRIPSI

DESAIN SISTEM NAVIGASI ROBOT DENGAN ISYARAT MATA MENGGUNAKAN METODE *CANNY* DAN *HOUGH TRANSFORM*

Oleh

Reda Anggra Distira
NIM 071910201079

Pembimbing :

Dosen Pembimbing Utama : Ir. Widyono Hadi, MT
Dosen Pembimbing Anggota : Sumardi, ST.,MT

PENGESAHAN

Skripsi berjudul **"Desain Sistem Navigasi Robot Dengan Isyarat Mata Menggunakan Metode *Canny* Dan *Hough Transform*"** telah diuji dan disahkan oleh Jurusan Teknik Elektro, Fakultas Teknik, Universitas Jember pada :

Hari : Rabu

Tanggal : 1 Februari 2012

Tempat : Laboratorium Jaringan Komputer Jurusan Teknik Elektro, Fakultas Teknik, Universitas Jember

Menyetujui,

Dosen Pembimbing Utama (Ketua)	Dosen Pembimbing Anggota (Sekretaris)
Ir. Widyono Hadi, MT NIP 19610414 198902 1 001	Sumardi, ST.,MT NIP 19670113 199802 1 001
Dosen Penguji I	Dosen Penguji II
Dr. Azmi Saleh, ST.,MT NIP 19710614 199702 1 001	Dr. Triwahju Hardianto, ST.,MT NIP 19700826 199702 1 001

Mengesahkan
Dekan Fakultas Teknik,

Ir. Widyono Hadi, MT
NIP 19610414 198902 1 001

DESAIN SISTEM NAVIGASI ROBOT DENGAN ISYARAT MATA MENGGUNAKAN METODE *CANNY* DAN *HOUGH TRANSFORM*

Reda Anggra Distira

Mahasiswa Jurusan Teknik Elektro.

Fakultas Teknik, Universitas Jember

ABSTRAK

Sistem navigasi robot merupakan sebuah sistem yang digunakan untuk mengendalikan pergerakan robot. Hasil dari keputusan sebuah navigasi robot diantaranya maju, mundur, belok kanan, belok kiri maupun berhenti. *Image Processing* atau sering disebut dengan pengolahan citra digital merupakan metode yang digunakan untuk mengolah atau memproses dari gambar asli sehingga menghasilkan gambar lain yang sesuai dengan kebutuhan. Tujuan penelitian ini adalah menggabungkan teknologi *image processing* dengan robotika. Merancang sebuah navigasi robot berdasarkan pergerakan bola mata. Sebagai input system digunakan *webcam* untuk input video (*real time*) yang mengambil object mata. Deteksi pergerakan bola mata dengan menggunakan penggabungan metode deteksi tepi *Canny* dan *Hough Transform.* Dari perancangan sistem yang telah dijelaskan diatas didapatkan hasil bahwa penggabungan Canny dan Hough Transform dapat berjalan baik dengan tingkat keberhasilan 100 %. Dan tingkat keberhasilan pada uji lapangan tanpa halangan sebesar 100% dan dengan halangan 80% dengan intensitas cahaya antara 10 – 160 lux.

Kata kunci : *Navigasi Robot, Webcam, Canny, Hough Transform, lux.*

ROBOT NAVIGATION SYSTEM DESIGN WITH EYE SIGNAL USING CANNY AND HOUGH TRANSFORM METHOD

Reda Anggra Distira

College Student of Department of Electrical Engineering

Engineering Faculty, Jember University

ABSTRACT

Robot navigation system is a system used to control the movement of the robot. The results from the decision of a robot navigation including forward, backward, turn right, turn left and stop. Image Processing or often referred to as digital image processing is a method used to process or processing of the original image to produce another image that as your needs. The purpose of this study is to combine image processing technology with robotics. Designing a robot navigation based on eye movement. As the input system used a webcam for video input (real time) that takes the eye object. Detection of eye movement by using a method combining Canny edge detection and Hough Transform. Based on system design described above showed that incorporation of Canny and Hough Transform can be run either with 100% success rate. And the success rate on field tests without hindrance by 100% and 80% with obstruction to the light intensity of 10-160 lux.

Keywords: Robot Navigation, Webcam, Canny, Hough Transform, lux.

RINGKASAN

Desain Sistem Navigasi Robot Dengan Isyarat Mata Menggunakan Metode *Canny* Dan *Hough Transform*; Reda Anggra Distira, 071910201079; 2012: 73 halaman; Jurusan Teknik Elektro Fakultas Teknik Universitas Jember.

Sistem navigasi robot merupakan sebuah sistem yang digunakan untuk mengendalikan pergerakan robot. Hasil dari keputusan sebuah navigasi robot diantaranya maju, mundur, belok kanan, belok kiri maupun berhenti. Untuk jenis robot otomatis, pergerakannya berdasarkan kondisi masukan sensor-sensor yang digunakan.

Image Processing atau sering disebut dengan pengolahan citra digital merupakan metode yang digunakan untuk mengolah atau memproses dari gambar asli sehingga menghasilkan gambar lain yang sesuai dengan kebutuhan. Operator Canny didesain untuk menjadi sebuah pendeteksi tepi yang optimal(berdasarkan kriteria tertentu). Deteksi tepi cany ini digunakan agar didapatkan tepian bentuk bola mata sehingga dapat dideteksi posisinya. Yang perlu diperhatikan pada deteksi tepi ini dalah pengaturan batas bawah threshold dan batas atas threshold. Transformasi Hough adalah standar algoritma pada *computer vision* yang digunakan untuk menentukan parameter objek geometri sederhana seperti garis dan lingkaranpada citra. Transformasi Hough pada citra iris dapat digunakan untuk mengisolasi lokasi citra iris dari bagian lainnya seperti pupil dan selera. Proses yang dilakukan adalah dengan mengambil parameter lingkaran yang melalui setiap titik pada citra tepi hasil pendeteksian tepi.

Dari hasil penelitian dapat diketahui bahwa kedua metode ini sangat bekerja dengan baik yakni dengan tingkat keberhasilan hampir 100 %. Dalam pengujian pada pengaruh intensitas cahaya sistem ini dapat bekerja diantara range 10 – 150 lumen. Dan untuk jarak maksimal transmisi data yang dapat dilakukan sistem yakni kurang dari 100 meter.

PRAKATA

Puji syukur kehadirat Allah SWT, atas hidayahnya dan rahmatnya sehingga kami dapat menyelesaikan skripsi ini sebagaimana mestinya. Shalawat serta salam semoga Allah SWT limpahkan kepada Nabi Muhammad SAW sebagai sumber inspirasi dan membuat kami lebih kuat dan menatap setiap hal yang penuh optimis dan berfikir positif, dalam menunjang kemampuan kami dalam menjalani persaingan globalisasi kerja nantinya.

Dalam pelaksanakanya kami tidak lepas dari kesulitan dan permasalahan dalam penyusunan skripsi ini, baik dari proses pembuatan proposal sampai penyusunan akhir skripsi , mengenai ilmu yang bermanfaat, moral dan sikap serta tanggung jawab dalam menyelesaikan skripsi ini. Dengan demikian kami mengucapkan terima kasih pada:

1. Ir. Widyono Hadi, MT selaku Dekan Fakultas Teknik Universitas Jember.
2. Bapak Sumardji, S.T., M.T., Selaku ketua Jurusan Teknik Elektro, Fakultas Teknik Universitas Jember.
3. Ir. Widyono Hadi, M.T., selaku Dosen Pembimbing Utama, dan Sumardi, S.T., M.T., selaku Dosen Pembimbing Anggota dan juga Ike Fibriani, S.T.,M.T selaku dosen pembimbing pembantu yang memberikan arahan dan saran-saran dalam penyelesaian skripsi ini.
4. Dr. Azmi Saleh, S.T., M.T., selaku penguji pertama dan Dr. Triwahju Hardianto, S.T., MT., selaku penguji kedua yang telah memberikan saran dan waktu.
5. Sumardi, S.T., M.T.,. selaku Dosen Pembimbing Akademik.
6. Seluruh Dosen Teknik Elektro Universitas Jember yang tidak dapat saya sebutkan satu-persatu, terima kasih atas bimbingan yang telah diberikan.
7. Bapak dan Ibu tercinta atas dukungan yang tak henti-hentinya .
8. Semua teman Elektro 2007 baik S1 maupun D3 yang telah menjadi saudara, rekan kuliah, teman main terima kasih atas segala doa, canda,

bantuan dan semuanya yang kalian berikan “hutang harta dibalas harta, hutang budi dibawa mati”

9. Teman-teman Teknik Elektro angkatan 2004 s/d 2011, manusia tidak pernah luput dari salah, mohon maaf jika selama kita bersama ada tindakan yang kurang berkenan. Terus semangat perjuangan di depan semakin berat.
10. Kepada seluruh pihak yang telah membantu menyelesaikan pendidikan di Universitas Jember ini yang tidak dapat saya sebutkan satu- persatu .

Dalam penyusunan skripsi ini tentunya masih banyak kekurangan baik dalam isi maupun analisisnya, oleh karena itu kami mengaharapkan pada para pembaca dapat merefisi dan manjadikan lebih baik, kami berharap semoga skripsi ini dapat berguna bagi pembaca, terima kasih.

Jember, Februari 2012

Penulis

DAFTAR ISI

DAFTAR GAMBAR

Halaman

DAFTAR TABEL

Halaman

BAB I
PENDAHULUAN

1.1 Latar Belakang

Perkembangan dunia robotika saat ini telah menjadi suatu hal yang menarik untuk dibicarakan. Penelitian dalam hal sistem navigasi robot juga sudah banyak melakukan. Sistem navigasi robot merupakan sebuah sistem yang digunakan untuk mengendalikan pergerakan robot. Hasil dari keputusan sebuah navigasi robot diantaranya maju, mundur, belok kanan, belok kiri maupun berhenti. Untuk jenis robot otomatis, pergerakannya berdasarkan kondisi masukan sensor-sensor yang digunakan. Selain robot otomatis, ada juga robot yang navigasinya berdasarkan perintah-perintah yang dikirimkan secara manual melalui PC, *remote control* atau *joystick* yang dinamakan *teleoperated.*

Image Processing atau sering disebut dengan pengolahan citra digital merupakan metode yang digunakan untuk mengolah atau memproses dari gambar asli sehingga menghasilkan gambar lain yang sesuai dengan kebutuhan. Penggunaan i*mage processing* ini sudah cukup berkembang sejak orang mengerti bahwa komputer tidak hanya mampu mengangani data teks, melainkan juga data citra. Pada awalnya pengolahan citra *(image processing)* dilakukan untuk memperbaiki kualitas citra, namun seiring berkembangnya dunia komputasi yang memungkinkan manusia mengambil informasi dari suatu citra, maka *image processing* tidak dapat dilepaskan dengan bidang *computer vision.* Dalam perkembangan lebih lanjut, *image processing* dan *computer vision* digunakan sebagai pengganti mata manusia, dengan perangkat *input image capture* seperti kamera dan *scanner* dijadikan sebagai mata dan mesin komputer (dengan program komputasinya) digunakan sebagai otak yang mengolah informasi.

Dari kedua bidang tersebut akhirnya muncul teknologi baru yakni *robot vision*. *Robot Vision* adalah robot yang menggunakan teknologi *image processing* dan *computer vision* dalam tugasnya, baik untuk pengenalan objek maupun navigasi

robot. Berdasarkan penjelasan diatas, penulis merancang dan membuat robot yang navigasi arahnya berdasarkan posisi bola mata manusia serta konfigurasi yang dapat dihasilkannya. Melalui sistem ini, diharapkan dapat menjadi inspirasi dalam dunia teknologi *Robotic Vision.* Salah satu penerapan aplikasinya yakni dalam pembuatan kursi dorong cerdas pada orang lumpuh yang navigasinya di tentukan oleh posisi mata penderita.

1.2 Rumusan Masalah

Berdasarkan latar belakang di atas, dapat di identifikasi masalah yang ada adalah sebagai berikut:

1. Bagaimana merancang system navigasi robot dengan menggunakan isyarat mata?
2. Bagaimana menentukan posisi bola mata berbasis *image processing* menggunkan metode *Canny* dan *Hough Transform*?
3. Bagaimana merancang komunikasi antara komputer dan robot aktual secara wireless?

1.3 Tujuan

Tujuan dari penelitian ini adalah :

1. Membuat sistem navigasi robot sebagai pengendali dengan teknologi wireless.
2. Merancang algoritma pendeteksian posisi bola mata menggunakan *image processing* dengan metode *Canny* dan *Hough Transform*.

1.4 Manfaat

Adapun manfaat yang penyusun ingin dicapai dari penelitian ini, adalah :

1. Dengan sistem yang dirancang, dapat diterapkan untuk navigator robot.
2. Dengan algoritma yang digunakan pada *image processing* dapat diimpementasikan sebagai proses dalam sitem navigasi robot.

1.5 Batasan Penelitian

Pada penelitian ini, penyusun membuat batasan masalah dalam membuat Desain Sistem Navigasi Robot Dengan Isyarat Mata Menggunakan Metode *Canny* dan *Hough Transform*, yaitu antara lain :

1. Sistem program pengendalian piranti menggunakan mikrokontroller AVR ATMega 8535 dan pemrogramnya menggunakan *MicrosoftVisualC++ Express Edition* untuk pengolahan citra dan *CodeVisionAVR* untuk mikrokontroller.
2. Metode yang digunakan untuk pendeteksian pergerakan bola mata menggunakan metode *Canny* dan *Hough Transform*.
3. Bola mata berwarna hitam.
4. Posisi kamera sudah ditentukan sebelumnya yang bertujuan agar fokus terhadap objek yang akan ditangkap.
5. Transfer data antara komputer dan robot aktual secara wireless dengan menggunakan modul wireless TLP 433.92A – RLP 43392-LC dengan jarak maksimal 100 m.
6. Robot aktualisasi menggunakan robot beroda dengan menggunakan servo continous sebagai penggeraknya.

1.6 Sistematika Penulisan

Dalam laporan penelitian ini untuk memudahkan pembahasan permasalahan penelitian tentang Desain Sistem Navigasi Robot Dengan Isyarat Mata Menggunakan Metode *Canny* dan *Hough Transform*, maka penyusun membuat sistematika penulisan sebagai berikut :

BAB I. PENDAHULUAN

Berisi uraian tentang latar belakang penelitian, rumusan masalah, metedologi pembahasan, sistematika pembahasan.

BAB II. TEORI PENUNJANG

Membahas teori-teori yang mendukung dalam perencanaan dan pembuatan sistem.

BAB III. METODELOGI PENULISAN

Menjelaskan tahap-tahap dan metode yang dilakukan dalam perencanaan pembuatan sistem.

BAB IV. HASIL DAN PEMBAHASAN

Bab ini membahas tentang hasil penelitian yang dilakukan serta analisa dan evaluasi terhadap data-data yang diperoleh selama perancangan sistem.

BAB V. PENUTUP

Bab ini berisikan kesimpulan dan evaluasi yang telah dilakukan dan saran yang membangun untuk pengembangan skripsi ini lebih lanjut.

DAFTAR PUSTAKA

LAMPIRAN

BAB II
TINJAUAN PUSTAKA

2.1 Kendali Robot

Robot berdasarkan kendalinya terbagi dalam dua kelompok. Kelompok yang pertama merupakan *automatic robot* yakni robot yang bergerak secara otomatis berdasarkan perintah-perintah yang telah diprogramkan sebelumnya atau berdasarkan masukan dari sensor-sensornya. Pada kelompok yang kedua yaitu *teleoperated* adalah robot yang sistem navigasinya berdasarkan perintah-perintah yang dikirimkan secara manual baik melalui *remote control,* PC atau *joystick.* (Dwi Mulyono,2011)

2.1.1 Sistem Gerak *Mobile Robot* Beroda

Robot beroda (wheel robot) dibagi menurut sistem penggeraknya, yaitu sistem gerak *differential drive*, *tricycle drive*, *synchronous drive*, dan *holomonic drive*

2. Differential Drive

Sistem gerak *differential drive* terdiri dari dua buah roda yang terpasang pada kiri dan kanan robot, sistem ini memungkinkan robot berputar ditempat dengan cara memutar kedua motor dengan arah berlawanan. Contoh sistem gerak ini pada kehidupan sehari-hari adalah pada garden belakang mobil dan mainan mobil *radio control*.

3. Tricycle Drive

Tricycle drive merupakan sistem gerak dengan tiga buah roda. Dua buah roda dengan satu poros dihubungkan pada sebuah penggerak, sedangkan sebuah roda diberlakukan sebagai kemudi yang dapat berputar (setir kemudi), ketika berbelok akan didapatkan radius sepanjang titik pertemuan antara roda depan dengan roda belakang, contoh sistem gerak ini adalah pada sistem transportasi becak dan bajaj

4. Synchronous Drive

Synchronous drive adalah sistem yang menggunakansemua roda yang terdapat pada robot untuk dapat bergerak. Pada satt robot berjalan pada permukaan yang tidak rata, maka roda yang terpengaruh pada kertidakrataan permukaan akan didukung oleh roda yang tidak terpengaruh, sehingga robot dapat bergerak dengan arah yang tetap. Contoh sisten gerak ini pada kehidupan sehari-hari adalah pada roda trolly pasar swalayan.

4. *Holonomic Drive*

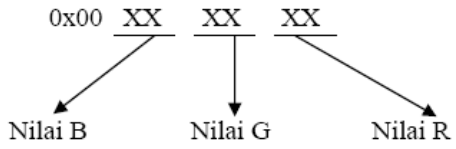
Holonomic drive adalah sistem gerak yang memungkinkan robot bergerak ke segala arah (dengan penggunaan roda omni-directional), konfigurasi ini memungkinkan gerakan rotasi dan translasi pada mobile robot.

2.2 Pengertian Citra Digital

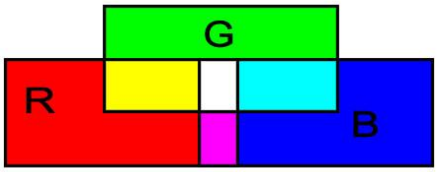
Citra dapat berbentuk foto hitam putih atau berwarna, sinyal-sinyal video seperti gambar pada monitor televisi, atau bersifat digital yang dapat langsung disimpan pada suatu pita magnetik. Menurut presisi yang digunakan untuk menyatakan titik-titik koordinat pada ranah waktu atau bidang dan untuk menyatakan nilai keabuan atau warna suatu citra, maka secara teoritis citra dapat dikelompokkan menjadi empat kelas citra, yaitu cara kontinu-kontinu, kontinu-diskret, diskret-kontinu, dan diskret-diskret; dengan label pertama menyatakan presisi dari titik-titik koordinat pada bidang citra sedangkan label kedua menyatakan presisi nilai keabuan atau warna. Kontinu dinyatakan dengan presisi takhingga, sedangkan diskret dinyatakan dengan presisi angka berhingga. Pengubahan citra yang bersifat kontinu menjadi citra yang bersifat diskret memerlukan pembuatan kisi-kisi arah vertikal dan horisontal, sehingga diperoleh citra dalam bentuk larik dua dimensi. Proses tersebut dikenal sebagai proses digitisasi atau pencuplikan (sampling). Setiap elemen larik tersebut dikenal sebagai elemen gambar atau piksel.

2.2.1 Pengolahan Citra Warna

Dasar dari pengolahan citra adalah pengolahan warna RGB pada posisi tertentu. Dalam pengolahan citra warna dipresentasikan dengan nilai hexadesimal dari 0x00000000 sampai 0x00ffffff. Warna hitam adalah 0x00000000 dan warna putih adalah 0x00ffffff. Definisi nilai warna di atas seperti gambar 2.1, variabel 0x00 menyatakan angka dibelakangnya adalah hexadecimal.



Gambar 2.1 Nilai warna RGB dalam hexadesimal Terlihat bahwa setiap warna mempunyai range nilai 00 (angka desimalnya adalah 0) dan ff (angka desimalnya adalah 255), atau mempunyai nilai derajat keabuan 256 = 2^8. Dengan demikian range warna yang digunakan adalah $(2^8)(2^8)(2^8) = 2^{24}$ (atau yang dikenal dengan istilah True Colour pada Windows). Nilai warna yang digunakan di atas merupakan gambungan warna cahaya merah, hijau dan biru seperti yang terlihat pada gambar 2.2. Sehingga untuk menentukan nilai dari suatu warna yang bukan warna dasar digunakan gabungan skala kecerahan dari setiap warnanya.



Gambar 2.2 Komposisi warna RGB

Dari definisi diatas untuk menyajikan warna tertentu dapat dengan mudah dilakukan, yaitu dengan mencampurkan ketiga warna dasar RGB.

Pengolahan citra dimulai dengan proses *thresholding*, yaitu proses pemisahan citra berdasarkan batas nilai tertentu, dalam proses *thresholding* citra warna diubah menjadi citra biner. Tujuan proses *thresholding* adalah untuk membedakan objek dengan latar belakangnya. Setelah proses *thresholding* proses selanjutnya adalah proses penghitungan nilai-nilai parameter antara lain R, G, B, RGB rata-rata (*color value*), dan indeks R (Ired), indeks G (Igreen), indeks B (Iblue), dari tiap-tiap *pixel*

a. Pengukuran Parameter RGB (*Red, Green* dan *Blue*)

   Paramater RGB diperoleh dari tiap-tiap *pixel* warna pada citra yang merupakan nilai intensitas untuk masing-masing warna merah, hijau, dan biru. Nilai rata-rata dari R,G dan B dijumlahkan untuk mendapatkan *color value* atau RGB rata-rata.

b. Pengukuran parameter Indeks R, Indeks G dan Indeks B

   Perhitungan indeks warna merah/indeksR (Ired), indeks warna hijau/indeks G (Igreen), dan indeks warna biru/indeks B (Iblue) menggunakan rumus pada persamaan (1), (2) dan (3). Intensitas warna merah dibagi dengan penjumlahan dari nilai intensitas warna merah, hijau, dan biru sehingga menghasilkan nilai parameter indeks R. Intensitas warna hijau dibagi dengan penjumlahan dari nilai intensitas warna merah, hijau, dan biru sehingga menghasilkan nilai parameter indeks G. Intensitas warna biru dibagi dengan penjumlahan dari nilai intensitas warna merah, hijau, dan biru sehingga menghasilkan nilai parameter indeks B. Perhitungan parameter Indeks R, G, dan B diperoleh dari tiap-tiap *pixel* pada citra kopi. Model warna RGB dapat juga dinyatakan dalam bentuk indeks warna RGB dengan rumus sebagai berikut (Ahmad, 2005; Arimurthy, dkk., 1992):

   Indeks warna merah:

$$(I_{red}) = \frac{R}{R+G+B} \quad \text{.....(1)}$$

   Indeks warna hijau:

$$(I_{green}) = \frac{G}{(R+G+B)}$$ .....(2)

Indeks warna biru:

$$(I_{blue}) = \frac{B}{(R+G+B)}$$ .....(3)

Dengan R, G dan B masing-masing merupakan besaran yang menyatakan nilai intensitas warna merah, hijau, dan biru.

2.2.2 Edge Detction

Perbedaan yang signifikan sebuah image brightness sangat menarik untuk beberapa alasan. Alasan yang pertama adalah sebuah tepian dari objek biasanya memiliki perbedaan intensitas cahaya (image yang terang bisa terdapat di latar belakang yang gelap atau sebaliknya image yang gelap bisa berada di latar belakang yang terang).dan alasan yang kedua adalah perbedaan tersebut dapat pula muncul akibat dari pola yang terbentuk dari perbedaan intensitas cahaya (zebra memiliki garis tubuh, macan tutul memiliki bintik-bintik pada tubuhnya atau garis-garis yang terbentuk karena bayangan). Obyek yang berada dalam bidang citra dan tidak bersinggungan dengan batas bidang citra, berarti obyek tersebut dikelilingi daerah yang bukan obyek yaitu latar belakang. Pertemuan antara bagian obyek dan bagian latar belakang disebut tepi obyek (dapat juga disebut tepi latar belakang, tetapi kita tidak tertarik pada latar belakang). Tepi merupakan salah satu fitur citra yang penting karena dapat mewakili informasi yang penting dari obyek dalam pemandangan. Poin-poin dimana sebuah image memiliki perbedaan intensitas cahaya yang tajam itulah yang sering disebut edges atau edge points.

2.2.2.1 *Canny Edge Detector*

Operator Canny didesain untuk menjadi sebuah pendeteksi tepi yang optimal(berdasarkan kriteria tertentu). Canny menggunakan sebuah gambar grayscale, dan menghasilkan sebuah gambar yang menampilkan posisi dari intensitas

dan akhir yang telah ditemukan. Operator Canny bekerja dalam sebuah proses bertingkat. Pertama gambar akan diperhalus dengan menggunakan konvolusi Gaussian. Kemudian sebuah operator turunan pertama dari 2-D digunakan untuk menghaluskan gambar pada daerah yang telah ditandai dengan sebagian turunan pertama yang tinggi. Tepi ini diberikan kenaikan menjadi lipatan dalam ukuran gradien gambar. Kemudian algoritma tersebut mencari puncak dari lipatan ini dan memberi nilai nol pada semua piksel yang bukan merupakan puncak lipatan yang menghasilkan garis tipis pada gambar keluaran, sebuah proses yang dikenal dengan non-maximal suppression.

Proses pencarian ini menampilkan hysteresis yang dikendalikan oleh dua thresholds: T1 dan T2 dengan T1 > T2. Pencarian hanya dapat dimulai pada titik dimana nilai lipatan lebih tinggi dari T1. Pencarian kemudian berlanjut dalam dua arah keluar dari titik tersebut hingga tinggi dari lipatan tersebut bernilai kurang dari T2. Hysteresis ini membantu untuk meyakinkan bahwa tepi yang memiliki noise tidak rusak menjadi banyak bagian tepi

(a) (b) (c)

Gambar 2.3 (a) citra koin asli (b) deteksi tepi dengan operator *sobel* (c) deteksi tepi dengan operator *canny*

2.2.3 Operasi Morfologi

Morphologi adalah teknik pengolahan citra digital dengan menggunakan bentuk *(shape)* sebagai pedoman dalam pengolahan. Nilai dari setiap pixel dalam citra digital hasil diperoleh melalui proses perbandingan antara pixel yang bersesuaian pada citra digital masukan dengan pixel tetangganya. Operasi morphologi bergantung pada

urutan kemunculan dari *pixel*, tidak memperhatikan nilai *numeric* dari *pixel* sehingga teknik morphologi sesuai apabila digunakan untuk melakukan pengolahan *binary image* dan *grayscale image*.

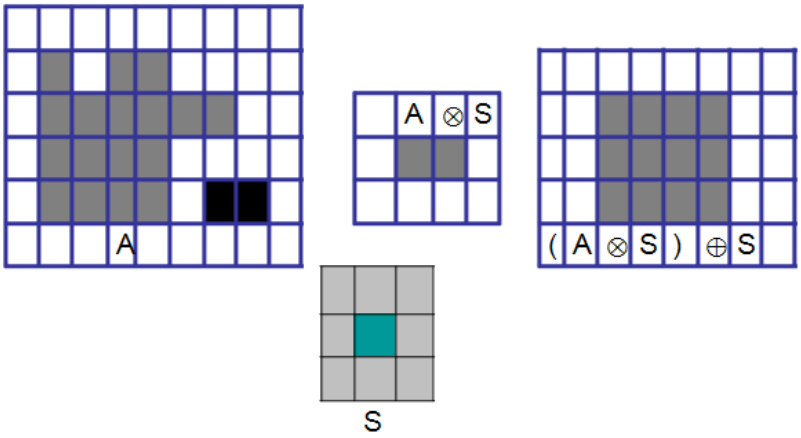
Dengan mengatur atau memilih ukuran dan bentuk dari matrik kernel (*structuring element)* yang digunakan maka kitadapat mengatur sensitivitas operasi morphologi terhadap bentuk tertentu (*spesifik)* pada citra digital masukan. Operasimorphologi standar yang dilakukan adalah proses erosi dan dilatasi. Dilatasi adalah proses penambahan *pixel* pada batasdari suatu objek pada citra digital masukan, sedangkan erosi adalah proses pemindahan/pengurangan *pixel* pada batasdari suatu objek. Jumlah *pixel* yang ditambahkan atau yang dihilangkan dari batas objek pada citra digital masukantergantung pada ukuran dan bentuk dari *structuring element* yang digunakan. Beberapa operasi morfologi yang dapat kita lakukan adalah Dilasi, Erosi, Opening, Closing, Thinning, shrinking, pruning, thickening, skeletonizing (Aniati Murni,2002). Pada penelitian ini hanya dilakukan operasi morfologi menggunakan operasi opening.

2.2.3.1 Operasi Opening

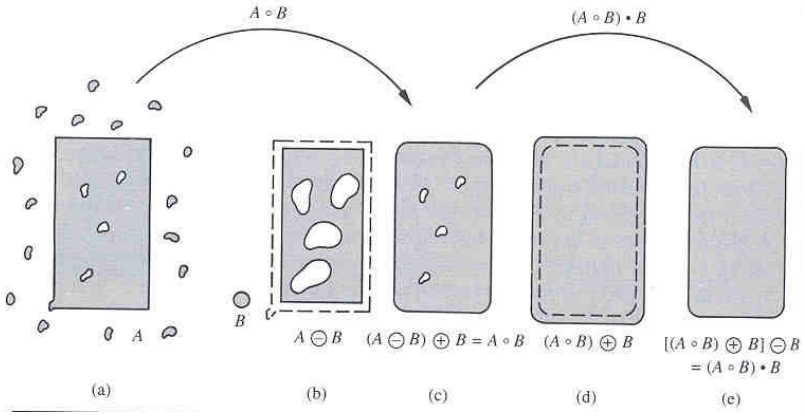
Seperti pada penjelasan sebelumnya operasi Dilatasi adalah proses penambahan *pixel* pada batasdari suatu objek pada citra digital masukan, sedangkan erosi adalah proses pemindahan/pengurangan *pixel* pada batasdari suatu objek. Jumlah *pixel* yang ditambahkan atau yang dihilangkan dari batas objek pada citra digital masukantergantung pada ukuran dan bentuk dari *structuring element* yang digunakan.

Pada penelitian ini akan menggunakan operasi opening yakni penggabungan antara erosi dan dilatasi. Operasi opening adalah proses erosi yang diikuti dengan dilatasi. Efek yang dihasilkan adalah menghilangnya objek-objek kecil dan kurus, memecah objek pada titik- titik yang kurus, dan secara umum memecah men-*smooth*-kan batas dari objek besar tanpa mengubah area objek yang signifikan.

$$A \circ S = (A \otimes S) \oplus S \quad \text{.....(4)}$$



Gambar 2.4 Pemodelan pada Operasi opening



Gambar 2.5 Contoh filter morfologi: (a) Gambar asli, dengan noise; (b) hasil dari operasi erosi; (c) hasil dari operasi opening; (d) hasil dilatasi pada operasi opening; (e) hasil akhir, operasi closing pada operasi opening (Giardina dan Dougherty[1988].)

2.2.4 Citra *Gray-Scale* (Keabuan)

Proses awal yang banyak dilakukan dalam image processing adalah mengubah citra berwarna menjadi citra gray-scale, hal ini digunakan untuk menyederhanakan model citra. Seperti telah dijelaskan di depan, citra berwarna terdiri dari 3 layer matrik yaitu R-layer, G-layer dan B-layer. Sehingga untuk

melakukan proses-proses selanjutnya tetap diperhatikan tiga layer di atas. Bila setiap proses perhitungan dilakukan menggunakan tiga layer, berarti dilakukan tiga perhitungan yang sama. Sehingga konsep itu diubah dengan mengubah 3 layer di atas menjadi 1 layer matrik gray-scale dan hasilnya adalah citra gray-scale. Dalam citra ini tidak ada lagi warna, yang ada adalah derajat keabuan.

Untuk mengubah citra berwarna yang mempunyai nilai matrik masing-masing r, g dan b menjadi citra gray scale dengan nilai s, maka konversi dapat dilakukan dengan mengambil rata-rata dari nilai r, g dan b sehingga dapat dituliskan menjadi:

$$s = \frac{r + g + b}{3} \quad \text{.....(5)}$$

Gambar 2.6 Contoh konversi citra ke Gray

2.2.5 Thresholding

Thresholding digunakan untuk mengatur jumlah derajat keabuan yang ada pada citra. Dengan menggunakan thresholding maka derajat keabuan bisa diubah sesuai keinginan, misalkan diinginkan menggunakan derajat keabuan 16, maka tinggal membagi nilai derajat keabuan dengan 16. Proses thresholding ini pada dasarnya adalah proses pengubahan kuantisasi pada citra, sehingga untuk melakukan thresholding dengan derajat keabuan dapat digunakan rumus:

$$x = b.\text{int}\left(w/b\right) \quad \text{.....(6)}$$

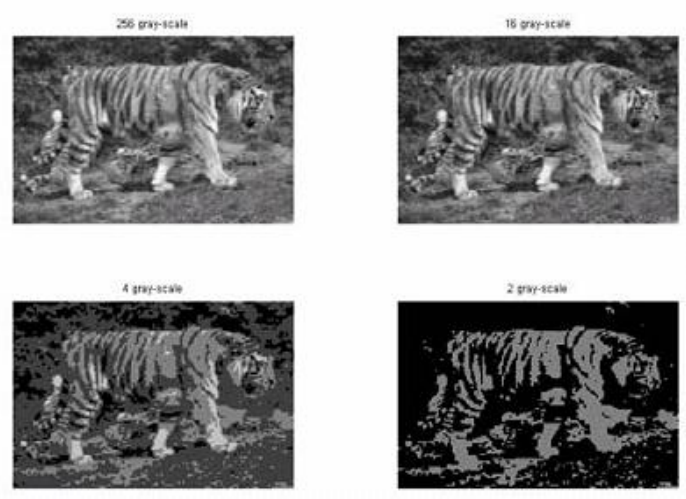
dimana :

w adalah nilai derajat keabuan sebelum thresholding

x adalah nilai derajat keabuan setelah thresholding

$$b = \text{int}\left(256/a\right)$$ .....(7)

Berikut ini contoh thresholding mulai di 256, 16, 4 dan 2.



Gambar 2.7 Contoh Derajat Kualitas *Tresholding*

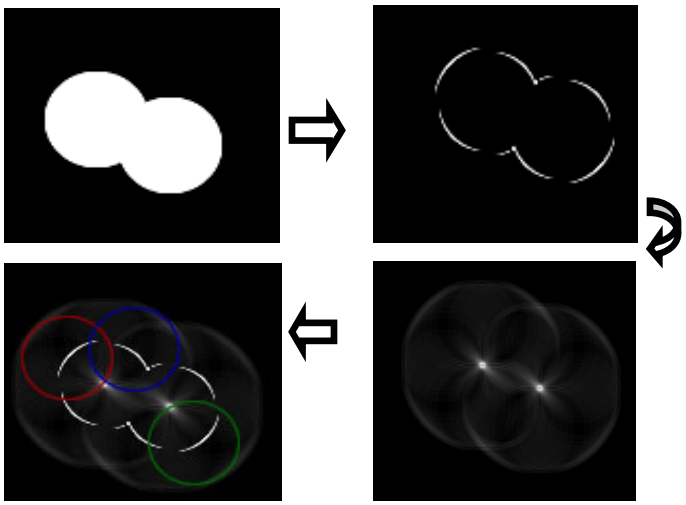
2.2.6 Transformasi Hough

Transformasi Hough adalah standar algoritma pada *computer vision* yang digunakan untuk menentukan parameter objek geometri sederhana seperti garis dan lingkaranpada citra. Transformasi Hough pada citra iris dapat digunakan untuk mengisolasi lokasi citra iris dari bagian lainnya seperti pupil dan selera.

Proses yang dilakukan adalah dengan mengambil parameter lingkaran yang melalui setiap titik pada citra tepi hasil pendeteksian tepi. Parameter ini adalah pusat koordinat x_c dan y_c dan radius r_c yang mampu untuk mendefinisikan setiap lingkaran yang sesuai dengan rumus

$x_c^2 + y_c^2 - r_c = 0$

Titiik maksimum pada ruang hough akan sama dengan radius dan pusat koordinat dari lingkaran terbai yang didefinisikan oleh pendetesian tepi.



Gambar 2.8 Proses *Hough Transform Circle* pada citra

2.3 AVR ATMega 8535

Atmel, salah satu vendor yang bergerak di bidang mikroelektronika telah mengembangkan AVR *(Alf and Vegard`s Riscprocessor)* sekitar tahun 1997. berbeda dengan mikrokontroller MCS51, AVR menggunakan arsitektur RISCK *(Reduce Instruction Set Compute)* yang mempunyai lebar bus data 8 bit. Perbedaan ini bisa dilihat dari frekuensi kerjanya. MCS51 memiliki frekuensi seperduabelas kali frekuensi osilator sedangkan frekuensi kerja AVR sama dengan frekuensi kerja osilator. Jadi dengan frekuensi osilator yang sama, kecepatan AVR dua belas kali lebih cepat dibanding kecepatan MCS51. Secara umum AVR dibagi menjadi 4 kelas, yaitu Attiny, AT90Sxx, ATMega dan AT86RFxx. Perbedaan antar tipe AVR terletak pada fitur-fitur yang ditawarkan, sementara dari segi set instruksi yang digunakan hampir sama.

2.3.1 Arsitektur ATMega

Fitur- fitur yang ada pada mikrokontroller AVR yakni :

1. 8 bit AVR berbasis *RISC* dengan performa tinggi dan konsumsi daya rendah dengan kecepatan 8 Mhz.

2. Kapabilitas memori flash 8 KB, SRAM sebesar 512 byte, dan *EEPROM (Electrcally Erasable Programmable Read Only Memory)* sebesar 512 byte.
3. *ADC* internal dengan fidelitas 10 bit sebanyak 8 channel.
4. Portal komunikasi serial *(USART)* dengan kecepatan maksimal 2,5 Mbps.
5. Enam pilihan mode *sleep* menghemat penggunaan daya listrik.
6. *Memory Flash* sebesar 8 Kb dengan kemampuan *Read While Write.*
7. Unit interupsi internal dan eksternal.
8. *Port* antarmuka SPI.
9. EEPROM sebesar 512 *byte* yang dapat diprogram saat operasi.
10. Antarmuka komparator analog.
11. CPU yang terdidi dari 3 buah register.

2.3.2 Konfigurasi Pin ATMega 8535

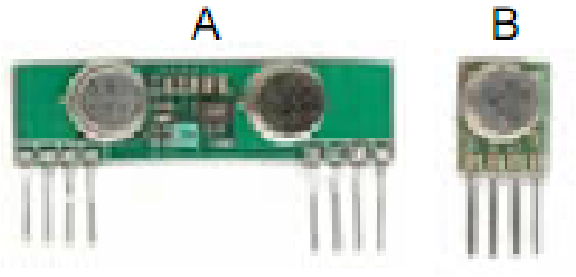
Konfigurasi pin ATMega bisa dilihat pada gambar dibawah ini.

Dari gambar tersebut dapat dijelaskan secara fungsional konfigurasi pin ATMega sebagai berikut :

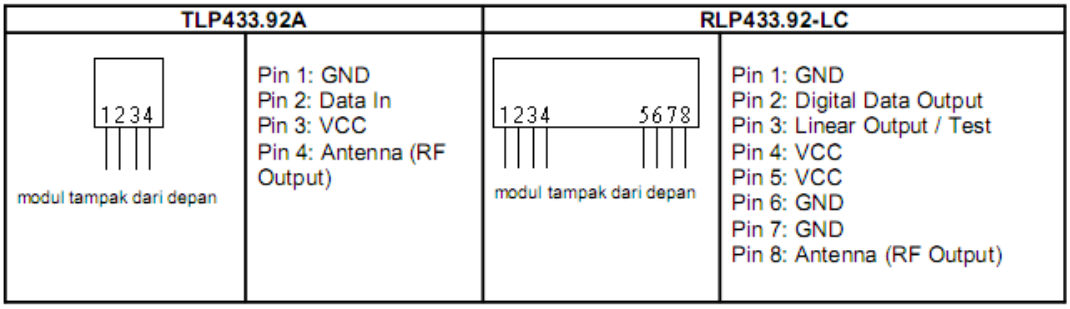
1. VCC merupakan pin yang berfungsi sebagai pin masukan catu daya.
2. GND merupakan pin ground.
3. Port A (PA.0..PA.7) merupakan pin I/O dua arah dan pin masukan ADC.
4. Port B (PB.0..PB.7) merupakan pin I/O dua arah dan pin fungsi khusus, yaitu Timer/ Counter, komparator analog, dan SPI.
5. Port C (PC.0..PC.7) merupakan pin I/O dua arah dan pin fungsi khusus, yaitu TWI, komparator analog, dan Timer Oscilator.
6. Port D (PD.0..PD.7) merupakan pin I/O dua arah dan pin fungsi khusus, yaitu komparator analog, interupsi eksternal, dan komunikasi serial.
7. RESET merupakan pin yang digunakan untuk me-reset mikrokontroller.
8. XTAL 1dan XTAL2 merupakan pin masukan *clock* eksternal.
9. AVCC merupakan pin masukan tegangan untuk ADC.
10. AREFF merupakan pin masukan tegangan referensi ADC.

2.4 Modul RF TLP433.92A dan Penerima RF RLP433.92-LC

Modul RF buatan LAIPAC ini sering sekali digunakan sebagai alat untuk komunikasi data secara wireless menggunakan media gelombang radio. Biasanya kedua modul ini dihubungkan dengan mikrokontroler atau peralatan digital yang lainnya. Masukan data untuk modul TLP adalah serial dengan level TTL (Transistor-transistor Logic). Jangkauan komunikasi maksimum dari pasangan modul RF ini adalah 100 meter tanpa halangan dan 30 meter di dalam gedung. Ukuran ini dapat dipengaruhi oleh faktor antena, kebisingan, dan tegangan kerja dari pemancar. Panjang antena yang digunakan adalah 17 cm, dan terbuat dari kawat besi



Gambar 2.9 Modul TLP433.92A dan Penerima RF RLP433.92-LC



Gambar 2.10 Konfigurasi pin TLP433.92A dan Penerima RF RLP433.92-LC

2.5 Motor Sevo

Motor servo adalah sebuah motor dengan sistem closed feedback di mana posisi dari motor akan diinformasikan kembali ke rangkaian kontrol yang ada di dalam motor servo. Motor ini terdiri dari motor dc, rangkaian gear , potensio meter dan rangkaian kontrol.

Gambar 2.11 Motor Servo

Motor servo mampu bekerja dua arah (CW dan CCW) dimana arah dan sudut pergerakan rotornya dapat dikendalikan hanya dengan memberikan pengaturan duty cycle sinyal PWM pada bagian pin kontrolnya. Terdapat 2 jenis motor servo yakni :

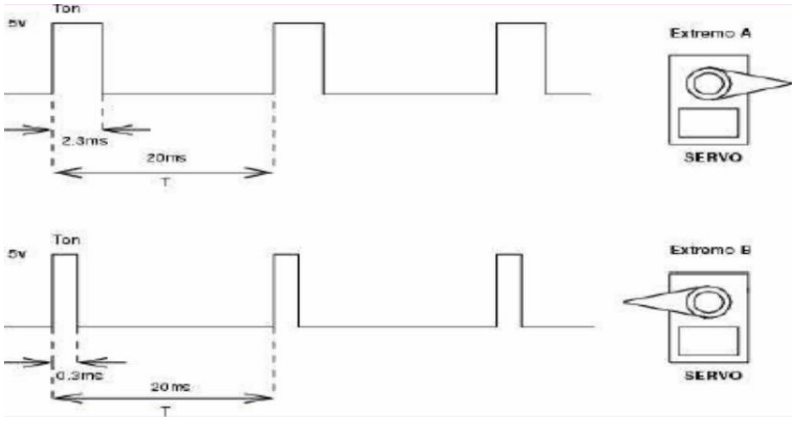
1. Motor Servo Standar 180°

   Motor servo jenis ini hanya mampu bergerak dua arah (CW dan CCW) dengan defleksi masing- masing sudut mencapai 90° sehingga total defleksisudut dari kanan – tengah – kiri adalah 180°.

2. Motor Servo Continuous

   Motor servo jenis ini mampu bergerak dua arah (CW dan CCW) tanpa batasan defleksi sudut putar (dapat berputar secara kontinyu).

Untuk menggerakkan motor sevo digunakan pensinyalan PWM yang sesuai agar didapatkan posisi sudut yang tepat. Gambar dibawah ini merupakan settingan PWM untuk motor servo.



Gambar 2.12 Pewaktuan sinyal pada motor sevo

Motor Servo akan bekerja secara baik jika pada bagian pin kontrolnya diberikan sinyal PWM dengan frekuensi 50Hz..Dimana pada saat sinyal dengan frekuensi 50Hz tersebut dicapai pada kondisi Ton duty cycle 1.5ms, maka rotor dari motor akan berhenti tepat di tengah-tengah (sudut 0° / netral). Pada saat Ton duty cycle dari sinyal yang diberikan kurang dari 1.5ms, maka rotor akan berputar ke arah kiri dengan membentuk sudut yang besarnya linier terhadap besarnya Ton duty cycle, dan akan bertahan diposisi tersebut.

BAB III
METODOLOGI PENELITIAN

3.1. Tempat dan Waktu Penelitian

Penelitian untuk membuat Desain Sistem Navigasi Robot Menggunakan Isyarat Mata Menggunakan Metode *Canny* Dan *Hough Transform* ini dilakukan di Laboratorium Instrumentasi dan Otomasi Pabrik, Kampus Fakultas Teknik, Universitas Jember di Jl. Slamet Riyadi no. 62 Patrang, Jember. Sedangkan waktu penelitian ini dilaksanakan mulai Mei 2011.

Tabel 3.1 Jadwal Kegiatan Penelitian

No	Kegiatan	Bulan					
		Mei	Juni	Juli	Aguts	Sep	Okt
1	Studi Literatur						
2	Pengerjaan Alat						
3	Pengujian Alat						
4	Analisa Alat						
5	Pembahasan						
6	Laporan						

3.2. Tahapan Perancangan

Dalam membuat desain sistem navigasi robot dengan isyarat mata menggunakan metode *Canny* dan *Hough Transform*, dibutuhkan langkah-langkah perancangan, antara lain:

7. Studi Literatur
8. Perancangan posisi penempatan kamera

   Menentukan posisi penempatan kamera yang akan digunakan sebagai pendeteksi posisi bola mata.
9. Perancangan perangkat keras dan perangkat lunak

Pembuatan perangkat penyusun sistem yang menghubungkan kamera, komputer dan robot beroda berupa perangkat keras dan perangkat lunak. Perancangan perangkat keras meliputi robot beroda, tempat peletakan kamera, 2 buah sistim minimum sebagai media transmisi data secara wireless. Sedangkan perangkat lunak meliputi program pendeteksi posisi bola mata dengan metode *canny edge detection* dan *hough transfrom* serta program pada robot beroda.

10. Pengujian perangkat keras dan perangkat lunak

    Pengujian perangkat penyusun sistem yang sudah dirancang, yaitu perangkat keras dan perangkat lunak sebelum diintegrasikan menjadi sistem keseluruhan.

11. Integrasi sistem

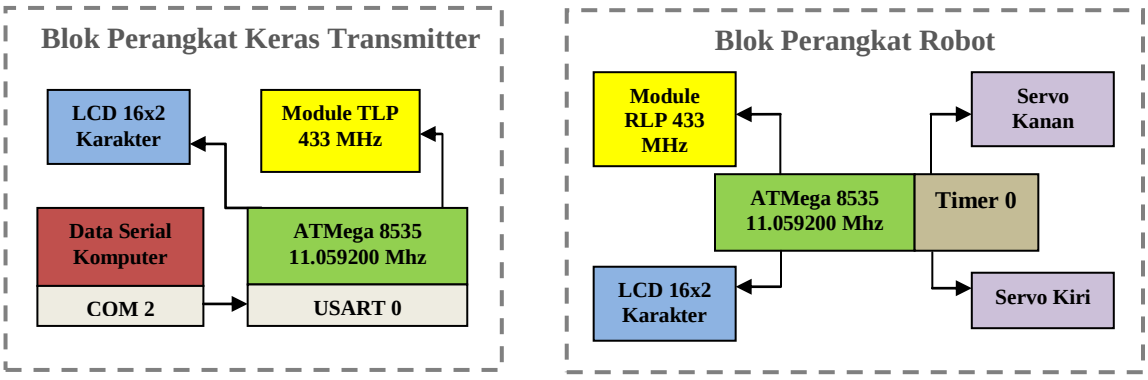
    Mengintegrasikan perangkat penyusun sistem yang sudah dirancang, yaitu perangkat keras dan perangkat lunak menjadi sistem keseluruhan.

12. Pengujian dan analisa sistem

    Menguji sistem yang telah terintegrasi secara menyeluruh untuk selanjutnya dilakukan analisa kinerja sesuai dengan fungsinya untuk navigasi robot dan kesesuaian komunikasi data.

3.3 Desain Perangkat Keras

Pembuatan perangkat keras ini dilakukan berdasarkan dengan blok diagram pada gambar 3.1 yang terdiri atas modul komunikasi nirkabel, system kontrolel, driver motor servo dan aktuator dua buah motor servo *continous* 360^0 serta desain penempatan posisi helm. Ke semua blok system perangkat keras ini harus terhubung dengan benar.



Gambar 3.1 Blok Diagram Perangkat Keras

3.3.1 Blok Perangkat Keras Transmitter

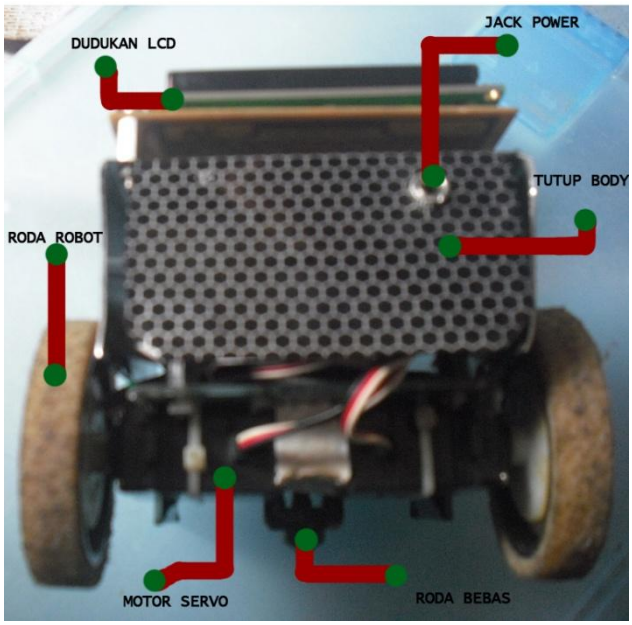
Data serial komputer merupakan data posisi bola mata yang telah diproses sebelumnya oleh perangkat lunak *Image Processing* yang dikirim melalui komunikasi serial dengan pemilihan COM 2. Data kemudian terima oleh mikrokontroller AT Mega 8535 melalui *USART 0* dan diolah sehingga dapat diterjemahkan sebagai perintah data aksi robot yang nantinya dikirim secara nirkabel melalui modul TLP 433 Mhz. Selain itu, data juga ditampilkan pada LCD 16x2 untuk memudahkan koreksi kesalahan pembacaan.

3.3.2 Blok Perangkat Keras Robot

Pada blok perangkat keras robot juga mengunakan mikrokontroller ATMega 8535 sebagai *main controller*. Data yang di pancarkan dari module TLP 433 MHZ akan di tangkap oleh penerima RLP 433 MHz yang kemudian diolah pada modul drivernya sehingga di dapatkan sebuah 4 buah komninasi input. 4 buah kombinasi input tersebut nantinya di terjemahkan oleh mikrokontroller sehingga didapatkan sebuah keputusan navigasi robot dengan memberikan sinyal PWM (*Pulse Width Modulation*) yang dibangkitkan oleh fitur *Timer 0* ATMega 8535 untuk kedua motor servo.

3.3.3 Mekanik Robot

Mekanik robot dibuat menggunakan bahan-bahan yang ringan namun cukup kuat seperti papan acrylic dan aluminium. Komponen gerak robot terdiri dari dua buah motor servo *continous* dengan kecepatan maksimal pada tegangan 5V yakni 0,14sec/60^0 . Pada bagian depan diberi *rol coaster* agar lebih memberikan kemudahan pergerakan pada robot.



Gambar 3.2 Desain Mekanik Robot

3.3.4 Sistem Komunikasi Wireless

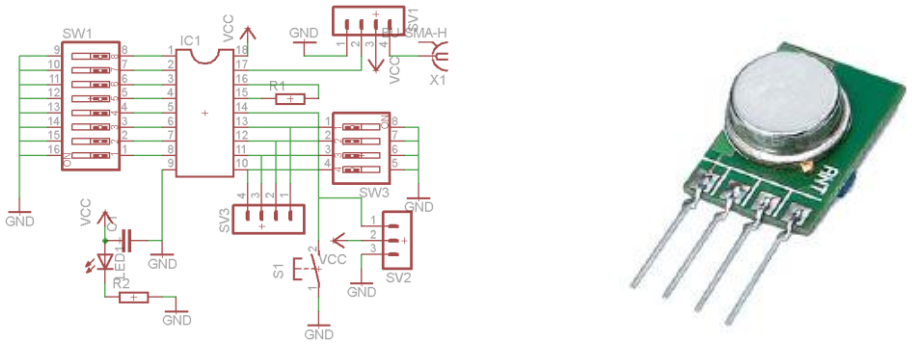
Sesuai dengan perancangan sebelumnya bahwa kendali robot ini menggunakan media komunikasi secara nirkabel. Media komunikasi nirkabel yang digunakan adalah TLP 433 dan RLP 433 yang kedua pemancar dan penerima tersebut bekerja pada frekuensi 433 MHz. Untuk dapat menggunakan kedua perangkat nirkable tersebut diperlukan driver module yang digunakan sebagai pengatur register alamat serta penerimaan data. Dari driver tersebut data yang dapat dikombinasikan yakni sebanyak 4 buah input data sehingga dapat menghasilkan 16 keadaan ouput yang ingin di olah.

Modul RF buatan LAIPAC ini sering sekali digunakan sebagai alat untuk komunikasi data secara wireless menggunakan media gelombang radio. Biasanya kedua modul ini dihubungkan dengan mikrokontroler atau peralatan digital yang lainnya. Masukan data untuk modul TLP adalah serial dengan level TTL (Transistor-transistor Logic). Jangkauan komunikasi maksimum dari pasangan modul RF ini adalah 100 meter tanpa halangan dan 30 meter di dalam gedung. Ukuran ini dapat

dipengaruhi oleh faktor antena, kebisingan, dan tegangan kerja dari pemancar. Panjang antena yang digunakan adalah 17 cm, dan terbuat dari kawat besi.

3.3.4.1 Driver TLP 433 MHz

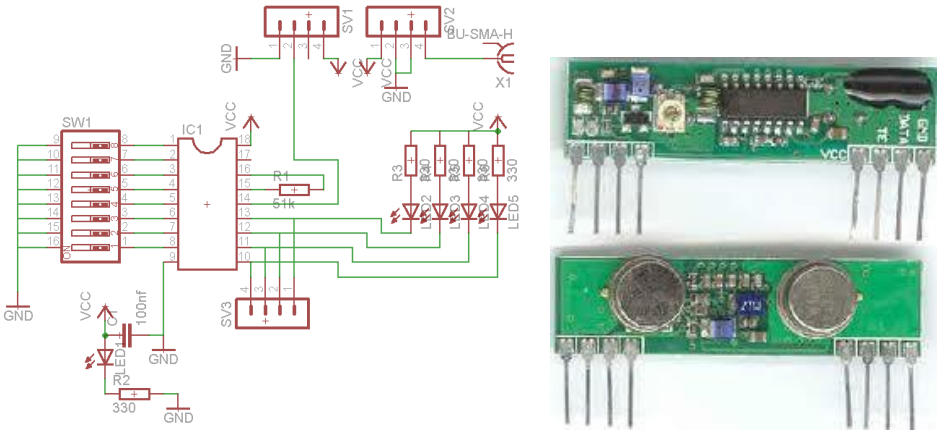
Modul TLP433 ini menggunakan modulasi ASK (Amplitudor Shift keying), dimana frekuensi kerja dari modul ini adalah 433 MHz. Modul ini berfungsi untuk mengirimkan data secara serial ke modul penerima RLP433.



Gambar 3.3 (a) Skema rangkaian driver TLP433.92A (b) TLP 433 MHz.

3.3.4.2 Driver RLP 433 MHz

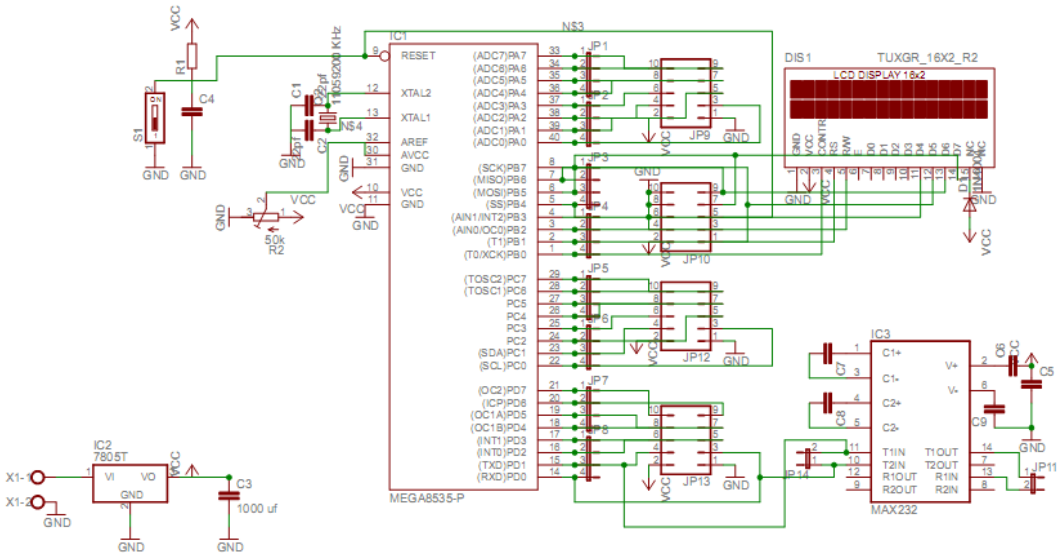
Modul RLP433 ini sama halnya dengan modul TLP yang menggunakan modulasi ASK (Amplifier Shift Keying) dengan frekuensi kerja dari modul ini adalah 433 MHz. Modul ini berfungsi untuk menerima data yang dikirim secara serial dari modul pemancar TLP433.



Gambar 3.4 (a) Skema rangkaian driver RLP433.92A (b) RLP 433 MHz.

3.3.5 Main Controller Board (Sistem Minimum AVR ATMega 8535)

Sistem minimum mikrokontroler adalah sistem elektronika yang terdiri dari komponen-komponen dasar yang dibutuhkan oleh suatu mikrokontroler untuk dapat berfungsi dengan baik. Pada umumnya, suatu mikrokontoler membutuhkan dua elemen (selain power supply) untuk berfungsi yakni Kristal Oscillator (XTAL), dan Rangkaian RESET. Analogi fungsi Kristal Oscillator adalah jantung pada tubuh manusia. Perbedaannya, jantung memompa darah dan seluruh kandungannya, sedangkan XTAL memompa data. Dan fungsi rangkaian RESET adalah untuk membuat mikrokontroler memulai kembali pembacaan program, hal tersebut dibutuhkan pada saat mikrokontroler mengalami gangguan dalam meng-eksekusi program. Pada sistem minimum AVR khususnya ATMEGA8535 terdapat elemen tambahan (optional), yaitu rangkaian pengendalian ADC: AGND (= GND ADC), AVCC (VCC ADC), dan AREF (= Tegangan Referensi ADC). Jangan lupa tambahkan konektor ISP untuk mengunduh (download) program ke mikrokontroler.



Gambar 3.5 Rangkaian Sistem Minimum Mikrokontroller ATMega 8535

Tabel 3.1 Alokasi Pin ATMega 8535 (Sisi Transmitter)

Pin	Keterangan	Koneksi
11,31	Ground	Ground Catu Daya
10	VCC	Catu daya +5 Volt
1-8	PORTB	LCD
12	Xtall 1	Crystal 11.0592 MHz
13	Xtall 2	Crystal 11.0592 MHz
30	AVCC	Tegangan Referensi ADC (+ 5V)
32	AREF	Tegangan Referensi ADC (+ 5V)
14	RXD	T2 Input IC MAX 232
15	TXD	T1 Input IC MAX 232

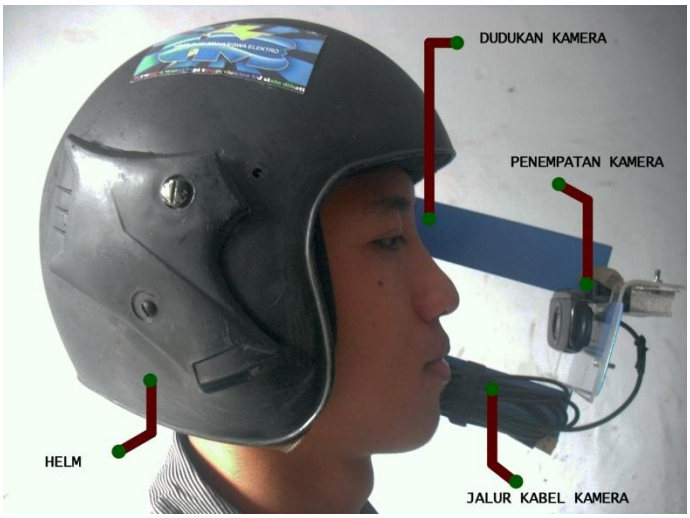
Tabel 3.2 Alokasi Pin ATMega 8535 (Controller Robot)

Pin	Keterangan	Koneksi
11,31	Ground	Ground Catu Daya
10	VCC	Catu daya +5 Volt
33-40	PORTA	LCD
12	Xtall 1	Crystal 11.0592 MHz
13	Xtall 2	Crystal 11.0592 MHz
30	AVCC	Tegangan Referensi ADC (+ 5V)
32	AREF	Tegangan Referensi ADC (+ 5V)
14-17	PD.0 – PD.3	INPUT DATA dari RLP 433 MHz
1-4	PB.0 – PB.3	Input Data Keypad

3.3.6 Desain Penempatan Posisi Kamera

Kamera merupakan objek vital yang ada pada system ini karena berfungsi sebagai input data gambar yang nantinya diolah oleh *software* pengolah citra. Untuk dapat menangkap objek mata sesuai dengan yang diharapkan, maka posisi kamera

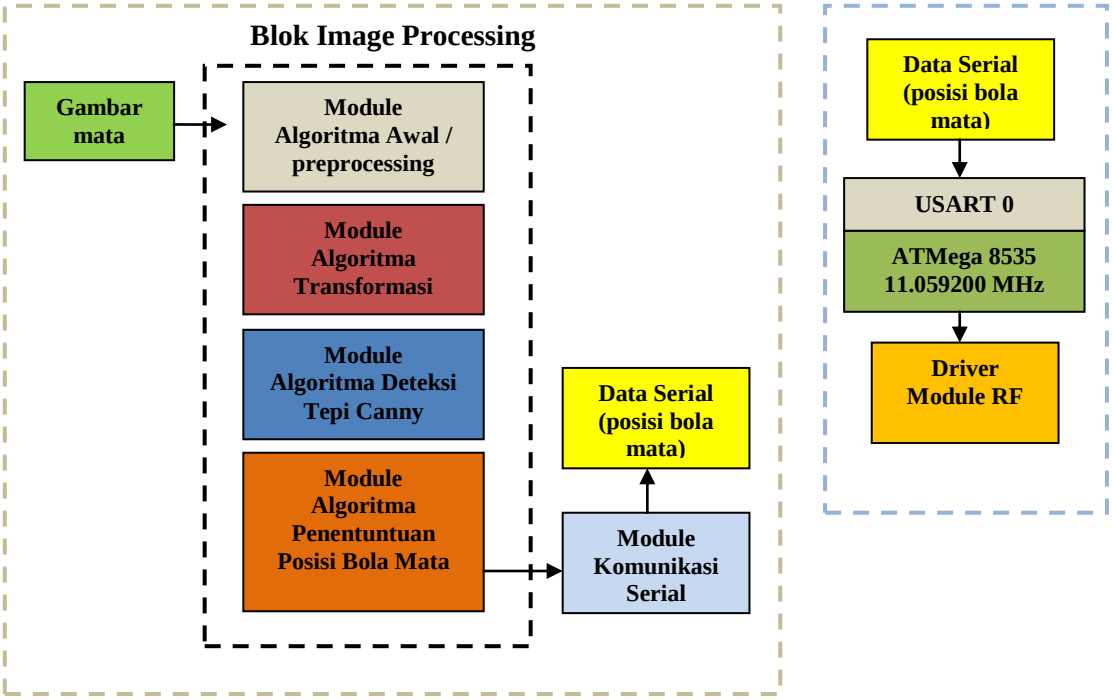
harus diatur sedemikian rupa agar objek bola mata tereteksi keseluruhan. Dari pertimbangan tersebut ada beberapa alternative yang salah satunya yakni dengan memberikan dudukan pada sebuah helm sehingga posisi kamera tersebut berada pada depan mata pengguna.



Gambar 3.6 Desain peletakan kamera pada helm

3.4 Desain Perangkat Lunak

Perangkat lunak pada system ini terbagi menjadi 2 blok bagian, yakni blok pada *image processing* yang terdiri dari beberapa modul-modul yakni modul algoritma awal, modul algoritma transformasi, modul algoritma deteksi tepi *Canny*, modul algoritma penentuan posisi bola mata dan blok perangkat lunak pada blok transmitter data. Perangkat lunak pada blok *image processing* merupakan system yang digunakan dalam mengolah data citra bola mata yang kemudian di dapatkan sebuah data posisi bola mata. Piranti ini menggunakan software *Visual C++ 2008 Ekspress Edition* dan juga menggunakan OpenCV 2.1. Sedangkan pada blok transmitter data menggunakan Mikrokontroller ATMega 8535 sebagai kontrollernya dengan compiler software CodeVisionAVR.



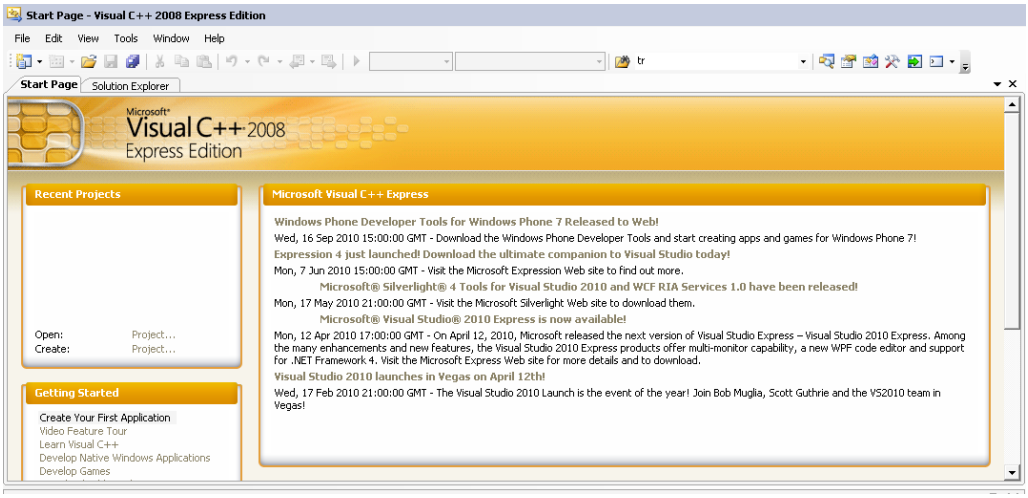
Gambar 3.7 Blok Diagram Perangkat Lunak

3.4.1 Blok piranti Image Processing

Dalam pemrograman pengolahan citra ini menggunakan software *Visual C++ 2008 Ekspress Edition. Visual C++ 2008 Ekspress Edition* ini merupakan jenis pengembangan terpadu (IDE) yang dikembangkan oleh Microsoft yang merupakan versi ringan dari Microsoft Visual Studio.

Visual C++ 2008 Express dapat digunakan untuk membangun aplikasi native dan non-managed. Termasuk Windows Platform SDK yang dapat digunakan untuk membangun aplikasi yang menggunakan Win32 API. Aplikasi menggunakan MFC atau ATL memerlukan Standard Edition atau lebih tinggi, dan tidak dapat dikompilasi dengan Express Edition. Visual C++ 2008 Express Edition ini juga dapat digunakan

untuk mengkompilasi .NET serta aplikasi Win32. Namun, mengkompilasi native aplikasi 64-bit melalui IDE tidak didukung tanpa konfigurasi terlebih dahulu.



Gambar 3.8 Tampilan *Work Space* pada Visual C++ 2008 Ekspress Edition

1. **Modul Algoritma Awal / Preprocessing**

   Pada Modul ini berisikan beberapa fungsi algoritma diantaranya fungsi untuk mendeteksi hardware kamera. Hal ini dilakukan agar sebelum program utama dijalankan diperlukan sebuah perangkat utama sebagai sensor yakni kamera. Berikut cuplikan program C untuk mendeteksi hardware kamera serta mencapture gambar.

```
capture = cvCaptureFromCAM(0);
    if (!capture)
            exit_nicely("Cannot initialize camera!");

    cvSetCaptureProperty(capture, CV_CAP_PROP_FRAME_WIDTH,  FRAME_WIDTH);
    cvSetCaptureProperty(capture, CV_CAP_PROP_FRAME_HEIGHT, FRAME_HEIGHT);

    frame = cvQueryFrame(capture);
    cvFlip(frame,NULL,-1);
    if (!frame)
            exit_nicely("cannot query frame!");
```

```
cvInitFont(&font, CV_FONT_HERSHEY_SIMPLEX, 0.4, 0.4, 0, 1, 8);
cvNamedWindow(wnd_name, 1);
```

Selanjutnya dilakukan fungsi perubahan gambar RGB ke dalam Grayscale. Hal ini dibutuhkan karena nantinya akan dilakukan deteksi tepi. Untuk dapat melukkan deteksi tepi maka fungsi grayscale ini perlu di lakukan terlebih dahulu.

```
cvCvtColor(ori, gray, CV_RGB2GRAY);  // fungsi untuk mengubah ke dalam grayscale
```

2. Modul Algoritma Transformasi

Modul ini berisikan beberapa fungsi algoritma diantaranya fungsi untuk mengubah tresholding yakni proses mengubah citra berderajat keabuan menjadi citra biner atau hitam putih sehingga dapat diketahui daerah mana yang termasuk obyek dan *background* dari citra secara jelas. Citra hasil *thresholding* ini nantinya digunakan lebih lanjut untuk proses pengenalan obyek serta ekstraksi fitur yang dilakukan oleh deteksi tepi Canny. Setelah itu dilakukan fungsi *smoothing* yakni dengan filter Gaussian. Filter Gaussian ini digunakan untuk mereduksi noise sehingga gambar lebih *smooth*. Dan yang terakhir yakni dilakukan sedikit operasi dilasi yakni proses penggabungan titik-titik latar (0) menjadi bagian dari objek (1), berdasarkan structuring element S yang digunakan.

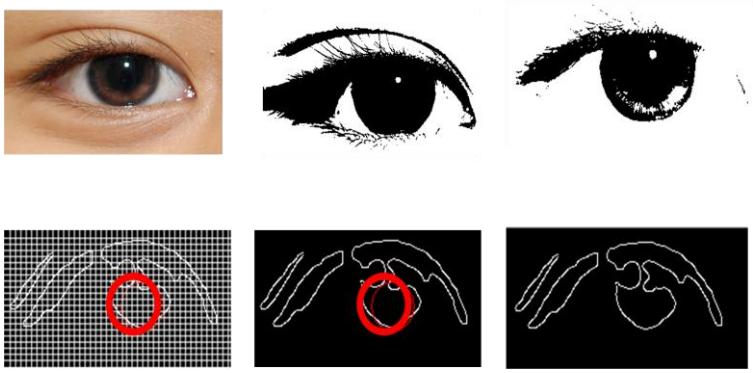
```
cvSmooth( gray, gray, CV_GAUSSIAN, 11, 11 );
cvDilate(gray,gray,0,10);
cvThreshold(gray,gray,lowSliderPosition,highSliderPosition,CV_THRESH_BINARY);
```

3. Modul Algoritma Deteksi Tepi Canny

Deteksi tepi cany ini digunakan agar didapatkan tepian bentuk bola mata sehingga dapat dideteksi posisinya. Yang perlu diperhatikan pada deteksi tepi ini dalah pengaturan batas bawah threshold dan batas atas threshold. Karena hanya

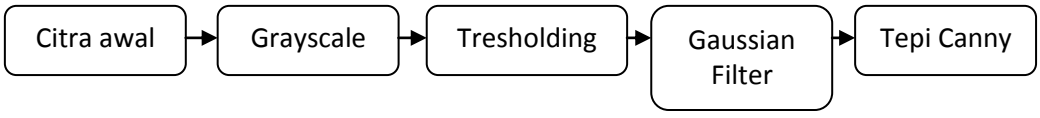
diantara batas bawah dan batas atas saja nantinya yang akan di deteksi tepi sehingga jika kita salah dalam pengaturan ke dua komponen tersebut bola mata tidak akan terbentuk tepiannya .

```
edges = cvCreateImage(cvGetSize(frame), IPL_DEPTH_8U, 1);
cvCanny( gray, edges, thelowThreshold, theHighThreshold, theAperture);
```



Gambar 3.9 Ilustrasi Deteksi Tepi Canny

Gambar diatas merupakan ilustrasi step-step algoritma deteksi tepi yakni citra awal mata kemudian di grayscale yakni pembentukan citra ke abuan. Selanjutnya tresholding sehingga menghasilkan cita hitam putih. Selanjutnya filter Gaussian diberikan agar memperhalus citra agar deteksi tepi lebih maksimal. Kemudian metode deteksi tepi canny dilakukan sehingga terdapat bentuk bulatan yang nantinya akan diproses selanjutnya agar dapat ditentukan posisi pixelnya.



Gambar 3.10 Step-step Modul Algoritma Tepi Canny

4. Modul Algoritma Penentuan Posisi Bola Mata

Untuk dapat menentukan posisi bola mata maka diperlukan sebuah metode untuk dapat mendeteksi bentuk bola mata, dalam hal ini yakni lingkaran. Dalam ilmu pengolahan citra dikenal sebuah fungsi untuk dapat mendeteksi bentuk-bentuk misalnya garis, lingkaran dll. Fungsi tersebut yakni Hough Transform. Karena pada penelitian ini bentuk yang di cari yakni lingkaran maka dalam Hough Transform ada cara untuk mendeteksi lingkaran yakni dengan fungsi Hough Circle Transform. Untuk dapat menggunakan fungsi ini agar lebih optimal maka harus dilakukan pendeteksian tepi pada citra sehingga karakter bentuk lingkaran akan lebih dapat mudah terdeteksi.

```
CvSeq* circles = cvHoughCircles(edges, storage,
CV_HOUGH_GRADIENT,0.5, edges->height,SliderPosition,sliderposisi);
int i;
for( i = 0; i < circles->total; i++ ) {
p = (float*)cvGetSeqElem( circles,i );
cvCircle( frame, cvPoint(cvRound(p[0]),cvRound(p[1])), 8,
CV_RGB(0,0,255), -1, 8, 0 );
cvCircle( frame,
cvPoint(cvRound(p[0]),cvRound(p[1])),cvRound(p[2]),CV_RGB(255,255,255),1
,8,0);}
```

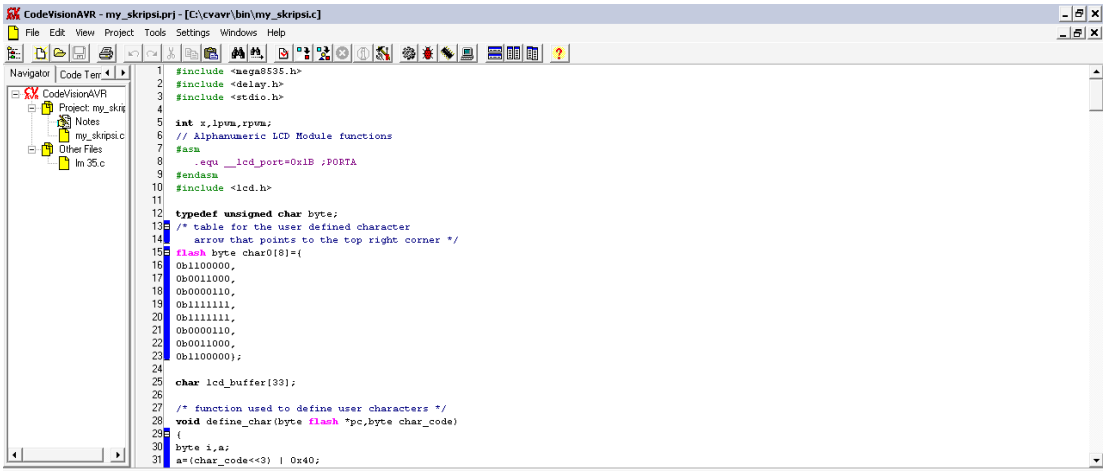
- **Langkah-langkah deteksi posisi bola mata**

1. Scanning lingkaran dengan fungi Hough Circle Transform
2. Kalibrasi nilai pixel pada saat bola mata ditengah → **p[0]**.
3. Untuk mendapatkan keputusan mata kanan, mata kiri maka ditentukan dengan pengurangan pixel sekarang dikurangi pixel posisi bola mata ditengah
    - Jika hasil dari **p[t] – p[0] > 20** maka **MATA KIRI**
    - Jika hasil dari **p[t] – p[0] > -20** maka **MATA KANAN**
    - Jika hasil dari **p[t] – p[tengah] >-20 dan < 20** maka **TENGAH**

Nilai 20 pixel adalah hasil dari kalibrasi sebelumnya. Kalibrasi ini dilakukan dengan *trial and error*. Dilakukan pengujian sebanyak 5 kali kemudian diambil rata-rata. Didapatkan nilai koefisien pengurangan yang paling baik adalah 20 pixel.

3.4.2 Compiler CodeVIsionAVR

CodeVisionAVR merupakan sebuah cross-compiler C, Integrated Development Environtment (IDE), dan Automatic Program Generator yang didesain untuk mikrokontroler buatan Atmel seri AVR. CodeVisionAVR dapat dijalankan pada ariab operasi Windows 95, 98, Me, NT4, 2000, dan XP. Cross-compiler C mampu menerjemahkan ariab semua perintah dari bahasa ANSI C, sejauh yang diijinkan oleh arsitektur dari AVR, dengan tambahan beberapa fitur untuk mengambil kelebihan khusus dari arsitektur AVR dan kebutuhan pada ariab embedded. File object COFF hasil kompilasi dapat digunakan untuk keperluan debugging pada tingkatan C, dengan pengamatan ariable, menggunakan debugger Atmel AVR Studio.



Gambar 3.10 Tampilan *Work Space* pada software CodeVisionAVR

IDE mempunyai fasilitas internal berupa software AVR Chip In-System Programmer yang memungkinkan Anda untuk melakukan transfer program kedalam chip mikrokontroler setelah sukses melakukan kompilasi/asembli secara otomatis. Software In-System Programmer didesain untuk bekerja dengan Atmel

STK500/AVRISP/AVRProg, Kanda Systems STK200+/300, Dontronics DT006, Vogel Elektronik VTEC-ISP, Futurlec JRAVR dan MicroTronics ATCPU/Mega2000 programmers/development boards. Untuk keperluan debugging sistem embedded, yang menggunakan komunikasi serial, IDE mempunyai fasilitas internal berupa sebuah Terminal.

3.5 Alogaritma, Skema Perangkat Keras Sistem Navigasi Robot, dan *Flowchart*

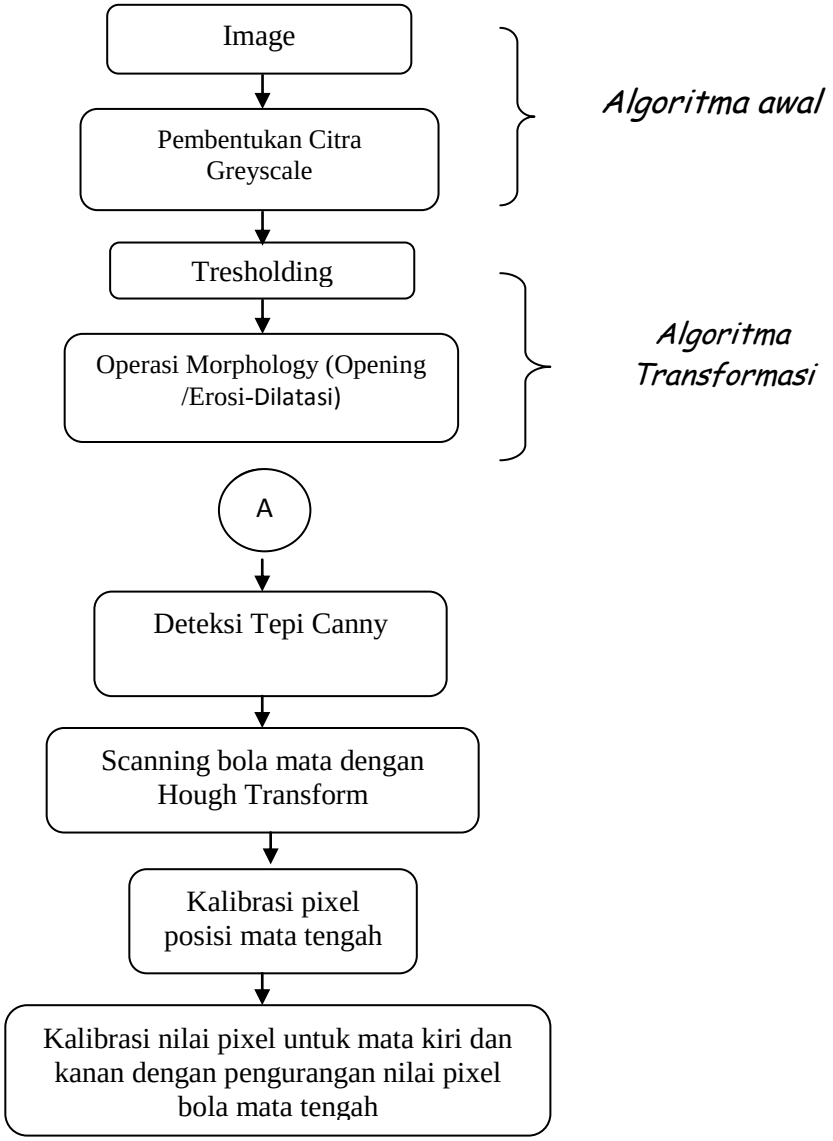
3.5.1. Algoritma Sistem Navigasi Robot

Adapun algoritma yang dipakai dalam pembutan sistem navigasi robot menggunakan isyarat mata adalah sebagai berikut :

2) Kamera menangkap gambar kedua mata.
3) Gambar mata di proses dengan pengolahan citra untuk mengetahui posisi bola mata dan kombinasinya.
4) Penentuan keputusan navigasi robot (kanan,kiri,maju).
5) Pengiriman data perintah navigasi dengan menggunakan module TLP433.92A dan Penerima RF RLP433.92-LC .
6) Robot menerima perintah navigasi dengan membaca data kiriman.
7) Robot melakukan navigasi sesuai perintah.

3.5.2 Algoritma Pendeteksian posisi bola mata menggunakan metode morphology

Dalam penelitian ini penulis menggunakan metode morphology yang sudah banyak digunakan pada pengolahan citra pada umumnya. Metode morphology ini di terapkan pada pendeteksian posisi bola mata dengan beberapa tahapan yakni:

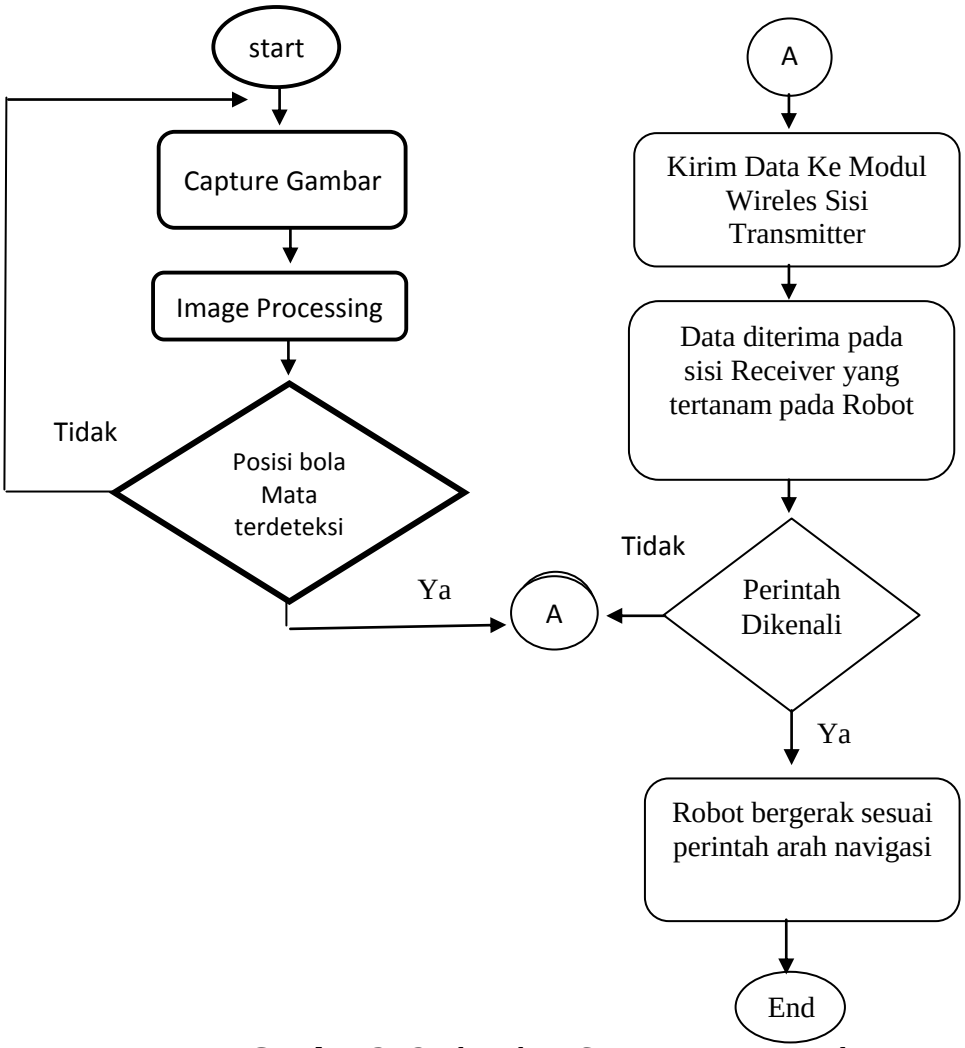


Gambar 3.11 Tahapan Algoritma Pendeteksian Posisi Bola Mata

Algoritma awal atau preprocessing merupakan algoritma awal yang digunakan sebelum mengolah ke jenjang selanjutnya. Algoritma awal ini terdiri atas deteksi perangkat kamera, algoritma pengambilan gambar (*capture*) dan juga pembentukan citra grayscale. Setelah dilakukan algoritma awal, maka dilakukan algoritma transformasi yakni berupa tresholding dan operasi morfologi. Tresholding

diberikan agar citra berada pada domain hitam dan putih sehingga nantinya dapt dilakukan deteksi tepi. Deteksi tepi dilakukan agar dapat memberikan bentuk citra bulat yang dihasilkan dari bola mata. Metode Canny dipilih karena keandalannya dari pada yang lain. Setelah terbentuk citra tepi maka dapat dilakukan scanning bola mata dengan menggunakan metode Hough Circle Transform. Dengan metode ini dapat menscanning posisi bola mata dengan tepat dan cepat yang nantinya akan dilakukan kalibrasi posisi bola mata tengah, kanan dan kiri untuk di terjemahkan kedalam sebuah data perintah navigasi robot.

3.5.3. *Flowchart* Sistem Alat



Gambar 3.12 *Flowchart* Sistem Kerja Perangkat

BAB 4. HASIL DAN PEMBAHASAN

4.1. Hasil Percobaan Perangkat Keras

4.1.1 Hasil Percobaan Rangkaian Pemancar TLP433.92A dan Penerima RF RLP433.92-LC (Tx dan Rx)

Untuk mengetahui bahwa modul RF bisa menerima atau mengirimkan data dengan baik, terlebih dahulu harus dilakukan suatu pengujian. Cara menguji modul RF biasanya hanya memberikan logika 1 atau 0 pada pin data modul transmitter, kemudian sinyal tersebut diterima oleh modul penerima dan output-nya juga 1 atau 0 sesuai dengan logika yang dikirimkan. Pada modul-modul tertentu hal tersebut tidak dapat dilakukan, karena adanya batas kecepatan minimal di dalam pengiriman data. Pada modul TLP433.92A ini menggunakan IC HT12E yang merupakan sebuah encoder. IC HT12E ini mengkonversi data parallel input kedalam serial output. IC ini meng-encodekan 12 bit data parallel kedalam data serial untuk transmisi data RF. 12 bit data itu kemudian dibagi menjadi 8 alamat data dan 4 bit data.

Sedangkan pada pengujian sensor receiver RF Module RLP433.92-LC membutuhkan IC HT12D dalam bekerjanya. IC HT12D ini merupakan sebuah decoder. IC ini biasanya digunakan pada aplikasi remote control seperti control pintu mobil, *security sistem* dan lain sebagainya. Ic ini mengkonversi data input serial kedalam parallel output.Ic ini akan mendecode data alamat serial dan data yang diterima dari modul Rx RLP433.92-LC kedalam sebuah parallel data dan mengirimkan kedalam pin output. Berikut ini adalah data hasil pengujian percobaan pengiriman data dengan mengkombinasikan mikrokontroller AVR ATMega8535.

Tabel. 4.1 Percobaan pengiriman tansmisi data modul TLP433.92A dan RLP433.92-LC.

No	Alamat Data	Data Tx	Data Rx	Keterangan
1	0b00000000	0b0001	0b0001	Berhasil
2	0b00000011	0b0011	0b0011	Berhasil
3	0b00000111	0b1011	0b1011	Berhasil
4	0b00011011	0b1001	0b1001	Berhasil
5	0b11111111	0b0111	0b0111	Berhasil

Dari table diatas dapat disimpulkan bahwa rangkaian pemancar dan penerima modul RF ini bekerja dengan baik. Dari 5 kali data percobaan pengiriman, kesemuanya berhasil diterima oleh rangkaian receiver dengan baik. Pengujian berikutnya adalah tingkat keberhasilan pembacaan data dengan jarak komunikasi yang diubah-ubah. Tabel dibawah ini merupakan hasil pengujian data tingkat keberhasilan pembacaan data dengan jarak komunikasi yang bervariasi.

Tabel 4.2 Pengujian transmisi data dengan halangan dinding dengan 5 kali percobaan

No	Alamat Data	Data Tx	Data Rx	Jarak (m)	% Keberhasilan	Keterangan
1	0b00000000	0b0001	0b0001	5	100 %	Berhasil
2	0b00000000	0b0011	0b0011	10	100 %	Berhasil
3	0b00000000	0b1100	0b1100	15	100 %	Berhasil
4	0b00000000	0b1111	0b1111	20	100 %	Berhasil
5	0b00000000	0b1010	0b1110	25	10 %	Gagal

Tabel 4.3 Pengujian transmisi data di tanah lapang (tidak ada halangan) dengan 5 kali percobaan

No	Alamat Data	Data Tx	Data Rx	Jarak (m)	% Keberhasilan	Keterangan

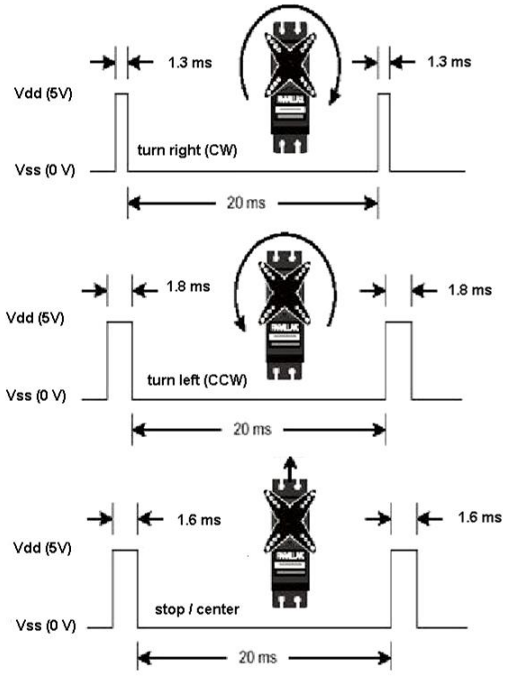
1	0b00000000	0b0001	0b0001	5	100 %	Berhasil
2	0b00000000	0b0010	0b0010	10	100 %	Berhasil
3	0b00000000	0b0011	0b0011	15	100 %	Berhasil
4	0b00000000	0b0100	0b0100	20	100 %	Berhasil
5	0b00000000	0b0101	0b0101	25	100 %	Berhasil
6	0b00000000	0b0110	0b0110	30	100 %	Berhasil
7	0b00000000	0b0111	0b0111	40	100 %	Berhasil
8	0b00000000	0b1000	0b1000	50	100 %	Berhasil
9	0b00000000	0b1001	0b1001	60	100 %	Berhasil
10	0b00000000	0b1010	0b1010	70	100 %	Berhasil
11	0b00000000	0b1011	0b1011	80	100 %	Berhasil
12	0b00000000	0b1100	0b1100	90	100 %	Berhasil
13	0b00000000	0b1101	0b1101	100	100 %	Berhasil
14	0b00000000	0b1110	0b1010	105	0 %	Gagal
15	0b00000000	0b1111	0b0011	110	0 %	Gagal

Dari data table diatas dapat disimpulkan bahwa dalam kondisi normal tanpa adanya halangan jarak maksimal yang dapat dikirim oleh rangkaian transmitter adalah ±100 m. Semakin banyak halangan maka tingkat kesuksesan pengiriman data semakin berkurang. Halangan yang diujikan di dalam penelitian ini berupa dinding, pohon, ranting.

4.1.2 Hasil Percobaan Pengujian Motor Servo Continous (360^o)

Gambar 4.1 Motor servo continous GWS S35 STD

Secara umum untuk menggerakkan motor servo kita memberikan pensinyalan PWM selama 20 ms. Karena motor servo yang saya gunakan adalah motor servo continous, maka motor servo jenis ini tidak dapat diatur derajatnya hanya saja kita dapat mengatur kecepatan putaran serta arah putarannya saja yakni CW atau CCW.



Gambar 4.2 Pewaktuan sinyal PWM untuk motor servo continous

Untuk membuat pewaktuan PWM yang tepat, maka saya menggunakan fitur timer 0 pada mikrokontroler ATMEGA 8535. Untuk mendapatkan PWM selama 20 ms maka diperlukan beberapa perhitungan kerja timer 0 yakni :

- Clock frekuensi 11059200 KHz
- T (Periode) = 1/f -> 1/11059200 = $9 . 10^{-8}$ second
- Karena timer0 merupakan 8 bit,maka Overflow timer 0 yakni :
  $256 \times 9 . 10^{-8} = 23{,}148 . 10^{-6}$ second
- Untuk menghasilkan 20 ms, maka :
  Counter = waktu yang diinginkan / waktu 1 x overflow
  $C = 20. 10^{-3} / 23{,}05 . 10^{-6} = 867{,}388$ dibulatkan **867** counter

```
// Timer 0 overflow interrupt service routine
unsigned int x,m_1,m_2;
interrupt [TIM0_OVF] void timer0_ovf_isr(void)
{
// Place your code here
x++;
if(x==867)       { x=0; }
else
{
if  (x <= m_1)  { PORTB.1=1; }
else        { PORTB.1=0; }
if  (x <= m_2)  { PORTB.2=1; }
else          { PORTB.2=0; }
}
}
```

Program diatas yakni program untuk membuat timer selama 20 ms dengan menggunakan fitur timer 0 pada mikrokontroler. Sesuai dengan perhitungan untuk menghasilkan nilai waktu 20 ms maka diperlukan counter sebayak 864 kali.

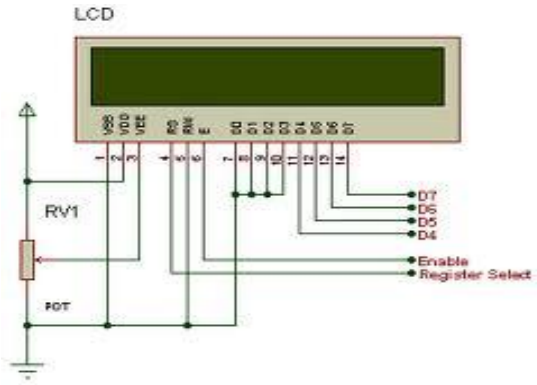
```
x++;
if(x==867)    { x=0; }
```

Untuk menggerakkan motor servo searah dengan jarum jam (CW) atau berlawanan arah jarum jam (CCW) maka pertama kali kita harus mengetahui berapa lama pewaktuan untuk servo berhenti ditengah. Dari percobaan yang dilakukan didapatkan yakni

Tabel 4.4 Pengujian perhitungan pwm servo serta nilai RPM yang dihasilkannya

No	PWM	Counter	Arah	RPM
1	0,9 ms	40	Turn Right (CW)	40,2
2	1,1 ms	50	Turn Right (CW)	38,2
3	1,38 ms	60	Turn Right (CW)	29
4	1,6 ms	70	Berhenti tengah	0
5	1,8 ms	80	Turn Left (CCW)	30,2
6	2,07 ms	90	Turn Left (CCW)	40,5
7	2,3 ms	100	Turn Left (CCW)	42,3

4.1.3 Hasil Percobaan Rangkaian Driver LCD 16 x 2



Gambar 4.3 Skema rangkaian driver LCD 16x2.

Pada rangkaian LCD yang digunakan telah bekerja sesuai dengan yang diharapkan, hal ini terbukti dengan munculnya karakter huruf dan angka yang ditulis pada program telah sesuai dengan yang dutampilkan pada LCD. Berikut ini merupakan contoh program LCD dengan menggunakan CodeVision AVR

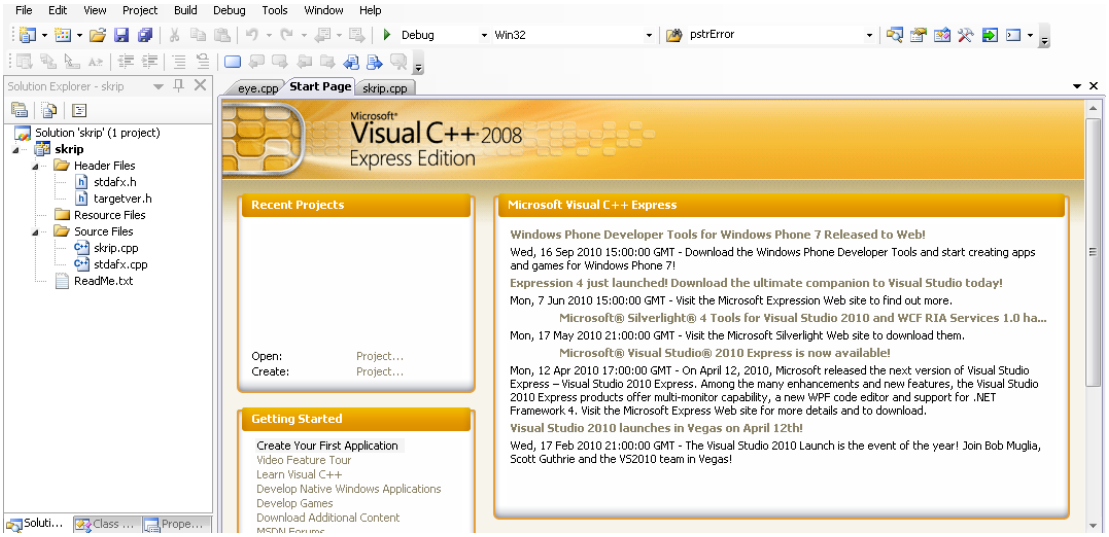
```
lcd_gotoxy(0,0);
putsf("Robot Eye Navi");
lcd_gotoxy(0,1);
putsf("Reda Anggra D");
delay_ms(500);
```

Program diatas akan menampilkan tulisan “Reda Anggra D”. Berikut ini contoh tampilan diatas.

Gambar 4.4 Tampilan program pada display LCD 16x2.

4.2 Hasil Percobaan Perangkat Lunak

Perangkat lunak yang digunakan pada skripsi ini terdiri atas dua bagian yakni, perangkat lunak untuk pengolahan citra dan perangkat lunak untuk hardware mikrokontroller. Untuk pengolahan citra menggunakan 2 buah software yakni *Microsoft Visual C++ 2008 Express Edition dan OpenCV 2.1.0. Microsoft Visual C++ 2008 Express Edition* adalah bagian dari *Visual Studio 2008* Express Edition yang berbasis C language. Kita dapat mengembangkan software dalam bentuk desktop didalamnya. Sedangkan *OpenCV 2.1.0* adalah compiler C++ yang digunakan untuk mengolah kode di dalamnya menjadi sebuah aplikasi berbasis *computer vision*. Dengan software ini kita dapat mengolah berbagai macam gambar dan video secara *real-time*. Untuk perangkat lunak mikrokontroller menggunakan *CodeVisionAVR C Compiler*



Gambar 4.5 Tampilan dinding kerja *Microsoft Visual C++ 2008 Ekspress Edition*

Untuk dapat menggunakan keduanya diperlukan beberapa pengaturan yakni :

1. Pada bagian **Configuration Properties** lalu masuk ke bagian **C/C++**, lalu pilih sub-bagian **General**. Pada properti **Additional Include Directories**nya klik dropdown menunya lalu pilih Edit.. lalu masukkandirektori **C:\OpenCV2.1\include\opencv**.
2. **Pada bagian Linker** lalu pilih sub-bagian **General**. Pada properti **Additional Include Directories**nya klik dropdown menunya lalu pilih Edit.. lalu masukkandirektori **C:\OpenCV2.1\lib**.
3. **Pada bagian Linker** lalu pilih sub-bagian **Input**. Pada properti **Additional Dependencies**nya tambahkan semua file *.lib yakni cv210d.lib, cvaux210d.lib, cxts210.lib, cxcore210d.lib, highgui210d.lib.

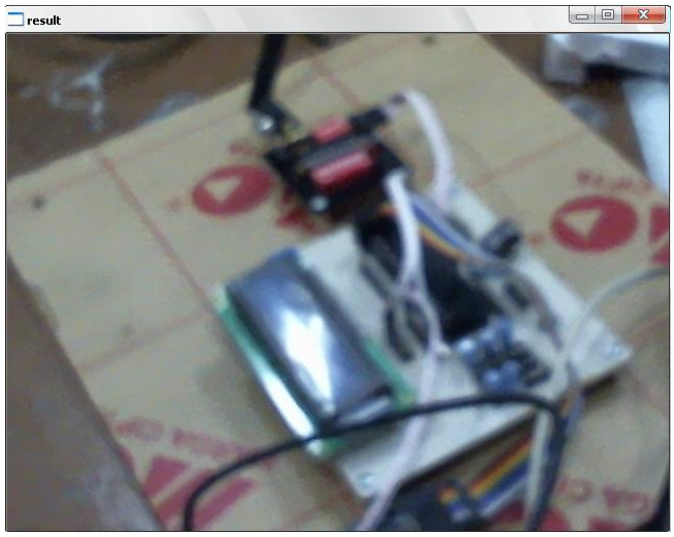
4.2.1 Pengujian Menampilkan Video Frame

Langkah untuk memunculkan video yang akan diolah, kita harus mendeteksi webcam dahulu. Berikut listing program untuk menampilkan frame video menggunakan *Microsoft Visual C++ 2008 Express Edition dan OpenCV 2.1.0.*

```
#include <stdio.h>
#include "cv.h"
#include "highgui.h"
int main( int argc, char **argv )
 {
   CvCapture *capture = 0;
    IplImage  *frame = 0;
    int       key = 0;

   /* initialize camera */
   capture = cvCaptureFromCAM( 0 );
```

```
  /* always check */
   if ( !capture ) {
      fprintf( stderr, "Cannot open initialize webcam!\n" );
      return 1;
   }

  /* create a window for the video */
 cvNamedWindow( "result", CV_WINDOW_AUTOSIZE );

        while( key != 'q' ) {
      /* get a frame */
      frame = cvQueryFrame( capture );

      /* always check */
      if( !frame ) break;

      /* display current frame */
      cvShowImage( "result", frame );

      /* exit if user press 'q' */
      key = cvWaitKey( 1 );
   }

   /* free memory */
   cvDestroyWindow( "result" );
   cvReleaseCapture( &capture );

   return 0;
}
```

Program diatas memerlukan include file header dari tool openCV yakni *#include "cv.h"* dan *#include "highgui.h"*. Fungsi utama dalam menagkap video dengan menggunakan openCV adalah *capture = cvCaptureFromCAM(0);*. Berikut ini hasil video frame yang dihasilkan.



Gambar 4.6 Contoh hasil frame video capture.

4.2.2 Pengujian Algoritma Deteksi Tepi Metode Canny

Deteksi tepi merupakan langkah pertama untuk melingkupi informasi di dalam citra. Tepi mencirikan batas-batas objek dan karena itu tepi berguna untuk proses segmentasi dan identifikasi objek di dalam citra. Tujuan operasi deteksi tepi adalah untuk meningkatkan penampakan garis batas suatu daerah atau objek di dalam citra (Munir, 2004). Secara garis besar, maka metode canny mempunyai tiga kemampuan dasar yaitu:

1. Rasio kesalahan yang kecil. Semua tepi harus ditemukan tanpa respon yang berlebihan. Tepi yang dideteksi harus berada sedekat mungkin dengan tepi sebenarnya.
2. Titik-titik tepi dilokalisasi dengan baik.

3. Respon pada titik tepi tunggal. Hal ini menunjukkan bahwa maksima local disekeliling tepi sebenarnya harus minimum. Sehingga tidak terdeteksi pixel tepi yang ganda jika hanya terdapat tepi tunggal.

Pada openCv metode deteksi tepi canny dapat dihasilkan dengan beberapa parameter yakni :

```
cvCanny(const CvArr* image, CvArr* edges, double threshold1, double threshold2, int aperture_size=3).
```

Dengan parameter sebagai berikut,

- *image* – Single-channel input image
- *edges* – Single-channel image to store the edges found by the function
- *threshold1* – The first threshold
- *threshold2* – The second threshold
- *aperture_size* – Aperture parameter for the Sobel operator

Berikut ini merupakan listing program untuk deteksi tepi menggunakan metode canny menggunakan OpenCV.

```
//----------------grayscale n edge detection-----------------
ori= (IplImage*)cvClone(frame);
        if (stage == STAGE_INIT)
        window = cvRect(0, 0, ori->width, frame->height);

        cvCvtColor(ori, gray, CV_BGR2GRAY);
        cvSmooth( gray, gray, CV_GAUSSIAN, 9, 9 );
// smooth it, otherwise a lot of false circles may be detected
        cvShowImage(wnd_nama,gray);


        edges = cvCreateImage(cvGetSize(frame), IPL_DEPTH_8U, 1);
```

```
tepi(gray,lowSliderPosition, highSliderPosition,3);
```

(a) (b)

Gambar 4.7 (a) Hasil deteksi tepi canny dengan *lowtrheshold* 10 *hightrheshold* 0. (b) Hasil deteksi tepi canny dengan *lowtrheshold* 50 *hightrheshold* 15.

Berdasarkan kriteria di atas, algoritme deteksi tepi Canny dilakukan dengan langkah-langkah sebagai berikut:

1. Pertama-tama dilakukan penghalusan (*smoothing*) citra untuk menghilangkan noise. Contohnya menggunakan filtering dengan Gaussian Filter.
2. Selanjutnya dicari *gradient magnitude* citra untuk melihat daerah-daerah yang memiliki turunan spasial yang tinggi.
3. Ditentukan arah dari tepi dengan menggunakan invers tangen dari *gradient magnitude* Y (Gy) dibagi gradient magnitude X (Gx). Arah yang diperoleh dari perhitungan ini kemudian dipetakan ke 0, 45, 90, atau 135 derajat berdasarkan kedekatannya dengan keempat derajat arah tadi.
4. Kemudian dilakukan **Non Maximum Suppression** yaitu, penghilangan nilai-nilai yang tidak maksimum. Ditelusuri daerah yang ditemukan pada langkah 2 (dengan arah seperti yang ditemukan pada langkah 3), dan menghilangkan (*suppress*) setiap piksel yang tidak maksimum.

5. Selanjutnya dilakukan **Hysteresis** yakni menggunakan dua *threshold* T1 (*threshold* bawah) dan T2 (*threshold* atas). Bila *magnitude* ada di bawah T1, titik tersebut di-set nol (dijadikan non-tepi). Bila *magnitude* ada di atas T2, maka termasuk tepi. Bila *magnitude* ada diantara T1 dan T2, di-set nol kecuali jika ada jalan (*path*) dari titik tersebut ke titik yang memiliki *magnitude* di atas T2.

4.2.3 Pengujian Algoritma Deteksi Bola Mata dengan metode Hough Transform

Untuk mendeteksi bola mata, ada beberapa tahapan yang harus dilakukan yakni:

4. Capture gambar dengan webcam
5. Mengubah ke gambar asli ke dalam citra grayscale
6. Memberikan filter Gaussian pada citra
7. Deteksi tepi dengan metode Canny
8. Scanning bola mata dengan fungi Hough Circle Transform
9. Kalibrasi nilai pixel pada saat bola mata ditengah → **p[0]**.
10. Untuk mendapatkan keputusan mata kanan, mata kiri maka ditentukan dengan pengurangan pixel sekarang dikurangi pixel posisi bola mata ditengah
    - Jika hasil dari **p[t] – p[0] > 20** maka **MATA KIRI**
    - Jika hasil dari **p[t] – p[0] > -20** maka **MATA KANAN**
    - Jika hasil dari **p[t] – p[tengah] >-20 dan < 20** maka **TENGAH**

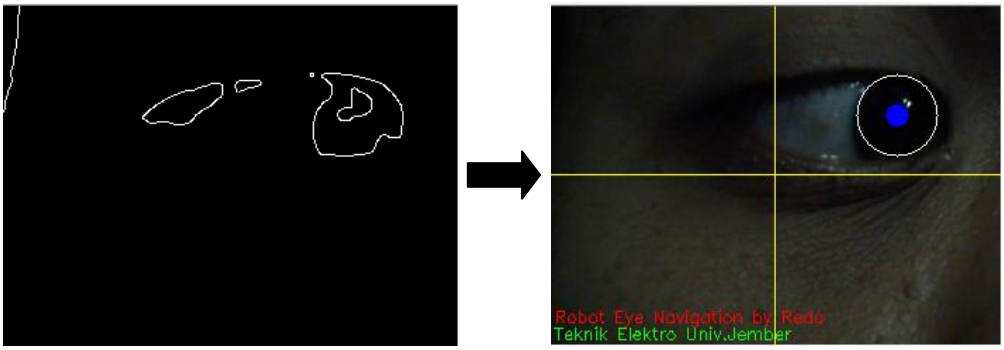
```
CvSeq* circles = cvHoughCircles(edges, storage, CV_HOUGH_GRADIENT,0.5, edges->height,SliderPosition,sliderposisi);

int i;
for( i = 0; i < circles->total; i++ )
{
```

```
p = (float*)cvGetSeqElem( circles,i );

cvCircle( frame, cvPoint(cvRound(p[0]),cvRound(p[1])), 8, CV_RGB(0,0,255), -1, 8,
0 );
cvCircle(                                                                    frame,
cvPoint(cvRound(p[0]),cvRound(p[1])),cvRound(p[2]),CV_RGB(255,255,255),1,8,0
);
}
```

```
                if (key == 's'){
                        tengah = p[0]; goto a;
                }
a:
                hasil = tengah - p[0];
                if (hasil > 20 )
                {printf("\r MATA KIRI \n");
                kirim(100);
                }
                else if (hasil < -20 ){
                printf("\r MATA KANAN \n");
                kirim(50);
                }
                else if (hasil> -20 && hasil <20)
                {printf("\r TENGAH \n");
          kirim(90);}
```

Deteksi tepi disini sangat berpengaruh terhadap keakuratan dalam menscanning posisi bola mata. Jika parameter-parameter deteksi tepi telah sesuai dengan yang diharapkan maka posisi bola mata akan dapat dibaca dengan baik.

Parameter-parameter yang dikatakan sesuai dengan harapan adalah jika hasil deteksi tepi tersebut telah dapat mendeteksi tepi bagian iris mata.



Gambar 4.8 Contoh hasil deteksi tepi dan posisi bola mata.

Parameter-paremeter yang di setting untuk mendapatkan posisi bola mata yang baik adalah batas threshold bawah dan batas threshold atas dari deteksi tepi. Berikut ini adalah data percobaan perubahan parameter tersebut dengan tingkat keberhasilannya.

4.2.3.1 Pengujian Pengaruh Perubahan Batas Threshold Bawah

Dalam pengujian ini bertujuan untuk mempengaruhi tingkat keberhasilan pendeteksian bola mata dengan menubah parameter *Low Threshold*

Tabel 4.5 Tingkat keberhasilan pendeteksian bola mata dengan perubahan batas bawah threshold dan batas atas 10

No	Batas Threshold Bawah	Batas Threshold Atas	Keterangan
1	2	10	x
2	4	10	x
3	6	10	x
4	8	10	√
5	10	10	√
6	12	10	√
7	14	10	√

8	16	10	√
9	18	10	√
10	20	10	√

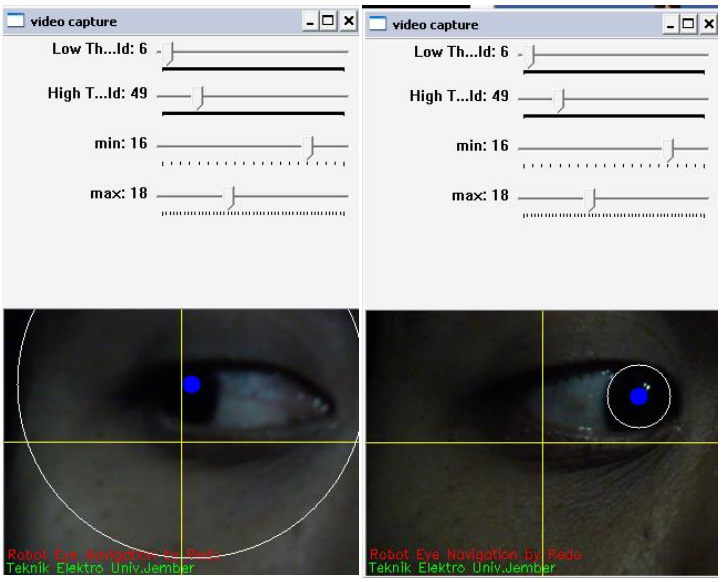
Tabel 4.6 Tingkat keberhasilan pendeteksian bola mata dengan perubahan batas bawah threshold dan batas atas 20

No	Batas Threshold Bawah	Batas Threshold Atas	Tingkat Keberhasilan
1	2	20	x
2	4	20	x
3	6	20	x
4	8	20	√
5	10	20	√
6	12	20	√
7	14	20	√
8	16	20	√
9	18	20	√
10	20	20	√

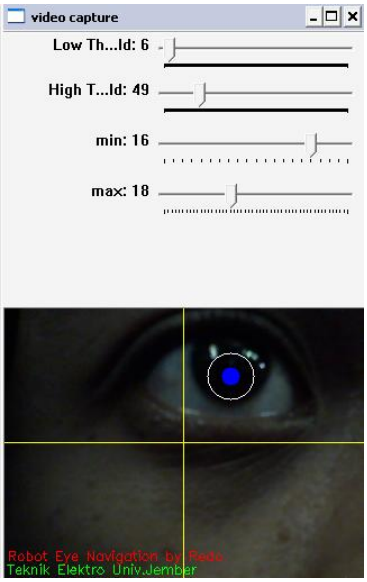
Tabel 4.7 Tingkat keberhasilan pendeteksian bola mata dengan perubahan batas bawah threshold dan batas atas 30

No	Batas Threshold Bawah	Batas Threshold Atas	Tingkat Keberhasilan
1	10	30	x
2	20	30	x
3	25	30	√
4	40	30	√
5	50	30	√
6	60	30	√

7	70	30	√
8	80	30	√
9	90	30	√
10	100	30	√

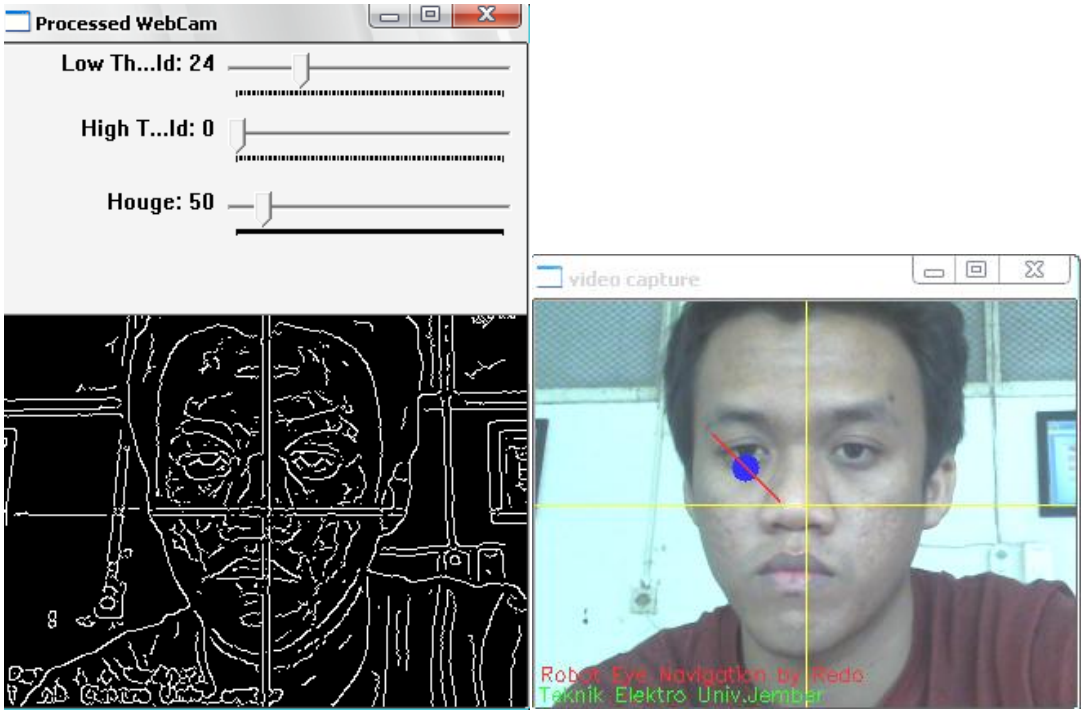


(a) (b)



(c)

Gambar 4.9 Contoh parameter deteksi bola (a) kiri (b) kanan (c) tengah



Gambar 4.10 Contoh parameter deteksi bola mata yang salah.

4.2.4 Pengujian Komunikasi Serial PC ke Mikrokontroller

4.2.4.1 Sisi Transmiter (Tx) dengan Microsoft Visual C++ 2008 Express Editiond

Komunikasi yang digunakan antara *Microsoft Visual C++ 2008 Express Editiond* dengan mikrokontroller yakni dengan menggunakan jenis komunikasi serial. Untuk dapat berkomunikasi serial dalam sistem operasi *Windows* digunakan fitur *Windows API*. *Windows API* merupakan singkatan dari *Windows Aplication Programming Interface* yakni sekumpulan antarmuka pemrograman aplikasi Interface yang dibuat oleh Microsoft . Berikut ini merupakan beberapa prosedur listing code komunikasi serial dengan WinAPI menggunakan C++.

1. Membuka serial port

```
HANDLE hSerial = CreateFile(L"COM2", GENERIC_WRITE, 0, 0, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, 0);

    if (hSerial == INVALID_HANDLE_VALUE) {
  if (GetLastError() == ERROR_FILE_NOT_FOUND)
    printf("COM port not found!\n");
  printf("Unknow error! - 1\n");
    }
```

Untuk membuka port serial kita harus mengetahui COM divais serial yang digunakan. Secara default COM serial yang digunakan pada PC adalah port serial COM1 yang berada pada port DB 9. Tetapi dalam penelitian ini menggunakan usb serial dengan nilai COM2.

2. Setting Parameter

```
DCB dcbSerialParams = {0};
  dcbSerialParams.DCBlength = sizeof(dcbSerialParams);

  if (!GetCommState(hSerial, &dcbSerialParams))
    printf("Unknow error! - 2\n");

  dcbSerialParams.BaudRate   = CBR_9600;
  dcbSerialParams.ByteSize   = 8;
  dcbSerialParams.StopBits   = ONESTOPBIT;
  dcbSerialParams.Parity     = NOPARITY;
  if(!SetCommState(hSerial, &dcbSerialParams))
    printf("Unknow error! - 3\n");
```

Parameter serial yang disetting adalah nilai baudrate, bytesize,stopbits dan parity. Parameter-parameter tersebut harus disesuaikan dengan parameter sisi

penerima / receiver. Dalam hal ini data akan dikirim serial ke dalam mikrokontroller 8535 dengan parameter yakni Communication Parameters 8 Data, 1 Stop, No Parity BaudRate 9600.

3. Setting Timeout

```
COMMTIMEOUTS timeouts               = {0};
timeouts.ReadIntervalTimeout        = 50;
timeouts.ReadTotalTimeoutConstant   = 50;
timeouts.ReadTotalTimeoutMultiplier = 10;
timeouts.WriteTotalTimeoutConstant  = 50;
timeouts.WriteTotalTimeoutMultiplier = 10;
if(!SetCommTimeouts(hSerial, &timeouts))
    printf("Timeout(s) error - 4\n");
```

Setting timeout ini diperlukan, karena jika salah satu problem pada komunikasi serial adalah jika dimana tidak ada data yang masuk kedalam port serial dalam kasus ini yakni *devise disconnect* atau *turn off*. Hal ini dapat menyebabkan program yang kita jalankan menjadi hang.

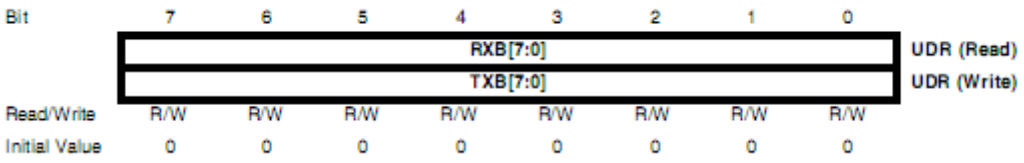
4. Pengiriman Data

```
char szBuff[12] = {100};
  char array[100];

  printf("Input string (max 100): ");
  gets(array);

  int len = strlen(array);
  int i, pos, delay;
```

```
  printf("Enter sleep/delay (ms): ");
  scanf("%d", &delay);

  for (pos = 0; ; pos++, pos %= len) {
     for (i = 0; i < 12; i++)
        szBuff[i + 1] = array[(pos + i) % len];

     DWORD dwBytesWrite = 0;
     if (!WriteFile(hSerial, szBuff,12, &dwBytesWrite, NULL))
        printf("Error writing to device - 5\n");

     printf("\r%d bytes written\r", dwBytesWrite);
     Sleep(delay);
  }

  CloseHandle(hSerial);

}
```

4.2.4.1 Sisi Receiver (Rx) Mikrokontroller AT Mega 8535

Pada sisi penerima data dari computer yakni menggunakan mikrokontroller AVR AT Mega 8535 dengan software compiler CodeVision AVR. Karena mikrokontroller ini digunakan sebagai komunikasi serial maka diperlukan beberapa hal yang harus diketahui dari register mikrokontroller AT Mega 8535 yakni UDR dan UCSRA

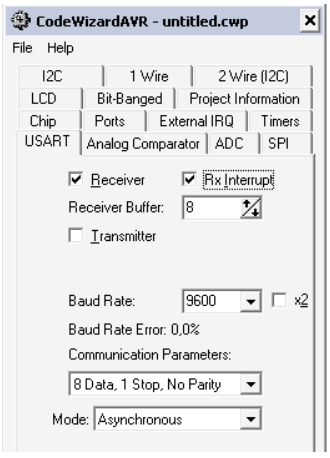


Bit	7	6	5	4	3	2	1	0	
	RXC	TXC	UDRE	FE	DOR	PE	U2X	MPCM	UCSRA
Read/Write	R	R/W	R	R	R	R	R/W	R/W	
Initial Value	0	0	1	0	0	0	0	0	

Gambar 4.11 Register UDR dan UCSRA pada mode USART AT Mega 8535

Register UDR adalah register penampung data dari mode USART. Register ini hanya dapat menampung 8 bit data yakni sebesar 255. Untuk penampungan data lebih dari 8 bit maka diperlukan teknik khusus untuk dapat menampungnya. Dalam alat yang saya gunakan hanya menampung 8 bit data saja. Pada register UDR terdapat dua buah jenis register yakni RXB untuk *read* dan TXB untuk *write* . Selain register UDR yang perlu diperhatikan pada mode USART yakni register UCSRA. Register UCSRA ini merupakan register control dari USART dimana terdapat RXC untuk mengetahui bahwa data receive sudah terpenuhi, TXC untuk mengetahui bahwa data transmit sudah terpenuhi serta UDRE untuk mengindikasikan bahwa mikrokontroller siap untuk menerima data serial baru.

Pada compiler mikrokontroller menggunakan CodeVision AVR diperlukan beberapa settingan mode USART yakni,



Gambar 4.12 Setting mode USART pada CodeVisionAVR

Karena difungsikan sebagai receiver maka hanya diaktifkan mode receiver saja dengan baudrate 9600 dan parameter komunikasi yakni 8 Data, 1 Stop, No Parity. Berikut ini merupakan penulisan register USART pada CodeVision hasil generator menunya.

```
// USART initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART Receiver: On
// USART Transmitter: Off
// USART Mode: Asynchronous
// USART Baud rate: 9600
UCSRA=0x00;
UCSRB=0x90;
UCSRC=0x86;
UBRRH=0x00;
        UBRRL=0x47;
```

Dan untuk code program untuk bisa mendapatkan data serial dari computer yakni :

```
// USART Receiver interrupt service routine
interrupt [USART_RXC] void usart_rx_isr(void)
{
char status,data;
status=UCSRA;
data=UDR;

if ((status & (FRAMING_ERROR | PARITY_ERROR | DATA_OVERRUN))==0)
  {
```

```
  rx_buffer[rx_wr_index]=data;
  receiv= rx_buffer[0];

  if ((++rx_wr_index == RX_BUFFER_SIZE)) rx_wr_index=0;
  if (++rx_counter == RX_BUFFER_SIZE)
     {
     rx_counter=0;
     rx_buffer_overflow=1;
     };
  };
}
```

Pada USART ini menggunakan mode *Rx Interrupt* karena dibutuhkan respon komunikasi yang sangat cepat agar tidak terjadi delay waktu transfer data. Sehingga jika ada perubahan data pada register UDR maka program secara otomatis menampungnya kedalam variable *receive* untuk segera diolah. Berikut ini merupakan tabel percobaan pengiriman data dari computer (*Ms Visual C++ Ekspress*) ke dalam mikrokontroler AT Mega 8535.

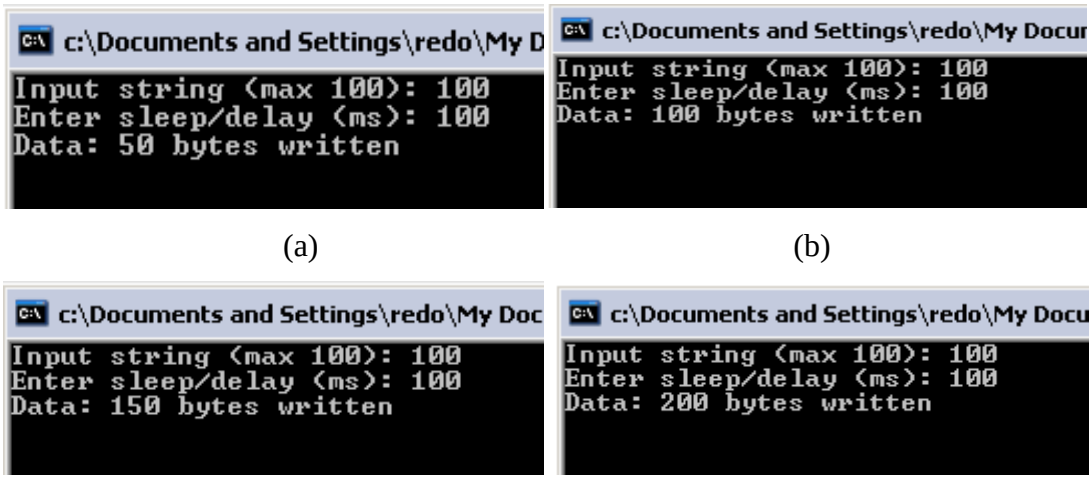
Tabel 4.8 Percobaan pengiriman data serial dengan *Microsoft Visual C++*

No	Data Tx	Data Rx	Keterangan
1	50	50	Berhasil
2	100	100	Berhasil
3	150	150	Berhasil
4	200	200	Berhasil

Untuk mengetahui prosentase keberhasilan pengiriman data maka diperlukan pengujian pengiriman data sebanyak 5 kali pada tiap-tiap data dan berikut ini mrupakan tabel prosentasi tingkat keberhasilan.

Tabel 4.9 Prosentasi keberhasilan dengan pengulangan 5 kali tiap percobaan

No	Data Tx	Data Rx	Pengiriman ke -					Keterangan
			1	2	3	4	5	
1	50	50	√	√	√	√	√	100%
2	100	100	√	√	√	√	√	100%
3	150	150	√	√	√	√	√	100%
4	200	200	√	√	√	√	√	100%

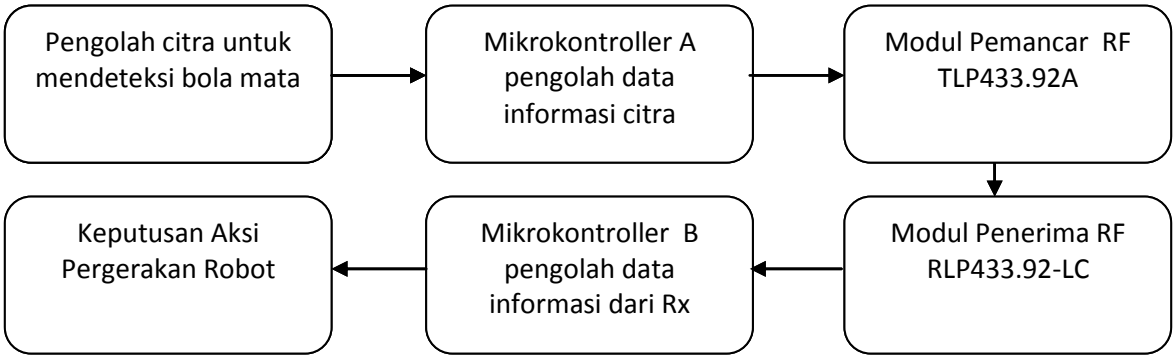


Gambar 4.13 Pengiriman data serial (a)data 50 (b)data 100 (c) data 150 (d) data 200.

Gambar 4.14 Tampilan LCD pada mikrokontroller (a) data 50 (b)data 100 (c) data 150 (d) data 200.

4.3 Pengujian Percobaan Sistem Secara Keseluruhan

Secara keseluruhan alur kerja sitem dimulai dari pengolahan citra mata untuk mendeteksi posisi bola mata yang kemudian hasilnya dikirim ke dalam mikrokontroller A melalui komunikasi serial. Hasil posisi mata tersebut dikodekan untuk dapat dikirim ke mikrokontroller B yang juga tertanam pada robot melalui wireless sebagai keputusan navigasi robot..

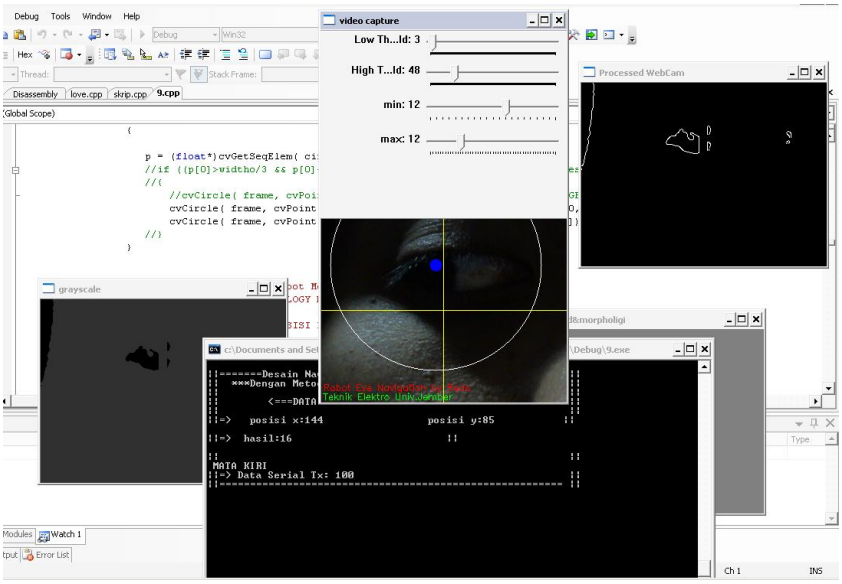


Gambar 4.15 Skema Alur Kerja Sistem Keseluruhan

Untuk mengetahui tingkat ketelitian serta keberhasilan sistem navigasi robot berdasarkan pergerakan mata ini perlu dilakukan percobaan sistem secara keseluruhan, yakni dengan menggabungkan antara hardware dan software. Ada 4 bagian penting yang harus berjalan dengan baik agar sistem keseluruhan dapat berjalan dengan baik yakni software pengolahan citra untuk mendeteksi posisi bola mata, wireless sisi pemancar data (Tx), wireless sisi penerima data (Rx), serta Robot.

Untuk dapat menjalan sistem ini secara keseluruhan pastikan keempat komponen tersebut berjalan dengan baik, maka dari itu sebelum melakukan percobaan dilakukan pengecekkan sistem secara keseluruhan serta pastikan kamera sebagai sensor mata terpasang dengan baik karena sistem pertama kali akan mendeteksi perangkat kamera. Berikut ini merupakan hasil percobaan sistem secara

keseluruhan dengan tingkat pencahayaan yang berbeda-beda yakni gelap, sedang, dan terang.



Gambar 4.16 Tampilan program saat dijalankan.

Tak dapat dipungkiri bahwa sebuah sistem pengolah citra sedikit banyak dipengaruhi oleh pengaruh luar dalam hal ini yakni pengaruh intensitas cahaya. Untuk menguji kehandalan sistem terhadap perubahan intensitas cahaya maka diperlukan pengujian sistem dalam beberapa kondisi intensitas cahaya sehingga didapatkan range intensitas cahaya yang dapat ditoleransi oleh sistem pengolah citra tersebut. Berikut ini merupakan table pengujian keberhasilan sistem pengolahan citra dalam mendeteksi bola mata pada intensitas yang diubah-ubah.

Tabel 4.10 Percobaan pengujian sistem terhadap perubahan intensitas cahaya

No.	Intensitas Cahaya	Keterangan
1	<10 lumen	Tak terdeteksi
2	10 – 50 lumen	Terdeteksi
3	51 – 80 lumen	Terdeteksi
4	81 – 100 lumen	Terdeteksi
5	101 – 130 lumen	Terdeteksi

6	131 – 160 lumen	Terdeteksi
7	>160	Tak terdeteksi

Tabel 4.11 Percobaan sistem keseluruhan dengan pencahayaan gelap dengan intensitas cahaya 12,5 lumen

No.	Perubahan Pergerakan Posisi Bola Mata	Aksi Robot	Keterangan
1	Kanan ke kiri	Belok Kiri	Berhasil
2	Kanan ke Tengah	Maju	Berhasil
3	Kiri ke Tengah	Maju	Berhasil
4	Kiri ke Kanan	Belok Kanan	Berhasil
5	Tengah ke Kanan	Belok Kanan	Berhasil
6	Tengah ke Kiri	Belok Kiri	Berhasil

Tabel 4.12 Percobaan sistem keseluruhan dengan pencahayaan Sedang dengan intensitas cahaya 65,5 lumen

No.	Perubahan Pergerakan Posisi Bola Mata	Aksi Robot	Keterangan
1	Kanan ke kiri	Belok Kiri	Berhasil
2	Kanan ke Tengah	Maju	Berhasil
3	Kiri ke Tengah	Maju	Berhasil
4	Kiri ke Kanan	Belok Kanan	Berhasil
5	Tengah ke Kanan	Belok Kanan	Berhasil
6	Tengah ke Kiri	Belok Kiri	Berhasil

Tabel 4.13 Percobaan sistem keseluruhan dengan pencahayaan Terang dengan intensitas cahaya145,5 lumen

No.	Perubahan Pergerakan Posisi Bola Mata	Aksi Robot	Keterangan

1	Kanan ke kiri	Belok Kiri	Berhasil
2	Kanan ke Tengah	Maju	Berhasil
3	Kiri ke Tengah	Maju	Berhasil
4	Kiri ke Kanan	Belok Kanan	Berhasil
5	Tengah ke Kanan	Belok Kanan	Berhasil
6	Tengah ke Kiri	Belok Kiri	Berhasil

Secara keseluruhan sistem berjalan dengan baik, dalam hal ini robot dapat bergerak sesuai dengan pergerakan bola mata baik di kondisi cahaya gelap, sedang dan terang. Untuk mendapatkan prosentase tingkat keberhasilan maka dilakukan 5 kali perngulangan pada tiap-tiap percobaan pada sistem diatas. Berikut ini merupakan prosentase tingkat keberhasilan tiap-tiap percobaan pada sistem.

Tabel 4.14 Prosentase tingkat keberhasilan tiap-tiap percobaan pada sistem dengan 5 kali sampling pada kondisi pencahayaan gelap (12,5 lumen)

No.	**Perubahan Pergerakan Posisi Bola Mata**	**Aksi Robot**	**Pengujian ke -**					**Prosentasi tingkat keberhasilan**
			1	**2**	**3**	**4**	**5**	
1	Kanan ke kiri	Belok Kiri	√	√	√	**x**	√	80%
2	Kanan ke Tengah	Maju	√	**x**	√	√	√	80%
3	Kiri ke Tengah	Maju	√	√	√	**x**	√	80%
4	Kiri ke Kanan	Belok Kanan	√	√	√	√	√	100%
5	Tengah ke Kanan	Belok Kanan	√	√	√	√	√	100%
6	Tengah ke Kiri	Belok Kiri	√	√	√	√	√	100%

Tabel 4.15 Prosentase tingkat keberhasilan tiap-tiap percobaan pada sistem dengan 5 kali sampling pada kondisi pencahayaan sedang (65,5 lumen)

No.	**Perubahan Pergerakan Posisi**	**Aksi Robot**	**Pengujian ke -**	**Prosentasi tingkat**

	Bola Mata						keberhasilan
			1	2	3	4	5	
1	Kanan ke kiri	Belok Kiri	√	√	√	√	√	100%
2	Kanan ke Tengah	Maju	√	√	√	√	√	100%
3	Kiri ke Tengah	Maju	√	√	√	√	√	100%
4	Kiri ke Kanan	Belok Kanan	√	√	√	√	√	100%
5	Tengah ke Kanan	Belok Kanan	√	√	√	√	√	100%
6	Tengah ke Kiri	Belok Kiri	√	√	√	√	√	100%

Tabel 4.16 Prosentase tingkat keberhasilan tiap-tiap percobaan pada sistem dengan 5 kali sampling pada kondisi pencahayaan terang (145,5 lumen)

No.	Perubahan Pergerakan Posisi Bola Mata	Aksi Robot	Pengujian ke -					Prosentasi tingkat keberhasilan
			1	2	3	4	5	
1	Kanan ke kiri	Belok Kiri	√	√	√	√	√	100%
2	Kanan ke Tengah	Maju	√	√	√	**x**	√	80%
3	Kiri ke Tengah	Maju	√	√	**x**	√	√	80%
4	Kiri ke Kanan	Belok Kanan	√	√	√	√	√	100%
5	Tengah ke Kanan	Belok Kanan	√	√	√	√	√	100%
6	Tengah ke Kiri	Belok Kiri	√	√	√	√	√	100%

Dari tabel percobaan diatas diketahui bahwa sistem dapat berjalan dengan baik pada pencahayaan sedang dengan tingkat keberhasilan tiap-tiap pergerakan yakni 100 %. Pada pencahayaan yang gelap tingkat keberhasilan deteksi bola mata tidak begitu bagus dikarenakan algoritma deteksi tepi tidak bekerja dengan baik sehingga tidak ada bentuk mata yang terdeteksi.

4.3.1 Pengujian delay waktu kerja sistem

Pengujian ini merupakan pengujian delay waktu saat perubahan pergerakan mata hingga terjadi aksi pada robot aktual. Dengan menggunakan timer stopwatch dilakukan percobaan sebanyak 5 kali pada tiap-tiap perubahan kondisi pergerakan. Berikut ini merupakan tabel hasip pengujian tersebut.

Tabel 4.17 Data tabel pengujian delay waktu system

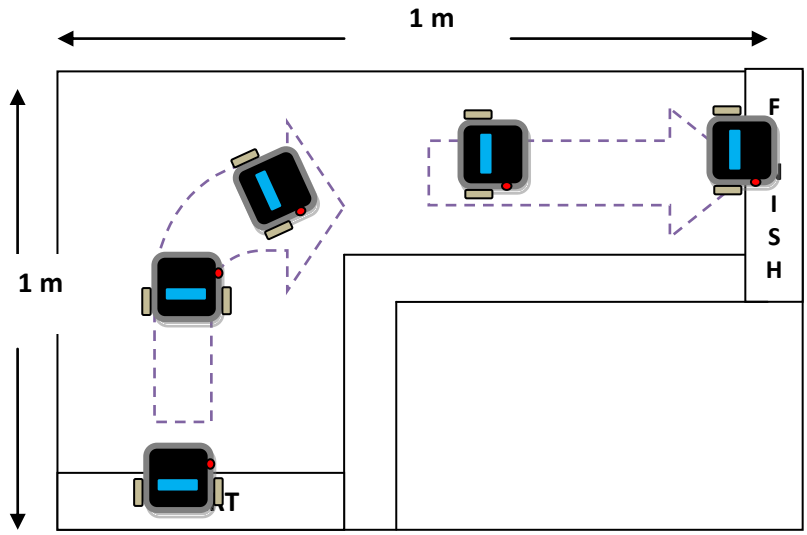
No.	**Perubahan Pergerakan Posisi Bola Mata**	**Aksi Robot**	**Pengujian ke – (second)**					**Rata – Rata Waktu (second)**
			1	**2**	**3**	**4**	**5**	
1	Kanan ke kiri	Belok Kiri	1	1	1	1.3	1	1.06
2	Kanan ke Tengah	Maju	1	0.8	1	0.9	1	0.94
3	Kiri ke Tengah	Maju	1	0.8	1	0.9	1	0.94
4	Kiri ke Kanan	Belok Kanan	1.5	1	1	1.2	1	1.14
5	Tengah ke Kanan	Belok Kanan	1.2	0.9	1	1	1	1.02
6	Tengah ke Kiri	Belok Kiri	1	1	1	0.8	0.9	0.94

Dari data tabel diatas dapat dilihat bahwa rata-rata waktu yang dihasilkan yakni sebesar 1 detik bahkan ad yang 1,14 detik. Ada beberapa factor yang mempengaruhi kecepatan transfer rate system ini memiliki delay waktu yakni algoritma program pada pengolahan cita yang mungkin kurang efisien, pencahayaan yang kurang pas sehingga menyebabkan kurang teliti dalam mendeteksi posisi bola mata serta delay waktu kerja yang dibutuhkan pada mikrokontroller untuk mengolah sebuah masukan perintah.

4.3.2 Pengujian pada lintasan berbentuk L tanpa halangan benda

Untuk mngetahui tingkat keberhasilan sistem maka diperlukan pengujian pada sebuah lapangan uji coba yakni berbentuk L dengan spesifikasi sesuai dengan gambar 4.19. Pada pengujian lapangan berbentuk L dengan tanpa halangan, dilakukan percobaan sebanyak 5 kali percobaan. Dari 5 kali percobaan tersebut memiliki

variasi waktu tempuh yang berbeda-beda dan dilakukan pada intensitas cahaya sebesar 110,5 lumen.



Gambar 4.17 Desain lapangan berbentuk L

Tabel 4.18 Tabel percobaan pada lapangan berbentuk L tanpa halangan

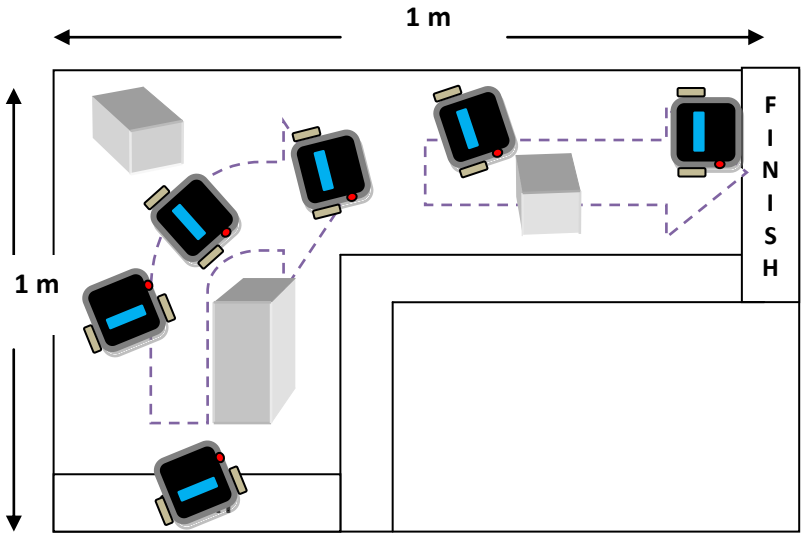
No	Percobaan ke -	Waktu	Keterangan
1	1	28 second	Berhasil
2	2	25 second	Berhasil
3	3	24 second	Berhasil
4	4	25 second	Berhasil
5	5	23 second	Berhasil
Rata - rata		**25 second**	**100 %**

Pada percobaan diatas dapat dilihat bahwa sistem berjalan dengan baik karena tidak ada kegagalan dalam percobaan . Rata –rata waktu yang diperlukan untuk melintasi lapangan uji coba tersebut adalah 25 second. Prosentase keberhasilan robot dalam bernavigasi pada lintasan tersebut yakni

- % keberhasilan = 5/5 * 100 % → **80 %**

- Rata –rata waktu tempuh = (28 + 25 +24 +25 +23) / 4 → **25**

4.3.3 Pengujian pada lintasan berbentuk L dengan halangan balok



Gambar 4.18 Desain lapangan berbentuk L dengan halangan balok

Pada pengujian lapangan berbentuk L dengan halangan, dilakukan percobaan sebanyak 5 kali percobaan. Untuk percobaan dengan halangan ini menggunakan 3 buah halangan berbentuk balok. Penempatan posisi halangan balok dapat dilihat pada gambar 4.20. Pada percobaan ketiga mengalami kegagalan hal ini dikarenakan sulitnya penyesuaian pergerakan mata dengan robot yang sering kali membentur halangan maupun dinding.

Tabel 4.19 Tabel percobaan pada lapangan berbentuk L dengan halangan

No	Percobaan ke -	Waktu	Keterangan
1	1	29 second	Berhasil
2	2	28 second	Berhasil
3	3	--	Gagal
4	4	31 second	Berhasil
5	5	29 second	Berhasil

Rata - rata	**29,25 second**	**80 %**

. Dari table diatas dapat disimpulkan bahwa dengan percobaan lapangan berbentuk L dengan halangan 3 buah balok robot dapat menempuh dengan rata-rata waktu sebesar 29,25 second. Dan prosentase ketercapaian finish sebesar 80%. Untuk pengujian dengan halangan dan tanpa halangan tidak terpaut waktu yang besar karena sistem transmisi wireless berjalan dengan baik dan cepat.

BAB 5. PENUTUP

5.1 Kesimpulan

Dari hasil penelitian yang telah dilakukan, maka dapat diambil kesimpulan sebagai berikut :

1. Secara keseluruhan system dapat berjalan dengan baik sesuai algoritma yang dirancang.
2. Komunikasi secara wireless menggunakan modul TLP 433.92A dan RLP 433.92-LC berjalan dengan baik dengan maksimum range jarak pembacaan yakni 100 meter tanpa halangan dan 25 meter dengan halangan tembok .
3. Kombinasi antara deteksi tepi canny dengan Hough Transform terbukti dapat mendeteksi bola mata dengan optimal (**lihat tabel 4.13 – 4.15**).
4. Delay waktu kerja system membutuhkan waktu rata-rata sebesar 1 detik (**lihat tabel 4.17**)
5. Pada uji lapangan berbentuk L robot dapat menempuhnya dengan rata-rata waktu tempuh sebesar 25 second sedangkan pada lapangan berbentuk L dengan halangan rata-rata waktu tempuh sebesar 29,25 second.
6. Sistem ini dapat berjalan optimal pada batas range intensitas pencahayaan 10 – 150 lumen.

5.2 Saran

1. Sistem ini mempunyai kelemahan pada pengaruh cahaya luar yang masuk, dikarenakan dapat merubah warna yang dedeteksi pada kamera webcam, karena system tidak akan bekerja jika kondisi pencahayaan yang sangat gelap.
2. Untuk kedepan diharapkan ada pengembangan agar sistem ini tidak lagi menggunakan helm.
3. Kualitas kamera webcam sangat menentukan hasil dari *Image* yang diperoleh, sehingga deteksi tepi yang diperoleh untuk diproses scanning bola mata bisa lebih smooth.

4. Penambahan GUI pada software image processing agar lebih memudahkan pengguna (*user friendly*) dan lebih menarik.

DAFTAR PUSTAKA

[1] Sankar, Amita. 2001. *Pattern Recognition From Classical to Modern*. Singapura: World Scientific Peblishing.

[2] Ahmad, Usman. 2005. *Pengolahan Citra Digital dan Teknik Pemrogramannya*. Yogyakarta : Graha Ilmu.

[3] Gunawan, Budi. 2003. *Deteksi Isyarat Tangan Oleh Komputer Dengan Digital Image Processing.* ISSN : 1979 - 6870

[4] R.L.Hsu and M.A.Mottaleb. *Face Detection in Color Image*, Appear In IEEE Tans. PAMI, Vol. 24, no.5, pp.696-706, may 2002.

[5] Puspita,Eru. Syamsiar, Trendy. Nur,Budi. 2006. *Sistem Identifikasi Scan Iris Mata Menggunakan Metode JST Propagansi Balik Untuk Aplikai* . Surabaya: PENS ITS.

[6] Murni, Aniati., *Diktat Kuliah : Pengolahan Citra Digital*, Universitas Indonesia, Jakarta, 2004.

[7] Mulyono, Dwi. 2011. *Pembuatan Mobile Robot Gripper Sebagai Pemindah Barang Otomatis Menggunakan Metode Fuzzy Logic*. Jember: Universitas Jember

[8]http://www.nashruddin.com/Real_Time_Eye_Tracking_and_Blink_Detection#template-creation : diakses pada tanggal 20 November 2011 : pada pukul 16.35 WIB.

LAMPIRAN

1. **Program Deteksi Bola Mata pada Microoft Visual C++**

```
// skripsi.cpp : Defines the entry point for the console
application.
#include "stdafx.h"
#include <tchar.h>
#include "cv.h"
#include "highgui.h"
#include "cxcore.h"
#include <iostream>
#include <stdlib.h>
#include <windows.h>
#include <stdio.h>
#include <string.h>
#include <time.h>

void serial(int data);
HANDLE hSerial;

#define FRAME_WIDTH           300
#define FRAME_HEIGHT          300
#define TPL_WIDTH             16
#define TPL_HEIGHT            16
#define WIN_WIDTH             TPL_WIDTH * 2
#define WIN_HEIGHT            TPL_HEIGHT * 2
#define TM_THRESHOLD          0.4
#define STAGE_INIT            1
#define STAGE_TRACKING        2

#define POINT_TL(r)           cvPoint(r.x, r.y)
#define POINT_BR(r)           cvPoint(r.x + r.width, r.y + r.height)
#define POINTS(r)             POINT_TL(r), POINT_BR(r)

#define PORTNUM 1
#define BAUDRATE 9600

#define DRAW_RECTS(f, d, rw, ro)
            \
do {
                        \
      cvRectangle(f, POINTS(rw), CV_RGB(255, 0, 0), 1, 8, 0);
      \
      cvRectangle(f, POINTS(ro), CV_RGB(0, 255, 0), 1, 8, 0);
      \
      cvRectangle(d, POINTS(rw), cvScalarAll(255),  1, 8, 0);
      \
      cvRectangle(d, POINTS(ro), cvScalarAll(255),  1, 8, 0);
      \
} while(0)

#define     DRAW_TEXT(f, t, d, use_bg)
                \
```

```
if (d)
                                \
{
                                \
      CvSize _size;
                          \
      cvGetTextSize(t, &font, &_size, NULL);
            \
      if (use_bg)
                          \
      {
                                \
            cvRectangle(f, cvPoint(0, f->height),
            \
                                cvPoint(_size.width + 5,
                  \
                                          f->height  -  _size.height  *
2),         \
                                CV_RGB(255, 100, 0), CV_FILLED, 8, 0);
            \
      }
                                \
      cvPutText(f, t, cvPoint(2, f->height - _size.height / 2),   \
                     &font, CV_RGB(255,255,0));
            \
      d--;
                          \
}

CvCapture*        capture;
IplImage*         frame, * gray, * prev, * diff, * tpl, *croppedka,
*crop ,*croppedki, *img, *eyes ,*origImg,*outImg,*ori;

IplImage* edges;

 IplImage* img_a;
CvMemStorage*     storage;
IplConvKernel*    kernel;
CvFont                  font;
CvPoint (pt1);
CvPoint (pt2);
CvPoint (pt3);
CvPoint (pt4);

char*             wnd_name    = "video capture";
char*             wnd_eye           = "face&eye detection";
char*             wnd_debug   = "treshold&morpholigi";
char*             wnd_nama    = "grayscale";
char*             wnd_prev    = "prev";
char*             wnd_cropka  = "croppedka";
char*             wnd_cropki  = "croppedki";
char*             crop_eye          = "crop_eye";
char*             wnd_crp           = "crop";
char*             wnd_crp1          = "crop";
```

```
int   get_connected_components(IplImage* img, IplImage* prev, CvRect
window, CvSeq** comp);

bool keepAspectRatio;
int i,scale;
int heighthsv,widthhsv,stephsv,channelshsv;
int heighthsv1,widthhsv1,stephsv1,channelshsv1;

int x,y,widtho,heighto,stepo,channelso;


uchar *datahsv,*datahsv1,*dataho ;
int         text_delay, stage = STAGE_INIT;
CvSeq*      comp = 0;
CvRect      window, eye;

char* kata  = "Robot Eye Navigation by Redo";
char* kata1 = "Teknik Elektro Univ.Jember";
int newWidth, newHeight;

void detectEyes(IplImage *eyes);
IplImage*  resizeImage(const  IplImage  *origImg,  int  newWidth,int
newHeight, bool keepAspectRatio);

void delay_frames(int nframes);
void init();

void exit_nicely(char* msg);
      int         key, nc, found;
int ka0,ka1,ki0,ki1;
int min,max;
float* p;
CvHaarClassifierCascade *cascade f;
CvHaarClassifierCascade *cascade_e;

int maxLowThreshold = 255;
int lowSliderPosition =11;

// Slider for the high threshold value of our edge detection
int  maxHighThreshold = 255;
int  highSliderPosition = 100;
int  maxSliderPosition=50;
int  minSliderPosition=20;
int  SliderPosition=23;
int  sliderposisi=18;
      int a,b, tengah, hasil;
void onLowThresholdSlide(int theSliderValue)
{
      lowSliderPosition = theSliderValue;
}

// Callback function to adjust the high threshold on slider movement
void onHighThresholdSlide(int theSliderValue)
{
```

```
      highSliderPosition = theSliderValue;
}

void     tepi(IplImage     *gray,double     thelowThreshold,     double
theHighThreshold,double theAperture){
      cvCanny(   gray,   edges,   thelowThreshold,   theHighThreshold,
theAperture);
}

void onMinSlide(int theSliderValue)
{
      SliderPosition = theSliderValue;
}

void onMaxSlide(int theSliderValue)
{
      sliderposisi = theSliderValue;
}


void kirim(char data){

 int n=12;
      char szBuff1[100]={data};
    DWORD dwByteswrote = 0;
      printf("||=>         Data         Serial         Tx:         %d
||\r\n",data);
    if(!WriteFile(hSerial, szBuff1, n, &dwByteswrote, NULL))
    {
            printf("||=>   Status   serial   :   error   writing   data
||\r\n");
    }
      return   ;
}
int main(int argc, char** argv)
{
    hSerial    =    CreateFile(L"COM2",    GENERIC_WRITE,    0,    0,
OPEN EXISTING, FILE ATTRIBUTE NORMAL, 0);
    if (hSerial == INVALID_HANDLE_VALUE) {
        if (GetLastError() == ERROR_FILE_NOT_FOUND)
            printf("COM port not found!\n");
            printf("Unknow error! - 1\n");
    }

    DCB dcbSerialParams = {0};
    dcbSerialParams.DCBlength = sizeof(dcbSerialParams);

    if (!GetCommState(hSerial, &dcbSerialParams))
        printf("Unknow error! - 2\n");

    dcbSerialParams.BaudRate   = CBR_9600;
    dcbSerialParams.ByteSize   = 8;
    dcbSerialParams.StopBits   = ONESTOPBIT;
    dcbSerialParams.Parity     = NOPARITY;
```

```
    if(!SetCommState(hSerial, &dcbSerialParams))
        printf("Unknow error! - 3\n");

    COMMTIMEOUTS timeouts                 = {0};
    timeouts.ReadIntervalTimeout          = 50;
    timeouts.ReadTotalTimeoutConstant     = 50;
    timeouts.ReadTotalTimeoutMultiplier   = 10;
    timeouts.WriteTotalTimeoutConstant    = 50;
    timeouts.WriteTotalTimeoutMultiplier  = 10;
    if(!SetCommTimeouts(hSerial, &timeouts))
       printf("Timeout(s) error - 4\n");

      init();

//============declarasi fungsi detect face &eye=================

      CvCapture *capture;

      //load image
      capture = cvCaptureFromCAM( 0 );

    cvNamedWindow("Processed WebCam", CV_WINDOW_AUTOSIZE);
      cvCreateTrackbar("Low  Threshold",wnd_name,  &lowSliderPosition,
maxLowThreshold, onLowThresholdSlide);

      // Create the high threshold slider
      cvCreateTrackbar("High          Threshold",          wnd_name,
&highSliderPosition, maxHighThreshold, onHighThresholdSlide);
      cvCreateTrackbar("min",wnd_name,&SliderPosition,
minSliderPosition, onMinSlide);
      cvCreateTrackbar("max",                wnd_name,&sliderposisi,
maxSliderPosition, onMaxSlide);
      while (key != 'a')

      {
            frame = cvQueryFrame(capture);
            if (!frame)
                  exit nicely("cannot query frame!");
                frame->origin = 0;

      //-------------buat garis x,y----------------------
                  pt1.x = (frame->width)/2;
                  pt1.y = 0;
                  pt2.x =(frame->width)/2;
                  pt2.y = frame->height;
                  pt3.x = 0;
                  pt3.y = (frame->height)/2;
                  pt4.x = (frame->width);
                  pt4.y = (frame->height)/2;

                  cvLine(frame,pt1,pt2,CV_RGB(255,255,0),1,4,0);
                  cvLine(frame,pt3,pt4,CV_RGB(255,255,0),1,4,0);

                  cvFlip(frame,NULL,-1);
```

```
//==========tampil tulisan---------

      CvSize _size;
                        \
      cvGetTextSize(kata, &font, &_size, NULL);
      cvGetTextSize(kata1, &font, &_size, NULL);
      cvPutText(frame,    kata,    cvPoint(2,    frame->height-12    -
_size.height / 2),      \
                  &font, CV_RGB(255,0,0));
      cvPutText(frame,    kata1,    cvPoint(2,    frame->height    -
_size.height / 2),      \
                  &font, CV_RGB(0,255,0));



//----------------grayscale n edge detection-----------------
            ori= (IplImage*)cvClone(frame);
            if (stage == STAGE_INIT)
                  window = cvRect(0, 0, ori->width, frame->height);

            cvCvtColor(ori, gray, CV_RGB2GRAY);

            cvSmooth( gray, gray, CV_GAUSSIAN, 11, 11 ); // smooth
it, otherwise a lot of false circles may be detected


      cvThreshold(gray,gray,lowSliderPosition,highSliderPosition,CV_
THRESH_BINARY);

            edges   =   cvCreateImage(cvGetSize(frame),   IPL_DEPTH_8U,
1);

            tepi(gray,lowSliderPosition, highSliderPosition,3);
            cvShowImage(wnd_nama,gray);

            widtho = edges->width;
            heighto= edges->height;
            stepo = edges->widthStep;
            channelso = edges->nChannels;

      // cvDilate(gray,gray,0,10);

            CvSeq*    circles    =    cvHoughCircles(edges,    storage,
CV_HOUGH_GRADIENT,0.5, edges->height,SliderPosition,sliderposisi);


                        int i;
                         for( i = 0; i < circles->total; i++ )
                         {

                              p = (float*)cvGetSeqElem( circles,i );
```

```
                                    cvCircle(                   frame,
cvPoint(cvRound(p[0]),cvRound(p[1])),  8,  CV_RGB(0,0,255),  -1,  8,  0
);
                                    cvCircle(                   frame,
cvPoint(cvRound(p[0]),cvRound(p[1])),cvRound(p[2]),CV_RGB(255,255,25
5),1,8,0);
                              //}
                         }


            printf("||=======Desain   Navigasi   Robot   Menggunakan
Mata========== ||\r\n");
            printf("||   ***Dengan  Metode  MORPHOLOGY  by  Reda  Anggra
D***        ||\r\n");
            printf("||
||\r\n");
            printf("||              <===DATA  NILAI  POSISI  PIXEL===>
||\r\n");
            printf("||
||\r\n");
            printf("||=>   posisi x:%d                posisi     y:%d
||\r\n\n",(int)p[0],(int)p[1]);
            printf("||=>  hasil:%d
||\r\n\n",hasil);
            printf("||
||\r\n");

//----------------kalibrasi posisi bola mata-----------
            if (key == 's'){
                  tengah = p[0]; goto a;
            }
a:
            hasil = tengah - p[0];
            if (hasil > 20 )
            {printf("\r MATA KIRI \n");
            kirim(100);
            }
            else if (hasil < -20 ){
            printf("\r MATA KANAN \n");
            kirim(50);
            }
            else if (hasil> -20 && hasil <20)
            {printf("\r TENGAH \n");
            kirim(90);}

printf("||=========================================================
||\r\n");

                  cvShowImage("Processed WebCam", edges);
                  cvShowImage(wnd_name, frame);

            int f;
            for (f=0;f<10;f++){
            printf("\n");
```

```
            }
            key  = cvWaitKey(15);
      }
      cvReleaseImage(&edges);
      exit_nicely(NULL);
}

int
get_connected_components(IplImage*  img,  IplImage*  prev,  CvRect
window, CvSeq** comp)
{
      IplImage* _diff;

      cvZero(diff);

      /* apply search window to images */
      cvSetImageROI(img, window);
      cvSetImageROI(prev, window);
      cvSetImageROI(diff, window);

      /* motion analysis */
      cvSub(img, prev, diff, NULL);
      cvThreshold(diff, diff, 0,255 , CV_THRESH_BINARY_INV);

      /* reset search window */
      cvResetImageROI(img);
      cvResetImageROI(prev);
      cvResetImageROI(diff);
      _diff = (IplImage*)cvClone(diff);

      /* get connected components */
      int     nc     =     cvFindContours(_diff,     storage,     comp,
sizeof(CvContour),CV RETR CCOMP,              CV CHAIN APPROX SIMPLE,
cvPoint(0,0));

      cvClearMemStorage(storage);
      //cvReleaseImage(&_diff);

      return nc;
}

void
delay_frames(int nframes)
{
      int i;

      for (i = 0; i < nframes; i++)
      {
            frame = cvQueryFrame(capture);
            if (!frame)
                  exit_nicely("cannot query frame");
            cvShowImage(wnd_name, frame);
            if (diff)
                  //cvShowImage(wnd_debug, diff);
```

```
            cvWaitKey(30);
      }
}

void
init()
{
      char* msg[] = { "Bismillahirrohmanirrohim",
                        "Reda Anggra Distira",
                        "071910201079",
                        "Fakultas Teknik UJ",
                        "Press 'q' to quit...",
                        "Enjoy it,,,," };
      int delay, i;

      capture = cvCaptureFromCAM(0);
      if (!capture)
            exit_nicely("Cannot initialize camera!");

      cvSetCaptureProperty(capture,          CV_CAP_PROP_FRAME_WIDTH,
FRAME_WIDTH);
      cvSetCaptureProperty(capture,         CV_CAP_PROP_FRAME_HEIGHT,
FRAME_HEIGHT);

      frame = cvQueryFrame(capture);
      cvFlip(frame,NULL,-1);
      if (!frame)
            exit_nicely("cannot query frame!");

      cvInitFont(&font, CV_FONT_HERSHEY_SIMPLEX, 0.4, 0.4, 0, 1, 8);
      cvNamedWindow(wnd_name, 1);

      for (delay = 30, i = 0; i < 6; i++, delay = 30)
            while (delay)
            {
                  frame = cvQueryFrame(capture);
                  if (!frame)
                        exit nicely("cannot query frame!");
                  DRAW_TEXT(frame, msg[i], delay, 0);
                  cvShowImage(wnd_name, frame);
                  cvWaitKey(30);
            }

      storage = cvCreateMemStorage(0);
      if (!storage)
            exit_nicely("cannot allocate memory storage!");

      kernel    =    cvCreateStructuringElementEx(3,    3,    1,    1,
CV_SHAPE_CROSS, NULL);
      gray   = cvCreateImage(cvGetSize(frame), 8, 1);
      prev   = cvCreateImage(cvGetSize(frame), 8, 1);
      diff   = cvCreateImage(cvGetSize(frame), 8, 1);
      croppedka   = cvCreateImage(cvGetSize(frame), 8, 1);
      croppedki   = cvCreateImage(cvGetSize(frame), 8, 1);
```

```
	eyes   = cvCreateImage(cvGetSize(frame), 8, 1);
	tpl      = cvCreateImage(cvSize(TPL_WIDTH, TPL_HEIGHT), 8, 1);

	if (!kernel || !gray || !prev || !diff || !tpl || !croppedka
|| !croppedki || !eyes)
		exit_nicely("system error.");

	gray->origin  = frame->origin;
	prev->origin  = frame->origin;
	diff->origin  = frame->origin;
	croppedka->origin  = frame->origin;
	croppedki->origin  = frame->origin;
	eyes->origin  = frame->origin;
	cvNamedWindow(wnd_debug, 1);
}

IplImage* cropImage(const IplImage *img, const CvRect region)
{

	IplImage *imageCropped;
	CvSize size;

	if (img->width <= 0 || img->height <= 0
		|| region.width <= 0 || region.height <= 0) {
		//cerr << "ERROR in cropImage(): invalid dimensions." <<
endl;
		exit(1);
	}

	if (img->depth != IPL_DEPTH_8U) {
		//cerr << "ERROR in cropImage(): image depth is not 8."
<< endl;
		exit(1);
	}

	// Set the desired region of interest.
	cvSetImageROI((IplImage*)img, region);
	// Copy region of interest into a new iplImage and return it.
	size.width = region.width;
	size.height = region.height;
	imageCropped   =   cvCreateImage(size,   IPL_DEPTH_8U,   img-
>nChannels);
	cvCopy(img, imageCropped);    // Copy just the region.

	return imageCropped;
}
IplImage*  resizeImage(const  IplImage  *origImg,  int  newWidth,int
newHeight, bool keepAspectRatio)
{
	//IplImage   resizeImage(const   IplImage   *origImg,   int
newWidth,int newHeight, bool keepAspectRatio)
	IplImage *outImg = 0;
	int origWidth;
	int origHeight;
```

```
	if (origImg) {
		origWidth = origImg->width;
		origHeight = origImg->height;
	}
	if (newWidth <= 0 || newHeight <= 0 || origImg == 0
		|| origWidth <= 0 || origHeight <= 0) {
		//cerr << "ERROR: Bad desired image size of " << newWidth
		//	<< "x" << newHeight << " in resizeImage().\n";
		exit(1);
	}

	if (keepAspectRatio) {
		// Resize the image without changing its aspect ratio,
		// by cropping off the edges and enlarging the middle section.
		CvRect r;
		// input aspect ratio
		float origAspect = (origWidth / (float)origHeight);
		// output aspect ratio
		float newAspect = (newWidth / (float)newHeight);
		// crop width to be origHeight * newAspect
		if (origAspect > newAspect) {
			int tw = (origHeight * newWidth) / newHeight;
			r = cvRect((origWidth - tw)/2, 0, tw, origHeight);
		}
		else {		// crop height to be origWidth / newAspect
			int th = (origWidth * newHeight) / newWidth;
			r = cvRect(0, (origHeight - th)/2, origWidth, th);
		}
		IplImage *croppedImg = cropImage(origImg, r);

		// Call this function again, with the new aspect ratio image.
		// Will do a scaled image resize with the correct aspect ratio.
		outImg = resizeImage(croppedImg, newWidth, newHeight, false);
		cvReleaseImage( &croppedImg );

	}
	else {

		// Scale the image to the new dimensions,
		// even if the aspect ratio will be changed.
		outImg = cvCreateImage(cvSize(newWidth, newHeight),origImg->depth, origImg->nChannels);
		if (newWidth > origImg->width && newHeight > origImg->height) {
			// Make the image larger
			cvResetImageROI((IplImage*)origImg);
			// CV_INTER_LINEAR: good at enlarging.
			// CV_INTER_CUBIC: good at enlarging.
```

```
                  cvResize(origImg, outImg, CV_INTER_LINEAR);
            }
            else {
                  // Make the image smaller
                  cvResetImageROI((IplImage*)origImg);
                  // CV_INTER_AREA: good at shrinking (decimation)
only.
                  cvResize(origImg, outImg, CV_INTER_AREA);
            }

      }
      return outImg;
}
void detectEyes(IplImage *eyes)
{
int i;

    //detect faces
      CvSeq *faces = cvHaarDetectObjects(
            eyes, cascade_f, storage,
            1.1, 3, 0, cvSize( 40, 40 ) );

    // return if not found
   if (faces->total == 0) return;

    // draw a rectangle
      CvRect *r = (CvRect*)cvGetSeqElem(faces, 0);
      cvRectangle(eyes,
                        cvPoint(r->x, r->y),
                        cvPoint(r->x + r->width, r->y + r->height),
                        CV_RGB(255, 0, 0), 1, 8, 0);

    // reset buffer for the next object detection
   cvClearMemStorage(storage);

    // Set the Region of Interest: estimate the eyes' position
  cvSetImageROI(eyes,  cvRect(r->x  +  (r->width/3),  r->y  +  (r-
>height/5), (r->width),( r->height/3)));

    // detect eyes
      CvSeq* eyes1 = cvHaarDetectObjects(
        eyes, cascade_e, storage,
            1.15, 3, 0, cvSize(15, 15));

    // draw a rectangle for each eye found
//    for( i = 0; i < (eyes1 ? eyes1->total : 0); i++ ) {

      if (eyes1 ?eyes1->total:0 ) {

      r = (CvRect*)cvGetSeqElem( eyes1,0 );
      //if (r->x > r->width/2){
            cvRectangle(eyes,
                              cvPoint(r->x, r->y),
```

```
                                cvPoint(r->x  +  r->width,  r->y  +  r-
>height),
                                CV_RGB(0, 255, 0), 1, 8, 0);

            IplImage* crop ;

            CvSize size;

      }

         cvResetImageROI(eyes);

}

void serial(int data){

      char szBuff[12] = {data};
      char array[100];


    int len = strlen(array);
    int i, pos;
      int delay=100;

   for (pos = 0; pos<100; pos++) {
        for (i = 0; i < 12; i++)
          szBuff[i + 1] = array[(pos + i) % 100];

        DWORD dwBytesWrite = 0;
        if (!WriteFile(hSerial, szBuff,12, &dwBytesWrite, NULL))
            printf("Error writing to device - 5\n");

       Sleep(delay);
    }

   CloseHandle(hSerial);
}

void
exit_nicely(char* msg)
{
      cvDestroyAllWindows();

      if (capture)
            cvReleaseCapture(&capture);
      if (gray)
            cvReleaseImage(&gray);
      if (prev)
            cvReleaseImage(&prev);
      if (diff)
            cvReleaseImage(&diff);
      if (croppedka)
            cvReleaseImage(&croppedka);
      if (croppedki)
```

```
            cvReleaseImage(&croppedki);
      if (tpl)
            cvReleaseImage(&tpl);
      if (storage)
            cvReleaseMemStorage(&storage);

      if (msg != NULL)
      {
            fprintf(stderr, msg);
            fprintf(stderr, "\n");
            exit(1);
      }

      exit(0);
}
```

2. Program Mikrokontroller pada sisi Transmitter

```
#include <mega8535.h>
#include <delay.h>
#include <stdio.h>
char buf[22];
// Alphanumeric LCD Module functions
#asm
  .equ __lcd_port=0x18 ;PORTB
#endasm
#include <lcd.h>

#define RXB8 1
#define TXB8 0
#define UPE 2
#define OVR 3
#define FE 4
#define UDRE 5
#define RXC 7

#define FRAMING_ERROR (1<<FE)
#define PARITY_ERROR (1<<UPE)
#define DATA_OVERRUN (1<<OVR)
#define DATA_REGISTER_EMPTY (1<<UDRE)
#define RX_COMPLETE (1<<RXC)

// USART Receiver buffer
#define RX_BUFFER_SIZE 8
char rx_buffer[RX_BUFFER_SIZE];

int receiv;
#if RX_BUFFER_SIZE<256
unsigned char rx_wr_index,rx_rd_index,rx_counter;
#else
unsigned int rx_wr_index,rx_rd_index,rx_counter;
#endif

// This flag is set on USART Receiver buffer overflow
```

```
bit rx_buffer_overflow;

// USART Receiver interrupt service routine
interrupt [USART_RXC] void usart_rx_isr(void)
{
char status,data;
status=UCSRA;
data=UDR;

if ((status & (FRAMING_ERROR | PARITY_ERROR | DATA_OVERRUN))==0)
   {
   rx_buffer[rx_wr_index]=data;
   receiv= rx_buffer[0];

   if ((++rx_wr_index == RX_BUFFER_SIZE)) rx_wr_index=0;
   if (++rx_counter == RX_BUFFER_SIZE)
      {
      rx_counter=0;
      rx_buffer_overflow=1;
      };
   };
}

#ifndef _DEBUG_TERMINAL_IO_
// Get a character from the USART Receiver buffer
#define _ALTERNATE_GETCHAR_
#pragma used+
char getchar(void)
{
char data;
while (rx_counter==0);
data=rx_buffer[rx_rd_index];
if (++rx_rd_index == RX_BUFFER_SIZE) rx_rd_index=0;
#asm("cli")
--rx_counter;
#asm("sei")
return data;
}
#pragma used-
#endif

// Declare your global variables here

void main(void)
{
PORTA=0x0f;
DDRA=0xff;

PORTB=0x00;
DDRB=0x00;

PORTC=0xff;
DDRC=0xFF;
```

```
PORTD=0xF0;
DDRD=0xFF;

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: Timer 0 Stopped
// Mode: Normal top=FFh
// OC0 output: Disconnected
TCCR0=0x00;
TCNT0=0x00;
OCR0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: Timer 1 Stopped
// Mode: Normal top=FFFFh
// OC1A output: Discon.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer 1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=0x00;
TCCR1B=0x00;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer 2 Stopped
// Mode: Normal top=FFh
// OC2 output: Disconnected
ASSR=0x00;
TCCR2=0x00;
TCNT2=0x00;
OCR2=0x00;

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
// INT2: Off
MCUCR=0x00;
MCUCSR=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x00;
```

```
// USART initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART Receiver: On
// USART Transmitter: On
// USART Mode: Asynchronous
// USART Baud rate: 9600
UCSRA=0x00;
UCSRB=0x98;
UCSRC=0x86;
UBRRH=0x00;
UBRRL=0x47;

// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
ACSR=0x80;
SFIOR=0x00;

// LCD module initialization
lcd_init(16);

// Global enable interrupts
#asm("sei")

while (1)
      {
   #asm("cli");
   lcd_clear();
   lcd_gotoxy(0,0);
   lcd_putsf("ComSer   :  AKSI");
   lcd_gotoxy(0,1);
   sprintf(buf,"rx:%d",receiv);
   lcd_puts(buf);
   delay_ms(10);
   #asm("sei");


 if(receiv== 90) {
  PORTD = 0b01110000;
  PORTC.5=0;
  delay_ms(500);
  //delay_ms(5000);
  lcd_gotoxy(9,1);
  lcd_putsf("MAJU");
  }

   else if(receiv == 100) {

  PORTD = 0b10110000;
  PORTC.5=0;
  delay_ms(500);
  lcd_gotoxy(9,1);
  lcd_putsf("B.KIRI");
```

```
  }

   else if(receiv== 50) {

  PORTD = 0b11010000;
  PORTC.5=0;
  delay_ms(500);
  lcd_gotoxy(9,1);
  lcd_putsf("B.KANAN");

  }
  else if(receiv== 80) {

  PORTD = 0b11100000;
  PORTC.5=0;
  delay_ms(500);
  lcd_gotoxy(9,1);
  lcd_putsf("B.KANAN");

  }
     };
}
```

3. Program pada mikrokontroller Robot (Sisis Penerima)

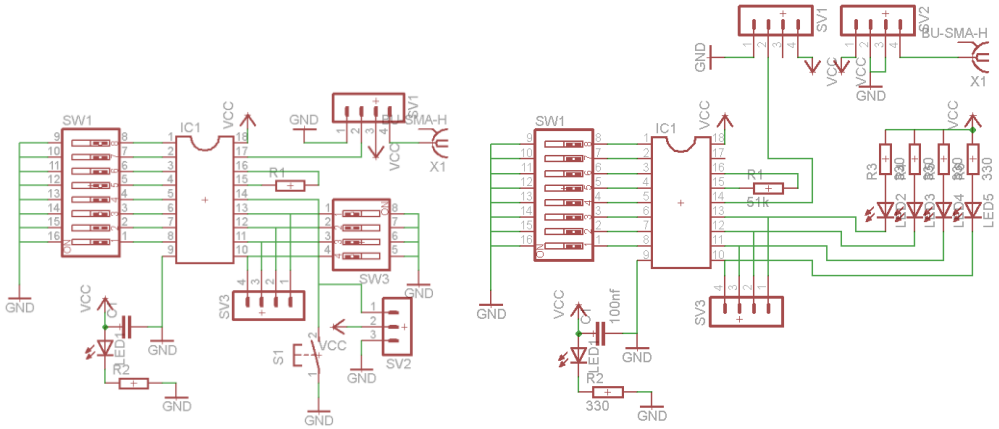
```
#include <mega8535.h>
#include <delay.h>
#include <stdio.h>

int x,lpwm,rpwm;
// Alphanumeric LCD Module functions
#asm
   .equ __lcd_port=0x1B ;PORTA
#endasm
#include <lcd.h>

char lcd_buffer[33];


// switch
#define sw_ok     PINB.0
#define sw_cancel PINB.1
#define sw_up     PINB.2
#define sw_down   PINB.3


// Timer 0 overflow interrupt service routine
interrupt [TIM0_OVF] void timer0_ovf_isr(void)
{
x++;
if (x>867){x=0;}
else {
    if (x<lpwm){PORTC.0=1;}
```

```
   else {PORTC.0=0;}
     if (x<rpwm){PORTC.1=1;}
   else {PORTC.1=0;}


}
}

void belok_kanan(){
       lpwm= 80;
       rpwm= 70;}

void berhenti(){
       lpwm= 70;
       rpwm= 70;}

void belok_kiri(){
       lpwm= 70;
       rpwm= 60;}

void maju(){
       lpwm= 100;
       rpwm= 40;}

void main(void)
{
PORTA=0x00;
DDRA=0x00;

PORTB=0x0f;
DDRB=0x00;

PORTC=0x00;
DDRC=0xff;

PORTD=0x00;
DDRD=0x00;

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: 12000,000 kHz
// Mode: Normal top=FFh
// OC0 output: Disconnected
TCCR0=0x01;
TCNT0=0x00;
OCR0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: Timer 1 Stopped
// Mode: Normal top=FFFFh
// OC1A output: Discon.
// OC1B output: Discon.
// Noise Canceler: Off
```

```
// Input Capture on Falling Edge
// Timer 1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=0x00;
TCCR1B=0x00;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer 2 Stopped
// Mode: Normal top=FFh
// OC2 output: Disconnected
ASSR=0x00;
TCCR2=0x00;
TCNT2=0x00;
OCR2=0x00;

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
// INT2: Off
MCUCR=0x00;
MCUCSR=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x01;

// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
ACSR=0x80;
SFIOR=0x00;

// LCD module initialization
lcd_init(16);

// Global enable interrupts
#asm("sei")
//showMenu();

while (1)
      {
      lcd_clear();
      lcd_gotoxy(0,0);
        lcd_putsf(" robot eye navi");
```

```
      lcd_gotoxy(0,1);
       lcd_putsf(" Aksi:");
     if (PIND.0==0 && PIND.1==1 &&PIND.2==1 &&PIND.3==1 ){
       maju(); lcd_gotoxy(9,1); lcd_putsf("maju"); delay_ms(50);        }
     else if (PIND.0==1 && PIND.1==0 &&PIND.2==1 &&PIND.3==1){
     belok_kiri();lcd_gotoxy(9,1);lcd_putsf("b.kiri"); delay_ms(50);
     }
     else if (PIND.0==1 && PIND.1==1 &&PIND.2==0 &&PIND.3==1){
     belok_kanan(); lcd_gotoxy(9,1);lcd_putsf("b.kanan");delay_ms(50);
     }
     else if (PIND.0==1 && PIND.1==1 &&PIND.2==1
&&PIND.3==0){berhenti();lcd_gotoxy(9,1);lcd_putsf("stop");delay_ms(50);}
     };
}
```

4. Skema Rangkaian Driver TLP 433 dan RLP 433 MHz



5. Skema Rangkaian sistim Minimum AVR 8535