SISTEM DETEKSI BOLA PADA ROBOT KRSBI BERDASARKAN DETEKSI WARNA HSV DENGAN PENAMBAHAN FUNGSI ROI (*REGION OF INTEREST*)

SKRIPSI

TEKNIK ELEKTRO KONSENTRASI TEKNIK ELEKTRONIKA

Ditujukan untuk memenuhi persyaratan

memperoleh gelar Sarjana Teknik

CHANDRA HALIM HARAHAP

NIM.135060301111046

UNIVERSITAS BRAWIJAYA

FAKULTAS TEKNIK

MALANG

2017

LEMBAR PENGESAHAN

SISTEM DETEKSI BOLA PADA ROBOT KRSBI BERDASARKAN DETEKSI WARNA HSV DENGAN PENAMBAHAN FUNGSI ROI (*REGION OF INTEREST*)

SKRIPSI

TEKNIK ELEKTRO KONSENTRASI TEKNIK ELEKTRONIKA

Ditujukan untuk memenuhi persyaratan
memperoleh gelar Sarjana Teknik

CHANDRA HALIM HARAHAP
NIM.135060301111046

Skripsi ini telah direvisi dan disetujui oleh dosen pembimbing
pada tanggal 24 Juli 2017

Dosen Pembimbing I	Dosen Pembimbing II
Dr. Eng. Panca Mudjirahardjo, S.T., M.T.	**Ir. Nurussa'adah, M.T.**
NIP. 19700329 200012 1 001	**NIP. 19680706 199203 2 001**

Mengetahui,
Ketua Jurusan

M. Aziz Muslim, S.T., M.T., Ph.D.
NIP. 19741203 200012 1 001

JUDUL SKRIPSI:

SISTEM DETEKSI BOLA PADA ROBOT KRSBI BERDASARKAN DETEKSI WARNA HSV DENGAN PENAMBAHAN FUNGSI ROI (*REGION OF INTEREST*)

Nama Mahasiswa : CHANDRA HALIM HARAHAP

NIM : 135060301111046

Program Studi : TEKNIK ELEKTRO

Konsentrasi : TEKNIK ELEKTRONIKA

Komisi Pembimbing :

Ketua : Dr. Eng. Panca Mudjirahardjo, S.T., M.T. ……

Anggota : Ir. Nurussa'adah, M.T. ……

Tim Dosen Penguji :

Dosen Penguji 1 : Dr. Ir. Ponco Siwindarto, M.Eng.Sc. ……

Dosen Penguji 2 : Dr. Ir. Muhammad Aswin, M.T. ……

Dosen Penguji 3 : Dr. Ing. Onny Setyawati, S.T., M.T., M.Sc. ……

Tanggal Ujian : 19 Juli 2017

SK Penguji : 846/UN.10.F07/SK/2017

PERNYATAAN ORISINALITAS SKRIPSI

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya dan berdasarkan hasil penelusuran berbagai karya ilmiah, gagasan dan masalah ilmiah yang diteliti dan diulas di dalam Naskah Skripsi ini adalah asli dari pemikiran saya. Tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu Perguruan Tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis dikutip dalam naskah ini dan disebutkan dalam sumber kutipan dan daftar pustaka.

Apabila ternyata di dalam naskah Skripsi ini dapat dibuktikan terdapat unsur-unsur jiplakan, saya bersedia Skripsi dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, pasal 25 ayat 2 dan pasal 70).

Malang, 6 Juli 2017

Mahasiswa,

CHANDRA HALIM HARAHAP

NIM. 135060301111046

Teriring Ucapan Terima Kasih kepada:

Ayahanda dan Ibunda tercinta

RINGKASAN

Chandra Halim Harahap, Jurusan Teknik Elektro, Fakultas Teknik Universitas Brawijaya, Juni 2017, *Sistem Deteksi Bola Pada Robot KRSBI Berdasarkan Deteksi Warna HSV Dengan Penambahan Fungsi ROI (Region Of Interest)*, Dosen Pembimbing: Panca Mudjirahardjo dan Nurussa'adah.

Kontes Robot Sepak Bola Indonesia (KRSBI) adalah salah satu kategori perlombaan robot yang sering diselenggarakan oleh Direktorat Jenderal Pendidikan Tinggi (Dirjen Dikti) setiap tahun. Untuk mengikuti kompetisi ini, setiap robot peserta diwajibkan mampu memiliki kemampuan dasar untuk deteksi bola menggunakan sensor kamera. Deteksi warna HSV merupakan cara untuk memfilter warna obyek yang dideteksi dengan warna yang lain berdasarkan lingkup warna *hue*, *saturation*, dan *value*. Untuk mempercepat proses filter warna pada deteksi warna HSV, proses deteksi HSV dapat ditambahkan fungsi ROI (*Region Of Interest*). ROI berfungsi untuk menyeleksi daerah dari obyek yang akan dideteksi untuk mengambil nilai dari parameter – parameter yang dibutuhkan untuk mendeteksi suatu obyek. *Library* yang digunakan adalah OpenCV (*Open source Computer Vision*) karena tersedia secara gratis serta banyaknya fitur yang bisa dilakukan.

Berdasarkan hasil penelitian, bola orange yang digunakan sebagai obyek yang dideteksi memiliki rata - rata nilai yaitu *Hue* (H) dengan nilai minimal sebesar 5 dan maksimal 17, *saturation* (S) dengan nilai minimal sebesar 148 dan maksimal sebesar 234, dan *value* (V) dengan nilai minimal sebesar 84 dan maksimal 254. Jarak maksimal bola terdeteksi sebesar 160,020 cm sedangkan secara praktik sebesar 158,50 cm. Persen error dari jarak bola secara teori dan praktik < 1,6 %. Pada sistem keseluruhan robot mampu mendeteksi bola pada 3 posisi bola yang diatur yaitu di timur robot, diselatan robot, dan di barat robot dengan jarak bola tetap 30 cm. Hasilnya, robot paling cepat menemukan bola pada posisi 1 (timur) dengan waktu rata-rata 4,12 detik sedangkan posisi 2 (selatan) robot menemukan bola dengan waktu rata-rata 6,10 detik dan posisi 3 (barat) robot menemukan bola dengan waktu rata-rata 8,05 detik. Posisi 1 memiliki waktu rata-rata lebih cepat dibandingkan posisi 2 dan posisi 3 dikarenakan algoritma pada mikrokontroler utama yang melakukan manuver gerakan putar kanan terlebih dahulu untuk mencari bola.

Kata kunci - KRSBI, Sensor kamera, Deteksi warna HSV, ROI, OpenCV

SUMMARY

Chandra Halim Harahap, *Department of Electrical Engineering, Faculty of Engineering, Brawijaya University, June 2017. Ball Detection System On KRSBI Robot Based on HSV Color Detection With Addition Function ROI (Region Of Interest), Academic Supervisor:* Panca Mudjirahardjo dan Nurussa'adah.

Kontes Robot Sepak Bola Indonesia (KRSBI) is one of the robot competition categories that are often organized by Direktorat Jenderal Pendidikan Tinggi (Dirjen Dikti) in every year. To participate in this competition, every participant's robot is required to have basic skills for ball detection using camera sensors. HSV color detection is a way to filter the color of objects detected with other colors based on the color scope of hue, saturation, and value. To speed up the color filter process on HSV color detection, the HSV detection process can be added to the ROI (Region Of Interest). ROI serves to select the area of the object to be detected to extract values from the parameters needed to detect an object. The library used is OpenCV (Open Source Computer Vision) because it is available for free as well as many features that can be done.

Based on the research results, the orange ball used as the detected object has an average value of Hue (H) with a minimum value of 5 and a maximum of 17, saturation (S) with a minimum value of 148 and a maximum of 234, and value (V) With a minimum value of 84 and a maximum of 254. The maximum distance of the ball is detected at 160,020 cm while in practice by 158,50 cm. Percent error from the ball spacing in theory and practice <1,6%. In the whole system the robot is able to detect the ball on 3 spherical position which is set in east robot, south of robot, and in west robot with fixed ball spacing 30 cm. The result, is the fastest robot finding the ball at position 1 (east) with an average time of 4,12 seconds while position 2 (south) robot finds the ball with an average time of 6,10 seconds and position 3 (west) robot finds the ball with an average time of 8,05 Seconds. Position 1 has an average time faster than position 2 and position 3 due to the algorithm on the main microcontroller that maneuvered the right-hand motion first to find the ball.

Keywords - KRSBI, Camera sensors, HSV Colour detection, ROI, OpenCV

PENGANTAR

Alhamdulillah, puji dan syukur penulis panjatkan kepada Allah SWT, karena atas segala petunjuk dan nikmat-Nya lah skripsi ini dapat diselesaikan. Skripsi berjudul "*SISTEM DETEKSI BOLA PADA ROBOT KRSBI BERDASARKAN DETEKSI WARNA HSV DENGAN PENAMBAHAN FUNGSI ROI (REGION OF INTEREST)*" ini disusun untuk memenuhi persyaratan memperoleh gelar Sarjana Teknik di Jurusan Teknik Elektro Universitas Brawijaya.

Penulis menyadari bahwa dalam penyusunan skripsi ini tidak terlepas dari bantuan berbagai pihak. Oleh karena itu, dengan ketulusan dan kerendahan hati penulis menyampaikan terima kasih kepada:

- Ayah Ridwan Harahap, Ibu Khairani selaku orang tua penulis atas segala inspirasi, nasehat, kasih sayang, perhatian dan kesabarannya didalam membesarkan dan mendidik penulis, serta telah banyak mendoakan kelancaran penulis hingga terselesaikannya skripsi ini,
- Seluruh Saudara dan Keluarga Besar di Subulussalam dan Padang Sidempuan.
- Yang Terhormat Bapak M. Aziz Muslim, ST., MT., Ph.D selaku Ketua Jurusan Teknik Elektro Universitas Brawijaya.
- Yang Terhormat Bapak Hadi Suyono, ST., MT., Ph.D selaku Sekretaris Jurusan Teknik Elektro Universitas Brawijaya.
- Yang Terhormat Bapak Dr. Eng. Panca Mudjirahardjo, S.T., M.T. selaku Dosen Pembimbing I atas segala bimbingan, pengarahan, saran, dan kritik yang telah diberikan selama proses pengerjaan skripsi.
- Yang Terhormat Ibu Ir. Nurussa'adah, M.T. selaku Ketua Kelompok Dosen Keahlian Elektronika Jurusan Teknik Elektro Universitas Brawijaya, Dosen Pembimbing II, dan Dosen Penasehat Akademik penulis atas segala bimbingan, pengarahan, saran, dan kritik yang telah diberikan selama perkuliahan dan pengerjaan skripsi.
- Seluruh dosen pengajar Jurusan Teknik Elektro Universitas Brawijaya,
- Seluruh staff recording Jurusan Teknik Elektro Universitas Brawijaya

- Keluarga Besar “Generasi Emas” atau asisten Laboratorium Mekatronika angkatan 2013 atas segala pengalaman, kebersamaan dan bantuan selama 3 tahun ini menjadi anggota tim robotika dan asisten.
- Keluarga Besar Laboratorium Sistem Digital atas segala pengalaman, kebersamaan dan bantuan selama menjadi asisten.
- Keluarga Besar Tim Robotika TEUB terkhusus sub divisi KRSBI atas segala pengalaman, kebersamaan dan bantuan selama ini.
- Keluarga Besar Mahasiswa Teknik Elektro Universitas Brawijaya,
- Saudara – sodari “Spectrum” angkatan 2013 atas segala bantuan dan kebersamaan yang telah diberikan selama 4 tahun ini.
- Seluruh teman-teman serta semua pihak yang tidak mungkin bagi penulis untuk mencantumkan satu-persatu, terimakasih banyak atas bantuan dan dukungannya.

Pada akhirnya, penulis menyadari bahwa skripsi ini masih belum sempurna. Oleh karena itu, penulis sangat mengharapkan kritik dan saran yang membangun. Penulis berharap semoga skripsi ini dapat bermanfaat bagi pengembangan ilmu pengetahuan dan teknologi serta bagi masyarakat.

Malang, Juli 2017

Penulis

DAFTAR ISI

Halaman

DAFTAR TABEL

No. Judul Halaman

DAFTAR GAMBAR

No. Judul Halaman

DAFTAR LAMPIRAN

BAB I
PENDAHULUAN

1.1 Latar Belakang

Kontes Robot Sepak Bola Indonesia (KRSBI) adalah Liga Sepak Bola Robot *Humanoid* (*Robosoccer Humanoid League*) yang mengacu pada cita-cita organisasi *RoboCup International* sebagai penyelenggara resmi kontes sepak bola robot di dunia, yang memiliki cita-cita utama yaitu tahun 2050 mampu mencetak tim sepak bola yang mampu melawan tim juara dunia. Peserta Kontes Robot Sepak Bola diwajibkan membuat minimal dua robot dan maksimal lima robot yang terdiri dari robot penjaga gawang dan robot penyerang yang telah dirancang menggunakan sensor, *processor*, dan aktuator yang sesuai dan berfungsi sesuai aturan perlombaan. Peserta kontes robot ini akan dibagi menjadi dua tim, yang terdiri dari tim merah dan tim biru, robot dari salah satu tim yang mampu menggiring dan memasukkan bola ke gawang lawan sebanyak mungkin selama waktu 2×10 menit akan dinyatakan sebagai pemenang (RISTEKDIKTI, 2016).

Untuk mengikuti kompetisi KRSBI, setiap robot peserta diwajibkan mampu memiliki kemampuan dasar untuk deteksi bola menggunakan sensor kamera. Pada saat perlombaan, tim robot KRSBI Teknik Elektro Universitas Brawijaya belum berhasil membuat sistem pendeteksian yang baik hal ini dilihat dari sistem kalibrasi untuk memperoleh nilai parameter-parameter warna bola yang lama yaitu dengan cara manual menggunakan *trackbar window* atau memanfaatkan aplikasi *photo editor* akibatnya kurang mengoptimalkan waktu untuk melakukan sesuatu yang lain. Selain itu sistem pendeteksian saat perlombaan yang dibuat terkadang juga masih menyebabkan kegagalan dari navigasi robot untuk menemukan posisi bola.

Berdasarkan ruang lingkup warna, obyek seperti bola dapat dideteksi dengan metode deteksi warna RGB & HSV. Deteksi latar belakang sebagai warna dominan telah berhasil dilakukan dengan menerapkan histogram S-RGB (Mudjirahardjo, 2016). Untuk deteksi warna HSV saat ini masih dikembangkan oleh tim robot KRSBI Teknik Elektro Universitas Brawijaya. Deteksi warna HSV merupakan cara untuk memfilter warna obyek yang dideteksi dengan warna yang lain berdasarkan lingkup warna *hue*, *saturation*, dan *value*. Untuk mempercepat proses filter warna pada deteksi warna HSV, proses deteksi HSV dapat ditambahkan fungsi ROI (*Region Of Interest*). ROI berfungsi untuk menyeleksi daerah dari obyek yang akan dideteksi

untuk mengambil nilai dari parameter – parameter yang dibutuhkan untuk mendeteksi suatu obyek. Fungsi ROI dapat dikombinasikan dengan fungsi *mouse event* untuk mendapatkan *region* pada obyek yang akan dideteksi dengan cepat.

Maka, pada skripsi ini dirancang suatu sistem untuk deteksi bola pada robot KRSBI berdasarkan deteksi warna HSV dengan penambahan fungsi ROI *(Region Of Interest*). *Library* yang digunakan adalah OpenCV (*Open source computer vision*). OpenCV digunakan karena *library* ini dapat tersedia secara gratis serta lebih dikhususkan untuk penglihatan komputer secara *real time* yang dikembangkan oleh pusat penelitian Intel di Nizhny Novgorod, Rusia. Penggunaan sensor kamera dan OpenCV sebenarnya pernah diteliti oleh Akhmad Tegar Fahreza F untuk deteksi warna lapangan KRSI menggunakan kamera dan sensor warna untuk mencari sudut hadap dan jarak (Tegar, 2015). Penulis menyadari bahwa metode yang digunakan pada skripsi tersebut berdasarkan deteksi warna HSV yang masih menggunakan cara manual untuk menentukan nilai dari parameter-parameter untuk deteksi obyek. Selain itu pada robot penari obyek yang dideteksi tidak berpindah tempat. Berbeda dengan kompetisi robot sepak bola obyek yang dideteksi yaitu bola, setiap waktu akan berpindah jika disentuh oleh robot lain atau oleh robot sendiri. Dengan menggunakan deteksi warna HSV dengan penambahan ROI diharapkan nantinya robot mampu deteksi obyek dengan baik dilihat dari respon navigasinya serta proses kalibrasi yang tidak lama.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan dapat disusun rumusan masalah sebagai berikut :

1. Bagaimana merancang dan membuat robot yang dapat mendeteksi bola dengan deteksi warna HSV dengan penambahan fungsi ROI ?
2. Bagaimana mengetahui jarak maksimal bola mampu dideteksi oleh robot?
3. Bagaimana mengetahui posisi bola yaitu kanan, kiri, dan tengah dari posisi robot ?
4. Bagaimana membuat robot bergerak sesuai posisi bola yang telah dideteksi ?

1.3 Batasan Masalah

Dengan mengacu pada permasalahan yang telah dirumuskan, batasan masalah untuk skripsi ini adalah sebagai berikut :

1. Tidak menggunakan robot jenis *humanoid* melainkan menggunakan dua buah roda sebagai aktuator,

2. Menggunakan *library* OpenCV,
3. Jumlah robot yang dibuat satu buah,
4. Warna bola yang digunakan warna orange,
5. Lapangan mencari bola digunakan karpet berwarna hijau,
6. Pembahasan mekanik robot hanya sekedar aspek praktis, tanpa membahas mekanika secara mendalam,
7. Sensor kamera yang digunakan harus dilakukan kalibrasi terlebih dahulu karena mudah terpengaruh oleh pencahayaan ruang.

1.4 Tujuan

Tujuan skripsi ini adalah untuk merancang dan membuat sistem untuk deteksi bola pada robot KRSBI menggunakan metode *color filtering* berdasarkan lingkup warna HSV dengan penambahan fungsi ROI (*Region Of Interest*).

1.5 Manfaat

Adapun manfaat dari hasil skripsi ini adalah sebagai berikut :

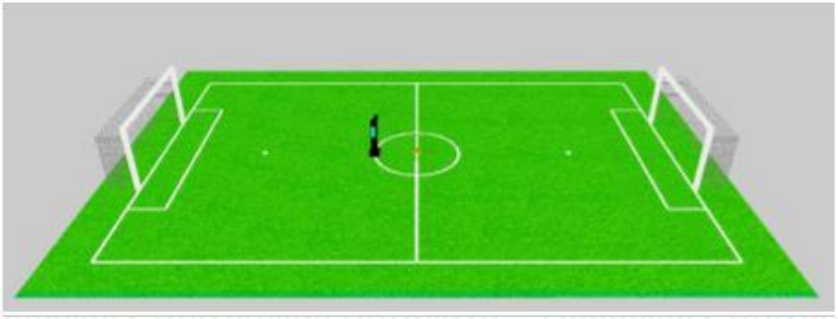
1. Manfaat utama dari skripsi ini yaitu membantu sistem deteksi dan mengikuti bola yang baik dan handal pada sistem robot sepak bola,
2. Karena skripsi ini menggunakan sensor utama berupa kamera diharapkan dapat dikembangkan lebih lanjut agar penggunaan sensor kamera dapat mengganti sensor-sensor lain seperti sensor jarak, *proximity*, dan lain-lain,
3. Hasil skripsi ini dapat dikembangkan kembali untuk aplikasi pada algoritma deteksi gawang pada kompetisi KRSBI.

BAB II
TINJAUAN PUSTAKA

2.1 Kontes Robot Sepak Bola Indonesia

Kontes Robot Sepak Bola Indonesia (KRSBI) merupakan sebuah kompetisi tahunan yang bertujuan untuk membuat perancangan dan pembuatan robot sepak bola yang diseleksi untuk mempersiapkan tim pilihan yang berhak menuju kompetisi internasional. Kompetisi ini mengacu pada cita-cita organisasi *RoboCup International* sebagai penyelenggara resmi kontes sepak bola robot di dunia, yang memiliki cita-cita utama yaitu tahun 2050 mampu mencetak tim sepak bola yang mampu melawan tim juara dunia. Peserta Kontes Robot Sepak Bola diwajibkan membuat minimal satu robot dan maksimal lima robot yang terdiri dari robot penjaga gawang dan robot penyerang yang telah dirancang menggunakan sensor, *processor*, dan aktuator yang sesuai dan berfungsi sesuai aturan perlombaan. Untuk dapat lolos dalam proses seleksi awal (Tahap II : Laporan perkembangan Robot dalam Video) tim harus menunjukkan bahwa tiap robot memiliki kemampuan dasar mencari bola dan menendang bola. Tanpa kemampuan dasar ini tim robot tidak akan diperkenankan mengikuti pertandingan regional (RISTEKDIKTI, 2016).

KRSBI menggunakan karpet berwarna hijau sebagai lapangan dan bola berwarna orange sebagai obyek yang akan dideteksi oleh masing-masing robot. Ilustrasi kondisi lapangan ditunjukkan dalam Gambar 2.1.



Gambar 2.1 Ilustrasi kondisi lapangan
Sumber : RISTEKDIKTI (2016,p.14)

Robot harus dilengkapi sensor yang bertujuan untuk merepresentasikan sensor-sensor alami dari manusia seperti kamera (mata), sensor sentuhan (kulit), sensor tekanan (untuk telapak kaki) dan lain sebagainya.

2.2 OpenCV

OpenCV *(Open Source Computer Vision Library)* merupakan *library* pemrograman pada komputer yang bersifat *open source* dan didalamnya terdapat ratusan algoritma untuk melakukan pemrograman *computer vision. Library* ini memiliki antarmuka C, C++, phyton, dan java, serta dapat digunakan oleh windows, linux, maupun android. OpenCV didesain untuk meningkatkan efisiensi pada komputasi dan memiliki fokus pada aplikasi *real-time* (Itseez, 2014).

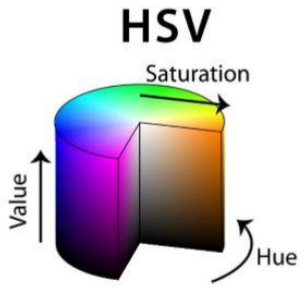
OpenCV memiliki beberapa fitur yaitu :

- Manipulasi data citra (alokasi, *copying*, *setting*, konversi).
- Citra dan video I/O (file dan kamera *based input*, *image/video file output*).
- Manipulasi Matriks dan Vektor beserta aljabar linear (*products*, *solvers*, *eigenvalues*, SVD).
- Data struktur dinamis (*lists*, *queues*, *sets*, *trees*, *graphs*).
- Pemroses Citra fundamental (*filtering*, *edge detection*, *corner detection*, *sampling and interpolation*, *color conversion*, *morphological operations*, *histograms*, *image pyramids*).
- Analisis struktur(*connected components*, *contour processing*, *distance transform*, *various moments*, *template matching*, *Hough transform*, *polygonal approximation*, *line fitting*, *ellipse fitting*, *Delaunay triangulation*).
- Kalibrasi kamera (*calibration patterns*, *estimasi fundamental matrix*, *estimasi homography*, *stereo correspondence*).
- Analisis gerakan (*optical flow*, *segmentation*, *tracking*).
- Pengenalan obyek (*eigen-methods*, HMM).
- Graphical User Interface (*display image/video*, penanganan keyboard dan *mouse event*, *scroll-bars*).
- Pelabelan citra (*line*, *conic*, *polygon*, *text drawing*).

2.3 Model Warna HSV

Ruang warna HSV terdiri dari tiga deskriptor yang berbeda, yang selalu diamati dari jumlah yang minimum untuk mengklasifikasikan warna. Pertama adalah *hue* yang dideskripasikan sebagai warna yang dapat dilihat oleh mata manusia. Kedua adalah *saturation*, dideskripsikan sebagai kemurnian warna atau pada *computer vision* merupakan representasi dari banyaknya

warna putih yang tercampur. Ketiga adalah *value*, digambarkan sebagai kecerahan warna (Ivask, 2015, p.14). Sistem silinder dari warna HSV ditunjukkan dalam Gambar 2.2.



Gambar 2.2 Sistem silinder warna HSV
Sumber : Ivask (2015,p.14)

2.4 Deteksi Warna HSV

Deteksi warna merupakan kemampuan untuk mengambil gambar, memisahkan warna tertentu, dan mengambil informasi berupa posisi dari warna yang dipisahkan tersebut. Sebagai contoh, jika seseorang melihat sebuah foto yang berisi gambar bola merah di jalan dan ingin melingkari warna merah yang ada di dalam foto, orang tersebut akan dengan mudah melingkari bola merah di dalam gambar. Inilah ide dasar dari deteksi warna. Seseorang tidak perlu mengetahui obyek yang ada pada foto adalah bola, dia hanya perlu mengetahui warna dari bola tersebut (Adam Goode, 2011). Adapun gambar hasil memisahkan satu warna dengan warna lainnya ditunjukkan dalam Gambar 2.3.

Gambar 2.3 Deteksi warna dengan memisahkan satu warna dengan warna lainnya
Sumber : Mordvintsev (2017,p.51)

Deteksi warna HSV merupakan cara untuk memfilter warna obyek yang dideteksi dengan warna yang lain berdasarkan lingkup warna *hue*, *saturation*, dan *value*. Pertama kali obyek input warna RGB akan di *convert* menjadi warna HSV. Kemudian warna HSV tersebut diberi batas tertentu dengan konsep binerisasi dengan instensitas 0 atau 255 sampai diperoleh hasil *thresholding*. Jika nilai *threshold* yang telah diatur adalah 200, maka semua *pixel* yang nilainya

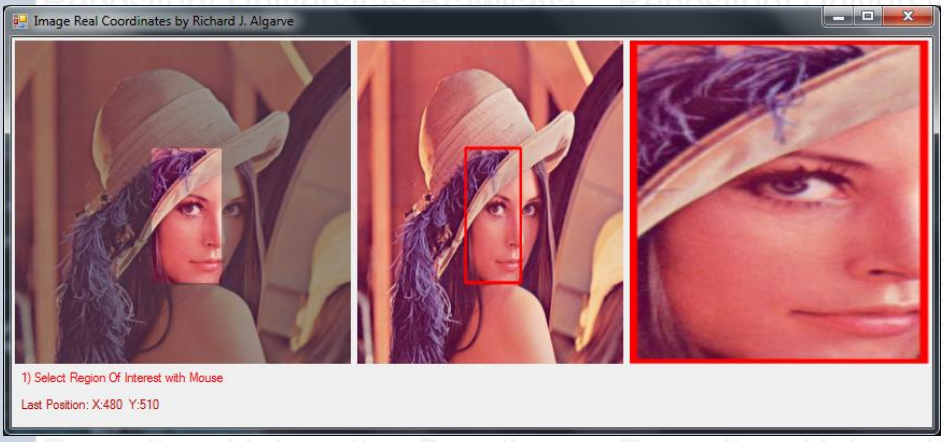
diatas atau sama dengan 200 diganti menjadi 255, sedangkan jika nilainya dibawah 200 diganti menjadi 0 seperti yang ditunjukkan pada persamaan (2-1).

$$f(x,y); if\ f(x,y) > Threshold\ Then, f(x,y) = 255\ else\ 0 \quad (2\text{-}1)$$

Salah satu cara dalam melakukan *tracking* warna, yaitu dengan menentukan nilai minimum dan maksimum untuk tiga jenis warna dasar. Setiap warna yang unik diwakili oleh nilai merah, hijau, dan biru yang dapat dicampur untuk menjadi warna baru. Bagian tersulit dalam menentukan warna adalah bahwa seseorang harus menentukan rentang nilai untuk semua tiga jalur warna. Karena pencahayaan yang berbeda-beda dan warna obyek yang tidak sempurna, sehingga diperlukan kalibrasi untuk mendapatkan rentang nilai yang optimal untuk suatu obyek tertentu (Tegar, 2015).

2.5 ROI (*Region Of Interest*)

ROI (*Region Of Interest*) merupakan salah satu fungsi untuk membuat *sub image* suatu gambar yang telah disediakan di *library* OpenCV. ROI digunakan untuk menyeleksi daerah dari obyek yang akan dideteksi kemudian mengambil nilai dari parameter – parameter yang dibutuhkan untuk mendeteksi suatu obyek. Dengan menggunakan *library* OpenCV fungsi ROI dapat dikombinasikan dengan fungsi *mouse event* untuk mendapatkan *region* pada obyek yang akan dideteksi dengan cepat. Adapun contoh dari hasil ROI dengan *mouse event* pada suatu Gambar ditunjukkan dalam Gambar 2.4.



Gambar 2.4 Hasil ROI dengan *mouse event* pada suatu gambar

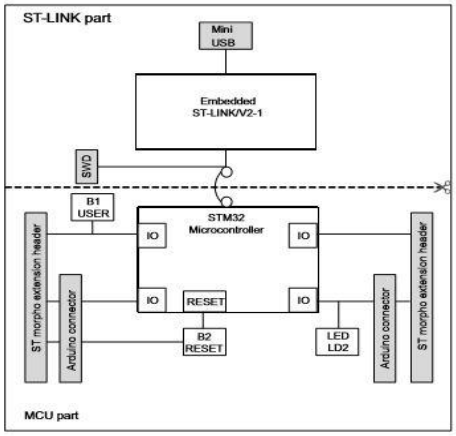
Sumber : Richard (2015)

2.6 Modul STM32F401RE Nucleo

STM32F401RE Nucleo merupakan board mikrokontroler berbasis ARM Cortex-M4 yang dikembangkan oleh STMicroelectronics. Modul mikrokontroler ini memiliki keunggulan yaitu

memiliki kecepatan proses data yang sangat tinggi dan RAM yang besar sehingga dapat diaplikasikan dalam skala yang lebih luas. Selain itu modul mikrokontroler ini juga kompatibel dengan *development tools* yang dikembangkan secara terbuka yaitu Mbed (STMicroelectronics, 2015).

Blok diagram *hardware* dari modul miktokontroler STM32F401RE Nucleo ditunjukkan dalam Gambar 2.5.



Gambar 2.5 Blok diagram *hardware* STM32 nucleo
Sumber : STMicroelectronic (2015,p.12)

Fitur-fitur yang dimiliki STM32F401RE Nucleo sebagai berikut:

1. Mikrokontroler ARM Cortex M4 32 bit memiliki kemampuan tinggi, dengan daya yang rendah serta kecepatan hingga 84 MHz.
2. Memiliki kapasitas *flash* memori 512 Kilobyte serta SRAM sebesar 96 Kilobyte.
3. Memiliki GPIO (50) dengan kemampuan *external interrupt.*
4. Fitur *Pheripheral*
   - Real Time Clock (RTC).
   - 7 buah *General Purpose Timers.*
   - 2 (dua) buah *Watchdog Timers*.
   - 4 (empat) buah USART/UART.
   - 16 *channel*s ADC 12-bit
   - *Advanced control timers* serta 7 (tujuh) *general purpose timers.*
   - 3 (tiga) buah I2C
   - 3 (tiga) buah SPI
   - 4 *channels* PWM

2.7 Raspberry Pi

Raspberry Pi adalah *personal computer* (PC) berukuran mini seukuran kartu kredit.Sistem operasinya ditanam pada sebuah SD Flash Card yang menjadikannya sangat mudah untuk diganti dan ditukar. Potensinya luar biasa, dari yang sudah maupun belum pernah dieksplorasi, tetapi telah diuji sebagai multimedia player dengan kemampuan streaming, sebagai perangkat mesin game, Internet dan sebagai mainboard pengembangan perangkat keras. Bentuk fisik dari Raspberry Pi-2 Model B ditunjukkan dalam Gambar 2.6.

Gambar 2.6 Raspberry Pi-2 model B
Sumber : Dokumentasi Penulis

Raspberry Pi-2 Model B ini adalah pengembangan dari tipe sebelumnya yaitu Rpi Model B. Beberapa keunggulan utama Raspberry Pi-2 ini adalah prosesornya yang lebih cepat yaitu *QuadCore* 900 MHz, memori yang lebih besar dari versi sebelumnya yaitu sebesar 1 GB. Spesifikasi adalah sebagai berikut:

Catu daya: 5 VDC, 800 mA melalui *micro* USB

2. Berbasis mikroprosesor Broadcom BCM2836 ARMv7 *Quad Core*, 900 MHz

3. Jumlah port I/O adalah 40 pin GPIO

4. *Port* antarmuka: UART TTL, SPI, I2C, 4x USB, RCA 3.5 mm, LCD *Panels* DSI, RJ-45 10/100, CSI (*Camera Serial Interface*), HDMI, MicroSD *Card Slot*

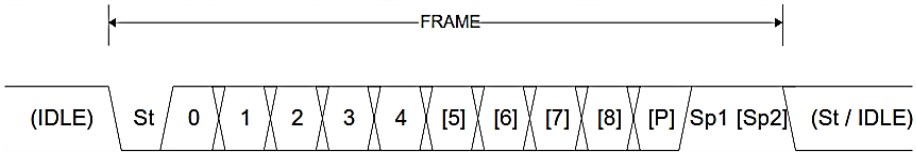
5. *Bootloader*: melalui sistem berbasis LINUX

6. Dimensi: 85.60 mm × 56.5 mm

2.8 Komunikasi Serial Asinkron

Komunikasi serial asinkron adalah suatu komunikasi data serial yang tidak memerlukan sinyal *clock* sebagai sinkronisasi. Pengiriman data serial ini harus diawali dengan *start bit* dan diakhiri dengan *stop bit*. Sinyal *clock* merupakan *baud rate* dari komunikasi data yang dibangkitkan oleh masing-masing baik penerima maupun pengirim data dengan frekuensi yang sama, jika nilai *baud rate* berbeda maka tidak akan pernah terjadi komunikasi (Durda, 2014).

Setiap pengiriman data pada UART menggunakan bit tanda *start bit* dan *stop bit*. Jalur data yang digunakan hanya satu untuk setiap pengiriman data. Data-data serial dikirim melewati jalur data satu persatu setiap satuan waktu. Paket data dimulai dengan *start bit* diikuti dengan *Least Significant Data Bit* (LSB). Data berikutnya adalah bit data yang dibentuk hingga 9 bit dan diakhiri dengan *Most Significant Bit* (MSB). Jika diaktifkan, maka *bit parity* akan diletakkan setelah bit data. Paket data diakhiri dengan *stop bit*. Setelah melakukan pengiriman data, maka komunikasi bisa langsung dimulai lagi atau jalur komunikasi bisa diset ke logika tinggi (*idle*) contohnya format pengiriman data komunikasi serial asinkron ditunjukkan dalam Gambar 2.7.

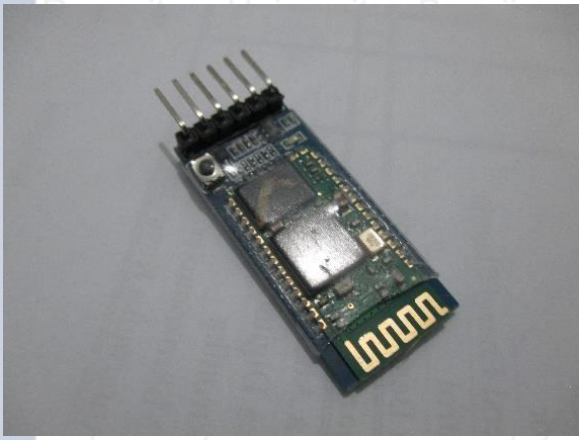


Gambar 2.7 Format pengiriman data komunikasi serial asinkron
Sumber : Atmel (2010,p.148)

2.9 Modul *Bluetooth* HC-05

Bluetooth Module HC-05 merupakan *module* komunikasi nirkabel pada frekuensi 2.4GHz dengan pilihan koneksi bisa sebagai *slave*, ataupun sebagai *master*. Dapat digunakan dengan mikrokontroler untuk membuat aplikasi *wireless* bentuk fisik modul *bluetooth* HC-05 ditunjukkan dalam Gambar 2.8. Adapun spesifikasinya adalah sebagai berikut:

- Frekuensi : 2.4GHz,
- Catu Daya : 5 V atau 3,3 V DC,
- Dimensi : 3.57cm x 1.52cm.



Gambar 2.8 Bentuk fisik modul *bluetooth* HC-05

Sumber : Dokumentasi Penulis

2.10 *Driver* L298N

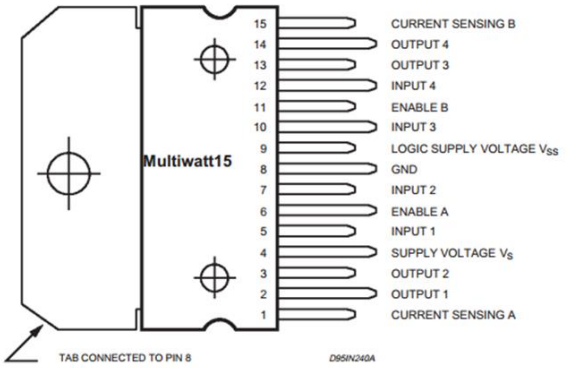
L298N adalah IC *driver* yang mengatur arah dan kecepatan motor DC. L298 dapat digunakan pada tegangan tinggi dan memiliki keluaran arus sampai dengan 2 A setiap *channel*-nya dengan ketahanan terhadap suhu yg cukup tinggi. Tabel 2.1 menunjukkan *absolute maximum rating* dari L298N. Gambar 2.9 menunjukkan *pin* konfigurasi dari L298.

Tabel 2.1

Absolute Maximum Rating Pin IC L298N

Symbol	Parameter	Value	Unit
V_S	*Power supply*	50	V
V_{SS}	*Logic Supply Voltage*	7	V
V_I, V_{en}	*Input and Enable Voltage*	-0.3 to 7	V
I_O	*Peak Output Current (Each Channel)*		
	-Non Repetitive	3	A
	-Repetitive (80% on -20% off; t_{on} = 10ms)	2.5	A
	-DC Operation	2	A
V_{sens}	*Sensing Voltage*	-1 to 2.3	V
P_{tot}	*Total Power Dissipation* (T_{case} = 75˚C)	25	W
T_{op}	*Junction Operation Temperature*	-25 to 130	C
T_{stg}, T_j	*Storage and Junction Temperature*	-40 to 150	C

Sumber : ST (2000,p.2)

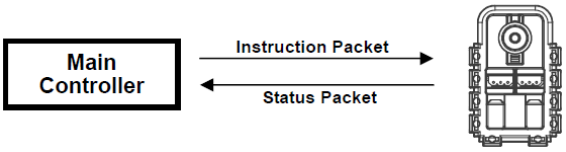


Gambar 2.9 Konfigurasi pin IC L298N

Sumber: ST (2000,p.2)

2.11 *Servo Serial Dynamixel*

Dynamixel adalah aktuator robot cerdas modular yang menggabungkan *gear reducer*, motor DC presisi dan sirkuit kontrol, semua dalam satu paket. Meskipun ukurannya ringkas, *servo serial dynamixel* dapat menghasilkan torsi tinggi dan dibuat dari bahan berkualitas tinggi untuk memberikan kekuatan yang diperlukan dan ketahanan untuk menahan kekuatan eksternal yang besar. *servo serial dynamixel* ini juga memiliki kemampuan untuk mendeteksi dan bertindak pada kondisi internal seperti perubahan suhu atau tegangan suplai. Aktuator *Dynamixel* memiliki banyak keunggulan dibandingkan produk sejenis. Posisi dan kecepatan dapat dikendalikan secara serial dengan resolusi 1024 *step*, adapun gambaran umum sistem pengontrolan dynamixel ditunjukkan dalam Gambar 2.10 (Robotis, 2006).



Gambar 2.10 Sistem pengontrolan *servo serial dynamixel*

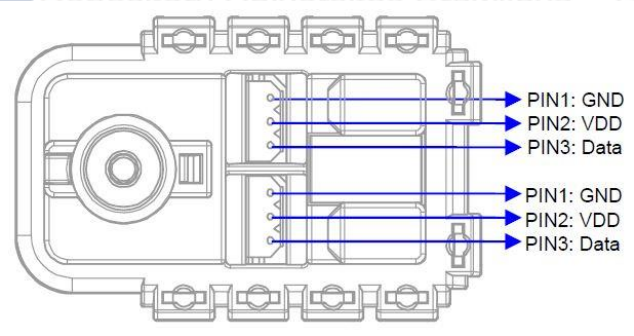
Sumber : Robotis (2006,p.9).

Kontroler utama berkomunikasi dengan unit *Dynamixel* dengan mengirim dan menerima paket data. Ada dua jenis paket; yang "*Instruction Packet*" (dikirim dari kontroler utama ke aktuator servo) dan "*Status Packet*" (dikirim dari Aktuator untuk mikrokontroler utama). Struktur paket instruksi pada pengiriman data ke aktuator servo serial dapat dilihat pada Gambar 2.11, *pinout* dari *servo serial dynamixel* ditunjukkan dalam Gambar 2.12.

OXFF	0XFF	ID	LENGTH	ERROR	PARAMETER1	PARAMETER2	...	PARAMETER N	CHECK SUM

Gambar 2.11 Paket instruksi pengiriman data pada *servo serial dynamixel*

Sumber : Robotis (2006,p.10)



Gambar 2.12 Pinout servo serial dynamixel

Sumber : Robotis (2006,p.6)

Kedua paket data pertama yaitu 0xFF menandakan awal dari paket yang akan datang. ID adalah nomor identitas yang unik dari aktuator yang berjumlah 254 ID dan memiliki *range* 0x00 - 0xFD. LENGTH merupakan panjang paket data dimana memiliki nilai jumlah parameter (N) + 2. INSTRUCTION merupakan instruksi yang harus dilakukan oleh aktuator. PARAMETER adalah informasi taambahan yang diberikan pada instruksi tertentu. CHECKSUM merupakan metode komputasi untuk pengecekan data. Persamaan untuk mencari nilai CHECKSUM dapat dilihat pada persamaan (2-2). Jika nilai yang terhitung lebih besar dari 255 maka bit terendah dianggap sebagai nilai CHECKSUM. Simbol "~" menandakan logika not.

$$\boldsymbol{CHECKSUM} = \sim (\boldsymbol{ID} + \boldsymbol{LENGTH} + \boldsymbol{INSTRUCTION} + \boldsymbol{PARAMETER\,1} + \cdots \boldsymbol{PARAMETER\,N}) \quad (2\text{-}2)$$

BAB III
METODE PENELITIAN

Penyusunan skripsi ini didasarkan pada masalah yang bersifat aplikatif, yaitu perencanaan dan perealisasian alat agar dapat bekerja sesuai dengan yang direncanakan dengan mengacu pada rumusan masalah. Oleh karena itu dibutuhkan suatu metode penelitian agar perencanaan dan perealisasian alat dapat dilakukan. Langkah – langkah yang perlu dilakukan untuk merealisasikan alat yang dirancang adalah penentuan spesifikasi alat, perancangan dan perealisasian alat, dan pengujian alat.

3.1 Penentuan Spesifikasi Alat

Spesifikasi alat secara global ditetapkan terlebih dahulu sebagai acuan dalam perancangan selanjutnya. Spesifikasi alat yang direncanakan adalah sebagai berikut :

1) Menggunakan dua buah motor DC sebagai aktuator,
2) Mekanika robot keseluruhan menggunakan bahan dasar *acrylic* dengan ketebalan 3 milimeter,
3) Driver motor DC menggunakan IC *driver* L298N,
4) Jari-jari roda robot diameter 8 cm dan tebal 1.2 cm,
5) Menggunakan kamera Logitech C270HD,
6) Sumber tegangan berupa baterai *Lithium Polymer* 3S 12.6V 2200mAH,
7) Rangkaian catu daya 5V berupa rangkaian regulator switching LM2576 *simple switcher* untuk STM32F401RE dan mini komputer Raspberry Pi,
8) Berat robot kurang dari 30 kg,
9) Ukuran robot diameter x tinggi ialah (40 cm x 30 cm),
10) Lapangan menggunakan karpet berwarna hijau,
11) Bola yang digunakan merupakan bola kasti berwarna orange,
12) Menggunakan servo serial dynamixel AX-12A untuk menggerakkan sensor kamera,
13) Menggunakan modul *bluetooth* HC-05 untuk komunikasi *wireless* robot,
14) USB to TTL *converter* yang digunakan bertipe CP2120,
15) Mikrokontroler utama robot menggunakan modul STM32F401RE Nucleo.

3.2 Perancangan Dan Perealisasian Alat

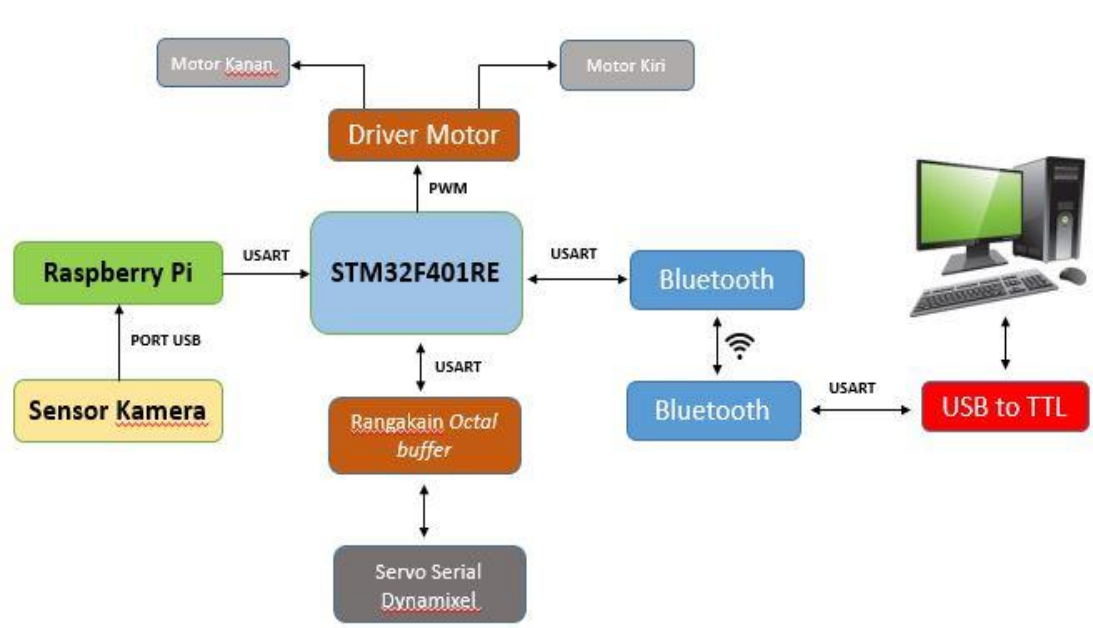
Perancangan dan pembuatan alat dalam skripsi ini dibagai menjadi dua bagian, yaitu pembuatan *hardware* dan *software*.

3.2.1 Perancangan dan Pembuatan Perangkat Keras (*Hardware*)

Secara garis besar perancangan dan pembuatan perangkat keras dapat dibagai menjadi beberapa bagian yaitu perancangan diagram blok sistem keseluruhan dan perancangan mekanika robot.

3.2.1.1 Perancangan Diagram Blok Sistem Keseluruhan

Diagram blok keseluruhan sistem yang dirancang ditunjukkan dalam Gambar 3.1.



Gambar 3.1 Diagram blok sistem keseluruhan

Fungsi masing-masing bagian dalam diagram blok ini adalah sebagai berikut :

1. STM32F401RE digunakan sebagai mikrokontroler utama dari sistem untuk mengolah hasil pengolahan menggunakan OpenCV dan menggerakkan aktuator,
2. *Driver* motor menggunakan driver L298N untuk menggerakkan motor berdasarkan PWM yang dikirim melalui mikrokontroler utama,
3. USB to TTL berfungsi sebagai konverter tegangan,
4. Raspberry Pi berfungi sebagai mini komputer yang nantinya ditambahkan *library* OpenCV yang kemudian akan digunakan untuk memproses olahan obyek yang akan dideteksi dan mengirimkan data tersebut ke mikrokontroler utama,

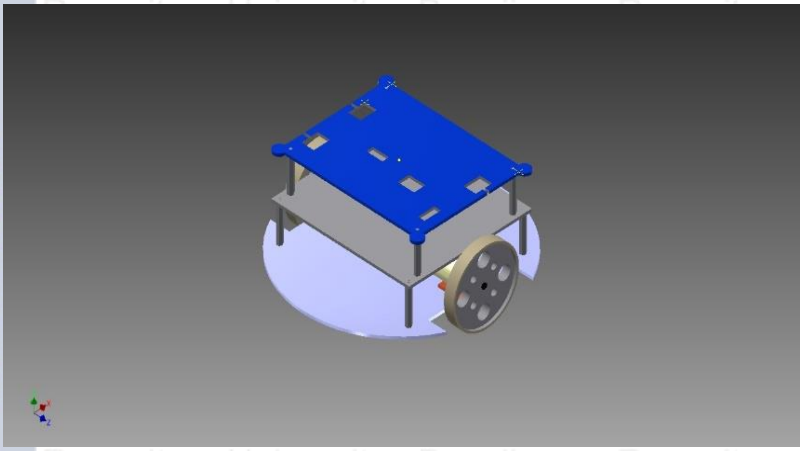
5. Motor kanan dan kiri sebagai aktuator penggerak robot,
6. Rangkaian *octal buffer* digunakan untuk menggerakkan servo serial dynamxel,
7. Servo serial dynamixel sebagai aktuator untuk menggerakkan sensor kamera,
8. Sensor kamera digunakan sebagai sensor utama untuk memperoleh gambar dari obyek yang akan dideteksi secara *real time*,
9. Komputer digunakan untuk pemantauan proses pemetaan yang dilakukan oleh robot,
10. Modul *bluetooth* HC-05 berfungsi untuk menerima data serial.

Prinsip kerja sistem ini adalah awalnya robot akan mencari bola dengan melakukan gerakan berputar di lapangan. Sembari mencari bola, robot akan melakukan proses pendeteksian warna yaitu warna orange dari bola. Jika menemukan bola berwarna orange melalui mini komputer yaitu Raspberry pi, robot akan memproses posisi bola yang telah dibagi dalam tiga buah keadaan yaitu tepat di depan robot, di kanan robot, dan di kiri robot. Selanjutnya jika hasil olahan dari raspberry pi telah menentukan keadaan posisi bola terhadap robot maka akan dikirim ke mikrokontroler utama. Mikrokontroler utama akan memproses untuk menggerakkan aktuator agar robot bergerak ke kanan jika bola pada posisi kanan dari robot, ke kiri jika bola pada posisi kiri dari robot, dan bergerak ke depan jika bola pada posisi tepat di depan dari posisi robot. Jika tidak menemukan bola berwarna orange robot akan terus melakukan proses pencarian dengan berputar dan penambahan variasi manuver gerakan agar pencarian tidak membuang waktu lama.

3.2.1.2 Perancangan Mekanika Robot

Perancangan mekanika pada skripsi ini hanya sebagai aspek praktis yaitu membuat robot *bi-drirectional* yaitu robot yang terdiri dari dua buah roda dan ditambahkan roda bebas *ball caster* sebagai penyeimbang ketika robot akan bergerak. Dalam perancangan ini mekanika robot menggunakan bahan dasar berupa *acrylic* dengan ketebalan 3 milimeter.

Berdasarkan peraturan perlombaan Kontes Robot Sepak Bola Indonesia (KRSBI) tahun 2016, batasan dimensi robot adalah 40×30 dengan urutan lebar dan tinggi. Dengan adanya peraturan tersebut, maka dimensi robot harus dirancang tidak melebihi batas dimensi yang telah ditetapkan. Gambar perspektif dari robot secara keseluruhan ditunjukkan dalam Gambar 3.2.



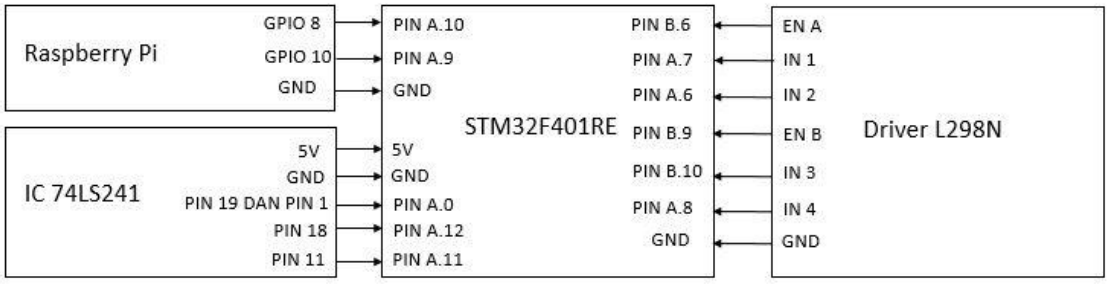
Gambar 3.2 Perspekstif robot secara keseluruhan

3.2.1.3 Perancangan Rangkaian Antarmuka Mikrokontroler Utama

Pada perancangan perangkat keras robot ini menggunakan modul mikrokontroller ARM sebagai pengolah utama untuk pemrosesan algoritma dan data sensor kamera dari raspberry pi. Mikrokontroller mempunyai 5 port dan 50 GPIO dapat diprogram menjadi masukan atau keluaran. Pada perancangan ini pin-pin yang digunakan adalah :

PIN A.10 = dihubungkan dengan pin TX GPIO nomor 8 pada raspberry pi
PIN A.9 = dihubungkan dengan pin RX GPIO nomor 10 pada raspberry pi
PIN B.6 = dihubungkan dengan masukan PWM *drive*r pada *ENABLE* A
PIN B.9 = dihubungkan dengan masukan PWM *drive*r pada *ENABLE* B
PIN A.7 = dihubungkan dengan pin INPUT 1 pada rangkaian *driver*
PIN A.6 = dihubungkan dengan pin INPUT 2 pada rangkaian *driver*
PIN B.10 = dihubungkan dengan pin INPUT 3 pada rangkaian *driver*
PIN A.8 = dihubungkan dengan pin INPUT 4 pada rangkaian *driver*
PIN A.0 = dihubungkan dengan pin IC 74LS241N nomor 1 dan 19
PIN A.12 = dihubungkan dengan pin IC 74LS241N nomor 18
PIN A.11 = dihubungkan dengan pin IC 74LS241N nomor 11

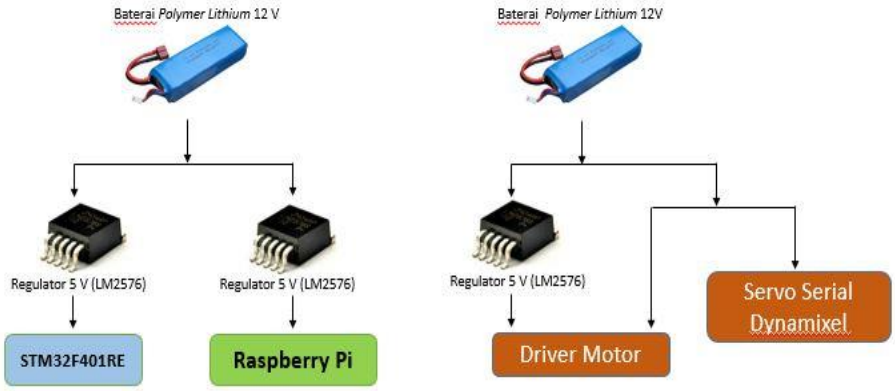
Mikrokontroler ARM merupakan kontroler utama yang nantinya akan memproses data dari sensor kamera yang data nya dikirim melalui raspberry pi. Mikrokontroler ARM juga berfungsi sebagai kontroler utama untuk mengatur pergerakan aktuator. Adapun diagram blok antarmuka mikrokontroler utama secara keseluruhan ditunjukkan dalam Gambar 3.3



Gambar 3.3 Diagram blok antarmuka mikrokontroler utama secara keseluruhan

3.2.1.4 Perancangan Catu Daya

Catu daya pada keseluruhan sistem dibagi menjadi dua, yaitu 5 V dan 12 V. Sumber catu daya yang dipakai adalah dua buah baterai lipo (*lithium polimer*) 12 V. Catu daya 5 V diperoleh dari rangkaian regulator simple switcher LM2576. Diagram blok catu daya pada sistem keseluruhan robot ditunjukkan dalam Gambar 3.4.

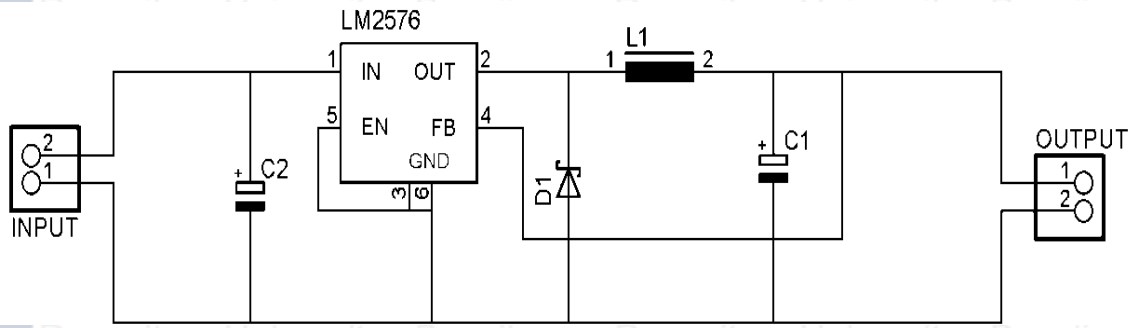


Gambar 3.4 Diagram blok catu daya dari sistem keseluruhan

Penggunaan dua buah baterai pada alat ini bertujuan agar robot yang dirancang layaknya robot yang akan berlomba, dengan menggunakan baterai khusus untuk aktuator serta baterai khusus untuk mikrokontroler dan mini komputer.

3.2.1.5 Perancangan Catu Daya 5 V

Alat ini membutuhkan catu daya 5V untuk mencatu mikrokontroler dan raspberry pi. Sedangkan sumber catu daya utama berupa baterai li-po 11.3 V, sehingga dibutuhkan suatu rangkaian untuk menurunkan tegangan menjadi 5V. Pada perancangan ini digunakan sebuah IC regulator *simple switcher* LM2576 dengan spesifikasi keluaran tegangan 5V dan arus maksimal 3A. Rangkaian regulator tegangan LM2576 ditunjukkan dalam Gambar 3.5.



Gambar 3.5 Rangkaian regulator tegangan LM2576

Rangkaian regulator tegangan dengan IC LM2576 membutuhkan komponen tambahan berupa kapasitor, induktor, dan dioda. Dioda D_1 adalah dioda tipe 1N5822 yang merupakan dioda *schottky*. Kapasitor C_1 bernilai 1000 µF sedangkan kapasitor C_2 bernilai 100 µF. Sedangkan Induktor L_1 memiliki nilai 100 µH. Rangkaian dan spesifikasi komponen didapatkan dari datasheet LM2576.

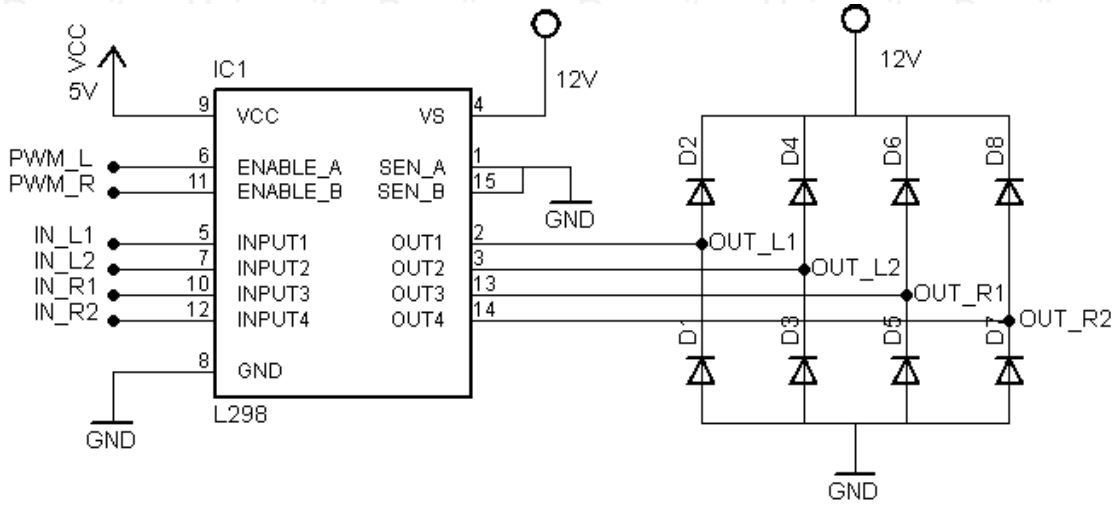
Regulator memiliki keluaran berupa sinyal kotak yang memiliki *duty cycle* atau t_{on} dan t_{off} yang dikontrol langsung berdasarkan besar tegangan umpan balik dari output. Kapasitor C_2 berfungsi untuk mengurangi *ripple* pada tegangan input. Dioda yang digunakan adalah dioda *schottky* karena memiliki respon yang sangat tinggi dibandingkan dioda biasa. Induktor digunakan untuk meningkatkan efisiensi rangkaian. Kapasitor C_1 digunakan untuk mengubah sinyal keluaran menjadi linear.

3.2.1.6 Perancangan Rangkaian *Driver* L298N

Untuk mengendalikan arah dan mengatur kecepatan motor DC, digunakan IC L298N. IC L298N adalah *driver* motor yang berbasis H-Bridge dengan memiliki dua buah kanal /*channel*, oleh sebab itu satu buah rangkaiam *driver* L298N mampu mengendalikan dua buah motor DC sekaligus. IC L298N mampu menangani beban setiap kanalnya hingga 2 A. Spesifikasi IC L298N meliputi:

- IO max = 2 A
- Vs = 2,5 – 46 V (*Power Supply*)
- Vss = 4,5 – 7 V (*Logic Supply Voltage*)
- Ven = -0,3 – 7 V (*Input and Enable Voltage*)

Driver dilengkapi dengan 1 indikator LED. Indikator LED berfungsi sebagai penanda adanya sumber tegangan 5 V dari catu daya yang merupakan masukkan pin VCC dari *driver*. Adapun skema rangkaian *driver* motor L298N ditunjukkan dengan Gambar 3.6.



Gambar 3.6 Rangkaian *driver* motor L298N

Pada perancangan ini masing-masing pin yang digunakan adalah:

INPUT1 = dihubungkan dengan keluaran logika arah motor kiri PIN A.7 pada rangkaian mikrokontroler utama *(Master)*

INPUT2 = dihubungkan dengan keluaran logika arah motor kiri PIN A.6 pada rangkaian mikrokontroler utama *(Master)*

INPUT3 = dihubungkan dengan keluaran logika arah motor kanan PIN B.10 pada rangkaian mikrokontroler utama *(Master)*

INPUT4 = dihubungkan dengan keluaran logika arah motor kanan PIN A.8 pada rangkaian mikrokontroler utama *(Master)*

ENABLE_A = dihubungkan dengan keluaran sinyal PWM motor kanan PIN B.6 pada rangkaian mikrokontroler utama *(Master)*

ENABLE_B = dihubungkan dengan keluaran sinyal PWM motor kiri PIN B.9 pada rangkaian mikrokontroler utama *(Master)*

OUT1, OUT2 = dihubungkan dengan masukan motor kiri

OUT3, OUT4 = dihubungkan dengan masukan motor kanan

PIN VS = dihubungkan dengan sumber 12V

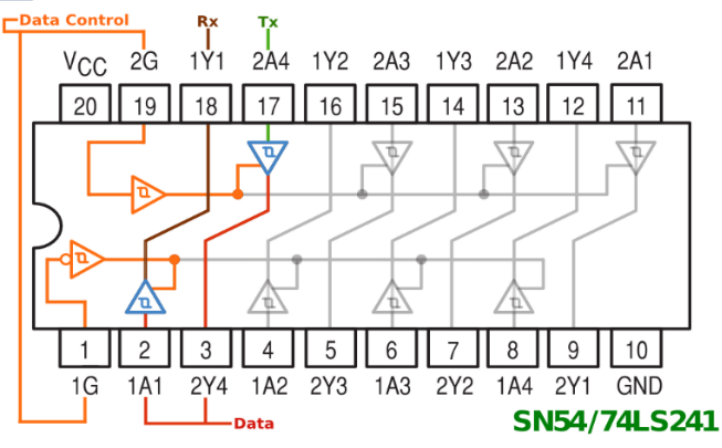
PIN VCC = dihubungkan dengan sumber 5V

PIN GND = dihubungkan dengan *ground*

3.2.1.7 Perancangan Rangkaian *Octal Buffer* 74LS241N

Untuk menggerakkan sensor kamera digunakan aktuator berupa *servo serial dynamixel*. Masalah pada *servo serial dynamixel* adlah komunikasi antara mikrokontroler utama dengan

servo serial dynamixel. Mikrokontroler utama menggunakan komunikasi berbasis *full duplex* atau dengan dua jalur data satu mengirim, dan satu menerima, sedangkan dynamixel membutuhkan komunikasi serial berbasis *half duplex*, satu jalur untuk berkomunikasi. Oleh sebab itu dibutuhkan sebuah *interface* lain untuk menghubungkan mikrokontroler utama dan dynamixel atau dengan kata lain *full duplex* ke *half duplex* menggunakan IC 74LS241N. IC 74LS241N adalah sebuah gerbang yang memilah-milah apakah data yang dialamatkan sebagai data *transfer* atau data *receiver*. Skema rangkain *octal buffer* 74LS241N ditunjukkan dalam Gambar 3.7.



Gambar 3.7 Rangkaian *octal buffer* 74LS241

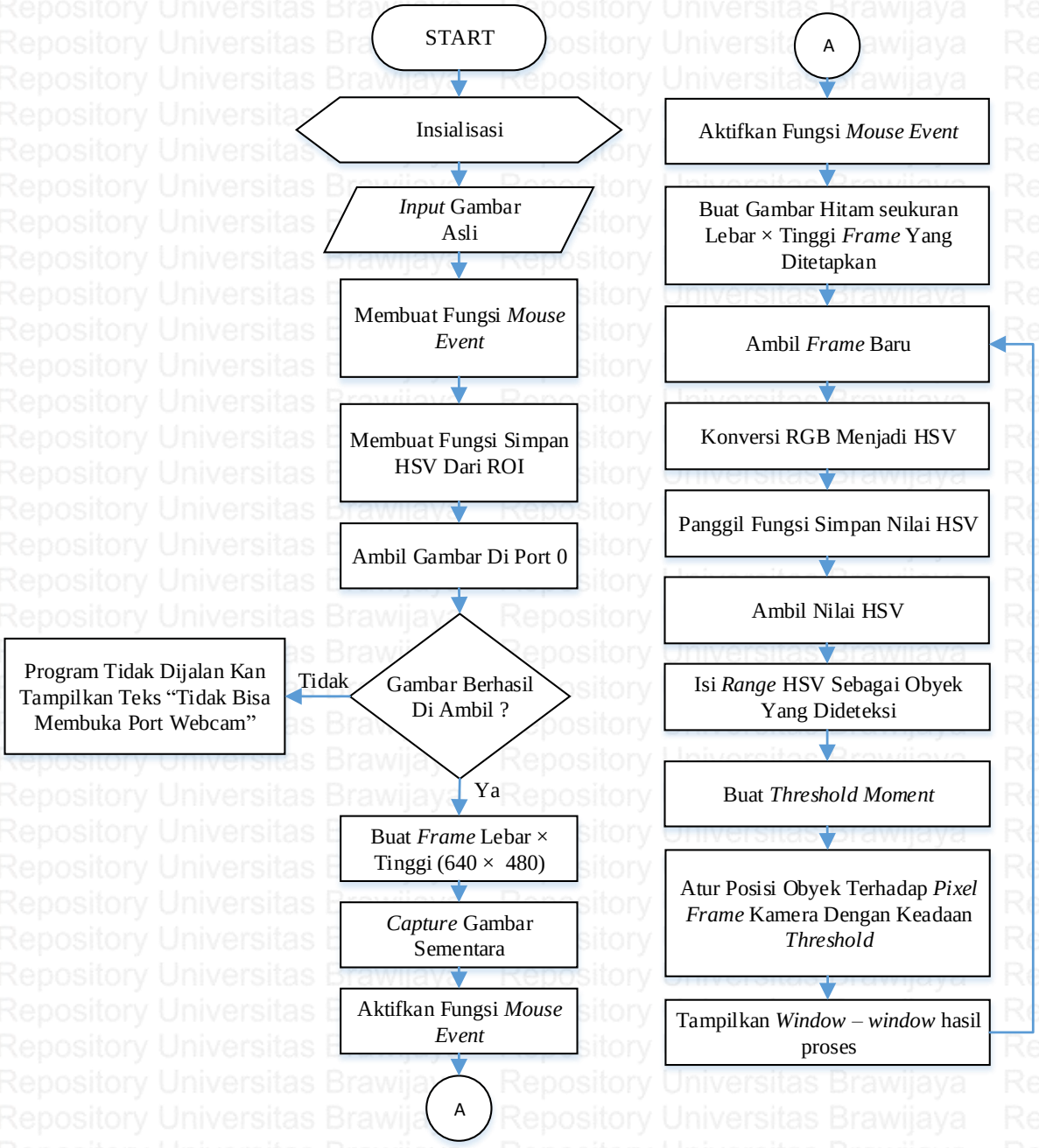
3.2.1.8 Perancangan *Board* Utama

Untuk mengoptimalkan dimensi yang telah diatur pada alat ini, maka dirancang *board* utama. *Board* utama merupakan bagian dari mekanika robot yang menggabungkan seluruh skema rangkaian elektrik yang akan digunakan. Rangkaian board utama terdiri dari beberapa *pinhead*, rangkaian LED indikator untuk *driver* L298N dan rangkaian *octal buffer*.

3.2.2 Perancangan dan Pembuatan Perangkat Lunak (*Software*)

3.2.2.1 Perancangan *Overview* Algoritma Deteksi Warna HSV dengan ROI

Untuk deteksi warna menggunakan HSV dengan ROI diperlukan beberapa proses sampai obyek bola yang dideteksi dapat diterima posisinya dalam koordinat *pixel* kamera. Tahap-tahap proses tersebut akan menjadi acuan untuk membuat program deteksi warna HSV. Adapun *overview* algoritma metode deteksi warna HSV ditunjukkan dalam Gambar 3.8.



Gambar 3.8 Overview algoritma deteksi warna HSV

Pada awalnya dilakukan inisialisasi variabel – variabel untuk fungsi UART, ROI, dan *Mouse Event*. Kemudian dibuat fungsi *mouse event* menggambar kotak dua dimensi untuk memperoleh ROI dengan mendrag *mouse*. Selanjutnya dibuat fungsi simpan nilai HSV dari hasil ROI untuk menampung nilai HSV dari area kotak yang diseleksi. Gambar asli akan diterima di port "0" yang merupakan nomor port kemera *webcam* pada Raspberry pi dengan lingkup warna dasar berupa RGB. Lebar × tinggi *frame* gambar diatur sebesar 640 x 480. Gambar sementara

akan di *capture*, hasil *capture* OpenCV pada umumnya bernilai 8 bit *unsigned integer* dengan *range* nilai 0 – 255. Kemudian fungsi *mouse event* diaktifkan untuk melakukan proses *dragging mouse* diawal. Setelah itu dibuat gambar hitam dengan ukuran *frame pixel* 640 × 480, hal ini dilakukan untuk menampilkan *window threshold* dari obyek yang di deteksi dengan konsep binerisasi dimana warna hitam merupakan *background* dan putih sebagai *foreground* yaitu obyek yang dideteksi. Kemudian dilakukan konversi dari RGB menjadi HSV. Untuk melakukan proses konversi RGB menjadi HSV awalnya 8 bit nilai dari R, G, dan B dikonversi kedalam format *floating point* dan menyesuaikan skala *range* 0 ke 1 kemudian dilakukan konversi secara berurutan meliputi persamaan (3-1) sampai persamaan (3-3) berikut:

Perhitungan Value :

$$V = \max(R, G, B) \tag{3-1}$$

Perhitungan Saturation

$$S = \begin{cases} \dfrac{V - \min(R,G,B)}{V} & Jika\ V \neq 0 \\ 0 & jika\ V = 0 \end{cases} \tag{3-2}$$

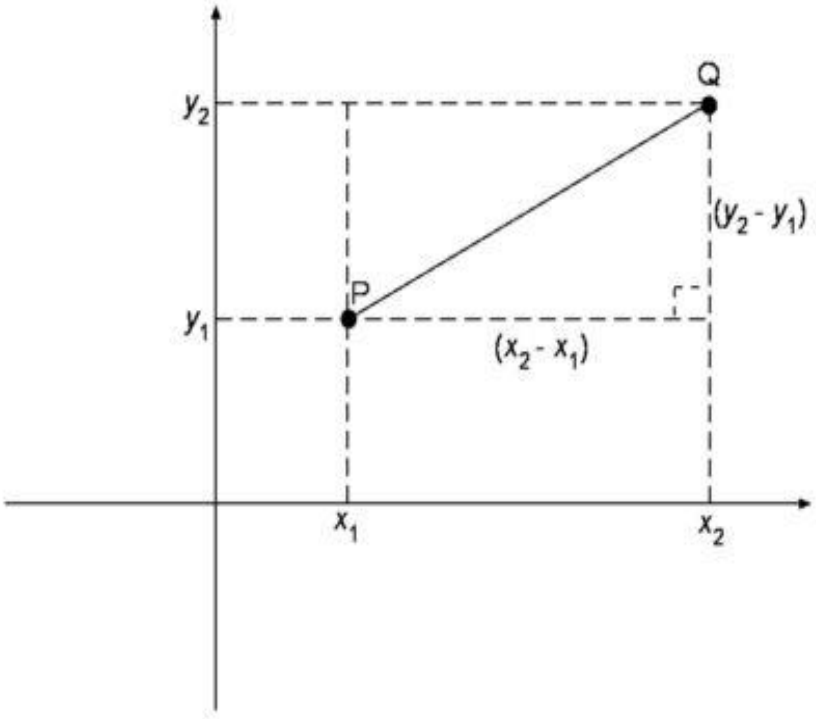
Perhitungan *Hue* :

$$H = \begin{cases} \dfrac{60\ (G-B)}{V - \min(R,G,B)}\ Jika\ V = R \\ \dfrac{120 + 60\ (B-R)}{V - \min(R,G,B)}\ Jika\ V = G \\ \dfrac{240 + 60\ (R-G)}{V - \min(R,G,B)}\ Jika\ V = B \end{cases} \tag{3-3}$$

Setelah proses konversi RGB ke HSV selesai dilakukan, diambil nilai HSV hasil ROI dengan mengaktifkan fungsi simpan nilai HSV yang telah dibuat sebelumnya. Setelah itu dilakukan proses morphologi erode dan dilasi, proses ini bertujuan agar obyek yang di deteksi yaitu *foreground* terhilang dari titik-titik kecil akibat dari pencampuran warna HSV lalu hasil warna HSV seleksi ROI ditampilkan pada *window threshold.* Hasil deteksi kemudian diterjemahkan kedalam koordinat X dan Y berdasarkan lebar dan tinggi *frame pixel* yang telah diatur sebelumnya.

3.2.2.2 Menggambar Kotak Dengan Menggunakan fungsi *mouse event* untuk Memperoleh ROI

Untuk memperoleh *region of interest* secara cepat digunakan fungsi *mouse event*. *Mouse event* berfungsi untuk menggambar kotak pada kulit bola atau pada bagian berwarna orange. Dengan menggunakan *mouse* maka area diseleksi awalnya akan didefinisi dalam bentuk koordinat X dan Y. Untuk menggambar kotak yang pertama harus diketahui ialah garis antara dua titik koordinat dalam dua dimensi sesuai dengan Gambar 3.9. Dengan menggunakan teorema Phytagoras maka dapat diperoleh persamaan (3-4).



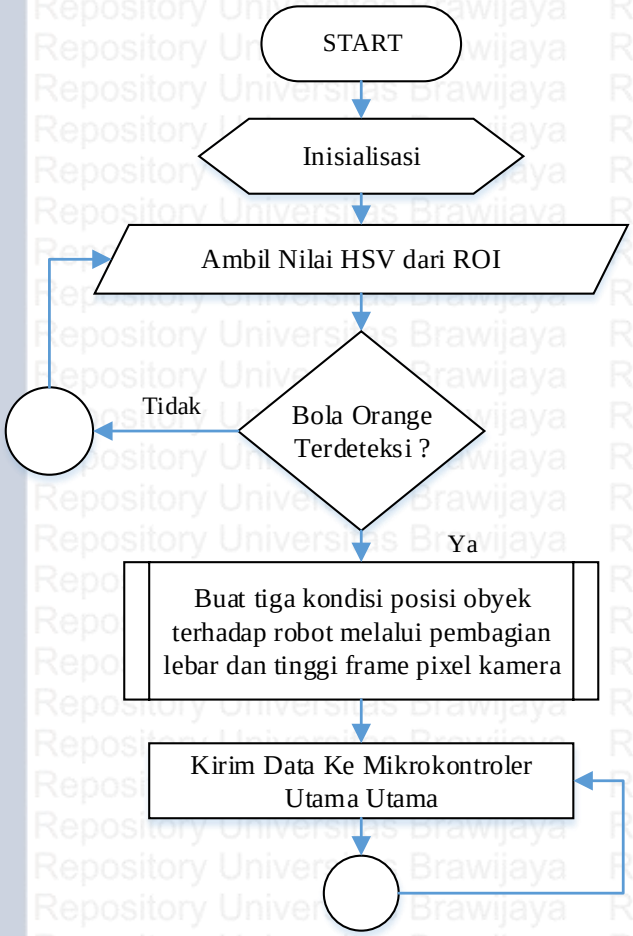
Gambar 3.9 Jarak antara kedua titik P dan Q dalam ruang dua dimensi

$$PQ = \sqrt{(X_2 - X_1)^2 + (Y_2 - Y_1)^2} \quad (3\text{-}4)$$

Berdasarkan persamaan (3-4) garis diagonal dari titik P dan Q dapat diperoleh berdasarkan koordinat X_1, Y_1 dan X_2, Y_2. Dengan menggunakan fungsi *mouse event* koordinat X_1, Y_1 dan X_2, Y_2 dapat diperoleh ketika mouse di drag pada *frame* gambar. Selanjutnya garis – garis untuk membentuk kotak berdasarkan garis diagonal diperoleh dengan pengurangan titik – titik koordinat maksimum dan minimum dari masing - masing koordinat X dan Y yang telah diketahui dan diproses kedalam fungsi menggambar kotak (*rectangle*) pada OpenCV.

3.2.2.3 Perancangan Diagram Alir Proses Pengiriman Data Hasil Deteksi warna HSV dengan ROI

Setelah obyek bola dapat dideteksi dengan menggunakan detksi warna HSV dengan ROI selanjutnya dirancang diagram alir pengiriman data nya ke mikrokontroler utama. Diagram alir proses pengiriman data hasil deteksi warna HSV dengan ROI pada raspberry pi ditunjukkan dalam Gambar 3.10.



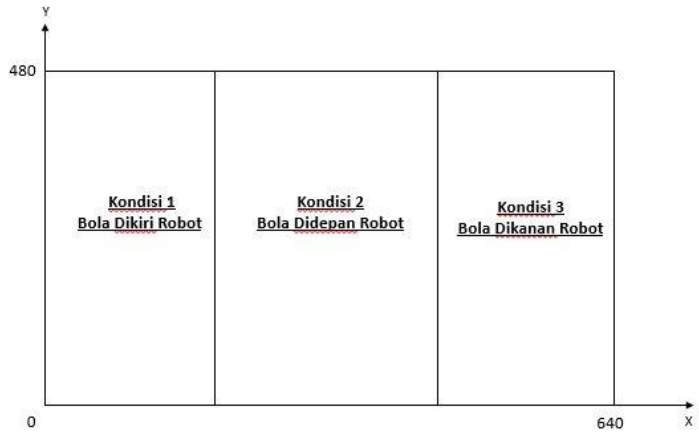
Gambar 3.10 Diagram alir pengiriman data hasil deteksi warna HSV pada raspberry pi

Ketika program deteksi warna HSV dijalankan maka nilai parameter HSV akan diatur pada kondisi awal yaitu Hmin = 0, Hmax = 179, Smin = 0, Smax = 255, Vmin = 0 , dan Vmax = 255. Dengan kondisi awal tersebut artinya belum ada pencampuran nilai HSV untuk seleksi warna tertentu. Selanjutnya nilai-nilai parameter tersebut akan diambil dari hasil ROI yang dilakukan dengan menggunakan fungsi *mouse event* untuk mendrag kotak pada kulit bola. Deteksi bola orange ditentukan oleh persamaan (3-5) berikut :

$$P_{(x,y)} \begin{cases} Bola, & H_{min} < H_{(x,y)} < H_{max} \ dan \\ & S_{min} < S_{(x,y)} < S_{max} \ dan \\ & V_{min} < V_{(x,y)} < V_{max} \\ Bukan\ bola\ ,\ kondisi\ lainnya \end{cases} \quad (3\text{-}5)$$

$P_{(x,y)}$ adalah *pixel* dikoordinat (x,y), jika bola dideteksi dalam *pixel* tersebut maka, H_{min} dan H_{max} berturut-turut adalah nilai inisialisasi (Hasil H ROI) minimal dan maksimal, kemudian S_{min} dan S_{max} berturut-turut adalah nilai inisialisasi (Hasil S ROI) minimal dan maksimal, lalu V_{min} dan V_{max} berturut-turut adalah nilai inisialisasi (Hasil V ROI) minimal dan maksimal. Jika bola orange sudah terdeteksi maka selanjutnya posisi bola terhadap robot akan diatur berdasarkan pembagian lebar dan tinggi *pixel* kamera.

Untuk mengetahui posisi bola terhadap robot diperlukan penentuan kondisi posisi bola. Penentuan kondisi posisi bola dilakukan dengan membagi lebar dan tinggi *frame pixel* kamera, lebar dan tinggi *frame pixel* kamera telah diatur sebelumnya yaitu lebar × tinggi yaitu 640 × 480. Adapun letak obyek terhadap robot melalui pembagian lebar dan tinggi *frame pixel* kamera ditunjukkan dalam Gambar 3.11.

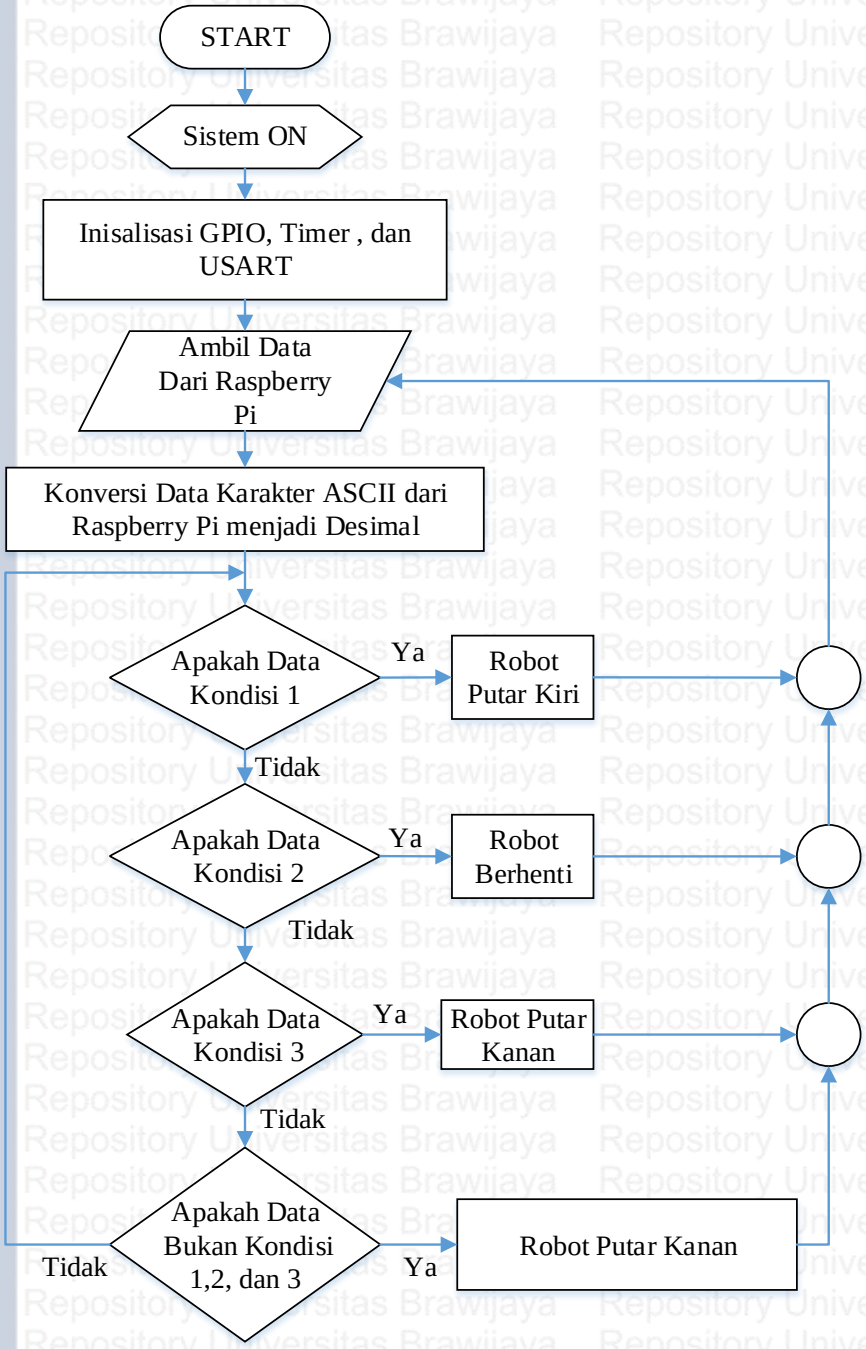


Gambar 3.11 Koordinat bola melalui pembagian lebar dan tinggi *frame pixel* kamera

Untuk pembagian lebar dan tinggi *frame pixel* di notasikan sebagai koordinat x dan y. Pembagian koordinat x dan y dilakukan pada Raspberry Pi yang telah diinstal *library* OpenCV. Setelah dilakukan pembagian koordinat x dan y maka masing – masing kondisi tersebut akan diwakilkan oleh karakter ASCII misalkan kondisi 1 yaitu bola di kiri robot diwakilkan sebagai karakter ASCII "A". Selanjutnya data berupa karater ASCII tersebut dikirimkan ke mikrokontroler utama melalui Raspberry Pi 2 menggunakan komunikasi serial.

3.2.2.4 Diagram Alir Perancangan Pada Mikrokontroler Utama

Mikrokontroler utama utama merupakan pusat pengolah data dan proses utama algoritma pergerakan atau bernavigasi. Adapun diagram alir perancangan pada mikrokontroler utama ditunjukkan dalam Gambar 3.12.



Gambar 3.12 Diagram alir perancangan pada mikrokontroler utama

Tahap pertama saat program dijalankan adalah inisialisasi seluruh variabel dan *peripheral* yang digunakan pada mikrokontroler. Selanjutnya mikrokontroler utama akan menerima data

yang dikirim dari Raspberry pi melalui komunikasi UART. Ketika mikrokontroler utama menerima data yang dikirim dari raspberry pi. dilakukan proses konversi data dari karakter ASCII menjadi desimal. Hasil konversi kemudian dibaca kedalam empat keadaan berbeda yaitu kondisi 1, kondisi 2, kondisi 3, dan bukan kondisi 1 , 2, dan 3. Masing-masing keadaan yang terbaca sebagai acuan navigasi robot yaitu, kondisi 1 maka robot putar kiri, kondisi 2 maka robot akan berhenti, kondisi 3 maka robot putar kanan, dan bukan kondisi 1, 2, dan 3 maka robot akan melakukan manuver mencari bola secara acak/*random* dengan cara melakukan gerakan putar kanan searah jarum jam.

3.3 Pengujian Alat

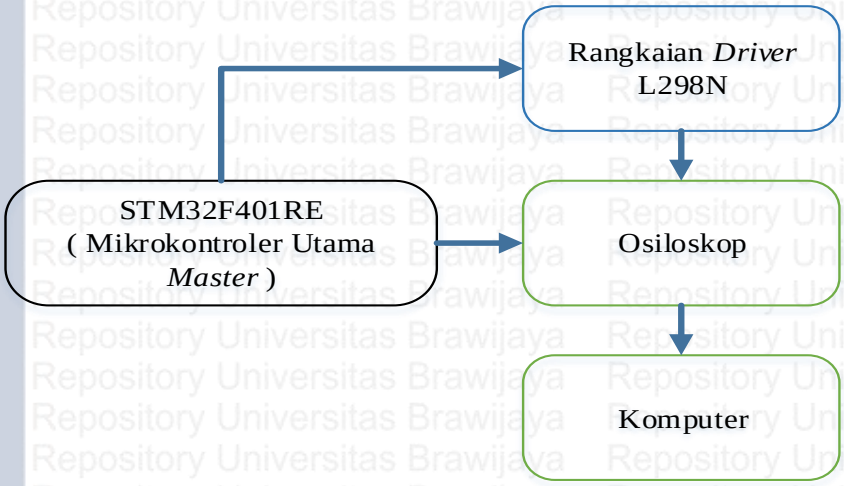
Untuk menganalisis kinerja alat maka perlu dilakukan pengujian sistem agar diketahui apakah sesuai dengan yang direncanakan. Pengujian dilakukan setiap bagian blok diagram untuk mengetahui hasil perancangan. Pengujian yang dilakukan meliputi :

3.3.1 Pengujian Rangkain *Driver* Motor L298N

3.3.1.1 Pengujian *Duty Cycle* Rangkaian Mikrokontroler dan *Driver* L298N

Pengujian ini dilakukan untuk mengetahui kinerja dan respon dari rangkaian *driver* motor L298N dengan membandingkan sinyal *duty cycle* pada rangkaian *driver* terhadap sinyal keluaran *duty cycle* PWM dari mikrokontroler. Alat yang digunakan untuk pengujian ini yaitu STM32F401RE (mikrokontroler utama sebagai *master*), catu daya 5V dan 12V, rangkaian *driver* L298N, osiloskop digital, dan komputer.

Prosedur pengujian dilakukan dengan menghubungkan rangkaian *driver* motor L298N, mikrokontroler utama dan osiloskop sesuai dengan skema pengujian pada Gambar 3.13. Dengan memberikan sinyal masukan *duty cycle* PWM dari range 0% sampai 100% pada pin masukan *driver* dengan kenaikan 10%. Maka dengan menggunakan osiloskop digital Velleman PCLAB PCSU1000 dan perangkat komputer sinyal persegi keluaran dari mikrokontroler dan *driver* L298 dapat diamati. Pengujian ini dilakukan dengan membandingkan nilai *duty cycle* masukan dari mikrokontroler dengan *duty cycle* yang dikeluarkan oleh rangkaian *driver* L298N.

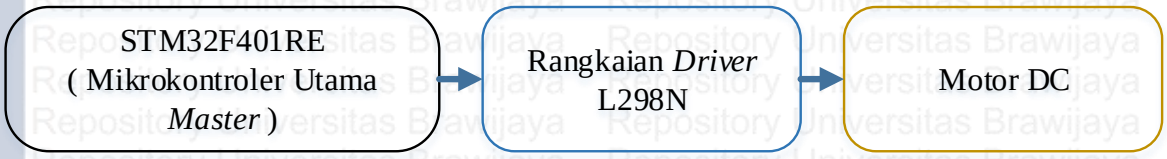


Gambar 3.13 Skema pengujian *duty cycle* rangkaian *driver* L298N terhadap sinyal masukan PWM

3.3.1.2 Pengujian Arah Gerak Motor

Pengujian ini dilakukan untuk mengetahui gerak motor apakah telah sesuai dengan masukan arah *driver* L298N terhadap pin keluaran digital dari mikrokontroler. Alat yang digunakan untuk pengujian ini yaitu STM32F401RE (mikrokontroler utama sebagai *master*), catu daya 5V dan 12V, rangkaian *driver* L298N, dan dua buah motor DC.

Prosedur pengujian dilakukan dengan menghubungkan *driver* motor L298N, mikrokontroler utama dan motor DC sesuai dengan skema pengujian pada Gambar 3.14. Mikrokontroler akan memberikan instruksi arah dan kecepatan pada rangkaian *driver* L298N. Dengan memberikan sinyal pin logika arah motor 0 atau 1 ke rangkaian *drive*r L298N maka gerak motor dapat diamati.



Gambar 3.14 Skema pengujian arah gerak motor

3.3.2 Pengujian *Servo Serial Dynamixel*

3.3.2.1 Pengujian Sudut *Servo Serial Dynamixel*

Pengujian ini dilakukan untuk mengamati apakah *servo serial dynamixel* dapat bergerak dengan perhitungan berdasarkan besar data serial dan dibuktikan dengan busur derajat. Data serial dikirim oleh mikrokontroler utama dengan menggunakan komunikasi UART data serial kemudian ditampilkan pada komputer. Alat yang digunakan untuk pengujian ini yaitu

STM32F401RE (mikrokontroler utama sebagai *master*), rangkaian *octal buffer* ,catu daya 5V dan 12V, busur 360°, dan komputer.

Prosedur pengujian dilakukan dengan mengirimkan data serial ke *servo dynamixel* dari 0 – 1023 dengan kenaikan 100. Karena *servo serial dynamixel* memiliki range sudut 0° - 300° maka besar perubahan sudut dapat dihitung dengan persamaan berikut :

$$Sudut\ Perhitungan\ (°) = \frac{Nilai\ Data\ Serial\ (0-1023)}{1023} \times 300° \quad (3\text{-}6)$$

Skema pengujian sudut servo serial dynamixel ditunjukkan dalam Gambar 3.15.

Gambar 3.15 Skema pengujian sudut *servo serial dynamixel*

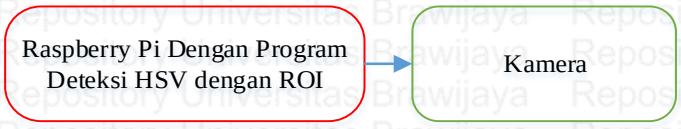
3.3.3 Pengujian Sensor Kamera Dengan Hasil Olahan OpenCV

3.3.3.1 Pengujian Jumlah *Frame* Yang Dapat Diproses Dalam Satu Detik

Pengujian ini dilakukan untuk mengetahui nilai rata-rata jumlah *frame* yang mampu diproses dalam satu detik dari sensor kamera pada Raspberry pi. Alat yang digunakan untuk pengujian ini yaitu raspberry pi, catu daya 5V dan kamera. *Frame per second* (fps) merupakan satuan untuk menyatakan Jumlah bingkai gambar atau *frame* yang ditunjukkan setiap detik dalam membuat gambar bergerak. Dalam OpenCV banyak *frame* yang mampu di proses dalam satu detik dapat diketahui menggunakan persamaan berikut :

$$fps = \frac{Jumlah\ frame\ yang\ di\ proses}{waktu\ yang\ dibutuhkan\ (\ detik\ )} \quad (3\text{-}7)$$

Pengujian dilakukan dengan menaikkan jumlah *frame* yang diproses dari 20 *frame* sampai 120 *frame* dengan kenaikan bertahap sebesar 20 *frame*. Setelah itu menghitung rata – rata banyak *frame* yang diproses dalam satu detik dari masing – masing jumlah *frame* yang telah ditentukan. Skema pengujian jumlah *frame* yang di proses dalam satu detik ditunjukkan dalam Gambar 3.16.



Gambar 3.16 Skema pengujian jumlah *frame* dalam satu detik

3.3.3.2 Pengujian Tampilan *Window* Proses Deteksi

Pengujian ini dilakukan untuk mengetahui apakah program deteksi yang telah dibuat telah sesuai dengan perancangan algoritma yang telah dibuat dan dapat bekerja dengan baik. Alat yang digunakan untuk pengujian ini yaitu raspberry pi, kamera, bola orange, catu daya 5V dan lapangan karpet berwarna hijau.

Prosedur pengujian dilakukan secara *real time* dengan menjalankan program pada raspberry pi dan mengikuti alur proses untuk memperoleh nilai HSV bola. Selanjutnya mengamati tampilan *window – window* dari proses deteksi sebagai acuan apakah proses deteksi telah sesuai dengan perancangan algoritma yang dibuat. Skema pengujian tampilan *window* proses deteksi ditunjukkan dalam Gambar 3.17.

Gambar 3.17 Skema pengujian tampilan *window* proses deteksi

3.3.3.3 Pengujian Nilai HSV Berdasarkan Deteksi Warna HSV Dengan ROI

Pengujian ini dilakukan untuk mengetahui nilai rata-rata *Hue* (H), *Saturation* (S), dan *Value* (V) dari bola berwarna orange pada lapangan karpet berwarna hijau dengan deteksi warna HSV dengan penambahan fungsi ROI. Alat yang digunakan untuk pengujian ini yaitu raspberry pi, kamera, bola orange, catu daya 5V dan lapangan karpet berwarna orange.

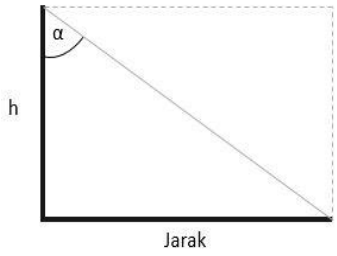
Prosedur pengujian dilakukan secara *real time* dengan mengambil nilai HSV dari ROI pada program sampai warna yang diinginkan bisa terdeteksi. Jarak bola dengan kamera di atur sebesar 30 cm kemudian dengan menggunakan fungsi *mouse event* kulit bola yang berwarna orange diseleksi dengan bentuk kotak untuk memperoleh nilai HSV berdasarkan fungsi ROI. Nilai HSV pada program memiliki lebar data sebesar 8 bit dengan *range*, yaitu *Hue* (0-255), *Saturation* (0-255), dan *Value* (0-255). Data hasil pengujian merupakan *range* data tertinggi dan terendah dari 10 kali pengujian yang dilakukan menggunakan kamera dan raspberry pi. Data yang diambil merupakan *range* nilai digital yang diambil pada 10 titik yang berbeda. Skema pengujian nilai HSV berdasarkan deteksi warna HSV dengan ROI ditunjukkan dalam Gambar 3.18.

Gambar 3.18 Skema pengujian nilai HSV berdasarkan deteksi warna HSV dengan ROI

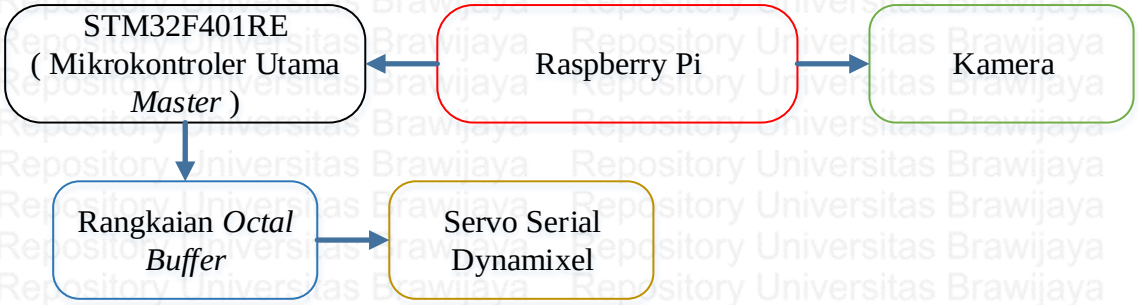
3.3.3.4 Pengujian Jarak Maksimal Bola Terdeteksi

Pengujian ini dilakukan untuk mengetahui jarak maksimum bola mampu di deteksi menggunakan program deteksi warna HSV dengan ROI. Alat yang digunakan untuk pengujian ini yaitu mekanika robot secara robot keseluruhan, raspberry pi, rangkaian *octal buffer*, *servo serial dynamixel*, catu daya 5V dan kamera.

Prosedur pengujian dilakukan dengan menempatkan sensor kamera di mekanika robot keseluruhan pada bagian kepala, kemudian menjalankan program deteksi warna HSV dengan ROI. *Range* nilai HSV yang digunakan merupakan rata-rata hasil pengujian nilai HSV sebelumnya. Perhitungan jarak menggunakan trigonometri pada persamaan (3-8) ilustrasi sistem trigonometri ditunjukkan dalam Gambar 3.19. Besar perubahan sudut pada sendi kepala dan tinggi pada kamera pada kepala robot dari permukaan tanah digunakan untuk menentukan jarak bola dari robot. Adapun skema pengujian jarak bola terdeteksi ditunjukkan dalam Gambar 3.20.



Gambar 3.19 Ilustrasi sistem trigonometri



Gambar 3.20 Skema pengujian jarak bola terdeteksi

$$Jarak = \frac{h}{\tan(90 - \alpha)} \quad (3\text{-}8)$$

Berdasarkan persamaan (3-7) diperoleh nilai h merupakan tinggi robot yang diukur menggunakan penggaris melalui permukaan datar. Sudut α merupakan sudut yang terukur pada nilai data serial servo dynamixel, sebagai kepala robot tempat sensor kamera dilekatkan. Jarak horizontal diukur dari titik tengah posisi kamera hingga titik tengah benda yang dideteksi.

3.3.3.5 Pengujian Posisi Bola Terhadap Robot

Pengujian ini dilakukan untuk mengetahui apakah *range* pembagian lebar dan tinggi *frame pixel* x dan y (640×480) kamera untuk menerjemahkan posisi bola terhadap robot telah sesuai dengan yang dirancang. Alat yang digunakan untuk pengujian yaitu mekanika robot secara keseluruhan, raspberry pi, catu daya 5V kamera, bola, dan lapangan karpet berwarna orange.

Proses pengujian dilakukan secara *real time* dengan menjalankan program deteksi warna HSV dengan ROI di raspberry pi. Kemudian memasukkan nilai rata – rata dari parameter HSV bola hasil dari ROI dan menempatkan bola sejauh 30 cm dari robot. Setelah itu memposisikan bola terhadap robot sesuai dengan tiga keadaan kanan, depan dan kiri dari robot. Selanjutnya menampilkan hasil pembagian kedalam tiga kondisi yaitu “kanan robot”, “depan robot”, dan “kiri robot” pada terminal Raspberry pi beserta nilai terbaru dari *frame pixel* kamera. Skema pengujian posisi bola terhadap robot ditunjukkan dalam Gambar 3.21.

Gambar 3.21 Skema pengujian posisi bola terhadap robot

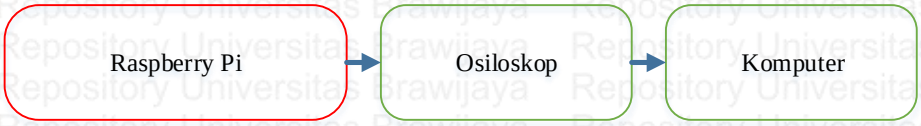
3.3.4 Pengujian Transmisi Data

3.3.4.1 Pengujian Sinyal Komunikasi Serial UART pada Mikrokontroler

Pengujian ini dilakukan untuk mengetahui apakah sinyal komunikasi serial UART telah sesuai dengan pengiriman data yang telah diprogram. Alat yang digunakan untuk pengujian ini yaitu catu daya 5V dan 12V, Raspberry pi (*slave*), osiloskop, dan komputer.

Proses pengujian dilakukan dengan mengatur perangkat UART pada raspberry pi dengan *baudrate* 9600 bps dan melakukan pengiriman data senilai 0xAA. Kemudian pin RX dari raspberry pi disambungkan pada osiloskop digital. Osiloskop digital yang digunakan adalah osiloskop Velleman PCLAB PCSU1000. Osiloskop digital digunakan untuk membaca bentuk

sinyal UART yang dikirimkan oleh Raspberry pi dan dibutuhkan komputer untuk menampilkan hasil pembacaan osiloskop. Skema pengujian sinyal komunikasi serial UART raspberry pi ditunjukan pada Gambar 3.22.



Gambar 3.22 Skema pengujian sinyal komunikasi UART Raspberry Pi

3.3.4.2 Pengujian Penerimaan Data Pada Mikrokontroler Utama

Pengujian ini dilakukan untuk mengetahui apakah data yang dikirim melalui raspberry pi sesuai dengan data yang diterima oleh mikrokontroler utama STM2F401RE. Alat yang digunakan untuk pengujian ini yaitu STM32F401RE (mikrokontroler utama sebagai *Master*), raspberry pi (*slave*), catu daya 5V, USB to TTL, dan komputer.

Proses pengujian dilakukan dengan mengatur perangkat UART pada raspberry pi dengan *baudrate* 9600 bps dan melakukan pengiriman data senilai 0XCC ke mikrokontroler utama STM32F401RE. Kemudian pin RX dari mikrokontroler utama disambungkan pada USB to TTL. Kemudian dengan menggunakan *aplikasi parallax serial terminal* diamati data yang telah diterima oleh mikrokontroler utama. Skema pengujian penerimaan data pada mikrokontroler utama ditunjukan pada Gambar 3.23.

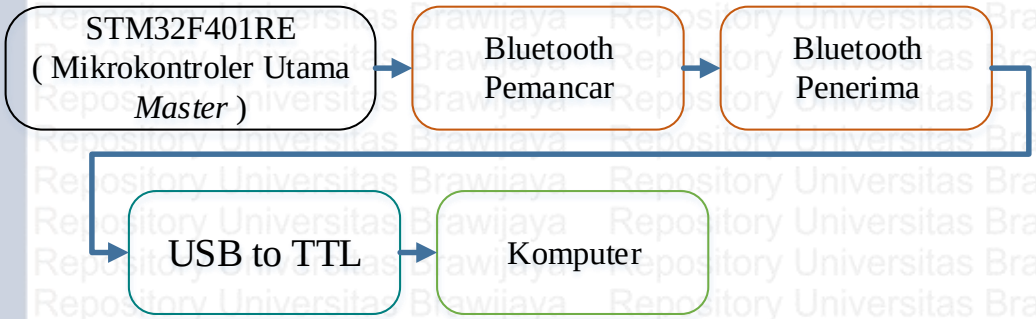
Gambar 3.23 Skema pengujian penerimaan data pada mikrokontroler utama

3.3.4.3 Pengujian Jarak Komunikasi *Bluetooth* HC-05

Pengujian ini dilakukan untuk mengetahui jarak maksimal yang dapat dijangkau oleh modul *Bluetooth* HC-05. Alat yang digunakan untuk pengujian ini yaitu STM32F401RE (mikrokontroler utama sebagai *Master*), USB to TTL, dua buah modul *bluetooth* HC-05 (*Transmitter* dan *Receiver*), catu daya 5V, dan komputer.

Prosedur pengujian dilakukan dengan mengirimkan data tulisan "Pengujian Jarak Komunikasi *Bluetooth* HC-05" dari perangkat mikrokontroler menggunakan modul *bluetooth* pemancar. Kemudian perangkat penerima yang berupa *bluetooth* dan USB to TTL akan menerima data yang dikirimkan oleh mikrokontroler yang dapat dilihat menggunakan parallax serial terminal pada komputer. Setelah itu jarak antara modul *bluetooth* HC-05 penerima dengan

modul *bluetooth* HC-05 pemancar yang ada pada robot diubah setiap satu meter hingga koneksi terputus. Skema pengujian jarak komunikasi *bluetooth* HC-05 dapat dilihat pada Gambar 3.24.

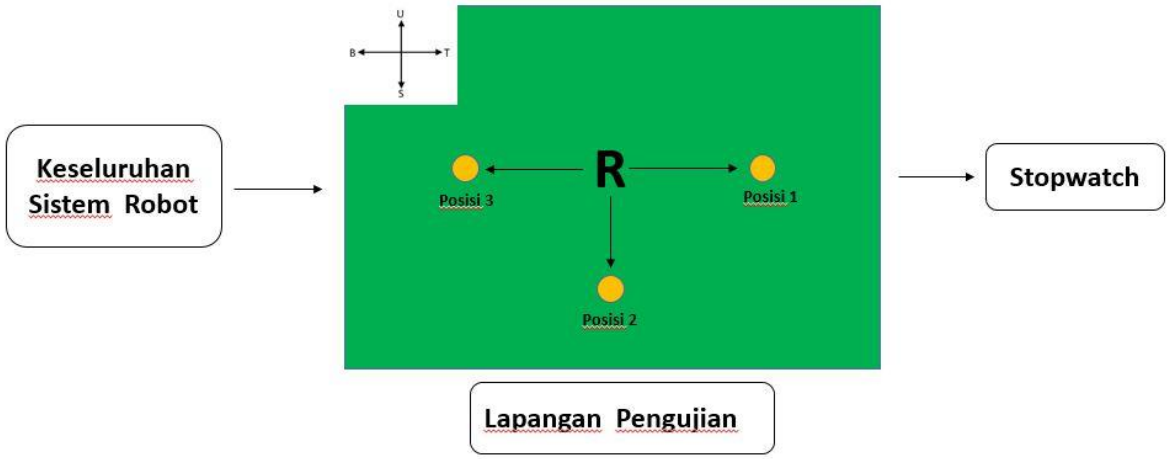


Gambar 3.24 Skema pengujian jarak komunikasi *bluetooth* HC-05

3.3.5 Pengujian Performa Sistem Keseluruhan

Pengujian ini dilakukan untuk mengetahui apakah setiap blok yang telah diuji dapat dirangkai menjadi satu sistem yang utuh dan dapat bekerja sesuai dengan perencanaan serta mengetahui performa robot dalam bernavigasi dengan mengikuti posisi bola. Alat yang digunakan untuk pengujian ini yaitu keseluruhan sistem robot, bola orange, lapangan karpet berwarna hijau, dan *stopwatch*. Ada beberapa parameter yang menentukan keberhasilan kerja robot. Parameter yang pertama adalah robot mampu melakukan navigasi putar kanan, putar kiri, dan jalan lurus. Parameter kedua adalah robot mampu mendeteksi bola orange dengan jarak tertentu. Parameter ketiga data yang dikirim dari raspberry pi ke mikrokontroler utama dapat selalu diamati di komputer. Parameter terakhir adalah robot mampu mengikuti pergerakan bola dengan melakukan navigasi sesuai posisi yang tepat.

Prosedur pengujian dilakukan dengan meletakan robot pada posisi awal (R) yaitu pada tengah-tengah lapangan pengujian dengan arah robot menghadap utara, lalu bola diletakan pada tiga lokasi yang telah diatur. Posisi bola diatur sesuai arah mata angin dengan robot sebagai acuan yaitu lokasi pertama bola diletakkan di arah timur, lokasi kedua bola diletakkan di arah selatan, dan lokasi ketiga bola diletakkan di arah barat. Masing – masing posisi bola diatur jarak nya terhadap robot sejauh 30 cm. Selanjutnya robot diaktifkan dan diamati lama waktu robot untuk untuk berhenti tepat di depan bola dari posisi awal menghadap utara menuju ketiga posisi bola yang telah diatur. Adapun skema pengujian performa sistem keseluruhan ditunjukkan dalam Gambar 3.25.



Gambar 3.25 Skema pengujian performa sistem keseluruhan

BAB IV
HASIL DAN PEMBAHASAN

Setelah melakukan perhitungan dan perancangan langkah selanjutnya adalah pembahasan hasil pengujian yang bertujuan untuk menganalisis alat yang telah dirancang dan di implementasikan telah bekerja sesuai dengan perancangan yang diharapkan. Pengujian dilakukan tiap-tiap blok dengan tujuan untuk mengamati apakah tiap blok sistem sudah sesuai dengan perancangan, kemudian dilanjutkan dengan pengujian secara keseluruhan sistem. Adapun pengujian yang perlu dilakukan sebagai berikut:

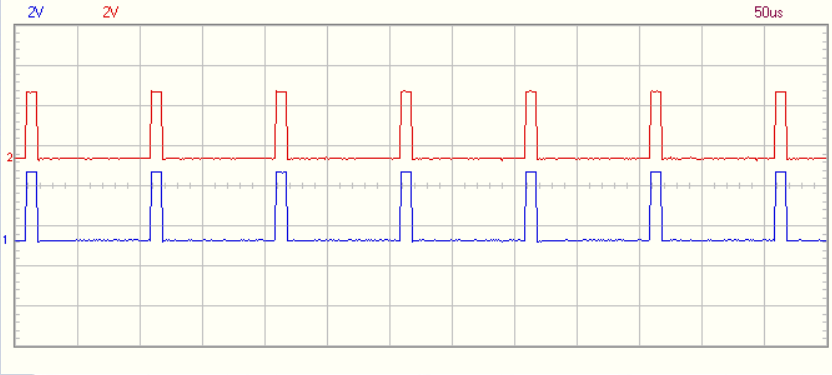
1. Pengujian Rangkaian *Driver* Motor L298N
   a) Pengujian *Duty Cycle* Rangkaian Mikrokontroler Dan *Driver* L298N
   b) Pengujian Arah Gerak Motor
2. Pengujian *Servo Serial Dynamixel*
   a) Pengujian Sudut *Servo Serial Dynamixel*
3. Pengujian Sensor Kamera Dengan Hasil Olahan OpenCV
   a) Pengujian Jumlah *Frame* Yang Dapat Diproses Dalam Satu Detik
   b) Pengujian Tampilan *Window* Proses Deteksi
   c) Pengujian Nilai HSV Berdasarkan Deteksi Warna HSV dengan ROI
   d) Pengujian Jarak Maksimal BolaTerdeteksi
   e) Pengujian Posisi Bola Terhadap Robot
4. Pengujian Transmisi Data
   a) Pengujian Sinyal Komunikasi Serial UART Pada Mikrokontroler
   b) Pengujian Penerimaan Data Pada Mikrokontroler Utama
   c) Pengujian Jarak Komunikasi *Bluetooth* HC-05
5. Pengujian Performa Sistem Keseluruhan

4.1 Pengujian Rangkaian *Driver* Motor L298N

4.1.1 Pengujian *Duty Cycle* Rangkaian Mikrokontroler dan *Driver* L298N

Pada pengujian, mikrokontroler memberikan masukan *duty cycle* sinyal PWM pada masukan *driver* L298N pada *range* 0% sampai 100% dengan kenaikan bertahap sebesar 10%. Hasil pengujian sinyal PWM mikrokontroler dengan keluaran *driver* L298N diamati

menggunakan osiloskop digital Vellemen PCLAB PCSU100. Adapun hasil pengujian *duty cycle* PWM sebesar 10% ditunjukkan dalam Gambar 4.1 dan Gambar 4.2.



Gambar 4.1 Sinyal keluaran *driver* L298N terhadap sinyal masukan PWM mikrokontroler sebesar 10% (warna merah: sinyal PWM mikrokontroler, warna biru: sinyal keluaran *driver* L298N)

Waveform Parameters

Amplitude:	CH1	CH2
DC Mean	0.25 V	0.19 V
Max	3.56 V	3.50 V
Min	-0.63 V	-0.56 V
Peak-to-Peak	4.19 V	4.06 V
High	3.38 V	3.25 V
Low	-0.06 V	-0.06 V
Amplitude	3.44 V	3.31 V
AC RMS	1.00 V	0.97 V
AC dBV	0.011 dBV	-0.229 dBV
AC dBm	2.23 dBm	1.99 dBm
AC+DC RMS	1.04 V	1.00 V
AC+DC dBV	0.301 dBV	0.003 dBV
AC+DC dBm	2.52 dBm	2.22 dBm
Timing:		
Duty Cycle	9.63 %	9.60 %
Positive Width	0.010 ms	0.010 ms
Negative Width	0.090 ms	0.090 ms
Rise Time	0.000 ms	0.000 ms
Fall Time	0.000 ms	0.000 ms
Period	0.100 ms	0.100 ms
Frequency	10.0 kHz	10.0 kHz
Phase	139.4 deg	-139.4 deg

Select All | Unselect All | Options... | Close | Help

Gambar 4.2 Tampilan waveform parameter osiloskop velleman PCSU100

Hasil pengujian keseluruhan data selisih *duty cycle* sinyal PWM mikrokontroler dengan keluaran *driver* motor L298N dapat dilihat pada Tabel 4.1. Berdasarkan tabel 4.1 diketahui bahwa pada pengujian terdapat selisih *duty cycle* keluaran rata-rata sebesar 0,67%. Selisih rata-rata 0,67% cukup baik dan efisien karena tidak memberikan pengaruh yang besar pada kinerja sistem yang dirancang, maka dapat disimpulkan bahwa keluaran *duty cycle driver* motor L298N dapat bekerja dengan baik saat menerima sinyal PWM dari mikrokontroler.

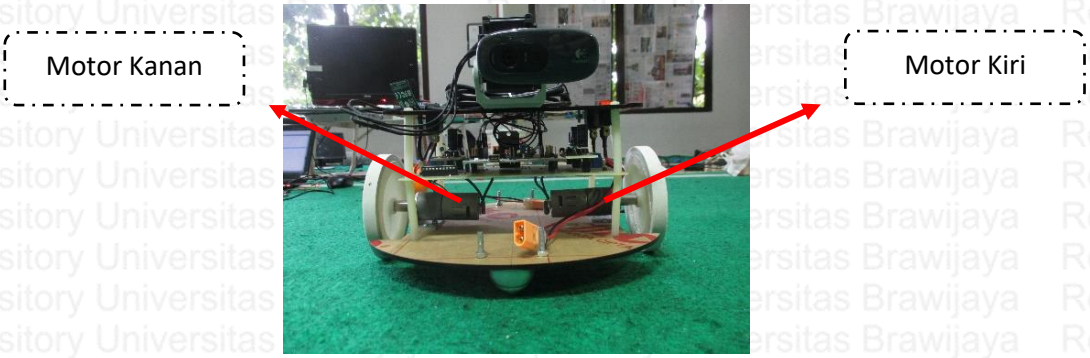
Tabel 4.1

Hasil Pengujian Data Selisih *Duty Cycle* MK Dengan *Driver* L298N

Duty Cycle **Keluaran** ***Driver***	***Duty Cycle*** **PWM Mikrokontroler**	***Duty Cycle*** **Selisih**
0%	0%	0%
9,6%	9,6%	0%
21%	19,3%	1,7%
29,4%	28,2%	1,2%
40,3%	38,4%	1,9%
49,8%	49,1%	0,7%
59,5%	59,3%	0,2%
70,1%	69,5%	0,6%
79,9%	79,7%	0,2%
89,8%	88,9%	0,9%
99,1%	99,1%	0%
Selisih *duty cycle* rata-rata 0,67%		

4.1.2 Pengujian Arah Gerak Motor

Pengujian arah gerak motor diatur dengan mengubah keadaan logika dari pin *direction* pada mikrokontroler. Untuk mengetahui respon navigasi robot penempatan motor dibagi menjadi dua bagian yaitu motor kanan dan motor kiri. Setiap motor menggunakan dua buah pin logika yang berfungsi untuk mengatur arah gerak motor melalui mikrokontroler utama. Penempatan motor kanan dan motor kiri tampak depan pada robot ditunjukkan dalam Gambar 4.3.



Gambar 4.3 Penempatan motor kanan dan motor kiri tampak depan pada robot

Adapun hasil pengujian arah gerak motor dapat dilihat pada Tabel 4.2.

Tabel 4.2

Hasil Pengujian Arah Gerak Motor

Motor Kanan		Motor Kiri		Respon Robot	
PORTA.8	**PORTB.10**	**PORTA.6**	**PORTA.7**	**Respon Yang Diharapkan**	**Hasil Pengujian**
1	0	0	1	Maju	Maju
0	1	1	0	Mundur	Mundur
1	0	1	0	Berlawanan Jarum Jam	Berlawanan Jarum Jam
0	1	0	1	Searah Jarum Jam	Searah Jarum Jam
0	0	0	0	*Brake*	*Brake*
1	1	1	1	*Stop*	*Stop*

Dalam Tabel 4.2 dapat diketahui bahwa respon *driver* motor L298N terhadap sinyal masukan arah dari mikrokontroler bekerja sesuai dengan perancangan yang diharapkan. Sehingga dapat disimpulkan bahwa driver L298N dapat bekerja dengan baik saat mendapatkan sinyal logika arah dari mikrokontroler.

4.2 Pengujian *Servo Serial Dynamixel*

4.2.1 Pengujian Sudut *Servo Serial Dynamixel*

Pada pengujian sudut dari *servo serial dynamixel* servo dipasangi dengan penunjuk sudut busur 360° seperti yang ditunjukkan dalam Gambar 4.4.

Gambar 4.4 Servo serial dynamixel yang telah dipasangi busur 360°

Pemasangan busur pada servo dilakukan dengan menyesuaikan posisi 150° yang diwakili oleh nilai data serial 512. Besar sudut hasil pengujian kemudian dibandingkan dengan besar sudut teori berdasarkan nilai data serial seperti pada Tabel 4.3.

Tabel 4.3

Hasil Pengujian Sudut *Servo Serial Dynamixel*

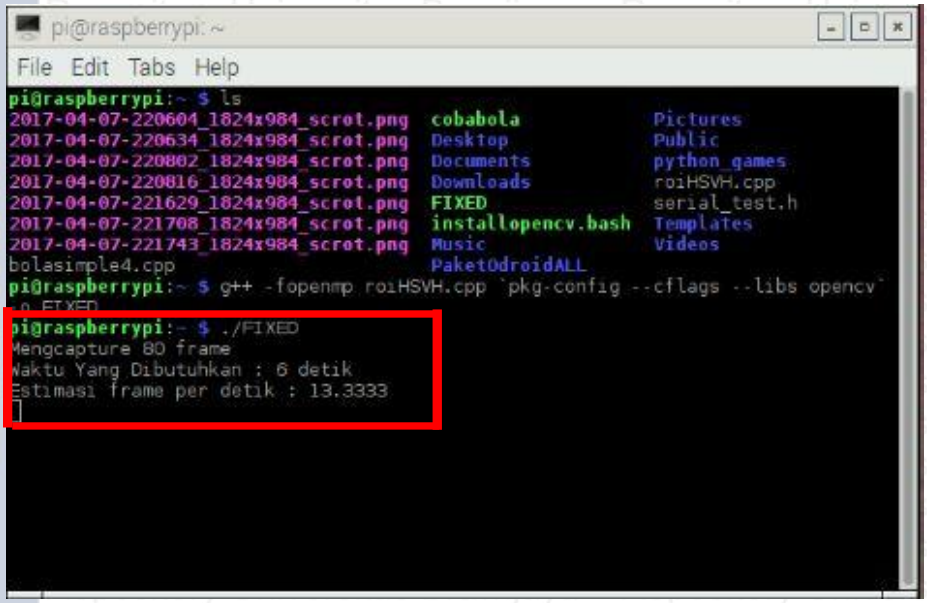
Nilai Serial	Sudut Teori (°)	Rata- Rata Sudut Praktik (°)	Persen *error* (%)
0	0	0	0
100	29,32551	28,66666	0,65884
200	58,65102	59	0,34897
300	87,97653	88,33333	0,35679
400	117,30205	117	0,30205
500	146,62756	145,66666	0,96089
600	175,95307	174,83333	1,11974
700	205,27859	204,33333	0,94525
800	234,60410	233,9	0,70410
900	263,92961	263	0,92961
1000	293,25513	292,63333	0,62179

Pengujian dilakukan sebanyak 3 kali dan data pengujian yang diambil adalah nilai rata-rata. Dari hasil pengujian pada Tabel 4.4 dapat dilihat bahwa nilai sudut perubahan servo berbanding lurus dengan sudut teori dan antara sudut teori dengan sudut sebenarnya memiliki *error* < 1,12%. Persen *error* terbesar perubahan sudut servo dari sudut teori ialah sebesar 1,11%, sedangkan yang terkecil adalah 0% pada sudut 0°. Dengan nilai persen *error* terbesar 1,11% dapat dikatakan cukup baik dan efisien karena tidak memberikan pengaruh yang besar pada kinerja sistem yang dirancang, maka dapat disimpulkan bahwa *servo serial dynamxel* baik digunakan untuk penggerak sensor kamera pada robot.

4.3 Pengujian Sensor Kamera Dengan Hasil Olahan OpenCV

4.3.1 Pengujian Jumlah *Frame* Yang Dapat Diproses Dalam Satu Detik

Pengujian dilakukan dengan menaikkan jumlah *frame* yang diproses dari 20 *frame* sampai 120 *frame* dengan kenaikan bertahap sebesar 20 *frame*. Setelah itu menghitung rata – rata banyak *frame* yang diproses dalam satu detik dari masing – masing jumlah *frame* yang telah ditentukan. Hasil estimasi *frame* per detik kemudian ditampilkan pada terminal raspberry pi. Estimasi *frame* per detik saat jumlah *frame* yang diproses sebesar 80 ditunjukkan dalam Gambar 4.5.



Gambar 4.5 Hasil estimasi *frame* per detik saat memproses 80 *frame*

Pengujian dilakukan sebanyak tiga kali pada setiap jumlah *frame* kemudian diambil nilai rata – rata dari hasil pengujian. Hasil pengujian jumlah *frame* yang diproses dalam satu detik ditunjukkan dalam Tabel 4.4

Tabel 4.4

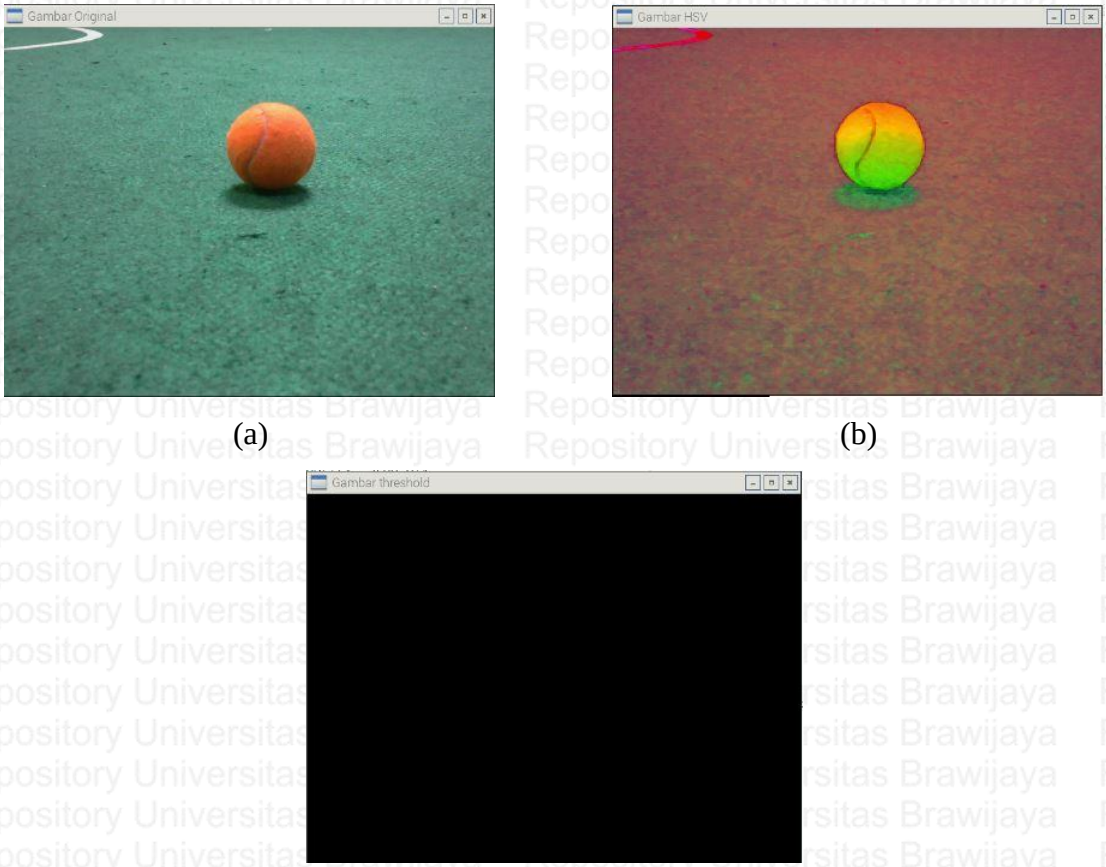
Hasil Pengujian Jumlah *Frame* Per Detik

Jumlah *Frame*	**Estimasi Jumlah *Frame* Per Detik**
20	10
40	13,3333
60	15
80	13,3333
100	14,2587
120	15
Fps rata – rata =	**13,48755 fps**

Seperti yang ditunjukkan dalam Tabel 4.4 kamera logitech C270 yang digunakan sebagai sensor utama yang dihubungkan dengan raspberry pi dalam satu detik mampu memproses jumlah *frame* hingga 13,48755 fps. Jumlah *frame* sebesar 13,48755 dinilai baik karena sudah cukup untuk membuat robot mampu merespon sesuai dengan yang di harapkan.

4.3.2 Pengujian Tampilan *Window* Proses Deteksi

Pengujian dilakukan secara *real time* dengan menjalankan program deteksi HSV dengan ROI pada raspberry pi dan mengikuti alur proses untuk memperoleh nilai HSV bola. Selanjutnya mengamati tampilan *window* – *window* dari proses deteksi sebagai acuan apakah proses deteksi telah sesuai dengan perancangan algoritma yang dibuat. Adapun hasil tampilan *window* - *window* awal saat program deteksi warna HSV dengan ROI dijalankan ditunjukkan dalam Gambar 4.6.



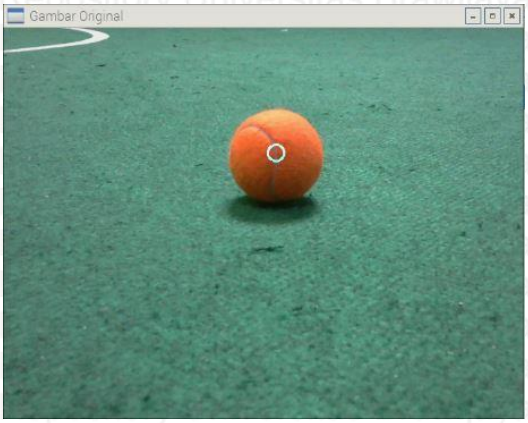
(a) (b)

(c)

Gambar 4.6 Hasil tampilan *window* – *window* awal program deteksi warna HSV dengan ROI ((a) gambar sebenarnya, (b) gambar hasil konversi gambar sebenarnya menjadi HSV, (c) gambar hitam *threshold* Awal)

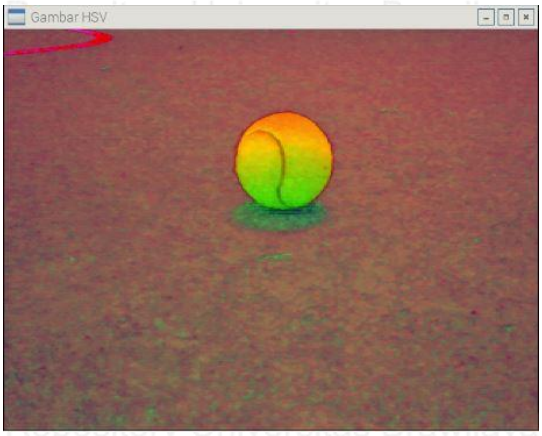
Seperti yang ditunjukkan dalam Gambar 4.6 awal program deteksi warna HSV dengan ROI dijalankan akan menampilkan tiga *window* utama yaitu *window* gambar sebenarnya, window gambar HSV, dan *window* gambar *threshold*. *Window* gambar sebenarnya diperoleh melalui hasil *capture* langsung melalui kamera tanpa adanya proses konversi. *Window* gambar HSV diperoleh dari hasil konversi RGB gambar sebenarnya menjadi gambar HSV. *Window* gambar *threshold* diperoleh dari *range* HSV warna yang nilaiya belum diatur sehingga akan

menampilkan gambar hitam sebagai kondisi awal. Dari hasil awal saat program dijalankan agar mampu mendeteksi warna pada bola dilakukan seleksi area dengan ROI pada gambar sebenarnya. Adapun hasil tempilan *window* setelah dilakukan seleksi area kotak dengan ROI ditunjukkan dalam Gambar 4.7.



(a)

(b)



(c)

(d)

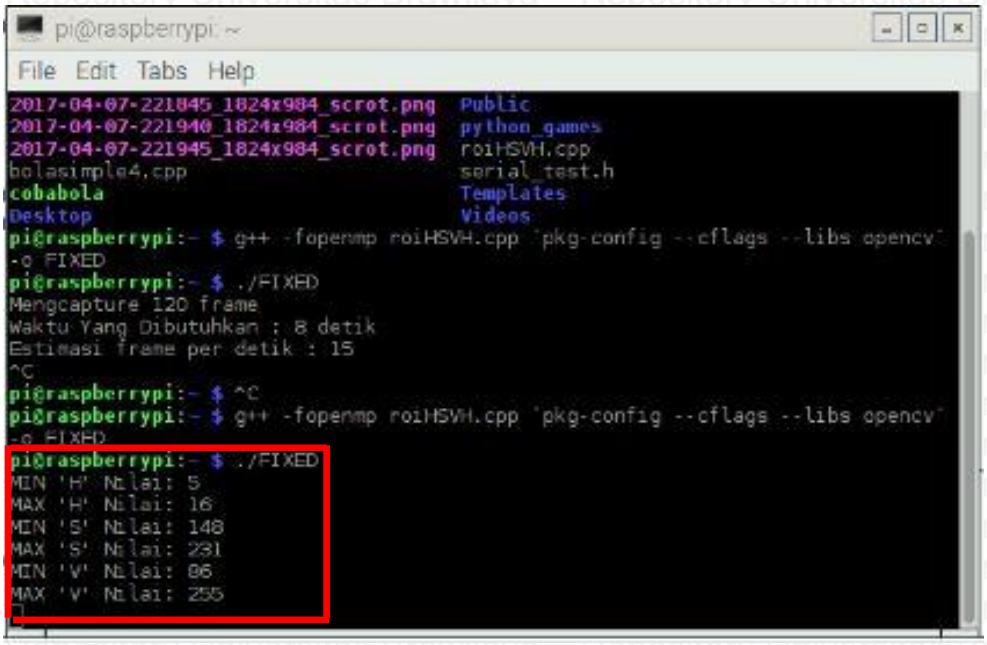
Gambar 4.7 Hasil tampilan *window* – *window* saat seleksi area dengan ROI selesai dilakukan ((a) gambar sebenarnya , (b) gambar hasil seleksi ROI, (c) gambar hasil konnversi gambar sebenarnya menjadi gambar HSV, (d) gambar *threshold*)

Dalam Gambar 4.7 setelah proses seleksi area dengan ROI seleksi dilakukan maka akan menampilkan satu *window* tambahan yaitu *window* hasil seleksi ROI. *Window* hasil seleksi ROI diperoleh berdasarkan area yang telah diseleksi menggunakan *mouse event* dengan ROI pada gambar sebenarnya. Hasil area seleksi ROI diambil nilai HSV dari warnanya kemudian dimasukkan dalam *range* warna yang dideteksi. Hasil obyek yang telah terdeteksi diamati pada *window* gambar *threshold* di mana warna putih merupakan warna yang dideteksi dan warna

hitam merupakan warna lain-lain yang tidak dideteksi. Pada gambar sebenarnya diberi penanda gambar lingkaran untuk lebih mempermudah hasil pengamatan obyek yang dideteksi. Dari hasil tampilan *window* – *window* pada Gambar 4.6 dan Gambar 4.7 diketahui bahwa program yang dirancang telah berjalan sesuai dengan perancangan algoritma yang dibuat.

4.3.3 Pengujian Nilai HSV Berdasarkan Deteksi Warna HSV dengan ROI

Pengujian dilakukan secara *real time* dengan mengambil nilai HSV dari ROI pada program sampai warna yang diinginkan bisa terdeteksi. Jarak bola dengan kamera di atur sebesar 30 cm kemudian dengan menggunakan fungsi *mouse event* kulit bola yang berwarna orange diseleksi dengan bentuk kotak untuk memperoleh nilai HSV berdasarkan fungsi ROI. Nilai HSV pada program memiliki lebar data sebesar 8 bit dengan *range*, yaitu *Hue* (0-255), *Saturation* (0-255), dan *Value* (0-255). Nilai HSV dari hasil seleksi kemudian ditampilkan pada terminal di Raspberry pi. Hasil tampilan *range* nilai HSV pada terminal Raspberry pi ditunjukkan dalam Gambar 4.8.



Gambar 4.8 Hasil tampilan *range* minimal dan maksimal nilai HSV pada terminal Raspberry pi

Data hasil pengujian merupakan *range* data tertinggi dan terendah dari 10 kali pengujian yang dilakukan menggunakan kamera dan raspberry pi. Data hasil pengujian nilai HSV ditunjukkan dalam Tabel 4.5.

Tabel 4.5

Hasil Pengujian Nilai HSV

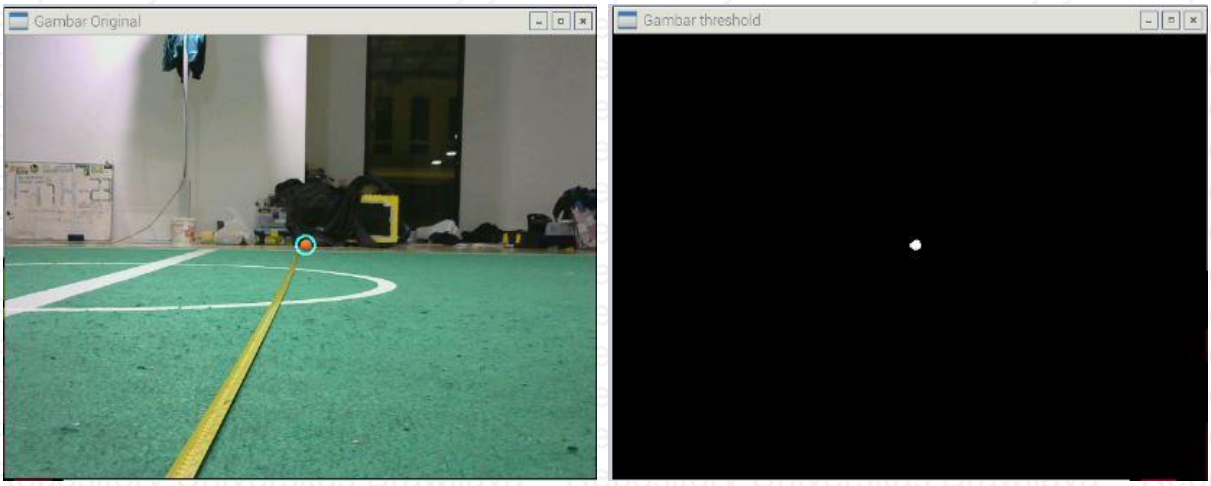
Pengujian ke	*Hue* (MIN – MAKS)	*Saturation* (MIN – MAKS)	*Value* (MIN – MAKS)
1	5 – 16	148 – 231	86 - 255
2	3 – 17	150 – 230	84 – 250
3	7 – 16	151 – 241	82 – 255
4	5 – 17	150 – 231	83 – 255
5	5 – 17	144 – 240	83 – 254
6	6 – 18	145 – 233	82 – 252
7	3 – 18	146 – 237	84 – 254
8	4 – 17	145 – 238	85 – 255
9	5 – 17	150 – 230	85 – 255
10	7 – 17	151 – 229	86 – 255
Rata – Rata	**5 - 17**	**148 - 234**	**84 - 254**

Pada Tabel 4.5 terlihat bahwa bola orange yang digunakan sebagai obyek yang dideteksi memiliki rata - rata nilai yaitu *Hue* (H) dengan nilai minimal sebesar 5 dan maksimal 17, *saturation* (S) dengan nilai minimal sebesar 148 dan maksimal sebesar 234, dan *value* (V) dengan nilai minimal sebesar 84 dan maksimal 254. Dari hasil pengujian terhadap nilai HSV dari tiga warna yang akan dideteksi oleh kamera, didapatkan bahwa nilai *hue* dan *saturation* memiliki *range* yang sempit. Nilai *value* memiliki *range* yang lebar karena mudah terpengaruh pencahayaan lingkungan. Sehingga pada pembuatan alat hanya nilai *hue* dan *saturation* yang akan digunakan sebagai pedoman warna.

4.3.4 Pengujian Jarak Maksimal BolaTerdeteksi

Pengujian dilakukan dengan menempatkan sensor kamera di mekanika robot keseluruhan dan menjalankan program deteksi warna HSV dengan ROI. *Range* nilai HSV yang digunakan merupakan rata-rata hasil pengujian nilai HSV sebelumnya yaitu *Hue* (H) dengan nilai minimal sebesar 5 dan maksimal 17, *saturation* (S) dengan nilai minimal sebesar 148 dan maksimal sebesar 234, dan *value* (V) dengan nilai minimal sebesar 84 dan maksimal 254. Besar perubahan sudut pada sendi kepala dan tinggi pada kamera pada kepala robot dari permukaan tanah digunakan untuk menentukan jarak bola dari robot secara teori. Parameter untuk mengetahui apakah bola masih terdeteksi atau tidak diamati melalui *window threshold* dimana warna putih yang mewakili warna yang dideteksi masih terlihat. Tampilan *window* gambar sebenarnya dan

window threshold saat pengujian jarak maksimal bola terdeteksi ditunjukkan dalam Gambar 4.9. Adapun hasil pengujian jarak maksimal bola terdeteksi ditunjukkan dalam Tabel 4.6



(a) (b)

Gambar 4.9 Tampilan *window* saat pengujian jarak maksimal bola terdeteksi ((a) *window* Gambar sebenarnya, (b) *window threshold*)

Tabel 4.6
Hasil Pengujian Jarak Maksimal Bola Terdeteksi

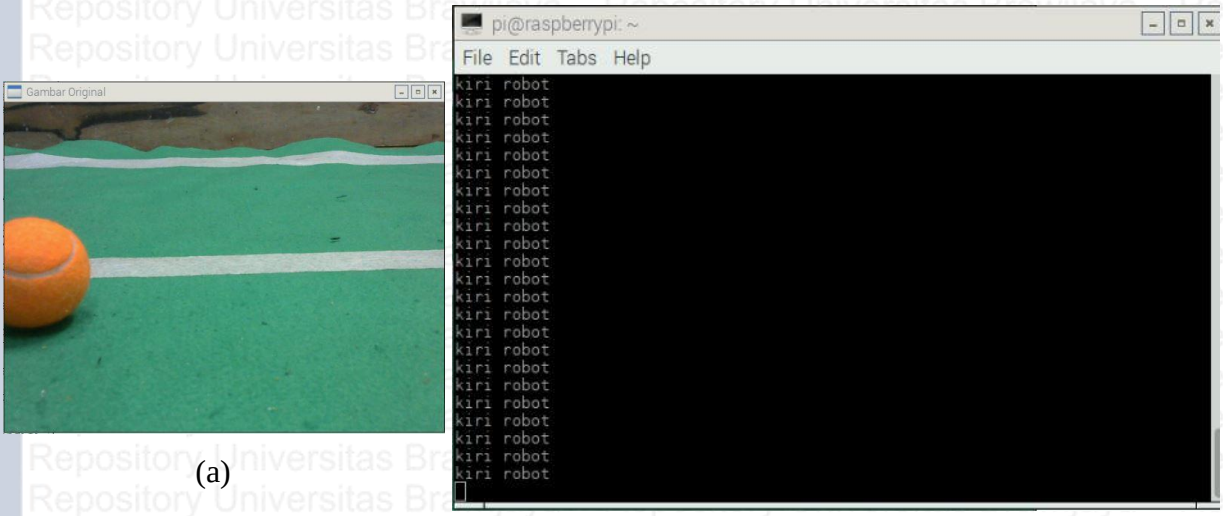
Sudut Kepala (°)	Jarak Teori (cm)	Jarak Rata – Rata Praktik (cm)	Persen *Error* (%)	Kondisi Bola
15	3,751	2,63	1,12	Terdeteksi
25	6,528	5,07	1,46	Terdeteksi
35	9,802	8,90	0,90	Terdeteksi
45	14,000	12,83	1,17	Terdeteksi
55	19,994	19,03	0,96	Terdeteksi
65	30,023	29,20	0,82	Terdeteksi
75	52,248	50,97	1,28	Terdeteksi
85	160,020	158,50	1,52	Terdeteksi
87	267,135	265,73	1,40	Tidak Terdeteksi

Pengujian dilakukan sebanyak 3 kali dan data pengujian yang diambil adalah nilai rata-rata. Berdasarkan hasil pengujian pada Tabel 4.6 secara teori jarak maksimal bola terdeteksi sebesar 160,020 cm sedangkan secara praktik sebesar 158,50 cm. Persen *error* dari jarak bola secara teori dan praktik < 1,6 %. Dengan jarak maksimal bola terdeteksi > 150 cm kemampuan robot untuk mendeteksi bola dinilai baik karena robot masih mampu mendeteksi bola saat posisi *kick*

off , saat *kick off* robot dan bola akan berjarak 150 cm sesuai dengan peraturan pertandingan KRSBI.

4.3.5 Pengujian Posisi Bola Terhadap Robot

Pengujian dilakukan secara *real time* dengan menjalankan program deteksi warna HSV dengan ROI di raspberry pi. Kemudian memasukkan nilai rata – rata dari parameter HSV bola hasil dari ROI dan menempatkan bola sejauh 30 cm dari robot. Setelah itu memposisikan bola terhadap robot sesuai dengan tiga keadaan kanan, depan dan kiri dari robot. Selanjutnya menampilkan hasil pembagian kedalam tiga kondisi yaitu "kanan robot", "depan robot", dan "kiri robot" pada terminal Raspberry pi beserta nilai terbaru dari *frame pixel* kamera. Tampilan hasil terminal dan *window* gambar sebenarnya dari posisi bola "kiri robot" ditunjukkan dalam Gambar 4.10. Adapun hasil pengujian pembagian *frame pixel* kamera untuk menentukan posisi bola terhadap robot ditunjukkan dalam Tabel 4.7.



(a) (b)

Gambar 4.10 Tampilan terminal dan *window* gambar sebenarnya saat posisi bola "Kiri Robot" ((a) *window* gambar sebenarnya, (b) tampilan teriminal)

Tabel 4.7
Hasil Pembagian *Frame Pixel* Kamera Untuk Menentukan Posisi Bola Terhadap Robot

Kiri Robot		**Depan Robot**		**Kanan Robot**	
X (min-maks)	**Y (min-maks)**	**X (min-maks)**	**Y (min-maks)**	**X (min-maks)**	**Y (min-maks)**
(1 - 64)	(1 - 480)	(65 - 519)	(1 - 480)	(520 - 640)	(1 - 480)

Berdasarkan Tabel 4.7 posisi bola kiri robot diperoleh dengan nilai sumbu x minimal 1 dan maksimal 64 sedangkan sumbu y minimal 1 dan maksimal 480. Selanjutnya posisi depan robot

diperoleh dengan nilai sumbu x minimal 65 dan maksimal 519 sedangkan sumbu y minimal 1 dan maksimal 480. Posisi kanan robot diperoleh dengan nilai sumbu x minimal 520 dan maksimal 640 sedangkan sumbu y minimal 1 dan maksimal 480. Posisi bola depan robot dibuat *range* pembagian *frame pixel* yang lebih besar dibandingkan kanan robot dan kiri robot agar robot tidak banyak manuver gerakan memutar untuk mencari bola. Pembagian *frame pixel* untuk menentukan posisi bola terhadap robot yang telah dibuat dinilai sudah cukup baik untuk keseluruhan sistem robot

4.4 Pengujian Transmisi Data

4.4.1 Pengujian Pengujian Sinyal Komunikasi Serial UART Pada Mikrokontroler

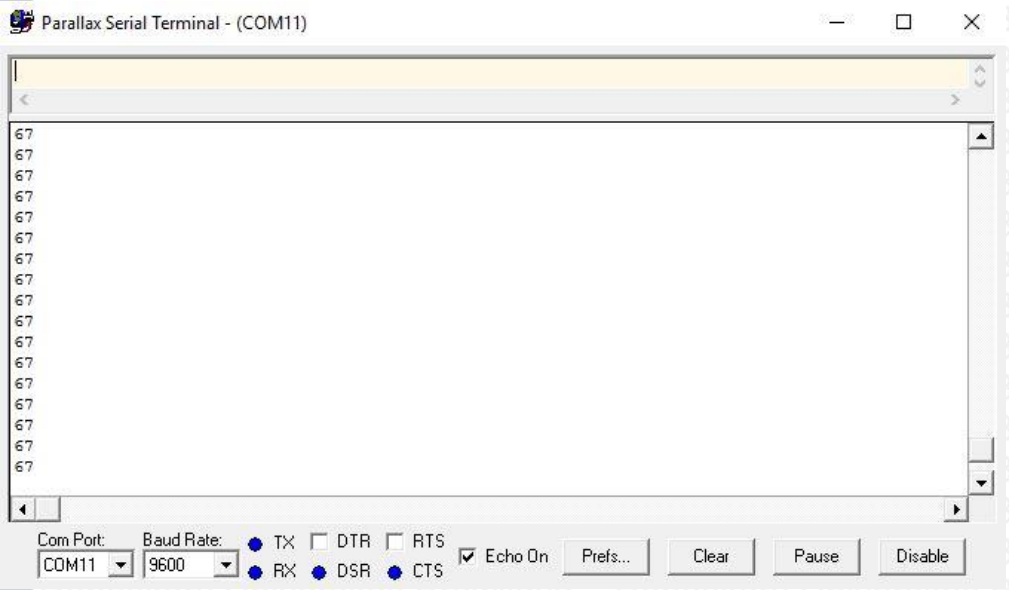
Pengujian dilakukan dengan mengirim karakter ‘A’ yaitu dengan cara mengatur perangkat UART dengan *baudrate* 9600 bps dan melakukan pengiriman data senilai 0xAA. Karena pengujian menggunakan osiloskop yang menampilkan data berupa sinyal kotak yang merepresentasikan bilangan biner, maka untuk mengetahui apakah data yang dikirimkan sudah benar, data harus dikonversi menjadi bilangan biner terlebih dahulu. Jika nilai hexadesimal 0xAA dikonversi menjadi kode biner maka hasilnya yaitu 0b10101010. Selanjutnya data diawali dengan bit *start* yang selalu berlogika *low* kemudian diikuti bit data ke 0 hingga ke 7 dan diakhiri dengan bit *stop* yang berlogika *high*. Hasil pengujian ditunjukkan dalam Gambar 4.4.

Gambar 4.11 Hasil pembacaan data dengan osiloskop

Pada Gambar 4.11 data hasil pembacaan data dengan osiloskop dapat disimpulkan bahwa sebuah frame data diawali dengan start bit kemudian diikuti data bit dengan total maksimum 9 bit yang diawali LSB hingga MSB. Dapat dikatakan data yang dikirimkan oleh mikrokontroler telah bekerja dengan baik dan siap dipakai dalam sistem.

4.4.2 Pengujian Penerimaan Data Pada Mikrokontroler Utama

Pengujian penerimaan data di mikrokontroler utama dilakukan dengan melihat data yang diterima menggunakan aplikasi *parallax serial terminal.* Awalnya perangkat UART pada raspberry pi diatur dengan *baudrate* 9600 bps dan melakukan pengiriman data senilai 0XCC ke mikrokontroler utama yang sebelumnya telah dimasukkan program penerimaan data. Kemudian pin RX dari mikrokontroler utama disambungkan pada USB to TTL. Selanjutnya data yang diterima diamati pada komputer. Adapun Gambar hasil pengujian penerimaan data pada mikrokontroler utama ditunjukkan dalam Gambar 12.



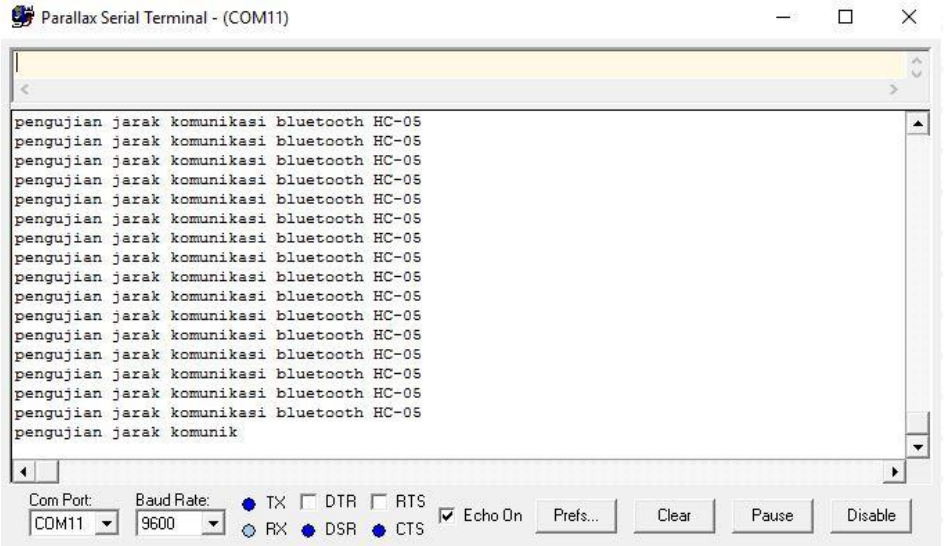
Gambar 4.12 Hasil pengujian penerimaan data pada mikrokontroler utama

Terlihat pada Gambar 12 mikrokontroler utama menerima data dengan nilai desimal "67" hal ini dikarenakan data yang dikirim melalui raspberry pi merupakan data dengan format berupa karakter, sedangkan pada mikrokontroler utama data dalam format karakter tersebut akan dikonversi kedalam bentuk nilai desimal. Pengkonversian data dengan format karakter ke nilai desimal dari pengujian telah sesuai dengan tabel ASCII, sehingga dapat disimpulkan bahwa data yang dikirim melalui raspberry pi ke mikrokontroler dapat langsung diolah dengan syarat melihat hasil konversinya melalui tabel ASCII.

4.4.3 Pengujian Jarak Komunikasi *Bluetooth* HC-05

Pengujian jarak komunikasi *bluetooth* HC-05 dilakukan dengan mengirimkan data kata "Pengujian Jarak Komunikasi *Bluetooth* HC-05". Gambar data yang dikirim kemudian dilihat

dengan menggunakan aplikasi *parallax serial terminal* di komputer seperti yang ditunjukkan dalam Gambar 4.13



Gambar 4.13 Hasil pengujian data kata yang dikirim menggunakan *bluetooth* HC-05

Setelah data kata seperti yang ditunjukkan dalam Gambar 4.13 berhasil dikirim selanjutnya *bluetooth* pemancar diperjauh jaraknya setiap 1 meter. Adapun hasil pengujian jarak maksimal pengiriman data *bluetooth* HC-05 ditunjukkan pada Tabel 4.8.

Tabel 4.8

Hasil Pengujian Jarak Maksimal Pengiriman Data *Bluetooth* HC-05

Jarak (Meter)	Kondisi Komunikasi
1-7	Terhubung
8	Terhubung
9	Terhubung
10	Terhubung
11	Terhubung
12	Terhubung
13	Terhubung
14	Terhubung
15	Tidak Terhubung
16	Tidak Terhubung
17	Tidak Terhubung
18	Tidak Terhubung

Berdasarkan hasil pengujian yang telah dilakukan, dapat disimpulkan bahwa *bluetooth* mampu mengirimkan data dengan baik sampai pada jarak 14 meter tanpa ada karakter yang hilang. Pada jarak 15 meter koneksi langsung terputus dan tidak menerima data lagi.

4.5 Pengujian Performa Sistem Keseluruhan

Pengujian dilakukan dengan meletakan robot pada posisi awal yaitu pada lapangan pengujian dengan arah robot menghadap utara, lalu bola diletakan pada tiga lokasi yang telah diatur. Posisi bola diatur sesuai arah mata angin dengan robot sebagai acuan yaitu lokasi pertama bola diletakkan di arah timur, lokasi kedua bola diletakkan di arah selatan, dan lokasi ketiga bola diletakkan di arah barat. Masing – masing posisi bola diatur jarak nya terhadap robot sejauh 30 cm. Adapun hasil pengujian performa sistem keseluruhan ditunjukkan dalam Tabel 4.9.

Tabel 4.9

Hasil Pengujian Performa Sistem Keseluruhan

Posisi Bola Terhadap Robot	Jarak Bola (cm)	Lama Waktu (Detik) Pengujian ke-					Rata - Rata
		1	2	3	4	5	
Posisi 1 (Timur)	30	4,21	3,89	4,11	4,15	4,24	4.12
Posisi 2 (Selatan)	30	6,17	5,94	6,23	5,97	6,19	6.10
Posisi 3 (Barat)	30	7,83	8,26	8,29	7,92	7,93	8.05

Pengujian dilakukan sebanyak 5 kali dan data pengujian yang diambil adalah nilai rata-rata. Berdasarkan hasil pengujian pada Tabel 4.9 robot paling cepat menemukan bola pada posisi 1 (timur) dengan waktu rata-rata 4,12 detik sedangkan posisi 2 (selatan) robot menemukan bola dengan waktu rata-rata 6,10 detik dan posisi 3 (barat) robot menemukan bola dengan waktu rata-rata 8,05 detik. Posisi 1 memiliki waktu rata-rata lebih cepat dibandingkan posisi 2 dan posisi 3 dikarenakan algoritma pada mikrokontroler utama yang melakukan manuver gerakan putar kanan terlebih dahulu untuk mencari bola.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil analisis dan pengujian setiap bagian dan keseluruhan sistem yang telah dilakukan, maka dapat dilambil kesimpulan sebagai berikut:

1. Robot mampu mendeteksi bola dengan deteksi warna HSVdan ROI dengan cara menyeleksi area yang akan dideteksi yaitu kulit bola dengan bentuk kotak, lalu diambil nilai rata-rata *range* HSV dari hasil seleksi tersebut sehingga hasil deteksi dapat diperoleh posisinya dalam bentuk koordinat x dan y dari *frame pixel* kamera. Bola orange yang digunakan sebagai obyek yang dideteksi memiliki rata - rata nilai yaitu *Hue* (H) dengan nilai minimal sebesar 5 dan maksimal 17, *saturation* (S) dengan nilai minimal sebesar 148 dan maksimal sebesar 234, dan *value* (V) dengan nilai minimal sebesar 84 dan maksimal 254.
2. Jarak bola yang dideteksi oleh robot dapat diketahui dengan cara memasang kamera dengan *servo serial dynamixel* untuk merubah sudut pada sendi kepala robot. Jarak diperoleh menggunakan perhitungan trigonometri, besar perubahan sudut pada sendi kepala dan tinggi pada kamera pada kepala robot dari permukaan tanah digunakan untuk menentukan jarak bola dari robot. *Servo serial dynamixel* pada robot memiliki sudut perubahan berbanding lurus dengan sudut teori dan antara sudut teori dengan sudut sebenarnya memiliki *error* < 1,12%. Persen *error* terbesar perubahan sudut servo dari sudut teori ialah sebesar 1,11%, sedangkan yang terkecil adalah 0% pada sudut 0°. Secara teori jarak maksimal bola terdeteksi sebesar 160,020 cm sedangkan secara praktik sebesar 158,50 cm. Persen *error* dari jarak bola secara teori dan praktik < 1,6 %.
3. Posisi bola “kanan robot”, “depan robot”, dan “kiri robot” diperoleh berdasarkan pembagian koordinat *frame pixel* kamera x dan y yaitu sebesar 640 dan 480. Posisi bola kiri robot diperoleh dengan nilai sumbu x minimal 1 dan maksimal 64 sedangkan sumbu y minimal 1 dan maksimal 480. Selanjutnya posisi depan robot diperoleh dengan nilai sumbu x minimal 65 dan maksimal 519 sedangkan sumbu y minimal 1 dan maksimal 480. Posisi kanan robot diperoleh dengan nilai sumbu x minimal 520 dan maksimal 640 sedangkan sumbu y minimal 1 dan maksimal 480.

4. Robot mampu bergerak sesuai dengan posisi bola dengan cara menerima data serial berupa karakter yang mewakili hasil pembagian *frame pixel* kamera. Kemudian data serial berupa karakter tersebut dikonversi kedalam bentuk desimal dan diproses untuk mendapatkan respon navigasi robot berdasarkan hasil pembacaan data. Pada pengujian robot diletakkan menghadap utara lalu posisi bola diatur sesuai arah mata angin dengan robot sebagai acuan yaitu lokasi pertama bola diletakkan di arah timur, lokasi kedua bola diletakkan di arah selatan, dan lokasi ketiga bola diletakkan di arah barat. Masing – masing posisi bola diatur jarak nya terhadap robot sejauh 30 cm. Robot paling cepat menemukan bola pada posisi 1 (timur) dengan waktu rata-rata 4,12 detik sedangkan posisi 2 (selatan) robot menemukan bola dengan waktu rata-rata 6,10 detik dan posisi 3 (barat) robot menemukan bola dengan waktu rata-rata 8,05 detik. Posisi 1 memiliki waktu rata-rata lebih cepat dibandingkan posisi 2 dan posisi 3 dikarenakan algoritma pada mikrokontroler utama yang melakukan manuver gerakan putar kanan terlebih dahulu untuk mencari bola.

5.2 Saran

Untuk pengembangan penelitian ini, ada beberapa saran yang dapat dilakukan antara lain:

1. Jika program deteksi digunakan pada robot humanoid di bagian sendi kepala robot tempat kamera dipasang sebaiknya dibuat *stabilizer* menggunakan sensor *gyroscope* atau sejenisnya agar proses deteksi tidak terganggu oleh guncangan dari lapangan yang kurang rata atau pergerakan robot yang kasar.
2. Untuk penggunaan pada robot beroda sebaiknya gunakan empat buah roda agar pergerakan lebih halus tidak banyak guncangan yang terjadi sehingga robot dapat mendeteksi dan mendekati bola lebih cepat dan akurat.
3. Menggunakan algoritma yang lebih baik dalam mengolah data dari sensor kamera untuk navigasi robot seperti PID dan lain-lain.

DAFTAR PUSTAKA

Adam Goode, A. R. (2011). Color-tracking Explanation. http://www.cmucam.org/projects/cmucam4/wiki/Color-tracking_Explanation. (Diakses 3 November 2016).

Atmel. (2007). *8-bit AVR with 16K Bytes In-System Programmable Flash ATMega16/16L*. San Jose: Atmel.

Durda, F. (2014). Serial And UART Tutorial . https://www.freebsd.org/doc/en_US.ISO8859-1/articles/serial-uart/. (Diakses 3 November 2016).

Itseez. (2014). *The OpenCV Tutorials.* Nizhny Novgorod: Itseez.

Ivask, M. (2015). *Raspberry Pi Based System For Visual Object Detection and Tracking*. Tallin : Tallin University of Technology.

Insyani, R.S.P.P., Mudjirahardjo, P. & Arief, R S. (2016). *IMPLEMENTASI SISTEM FIRE FOLLOWING MENGGUNAKAN KONTROLER PID DENGAN METODE TUNING PADA ROBOT KONTES PEMADAM API INDONESIA (KRPAI) DIVISI BERODA*. Malang : Skripsi Jurusan Teknik Elektro FT-UB.

Kour, H. (2015). Analysis on Image Color Model. *International Journal Of Adavanced in Computer and Communication Engineering* (IJARCCE). 4(12): 233-235.

Mordvintsev, A. & Abid, K. (2017). *OpenCV-Pyton Tutorials Documentation*. OpenCV.

Mudjirahardjo, P., Nurussa'adah., & Siwindarto, P. (2016). Soccer Field Detection Based on Histogram of S-RGB. *ARPN Journal of Engineering and Applied Science*. 21(XI), pp: 12405–12408.

Richard, J. A. (2015). EMGU CV – Select ROI (Region Of Interest) With Mouse . https://www.codeproject.com/Tips/859100/Emgu-CV-Select-ROI-Region-Of-Interest-With-Mouse-C. (Diakses 21 Maret 2017).

RISTEKDIKTI. (2016). *PanduanKRSBI2016Beta*. Jakarta Pusat: Dikti.

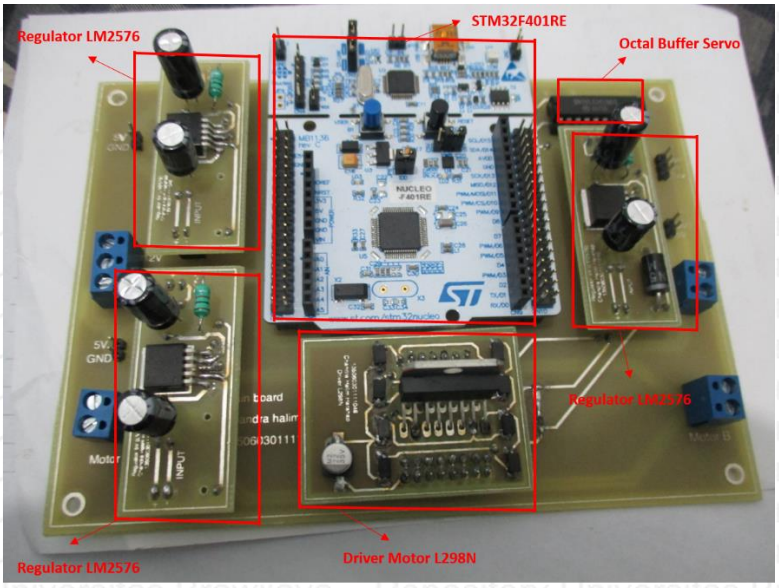
Robotis. (2006). *AX-12/AX-12+/AX-12A - ROBOTIS e-MANUAL*. Robotis.

STMicroelectronics. (2015). *UM1724 User Manual Datasheet.* STMicroelectronics.

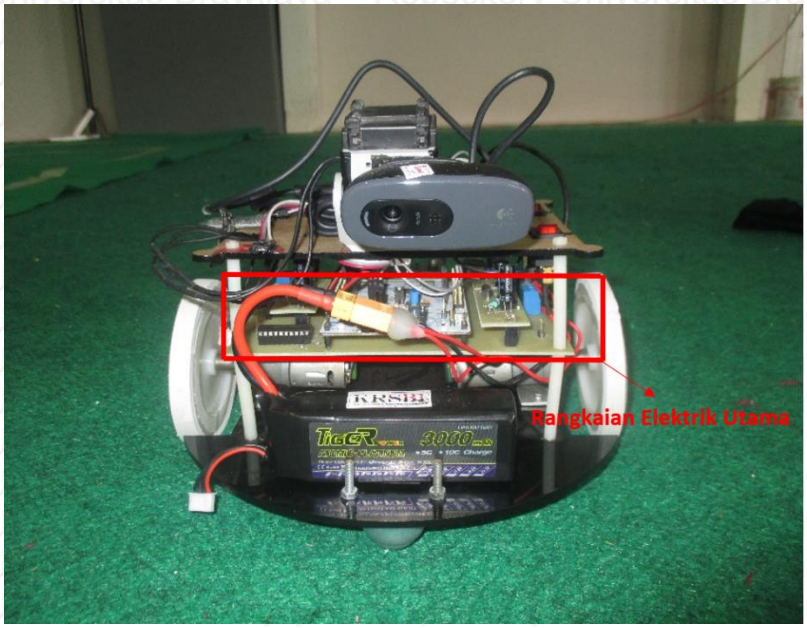
Tegar, A.F.F., Sulistiyanto, N. & Maulana, E. (2015)**.** *DETEKSI WARNA LAPANGAN KRSI MENGGUNAKAN KAMERA DAN SENSOR WARNA UNTUK MENCARI SUDUT HADAP DAN JARAK*. Malang : Skripsi Jurusan Teknik Elektro FT-UB.

LAMPIRAN 1

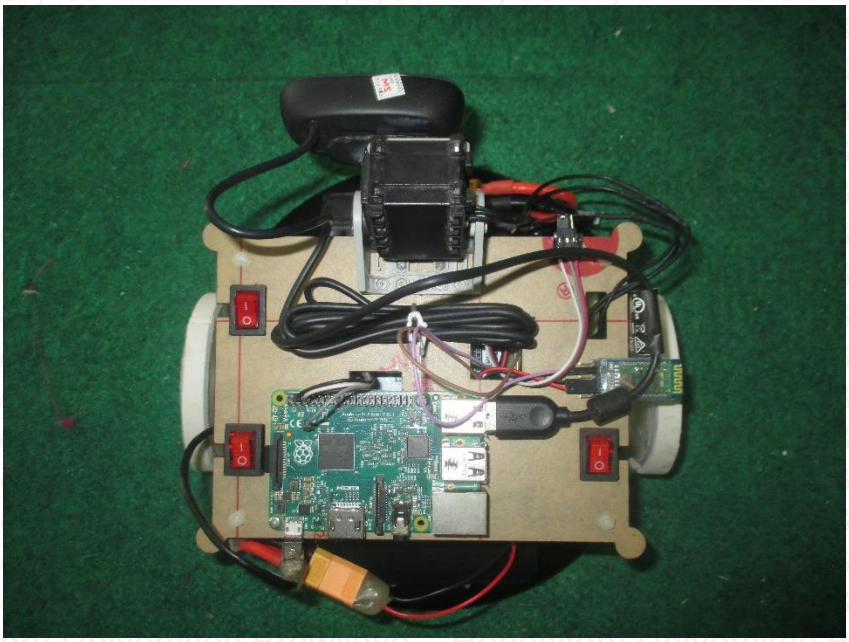
DOKUMENTASI ALAT



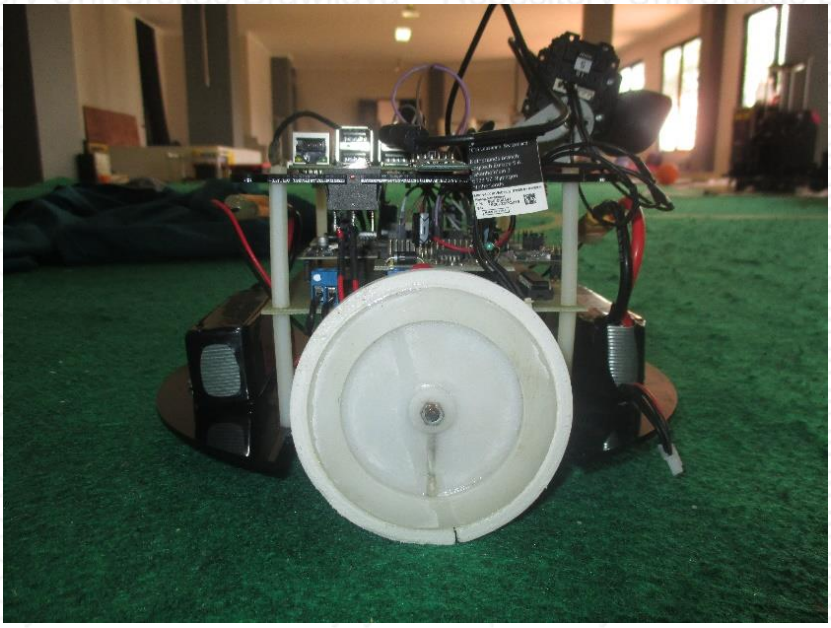
Gambar 1.1 Rangkain elektrik utama robot keseluruhan



Gambar 1.2 Robot tampak depan



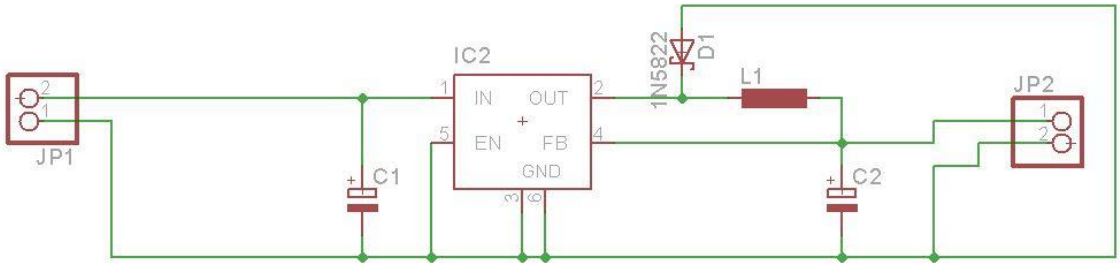
Gambar 1.3 Robot tampak atas



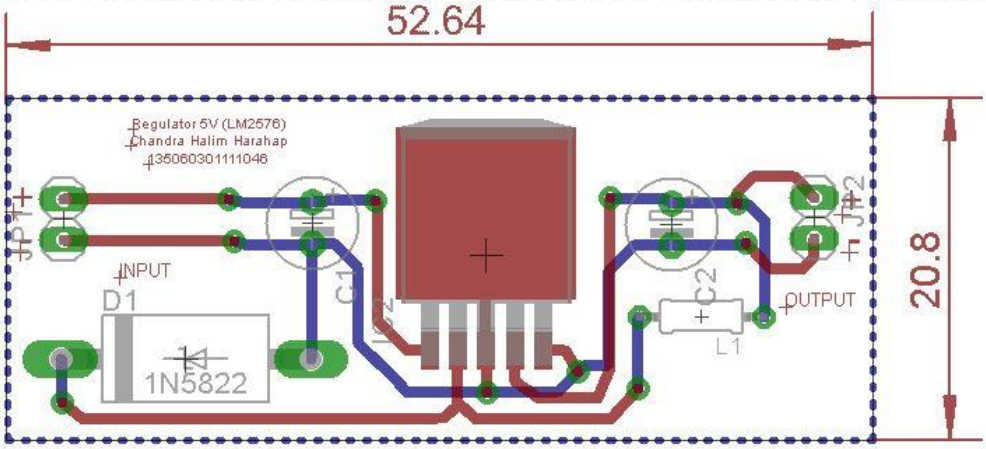
Gambar 1.4 Robot tampak samping

LAMPIRAN 2

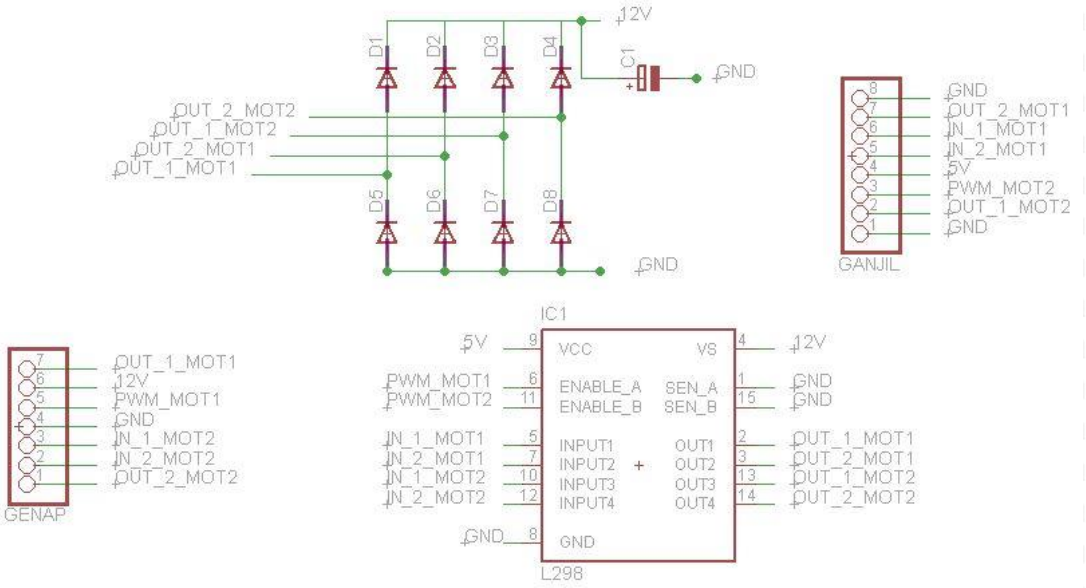
SKEMATIK RANGKAIAN



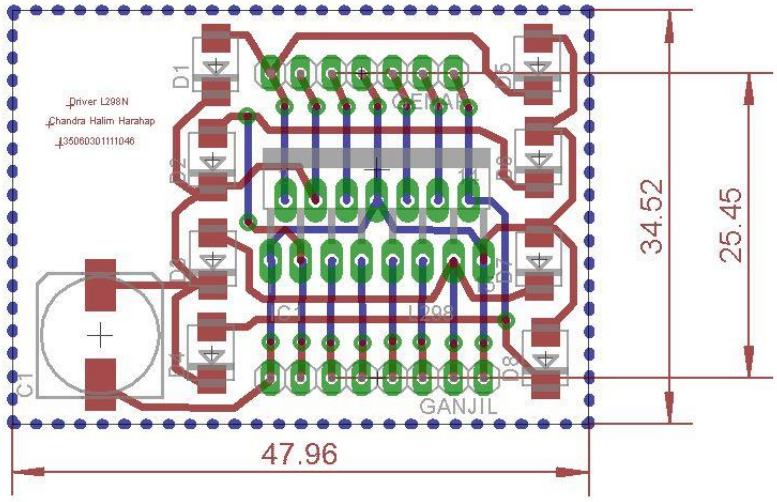
Gambar 1.5 Skematik rangkaian regulator LM2576



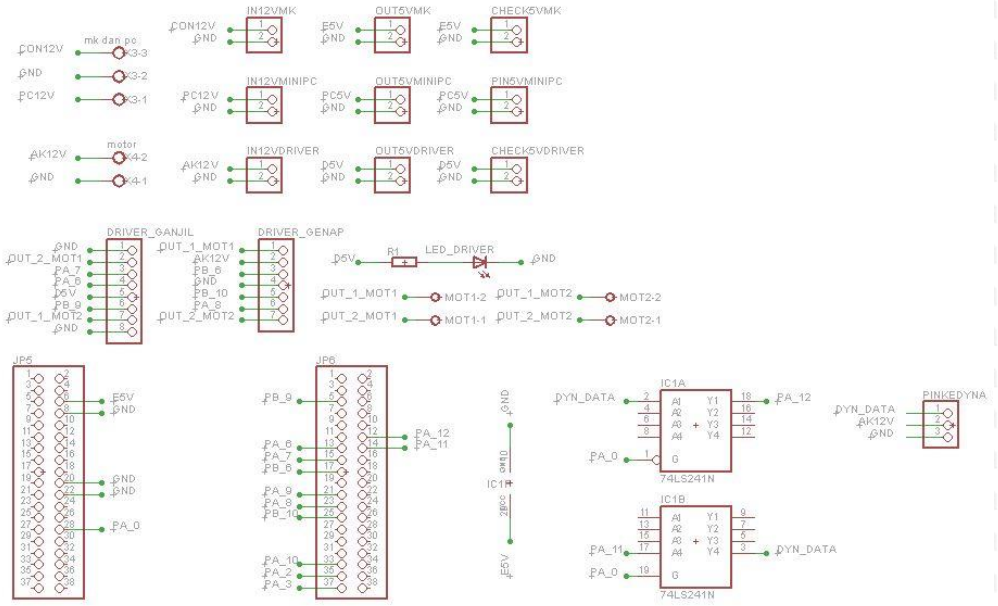
Gambar 1.6 *Board* rangkaian regulator LM2576



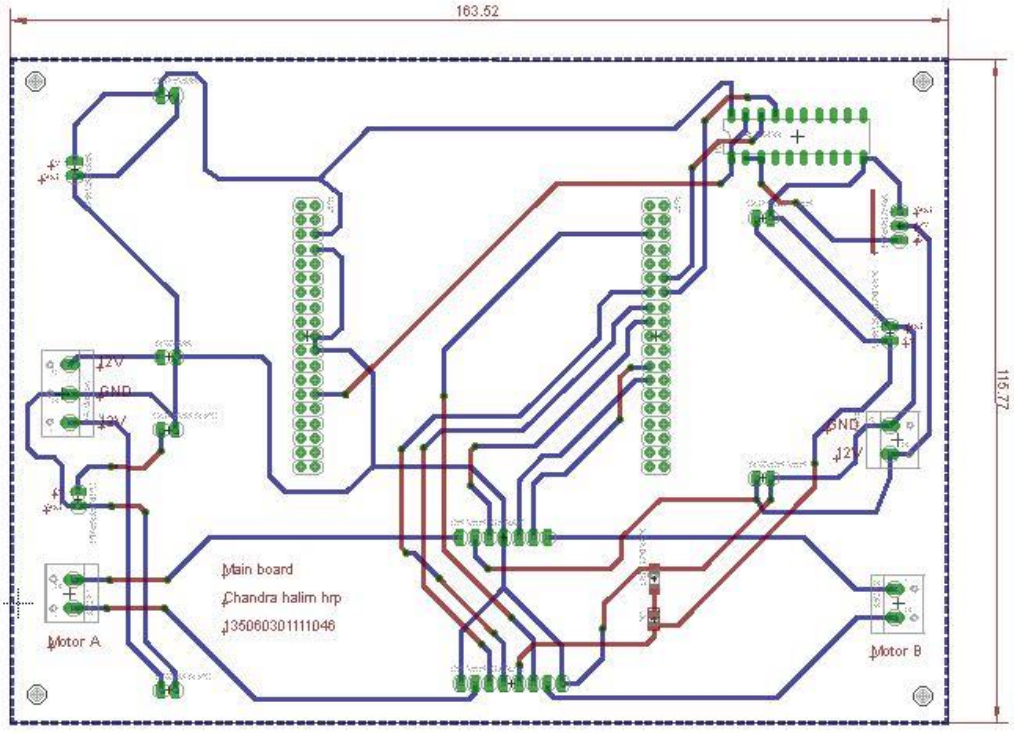
Gambar 1.7 Skematik rangkaian *driver* L298N



Gambar 1.8 *Board* rangkaian *driver* L298N



Gambar 1.9 Skematik rangkaian *board* utama



Gambar 1.10 *Board* Rangkaian *Board* Utama

LAMPIRAN 3

LISTING PROGRAM

1. Listing Program Mikrokontroler Utama

```
#include "stm32f4xx.h"
#include "stm32f4xx_tim.h"
#include "stm32f4xx_rcc.h"
#include "stm32f4xx_gpio.h"
#include "stm32f4xx_usart.h"
#include "stm32f4xx_adc.h"
#include "stm32f4xx_exti.h"
#include "stm32f4xx_syscfg.h"
#include "stm32f4xx_i2c.h"
#include "misc.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <stdarg.h>


//-------------------------------------------------------------
-----//
// EEPROM AREA  //
#define AX_MODEL_NUMBER_L         0
#define AX_MODEL_NUMBER_H         1
#define AX_VERSION                2
#define AX_ID                     3
#define AX_BAUD_RATE              4
#define AX_RETURN_DELAY_TIME      5
#define AX_CW_ANGLE_LIMIT_L       6
#define AX_CW_ANGLE_LIMIT_H       7
#define AX_CCW_ANGLE_LIMIT_L      8
#define AX_CCW_ANGLE_LIMIT_H      9
#define AX_SYSTEM_DATA2           10
#define AX_LIMIT_TEMPERATURE      11
#define AX_DOWN_LIMIT_VOLTAGE     12
#define AX_UP_LIMIT_VOLTAGE       13
#define AX_MAX_TORQUE_L           14
#define AX_MAX_TORQUE_H           15
#define AX_RETURN_LEVEL           16
#define AX_ALARM_LED              17
#define AX_ALARM_SHUTDOWN         18
#define AX_OPERATING_MODE         19
#define AX_DOWN_CALIBRATION_L     20
#define AX_DOWN_CALIBRATION_H     21
#define AX_UP_CALIBRATION_L       22
#define AX_UP_CALIBRATION_H       23

// RAM AREA  //
#define AX_TORQUE_ENABLE          24
#define AX_LED                    25
#define AX_CW_COMPLIANCE_MARGIN   26
#define AX_CCW_COMPLIANCE_MARGIN
27
#define AX_CW_COMPLIANCE_SLOPE    28
#define AX_CCW_COMPLIANCE_SLOPE   29
#define AX_GOAL_POSITION_L        30
#define AX_GOAL_POSITION_H        31
#define AX_GOAL_SPEED_L           32
#define AX_GOAL_SPEED_H           33
#define AX_TORQUE_LIMIT_L         34
#define AX_TORQUE_LIMIT_H         35
#define AX_PRESENT_POSITION_L     36
#define AX_PRESENT_POSITION_H     37
#define AX_PRESENT_SPEED_L        38
#define AX_PRESENT_SPEED_H        39
#define AX_PRESENT_LOAD_L         40
#define AX_PRESENT_LOAD_H         41
#define AX_PRESENT_VOLTAGE        42
#define AX_PRESENT_TEMPERATURE    43
#define AX_REGISTERED_INSTRUCTION 44
#define AX_PAUSE_TIME             45
#define AX_MOVING                 46
#define AX_LOCK                   47
#define AX_PUNCH_L                48
#define AX_PUNCH_H                49

// Status Return Levels //
#define AX_RETURN_NONE            0
#define AX_RETURN_READ            1
#define AX_RETURN_ALL             2

// Instruction Set //
#define AX_PING                   1
#define AX_READ_DATA              2
#define AX_WRITE_DATA             3
#define AX_REG_WRITE              4
#define AX_ACTION                 5
#define AX_RESET                  6
#define AX_SYNC_WRITE             131
```

```
// Specials //
#define OFF                0
#define ON                 1
#define LEFT
        0
#define RIGTH              1
#define AX_BYTE_READ       1
#define AX_BYTE_READ_POS       2
#define AX_RESET_LENGTH
        2
#define AX_ACTION_LENGTH
    2
#define AX_ID_LENGTH       4
#define AX_LR_LENGTH       4
#define AX_SRL_LENGTH      4
#define AX_RDT_LENGTH      4
#define AX_LEDALARM_LENGTH     4
#define AX_SALARM_LENGTH       4
#define AX_TL_LENGTH       4
#define AX_VL_LENGTH       6
#define AX_CM_LENGTH       6
#define AX_CS_LENGTH       5
#define AX_CCW_CW_LENGTH       8
#define AX_BD_LENGTH       4
#define AX_TEM_LENGTH      4
#define AX_MOVING_LENGTH       4
#define AX_RWS_LENGTH      4
#define AX_VOLT_LENGTH     4
#define AX_LED_LENGTH      4
#define AX_TORQUE_LENGTH       4
#define AX_POS_LENGTH      4
#define AX_GOAL_LENGTH     5
#define AX_MT_LENGTH       5
#define AX_PUNCH_LENGTH        5
#define AX_SPEED_LENGTH        5
#define AX_GOAL_SP_LENGTH      7
#define AX_ACTION_CHECKSUM
        250
#define BROADCAST_ID       254
#define AX_START           255
#define AX_CCW_AL_L        255
#define AX_CCW_AL_H        3
#define TIME_OUT           10
#define TX_DELAY_TIME
        400
#define Tx_MODE            1
#define Rx_MODE            0
#define LOCK               1


#define sendData(args)  (Serial3.write(args))
#define availableData() (Serial3.available())
#define readData()     (Serial3.read())
#define peekData()     (Serial3.peek())
#define beginCom(args)  (Serial3.begin(args))
#define endCom()       (Serial3.end())
```

```
#include <inttypes.h>

unsigned char Checksum;
unsigned char Direction_Pin;
unsigned char Time_Counter;
unsigned char Incoming_Byte;
unsigned char Position_High_Byte;
unsigned char Position_Low_Byte;
unsigned char Speed_High_Byte;
unsigned char Speed_Low_Byte;
unsigned char Load_High_Byte;
unsigned char Load_Low_Byte;

int Moving_Byte;
int RWS_Byte;
int Speed_Long_Byte;
int Load_Long_Byte;
int Position_Long_Byte;
int Temperature_Byte;
int Voltage_Byte;
int Error_Byte;

void init();
//-------------------------------------------------------------
-----//
int Dynamixel_moveSpeed(unsigned char ID, int
Position, int Speed);
void konfigurasipindynamixel(void);
//-------------------------------------------------------------
-----//
TIM_ICInitTypeDef
    TIM_ICInitStructure;
TIM_TimeBaseInitTypeDef
    TIM_TimeBaseStructure;
TIM_OCInitTypeDef
    TIM_OCInitStructure;
void konfigurasitimer4(void);
void motor1(int speedm1);
void motor2(int speedm2);
void stop();
void jalanlurus();
void putarkanan();
void putarkiri();
//-------------------------------------------------------------
-----//
void usart1(uint32_t baudrate);
void usart2(uint32_t baudrate);
void usart6(uint32_t baudrate);
void USART6_send(int c);
void USART2_send(int c);
void USART2_sendString(char *ptr, int end_cmd);
int USART6_receiveString(char *ptr);
void cetak (USART_TypeDef* USARTx, const
char *pFormat, ... );
void USART1_IRQHandler(void);
```

```
void USART2_IRQHandler(void);
//-------------------------------------------------------------
-----//
static __IO uint32_t sysTickCounter;
void SysTick_Init(void);
void TimeTick_Decrement(void);
void delay_us(u32 n);
void delay_1ms(void);
void delay_ms(u32 n);
//-------------------------------------------------------------
-----//
#define P_0       GPIO_Pin_0
#define P_1       GPIO_Pin_1
#define P_2       GPIO_Pin_2
#define P_3       GPIO_Pin_3
#define P_4       GPIO_Pin_4
#define P_5       GPIO_Pin_5
#define P_6       GPIO_Pin_6
#define P_7       GPIO_Pin_7
#define P_8       GPIO_Pin_8
#define P_9       GPIO_Pin_9
#define P_10      GPIO_Pin_10
#define P_11      GPIO_Pin_11
#define P_12      GPIO_Pin_12
#define P_13      GPIO_Pin_13
#define P_14      GPIO_Pin_14
#define P_15      GPIO_Pin_15
#define On        0x01
#define Off 0x02
#define TO        0x04
#define SYSTICK_RELOAD_VAL 0xA80000
#define SYSTICK_DIV(x) ((x*0x0186)>>16)
GPIO_TypeDef*           GPIO;
GPIO_InitTypeDef        GPIO_InitStructure;
static __IO uint32_t                TimmingDelay;
unsigned int            Tick;
uint32_t                mlls;
uint32_t                mcrs;
uint32_t                Count;
void init_IO(uint32_t RCC_AHB1Periph_GPIO,
uint16_t Pin, GPIOMode_TypeDef
GPIO_Mode,GPIOOType_TypeDef GPIO_OType,
GPIOPuPd_TypeDef GPIO_PuPd );
void Pin(GPIO_TypeDef* GPIO,uint8_t Stat,
uint16_t Pin);
uint8_t Pin_In(GPIO_TypeDef* GPIO, uint16_t
Pin);
uint8_t Pin_Out(GPIO_TypeDef* GPIO, uint16_t
Pin);
//-------------------------------------------------------------
-----//
void trackball();
void dataolahrawcam();
void searchballrotate();
int cam_check=0,cam_pros=0,cam_end; //interupt
usart 1
int datausart2; //interupt usart 2
//-------------------------------------------------------------
-----//
int main(void)
{
    init();

    while(1)
    {
        //**********Check Data Yang Dikirim
Oleh Raspberry Pi*****************//
                    dataolahrawcam();
//        cetak(USART1,"test");

    cetak(USART1,"%d\t%d\t%d\r",cam_check,ca
m_pros,cam_end);
//            cetak(USART2,"%d %d
%d\n",cam_check,cam_pros,cam_end);
        //************Check
motor**************//
//                  Motor 1
//                  motor1(70);
//                  Pin(GPIOA,On,P_7);    //IN1
//                Pin(GPIOA,Off,P_6);   //IN2
//                  motor 2
//                  motor2(70);
//                  Pin(GPIOB,On,P_10);   //IN3
//                  Pin(GPIOA,Off,P_8);   //IN4
//                        jalanlurus();
//                  putarkiri();
//                  putarkanan();

        //********Check servo serial
dynamixel***********//
//
    Dynamixel_moveSpeed(14,1000,1000);
//                  delay_ms(1000);
//
    Dynamixel_moveSpeed(14,512,1000);
//                  delay_ms(1000);
//
    Dynamixel_moveSpeed(14,1000,1000);
//                  delay_ms(1000);

        //************Uji
Bluetooth***************//
//                  cetak(USART1,"pengujian jarak
komunikasi bluetooth HC-05\r");
//                  cetak(USART1,"Jarak Maksimum
Komunikasi Bluetooth HC-05\r");

        //**************Check
GPIO*****************//
//                  Pin(GPIOA,On,GPIO_Pin_5);
//                  delay_ms(1000);
```

```
//                  Pin(GPIOA,Off,GPIO_Pin_5);
//                  delay_ms(1000);

//                          searchballrotate();
                            trackball();

    }
}
void trackball()
{
    dataolahrawcam();
    if (cam_pros==800)
    {
        dataolahrawcam();
        if(cam_end > 0)
        {

            dataolahrawcam();
            putarkiri();
        }
        if(cam_end < 0)
        {

            dataolahrawcam();
            putarkanan();
        }
    }
    else
    {
        searchballrotate();
    }

}
void searchballrotate()
{
    dataolahrawcam();
    if (cam_pros==1)//1
    {
        putarkiri();
    }
    else if (cam_pros==2)//2
    {
        stop();
    }
    else if (cam_pros==3)//3
    {
//      jalanlurus();
        stop();
    }
    else if (cam_pros==12)//4 //BATAS TENGAH
    {
//      jalanlurus();
        stop();
    }
    else if (cam_pros==12)//5 //BATAS TENGAH
    {
//      jalanlurus();
        stop();
    }
    else if (cam_pros==12)//6 //BATAS TENGAH
    {
//      jalanlurus();
        stop();
    }
    else if (cam_pros==-1)//7
    {
//      jalanlurus();
        stop();
    }
    else if (cam_pros==-2)//8
    {
        stop();;
    }
    else if (cam_pros==-3)//9
    {
        putarkanan();
    }
    else if (cam_pros==800)//A
    {
        stop();
    }
}
void dataolahrawcam()
{
    if (cam_check==49)//1
    {
        cam_pros=1;
        cam_end=cam_pros;
    }
    else if (cam_check==50)//2
    {
        cam_pros=2;
        cam_end=cam_pros;
    }
    else if (cam_check==51)//3
    {
        cam_pros=3;
        cam_end=cam_pros;
    }
    else if (cam_check==52)//4 //BATAS TENGAH
    {
        cam_pros=12;
        cam_end=cam_pros;
    }
    else if (cam_check==53)//5 //BATAS TENGAH
    {
        cam_pros=12;
        cam_end=cam_pros;
    }
    else if (cam_check==54)//6 //BATAS TENGAH
    {
        cam_pros=12;
```

```
        cam_end=cam_pros;
    }
    else if (cam_check==55)//7
    {
        cam_pros=-1;
        cam_end=cam_pros;
    }
    else if (cam_check==56)//8
    {
        cam_pros=-2;
        cam_end=cam_pros;
    }
    else if (cam_check==57)//9
    {
        cam_pros=-3;
        cam_end=cam_pros;
    }
    else if (cam_check==67)//C
    {
        cam_pros=800;
    }
}
void jalanlurus()
{
//  lumayan lurus
    motor1(60);//75
    Pin(GPIOA, Off, GPIO_Pin_6);
    Pin(GPIOA, On, GPIO_Pin_7);
    motor2(55);
    Pin(GPIOA, On, GPIO_Pin_8);
    Pin(GPIOB, Off, GPIO_Pin_10);
}

void putarkiri()
{
//  putar kiri
    motor1(60);
    Pin(GPIOA, On, GPIO_Pin_6);
    Pin(GPIOA, Off, GPIO_Pin_7);
    motor2(60);
    Pin(GPIOA, On, GPIO_Pin_8);
    Pin(GPIOB, Off, GPIO_Pin_10);
}

void putarkanan()
{
//  putar kiri
    motor1(60);
    Pin(GPIOA, Off, GPIO_Pin_6);
    Pin(GPIOA, On, GPIO_Pin_7);
    motor2(61);
    Pin(GPIOA, Off, GPIO_Pin_8);
    Pin(GPIOB, On, GPIO_Pin_10);
}

void stop()
{
//  lumayan lurus
    motor1(62);//75
    Pin(GPIOA, Off, GPIO_Pin_6);
    Pin(GPIOA, Off, GPIO_Pin_7);
    motor2(62);
    Pin(GPIOA, Off, GPIO_Pin_8);
    Pin(GPIOB, Off, GPIO_Pin_10);
}
void motor1(int speedm1)
{
    int kecMOTOne;

    kecMOTOne = (uint16_t) (( speedm1 * (279 -
1)) / 100);
    TIM4->CCR1 = kecMOTOne;
}

void motor2(int speedm2)
{
    int kecMOTTwo;

    kecMOTTwo = (uint16_t) (( speedm2 * (279 -
1)) / 100);
    TIM4->CCR4 = kecMOTTwo;
}

void konfigurasitimer4(void)
{
    //clock enable

RCC_APB1PeriphClockCmd(RCC_APB1Periph_T
IM4, ENABLE);

RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_
GPIOB, ENABLE);

    //GPIO__TIM_init
    GPIO_InitStructure.GPIO_Pin   =
P_6|P_7|P_8|P_9;
    GPIO_InitStructure.GPIO_Mode  =
GPIO_Mode_AF;
    GPIO_InitStructure.GPIO_Speed =
GPIO_Speed_100MHz;
    GPIO_InitStructure.GPIO_OType =
GPIO_OType_PP;
    GPIO_InitStructure.GPIO_PuPd  =
GPIO_PuPd_NOPULL;
    GPIO_Init(GPIOB, &GPIO_InitStructure);
    GPIO_PinAFConfig(GPIOB,
GPIO_PinSource6, GPIO_AF_TIM4);
    GPIO_PinAFConfig(GPIOB,
GPIO_PinSource7, GPIO_AF_TIM4);
    GPIO_PinAFConfig(GPIOB,
GPIO_PinSource8, GPIO_AF_TIM4);
```

```
    GPIO_PinAFConfig(GPIOB,
GPIO_PinSource9, GPIO_AF_TIM4);

    //timer_init
    TIM_TimeBaseStructure.TIM_ClockDivision
= 0;
    TIM_TimeBaseStructure.TIM_CounterMode
= TIM_CounterMode_Up;
    TIM_TimeBaseStructure.TIM_Period        =
279;
    TIM_TimeBaseStructure.TIM_Prescaler     =
14;

TIM_TimeBaseStructure.TIM_RepetitionCounter =
0;
    TIM_TimeBaseInit(TIM4,
&TIM_TimeBaseStructure);

    //Channel 1,2,3, and 4//
    TIM_OCInitStructure.TIM_OCMode          =
TIM_OCMode_PWM2;
    TIM_OCInitStructure.TIM_OutputState     =
TIM_OutputState_Enable;
    TIM_OCInitStructure.TIM_OutputNState
= TIM_OutputNState_Enable;
    TIM_OCInitStructure.TIM_Pulse           = 0;
    TIM_OCInitStructure.TIM_OCPolarity      =
TIM_OCPolarity_Low;
    TIM_OCInitStructure.TIM_OCNPolarity
= TIM_OCNPolarity_Low;
    TIM_OCInitStructure.TIM_OCIdleState     =
TIM_OCIdleState_Set;
    TIM_OCInitStructure.TIM_OCNIdleState
= TIM_OCIdleState_Reset;

    TIM_OC1Init(TIM4, &TIM_OCInitStructure);
//  TIM_OC1PreloadConfig(TIM4,
TIM_OCPreload_Enable);
                        //ditambah
    TIM_OCInitStructure.TIM_OutputState     =
TIM_OutputState_Enable;
    TIM_OCInitStructure.TIM_Pulse           = 0;

    TIM_OC2Init(TIM4, &TIM_OCInitStructure);
    TIM_OC2PreloadConfig(TIM4,
TIM_OCPreload_Enable);
    TIM_OCInitStructure.TIM_OutputState     =
TIM_OutputState_Enable;
    TIM_OCInitStructure.TIM_Pulse           = 0;

    TIM_OC3Init(TIM4, &TIM_OCInitStructure);
    TIM_OC3PreloadConfig(TIM4,
TIM_OCPreload_Enable);
//  TIM_OCInitStructure.TIM_OutputState     =
TIM_OutputState_Enable;     //ditambah
//  TIM_OCInitStructure.TIM_Pulse           = 0;
    //ditambah

    TIM_OC4Init(TIM4, &TIM_OCInitStructure);
    TIM_OC4PreloadConfig(TIM4,
TIM_OCPreload_Enable);
    TIM_OCInitStructure.TIM_OutputState     =
TIM_OutputState_Enable;
    TIM_OCInitStructure.TIM_Pulse           = 0;


    //timer enable
    TIM_Cmd(TIM4, ENABLE);
    TIM_CtrlPWMOutputs(TIM4, ENABLE);
}
void cetak(USART_TypeDef* USARTx, const char
*pFormat, ... )
{
  va_list ap;
  char pStr[100];

  va_start(ap, pFormat);
  vsprintf(pStr, pFormat, ap);
  va_end(ap);

  int i=0;
  int n = strlen(pStr);
  for(i=0;i<n;i++)
  {
    USART_SendData(USARTx, (uint8_t)pStr[i]);
    while (USART_GetFlagStatus(USARTx,
USART_FLAG_TC) == RESET);
  }
}
void USART1_IRQHandler(void)
{
  if(USART_GetITStatus(USART1,
USART_IT_RXNE) )
    {
      {
            cam_check = USART1->DR;
      }
    }
}
void USART2_IRQHandler(void)
{
  if(USART_GetITStatus(USART2,
USART_IT_RXNE) )
    {
      {
            datausart2 = USART2->DR;
      }
    }
}
void usart1(uint32_t baudrate)
```

```
{
  USART_InitTypeDef USART_InitStructure;
  GPIO_InitTypeDef  GPIO_InitStructure;
  NVIC_InitTypeDef  NVIC_InitStructure;

  // mengaktifkan APB2 peripheral clock untuk
USART1

RCC_APB2PeriphClockCmd(RCC_APB2Periph_
USART1,ENABLE);
  // mengaktifkan APB1 peripheral clock untuk
GPIOA

RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_
GPIOA, ENABLE);

  // mengkoneksikan pins USART1 ke AF
  GPIO_PinAFConfig(GPIOA,
GPIO_PinSource10, GPIO_AF_USART1);
  GPIO_PinAFConfig(GPIOA, GPIO_PinSource9,
GPIO_AF_USART1);

  // konfigurasi pin USART1 :  PA.12 Tx, PA.11
Rx //
  GPIO_InitStructure.GPIO_Pin          =
GPIO_Pin_10 | GPIO_Pin_9;
  GPIO_InitStructure.GPIO_Speed        =
GPIO_Speed_50MHz;
  GPIO_InitStructure.GPIO_Mode         =
GPIO_Mode_AF;
  GPIO_InitStructure.GPIO_OType        =
GPIO_OType_PP;
  GPIO_InitStructure.GPIO_PuPd         =
GPIO_PuPd_UP;
  GPIO_Init(GPIOA, &GPIO_InitStructure);

  // konfigurasi USART
  USART_InitStructure.USART_BaudRate
= baudrate;
  USART_InitStructure.USART_WordLength
= USART_WordLength_8b;
  USART_InitStructure.USART_StopBits       =
USART_StopBits_1;
  USART_InitStructure.USART_Parity         =
USART_Parity_No;

USART_InitStructure.USART_HardwareFlowCont
rol = USART_HardwareFlowControl_None;
  USART_InitStructure.USART_Mode       =
USART_Mode_Rx | USART_Mode_Tx;
  USART_Init(USART1, &USART_InitStructure);

  //inisialisasi Usart Interupt//
   USART_ITConfig(USART1,
USART_IT_RXNE, ENABLE);
    // enable the USART1 receive interrupt

    NVIC_InitStructure.NVIC_IRQChannel
= USART1_IRQn;
          // we want to configure the USART3
interrupts
    NVIC_InitStructure.NVIC_IRQChannelPreemp
tionPriority = 0;          // this sets the
priority group of the USART3 interrupts
    NVIC_InitStructure.NVIC_IRQChannelSubPrio
rity = 1;
                // this sets the subpriority inside
the group
    NVIC_InitStructure.NVIC_IRQChannelCmd
= ENABLE;
                // the USART3 interrupts are
globally enabled
    NVIC_Init(&NVIC_InitStructure);

                                    // the properties
are passed to the NVIC_Init function which takes
care of the low level stuff
    ////////////////////////////
  // Enable USART1
  USART_Cmd(USART1, ENABLE);
}
void usart2(uint32_t baudrate)
{
  USART_InitTypeDef USART_InitStructure;
  GPIO_InitTypeDef GPIO_InitStructure;
  NVIC_InitTypeDef  NVIC_InitStructure;

  // mengaktifkan APB1 peripheral clock untuk
USART2

RCC_APB1PeriphClockCmd(RCC_APB1Periph_
USART2,ENABLE);
  // mengaktifkan APB1 peripheral clock untuk
GPIOA

RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_
GPIOA, ENABLE);

  // mengkoneksikan pin USART2 ke AF
  GPIO_PinAFConfig(GPIOA, GPIO_PinSource2,
GPIO_AF_USART2);
  GPIO_PinAFConfig(GPIOA, GPIO_PinSource3,
GPIO_AF_USART2);

  // konfigurasi pin USART2 :  PA.2 Tx, PA.3 Rx
//
  GPIO_InitStructure.GPIO_Pin          =
GPIO_Pin_2 | GPIO_Pin_3;
```

```
   GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
   GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF;
   GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
   GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_UP;
   GPIO_Init(GPIOA, &GPIO_InitStructure);

   // konfigurasi USART
   USART_InitStructure.USART_BaudRate = baudrate;
   USART_InitStructure.USART_WordLength = USART_WordLength_8b;
   USART_InitStructure.USART_StopBits = USART_StopBits_1;
   USART_InitStructure.USART_Parity = USART_Parity_No;

USART_InitStructure.USART_HardwareFlowControl = USART_HardwareFlowControl_None;
   USART_InitStructure.USART_Mode = USART_Mode_Rx | USART_Mode_Tx;
   USART_Init(USART2, &USART_InitStructure);

   //inisialisasi Usart Interupt//
  USART_ITConfig(USART2, USART_IT_RXNE, ENABLE);
// enable the USART1 receive interrupt

   NVIC_InitStructure.NVIC_IRQChannel = USART2_IRQn;          // we want to configure the USART3 interrupts

NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;          // this sets the priority group of the USART3 interrupts

NVIC_InitStructure.NVIC_IRQChannelSubPriority = 1;          // this sets the subpriority inside the group
   NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;          // the USART3 interrupts are globally enabled
   NVIC_Init(&NVIC_InitStructure);
// the properties are passed to the NVIC_Init function which takes care of the low level stuff
   ////////////////////////////////

   // Enable USART2
   USART_Cmd(USART2, ENABLE);

}
void usart6(uint32_t baudrate)
{
   USART_InitTypeDef USART_InitStructure;
   GPIO_InitTypeDef GPIO_InitStructure;
   NVIC_InitTypeDef  NVIC_InitStructure;

   // mengaktifkan APB2 peripheral clock untuk USART6
   RCC_APB2PeriphClockCmd(RCC_APB2Periph_USART6,ENABLE);
   RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOA, ENABLE);


   // mengkoneksikan pin USART6 ke AF
   GPIO_PinAFConfig(GPIOA, GPIO_PinSource12, GPIO_AF_USART6);
   GPIO_PinAFConfig(GPIOA, GPIO_PinSource11, GPIO_AF_USART6);

   /* konfigurasi pin USART6:  PA.12 Tx, PA.11 Rx */
   GPIO_InitStructure.GPIO_Pin = GPIO_Pin_12 | GPIO_Pin_11;
   GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
   GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF;
   GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
   GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_UP;
   GPIO_Init(GPIOA, &GPIO_InitStructure);

   // konfigurasi USART
   USART_InitStructure.USART_BaudRate = baudrate;
   USART_InitStructure.USART_WordLength = USART_WordLength_8b;
   USART_InitStructure.USART_StopBits = USART_StopBits_1;
   USART_InitStructure.USART_Parity = USART_Parity_No;
   USART_InitStructure.USART_HardwareFlowControl = USART_HardwareFlowControl_None;
   USART_InitStructure.USART_Mode = USART_Mode_Rx | USART_Mode_Tx;
   USART_Init(USART6, &USART_InitStructure);

   //inisialisasi Usart Interupt//
   USART_ITConfig(USART6, USART_IT_RXNE, ENABLE);

         // enable the USART1 receive interrupt

   NVIC_InitStructure.NVIC_IRQChannel = USART6_IRQn;
```

```
        // we want to configure the USART3
interrupts
    NVIC_InitStructure.NVIC_IRQChannelPreemp
tionPriority = 0;
                // this sets the priority group of
the USART3 interrupts
    NVIC_InitStructure.NVIC_IRQChannelSubPrio
rity       = 1;
                // this sets the subpriority inside
the group
    NVIC_InitStructure.NVIC_IRQChannelCmd
= ENABLE;
        // the USART3 interrupts are globally
enabled
    NVIC_Init(&NVIC_InitStructure);

    // the properties are passed to the NVIC_Init
function which takes care of the low level stuff
    ////////////////////////////

    // Enable USART6
    USART_Cmd(USART6, ENABLE);

}
// Send USART6
void USART6_send(int c)
{
   while(USART_GetFlagStatus(USART6,
USART_FLAG_TXE) == RESET); // Wait empty
   USART_SendData(USART6, c); // Send char
}
// Send USART
void USART2_send(int c)
{
   while (USART_GetFlagStatus(USART2,
USART_FLAG_TXE) == RESET); // Wait empty
   USART_SendData(USART2, c); // Send char
}

// Send string data
void USART2_sendString(char *ptr, int end_cmd)
{
   while (*ptr != 0)
   {
      USART2_send(*ptr);
      ptr++;
   }

   if (end_cmd == 0)
   {
      USART2_send(0x0A); // send LineFeed
      USART2_send(0x0D); // send CariageReturn
   }
   else if (end_cmd == 1)
   {
      USART2_send(0x0D); // send CariageReturn
      USART2_send(0x0A); // send LineFeed
   }
   else if (end_cmd == 2)
   {
      USART2_send(0x0A); // send LineFeed
   }
   else if (end_cmd == 3)
   {
      USART2_send(0x0D); // send CariageReturn
   }
}
void SysTick_Init(void) {

/****************************************
    *SystemFrequency = 84Mhz
    *SystemFrequency/1000      1ms         *
    *SystemFrequency/100000    10us        *
    *SystemFrequency/1000000   1us         *

****************************************/
   while (SysTick_Config(SystemCoreClock /
1000000) != 0) {
   }

}
void SysTick_Handler(void)
{
    TimeTick_Decrement();
}
void TimeTick_Decrement(void) {
   if (sysTickCounter != 0x00) {
      sysTickCounter--;
   }
}
void delay_us(u32 n) {
   sysTickCounter = n;
   while (sysTickCounter != 0  ) {
   }
}
void delay_1ms(void) {
   sysTickCounter = 1000;
   while (sysTickCounter != 0) {
   }
}
void delay_ms(u32 n) {
   while (n--) {
      delay_1ms();
   }
}
void init_IO(uint32_t RCC_AHB1Periph_GPIO,
uint16_t Pin, GPIOMode_TypeDef
GPIO_Mode,GPIOOType_TypeDef GPIO_OType,
GPIOPuPd_TypeDef GPIO_PuPd )
{
```

```
    RCC_AHB1PeriphClockCmd(RCC_AHB1Peri
ph_GPIO, ENABLE);
    switch(RCC_AHB1Periph_GPIO){
        case ((uint32_t)0x00000001) : GPIO =
GPIOA; break;
        case ((uint32_t)0x00000002) : GPIO =
GPIOB; break;
        case ((uint32_t)0x00000004) : GPIO =
GPIOC; break;
        case ((uint32_t)0x00000008) : GPIO =
GPIOD; break;
        case ((uint32_t)0x00000010) : GPIO =
GPIOE; break;
        case ((uint32_t)0x00000020) : GPIO =
GPIOF;  break;
    }

    GPIO_InitStructure.GPIO_Pin = Pin;
    GPIO_InitStructure.GPIO_Mode =
GPIO_Mode;
    GPIO_InitStructure.GPIO_OType =
GPIO_OType;
    GPIO_InitStructure.GPIO_Speed =
GPIO_Speed_100MHz;
    GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd;
    GPIO_Init(GPIO, &GPIO_InitStructure);
}
void Pin(GPIO_TypeDef* GPIO,uint8_t Stat,
uint16_t Pin){
    switch(Stat){
        case 0x01 : GPIO->BSRRL = Pin; break;
        case 0x02 : GPIO->BSRRH = Pin; break;
        case 0x04 : GPIO->ODR  ^= Pin; break;
    }
}
uint8_t Pin_In(GPIO_TypeDef* GPIO, uint16_t
Pin){
    uint8_t bit = 0;
        if((GPIO->IDR & Pin) != 0) bit = 1;
    else bit = 0;
    return bit;
}
uint8_t Pin_Out(GPIO_TypeDef* GPIO, uint16_t
Pin){
    uint8_t bit = 0;
        if((GPIO->ODR & Pin) != 0) bit = 1;
    else bit = 0;
    return bit;
}
int Dynamixel_moveSpeed(unsigned char ID, int
Position, int Speed)
{
  char Position_H,Position_L,Speed_H,Speed_L;
  Position_H = Position >> 8;
  Position_L = Position;            // 16 bits - 2 x 8
bits variables
  Speed_H = Speed >> 8;
  Speed_L = Speed;                  // 16 bits - 2 x 8
bits variables
  Checksum = (~(ID + AX_GOAL_SP_LENGTH
+ AX_WRITE_DATA +
AX_GOAL_POSITION_L + Position_L +
Position_H + Speed_L + Speed_H))&0xFF;

  GPIO_SetBits(GPIOA, GPIO_Pin_0);

  USART6_send(AX_START); // Send
Instructions over Serial
  USART6_send(AX_START);
  USART6_send(ID);
  USART6_send(AX_GOAL_SP_LENGTH);
  USART6_send(AX_WRITE_DATA);
  USART6_send(AX_GOAL_POSITION_L);
  USART6_send(Position_L);
  USART6_send(Position_H);
  USART6_send(Speed_L);
  USART6_send(Speed_H);
  USART6_send(Checksum);

  delay_ms(TX_DELAY_TIME);

  GPIO_ResetBits(GPIOA, GPIO_Pin_0);

  return -1 ;
}
void konfigurasipindynamixel(void)
{
  GPIO_InitTypeDef GPIO_InitStructure;

RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_
GPIOA, ENABLE);

  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0;
  GPIO_InitStructure.GPIO_Mode =
GPIO_Mode_OUT;
  GPIO_InitStructure.GPIO_Speed =
GPIO_Speed_50MHz;
  GPIO_Init(GPIOA, &GPIO_InitStructure);
}
void init()
{
   //Inisialisasi untuk delay
   SystemInit();
   SysTick_Init();
   //Inisialisasi Usart
   usart1(9600);
   usart2(9600);
   usart6(9600);
   //Inisialisasi pin led
```

```
    init_IO(RCC_AHB1Periph_GPIOA,
GPIO_Pin_5, GPIO_Mode_OUT,
GPIO_OType_PP, GPIO_PuPd_NOPULL );//led
    //inisialisasi pin direction motor 1
    init_IO(RCC_AHB1Periph_GPIOA,
GPIO_Pin_6, GPIO_Mode_OUT,
GPIO_OType_PP, GPIO_PuPd_NOPULL );//IN2
    init_IO(RCC_AHB1Periph_GPIOA,
GPIO_Pin_7, GPIO_Mode_OUT,
GPIO_OType_PP, GPIO_PuPd_NOPULL );//IN1
    //inisialisasi pin direction motor 2
    init_IO(RCC_AHB1Periph_GPIOA,
GPIO_Pin_8, GPIO_Mode_OUT,
GPIO_OType_PP, GPIO_PuPd_NOPULL );//IN4
    init_IO(RCC_AHB1Periph_GPIOB,
GPIO_Pin_10,GPIO_Mode_OUT,
GPIO_OType_PP, GPIO_PuPd_NOPULL );//IN3
    //Inisialisasi timer
    konfigurasitimer4();
    //  Inisialisasi pin Dynamixel
    konfigurasipindynamixel();
}
```

2. Listing Program Raspberry Pi

```
#include "opencv2/highgui/highgui.hpp"
#include "opencv2/imgproc/imgproc.hpp"
#include <iostream>
#include <stdlib.h>
#include <stdio.h>
#include <fcntl.h>
#include <sys/time.h>
#include <cv.h>
#include <termios.h>
#include <unistd.h>
#include <sstream>
#include <string>
#include <vector>
#include <highgui.h>
#include <time.h>
#include "serial_test.h"

using namespace cv;
using namespace std;

//Parameter HSV awal
int H_MIN;
int H_MAX;
int S_MIN;
int S_MAX;
int V_MIN;
int V_MAX;

//Variabel Mouse Dan Roi
bool modekalibrasi;
bool mousedidrag;
bool mousebergerak;
bool kotakseleksi;
cv::Point koordinatawalklikmouse,
posisiterbarukoordinatklikmouse;
cv::Rect kotakROI;
vector<int> HB_ROI, SB_ROI, VB_ROI;

//Variabel window yang akan ditampikan
const string namaTrackbarwindow = "Trackbar";
const string namaWindoworiginal = "Gambar
Original";
const string namaWindowHSV = "Gambar HSV";
const string namaWindowHSVROI = "HSVROI";

// konstanta alamat PORT Usart.
static const char* portName = "/dev/ttySAC0";
char dataIn = 0;

//fungsi klik dan drag kotak
void klikdandragkotak(int event, int x, int y, int
flags, void* param)
{
    if (modekalibrasi == true)
    {
    Mat* videoFeed = (Mat*)param;
    if (event == CV_EVENT_LBUTTONDOWN
&& mousedidrag == false)
```

```
    {
        koordinatawalklikmouse = cv::Point(x, y);
        mousedidrag = true;
    }
    if (event == CV_EVENT_MOUSEMOVE &&
mousedidrag == true)
    {
        posisiterbarukoordinatklikmouse =
cv::Point(x, y);
        mousebergerak = true;
    }
    if (event == CV_EVENT_LBUTTONUP &&
mousedidrag == true)
    {
        kotakROI     =
Rect(koordinatawalklikmouse,
posisiterbarukoordinatklikmouse);
        mousedidrag   = false;
        mousebergerak = false;
        kotakseleksi  = true;
    }
    if (event == CV_EVENT_RBUTTONDOWN)
    {
        H_MIN = 0;
        S_MIN = 0;
        V_MIN = 0;
        H_MAX = 255;
        S_MAX = 255;
        V_MAX = 255;
    }
    if (event == CV_EVENT_MBUTTONDOWN)
    {
        //Tidak melakukan apa - apa
    }
    }

}

//fungsi simpan nilai HSV
void simpannilaiHSV(cv::Mat frame, cv::Mat
hsv_frame)
{
    //jika mouse sudah tidak bergerak dan kotak
telah terseleksi
    if (mousebergerak == false &&  kotakseleksi ==
true)
    {
        if (HB_ROI.size()>0) HB_ROI.clear();
        if (SB_ROI.size()>0) SB_ROI.clear();
        if (VB_ROI.size()>0 )VB_ROI.clear();
        if (kotakROI.width<1 ||
kotakROI.height<1) cout << "drag bentuk kotak
jangan garis" << endl;
        else
        {
                for (int i = kotakROI.x;
i<kotakROI.x + kotakROI.width; i++){
                        for (int j = kotakROI.y;
j<kotakROI.y + kotakROI.height; j++)
                        {

    HB_ROI.push_back((int)hsv_frame.at<cv::Vec3
b>(j, i)[0]);

    SB_ROI.push_back((int)hsv_frame.at<cv::Vec3
b>(j, i)[1]);

    VB_ROI.push_back((int)hsv_frame.at<cv::Vec3
b>(j, i)[2]);
                        }
                }
        }
        kotakseleksi = false;

        //Tampilkan nilai - nilai parameter HSV di
terminal
        if (HB_ROI.size()>0)
        {
                H_MIN =
*std::min_element(HB_ROI.begin(),
HB_ROI.end());
                H_MAX =
*std::max_element(HB_ROI.begin(),
HB_ROI.end());
                //cout << "MIN 'H' Nilai: " <<
H_MIN << endl;
                //cout << "MAX 'H' Nilai: " <<
H_MAX << endl;
        }
        if (SB_ROI.size()>0)
        {
                S_MIN =
*std::min_element(SB_ROI.begin(),
SB_ROI.end());
                S_MAX =
*std::max_element(SB_ROI.begin(),
SB_ROI.end());
                //cout << "MIN 'S' Nilai: " <<
S_MIN << endl;
                //cout << "MAX 'S' Nilai: " <<
S_MAX << endl;
        }
        if (VB_ROI.size()>0)
        {
                V_MIN =
*std::min_element(VB_ROI.begin(),
VB_ROI.end());
                V_MAX =
*std::max_element(VB_ROI.begin(),
VB_ROI.end());
```

```
                //cout << "MIN 'V' Nilai: " <<
V_MIN << endl;
                //cout << "MAX 'V' Nilai: " <<
V_MAX << endl;
        }

    }

    if (mousebergerak == true)
    {
        rectangle(frame, koordinatawalklikmouse,
cv::Point(posisiterbarukoordinatklikmouse.x,
posisiterbarukoordinatklikmouse.y), cv::Scalar(255,
255, 255), 1, 8, 0);

        //Menampilkan window ROI yang
dipotong
        Mat roi =
frame(Rect(koordinatawalklikmouse,
posisiterbarukoordinatklikmouse));
        namedWindow("ROI",
WINDOW_AUTOSIZE);
        imshow("ROI", roi);

    }

}


int main(int argc, char* argv[])
 {
//*****************************Konfigurasi
Usart*********************************//
    char path[256];
    for (int i = 0; i<=1; i++)
    {
      // reset variabel path
      memset(path,0,sizeof(path));

    // menggabung variabel path dengan pin mode
sementara dari file
      sprintf(path, "%s%s%d",
GPIO_MODE_PATH, GPIO_FILENAME, i);

    // membuat file descriptor
      int pinMode = open(path, O_RDWR);

    // atur set pin mode menjadi serial
      setPinMode(pinMode, SERIAL);

      // menutup pinmode
      close(pinMode);
    }

      // varibel untuk Port serial
      int serialPort;

    //struct untuk menahan settingan port
      struct termios portOptions;

    // Membuka serial port sebagai read.write,
bukan sebagai controlling terminal dan
    // tidak memblok CPU karena terlalu lama untuk
membuka port serial
    serialPort = open(portName, O_RDWR |
O_NOCTTY | O_NDELAY );

    // Mengambil settingan port saat ini
      tcgetattr(serialPort, &portOptions);

      // Flush port buffers (in dan out) sebelum mulai
menggunakan
      tcflush(serialPort, TCIOFLUSH);

      // Atur baudrates input output
      cfsetispeed(&portOptions, B9600);
      cfsetospeed(&portOptions, B9600);

    // Atur c_flag dari port agar menahan settingan
dari data bits, parity, stop bits
      // Dan flow kontrol hardware
      portOptions.c_cflag |= CLOCAL;
      portOptions.c_cflag |= CREAD;

      // atur informasi frame
      portOptions.c_cflag &= ~CSIZE; // info
ukuran frame diclear
      portOptions.c_cflag |= CS8;    // atur 8 bit
frames
      portOptions.c_cflag &= ~PARENB;// no parity
      portOptions.c_cflag &= ~CSTOPB;// satu stop
bit

      // panggil kembali ke system
      tcsetattr(serialPort, TCSANOW,
&portOptions);

      // Flush buffer satu kali lagi.
      tcflush(serialPort, TCIOFLUSH);


    FILE*           serial = fopen("/dev/ttySAC0",
"w");
     if (serial == 0)
     {
          printf("Failed to open serial port\n");
     }
     sleep(1);
```

```
//****************************************
****************************************
**********//

   //deklarasi gambar sebenarnya
   Mat imgOriginalB;

   //capture gambar video secara real-time pada
webcam melalui port 0 raspberry
   VideoCapture cap(0);

   if ( !cap.isOpened() )  // jika tidak berhasil
program tidak dijalankan
   {
      cout << "Tidak bisa membuka port webcam"
<< endl;
      return -1;
   }

//**************************************Pen
gujian FPS
****************************************//
   /*
   // Deklarasi FPS
   double fps = cap.get(CV_CAP_PROP_FPS);

   // banyak frame yang diambil
   int num_frames = 20;

   // mulai dan stop waktu
   time_t start, end;

   cout << "Mengcapture " << num_frames << "
frame" << endl ;

   // Mulai waktu
   time(&start);

   // Ambil frame
   for(int i = 0; i < num_frames; i++)
   {
      cap >> imgOriginalB;
   }

   // Waktu berhenti
   time(&end);

   // Waktu yang berlalu
   double seconds = difftime (end, start);
   cout << "Waktu Yang Dibutuhkan : " << seconds
<< " detik" << endl;

   // Calculate frames per second
   fps  = num_frames / seconds;
   cout << "Estimasi frame per detik : " << fps <<
endl;

   */
//****************************************
****************************************
**//

   //atur x,y frame (480,320)
   cap.set(CV_CAP_PROP_FRAME_WIDTH,
640);//425
   cap.set(CV_CAP_PROP_FRAME_HEIGHT,
480);//233

   //deklarasi variabel akhir dari nilai frame x,y

   int iLastXB = 0;
   int iLastYB = 0;

   //deklarasi nama window
   cv::namedWindow(namaWindoworiginal);

   //aktifkan mode kalibrasi
   modekalibrasi = true;

   //aktifkan seleksi kotak menggunakan mouse pada
gambar sebenarnya
   cv::setMouseCallback(namaWindoworiginal,
klikdandragkotak, &imgOriginalB);

   //kondisi awal di non aktifkan
   mousedidrag   = false;
   mousebergerak = false;
   kotakseleksi  = false;

   //Capture gambar sementara dari kamera
   Mat imgTmpB;
   cap.read(imgTmpB);

while (true)
 {
   //Ambil frame baru
   bool bSuccessB = cap.read(imgOriginalB);

   //deklarasi HSV Bola
   Mat imgHSVB;

   //konversi RGB ke HSV
   cvtColor(imgOriginalB, imgHSVB,
COLOR_BGR2HSV);

   //Ambil nilai HSV ROI
   simpannilaiHSV(imgOriginalB, imgHSVB);

   //deklarasi threshold BOLA
```

```
   Mat imgThresholdedB;

   //Isi nilai HSV bola berdasarkan range hasil
ambil HSV ROI dan buat threshold
   inRange(imgHSVB, Scalar(H_MIN, S_MIN,
V_MIN), Scalar(H_MAX, S_MAX, V_MAX),
imgThresholdedB);

   //morphological opening (menghapus obyek -
obyek kecil pada foreground)
   erode(imgThresholdedB, imgThresholdedB,
getStructuringElement(MORPH_ELLIPSE, Size(5,
5)) );
   dilate( imgThresholdedB, imgThresholdedB,
getStructuringElement(MORPH_ELLIPSE, Size(5,
5)) );

   //morphological closing (menghapus lubang -
lubang kecil pada foreground)
   dilate( imgThresholdedB, imgThresholdedB,
getStructuringElement(MORPH_ELLIPSE, Size(5,
5)) );
   erode(imgThresholdedB, imgThresholdedB,
getStructuringElement(MORPH_ELLIPSE, Size(5,
5)) );

   //deklarasi moment threshold
   Moments oMomentsB =
moments(imgThresholdedB);

   double dM01B  = oMomentsB.m01;
   double dM10B  = oMomentsB.m10;
   double dAreaB = oMomentsB.m00;

   //Variabel penanda gambar hasil deteksi
   Mat hasildeteksi = Mat::zeros( imgTmpB.size(),
CV_8UC3 );;

 if (dAreaB>10000)
 {

   //Variabel baru untuk memperoleh nilai frame
x,y berdasarkan moment threshold
    int posXB = dM10B / dAreaB;
    int posYB = dM01B / dAreaB;

    //Update variabel lama frame x,y dengan
variabel baru
    iLastXB = posXB;
    iLastYB = posYB;

    if (iLastXB > 0 && iLastYB > 0)
    {
          //Menggambar lingkaran sebagai penanda
hasil deteksi
          //circle(hasildeteksi, Point(iLastXB,
iLastYB), 10, Scalar(0, 0, 0), 2);
          circle(hasildeteksi, Point(iLastXB,
iLastYB), 10, Scalar(255, 255, 0), 2);
    }

    //************************Cetak Data X dan
Y*************************//
    printf("posxB = %d posyB = %d\n", iLastXB,
iLastYB);


    //**********************kirim data 3
Kondisi*************************//
    //Kondisi 1
    if((iLastXB>=1&&iLastXB<=64)&&(iLastYB>
=1&& iLastYB<=480))
    {
    write(serialPort, "1", 1);
    }
    else if((iLastXB>=65&&
iLastXB<=130)&&(iLastYB>=1&&
iLastYB<=480))
    {
    write(serialPort, "2",1);
    }
    else
if((iLastXB>=131&&iLastXB<=195)&&(iLastYB
>=1&& iLastYB<=480))
    {
    write(serialPort, "3",1);
    }
    else
if((iLastXB>=196&&iLastXB<=260)&&(iLastYB
>=1&& iLastYB<=480))
    {
    write(serialPort, "4", 1);
    }
    else if((iLastXB>=261&&
iLastXB<=325)&&(iLastYB>=1&&
iLastYB<=480))
    {
    write(serialPort, "5",1);
    }
    else
if((iLastXB>=326&&iLastXB<=390)&&(iLastYB
>=1&& iLastYB<=480))
    {
    write(serialPort, "6",1);
    }
    else
if((iLastXB>=391&&iLastXB<=455)&&(iLastYB
>=1&& iLastYB<=480))
    {
    write(serialPort, "7", 1);
    }
```

```
    else if((iLastXB>=456&&
iLastXB<=520)&&(iLastYB>=1&&
iLastYB<=480))
    {
    write(serialPort, "8",1);
    }
    else
if((iLastXB>=521&&iLastXB<=640)&&(iLastYB
>=1&& iLastYB<=480))
    {
    write(serialPort, "9",1);
    }

 }
    else
    {
    //Bukan Kondisi 1,2, dan 3
    iLastXB = 0;
    iLastYB = 0;
    printf("posxB = %d posyB = %d\n", iLastXB,
iLastYB);
    write(serialPort, "A",1);
    }

    //Menampilkan gambar penanda hasil deteksi
pada gambar original
    imgOriginalB = imgOriginalB + hasildeteksi;

    //Tampilkan window threshold
    imshow("Gambar threshold",
imgThresholdedB);

    //Tampilka window gambar original
    imshow(namaWindoworiginal, imgOriginalB );
  //show the original image

     //Tampilkan window HSV
    imshow(namaWindowHSV, imgHSVB);

     if (waitKey(30) == 27)
      {
         cout << "tombol esc di tekan" << endl;
         break;
      }
    }

   return 0;
}

//*****************************USART******
***********************************//
void setPinMode(int pinID, int mode)
{
  writeFile(pinID, mode);
}

void writeFile(int fileID, int value)
{
  char buffer[4];  //atur buffer 4 byte string
  memset((void *)buffer, 0, sizeof(buffer)); // buffer
out diclear
  sprintf(buffer, "%d", value);
  lseek(fileID, 0, SEEK_SET);
  int res = write(fileID, buffer, sizeof(buffer));
}
```

LAMPIRAN 4

DATASHEET

1. Datasheet Mikrokontroler Utama STM32F401RE

life.augmented

UM1724
User manual
STM32 Nucleo-64 boards

Introduction

The STM32 Nucleo-64 board (NUCLEO-F030R8, NUCLEO-F070RB, NUCLEO-F072RB, NUCLEO-F091RC, NUCLEO-F103RB, NUCLEO-F302R8, NUCLEO-F303RE, NUCLEO-F334R8, NUCLEO-F401RE, NUCLEO-F410RB, NUCLEO-F411RE, NUCLEO-F446RE, NUCLEO-L053R8, NUCLEO-L073RZ, NUCLEO-L152RE, NUCLEO-L476RG) provides an affordable and flexible way for users to try out new concepts and build prototypes with the STM32 microcontrollers in LQFP64 package, choosing from the various combinations of performance, power consumption and features. The ARDUINO® connectivity support and ST morpho headers allow easy expansion of the functionality of the Nucleo open development platform with a wide choice of specialized shields. The STM32 Nucleo board does not require any separate probe as it integrates the ST-LINK/V2-1 debugger/programmer. The STM32 Nucleo board comes with the STM32 comprehensive software HAL library together with various packaged software examples, as well as direct access to mbed™ online resources at http://mbed.org/.

Figure 1. STM32 Nucleo-64 board (1)

1. Picture not contractual.

3 Conventions

Table 2 provides the conventions used for the ON and OFF settings in the present document.

Table 2. ON/OFF conventions

Convention	Definition
Jumper JP1 ON	Jumper fitted
Jumper JP1 OFF	Jumper not fitted
Solder bridge SBx ON	SBx connections closed by solder or 0 ohm resistor
Solder bridge SBx OFF	SBx connections left open

In this document the references are "STM32 Nucleo board" and "STM32 Nucleo boards" for all information that is common to all sale types.

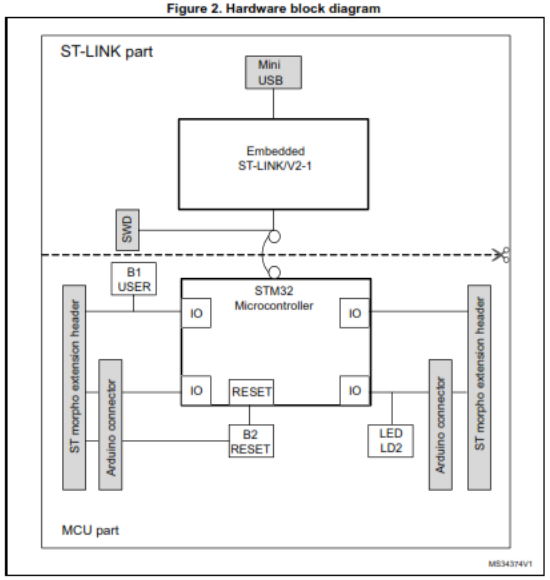
6 Hardware layout and configuration

The STM32 Nucleo board is designed around the STM32 microcontrollers in a 64-pin LQFP package.

Figure 2 shows the connections between the STM32 and its peripherals (ST-LINK/V2-1, pushbutton, LED, Arduino connectors and ST morpho connector).

Figure 3 and *Figure 4* show the location of these features on the STM32 Nucleo board.

Figure 2. Hardware block diagram



6.3 Power supply and power selection

The power supply is provided either by the host PC through the USB cable, or by an external Source: VIN (7V-12V), E5V (5V) or +3V3 power supply pins on CN6 or CN7. In case VIN, E5V or +3V3 is used to power the Nucleo board, using an external power supply unit or an auxiliary equipment, this power source must comply with the standard EN-60950-1: 2006+A11/2009, and must be Safety Extra Low Voltage (SELV) with limited power capability.

6.3.1 Power supply input from the USB connector

The ST-LINK/V2-1 supports USB power management allowing to request more than 100 mA current to the host PC.

All parts of the STM32 Nucleo board and shield can be powered from the ST-LINK USB connector CN1 (U5V or VBUS). Note that only the ST-LINK part is power supplied before the USB enumeration as the host PC only provides 100 mA to the board at that time. During the USB enumeration, the STM32 Nucleo board requires 300 mA of current to the host PC. If the host is able to provide the required power, the targeted STM32 microcontroller is powered and the red LED LD3 is turned ON, thus the STM32 Nucleo board and its shield can consume a maximum of 300 mA current, not more. If the host is not able to provide the required current, the targeted STM32 microcontroller and the MCU part including the extension board are not power supplied. As a consequence the red LED LD3 remains turned OFF. In such case it is mandatory to use an external power supply as explained in the next *Section 6.3.2: External power supply inputs: VIN and E5V*.

When the board is power supplied by USB (U5V) a jumper must be connected between pin 1 and pin 2 of JP5 as shown in *Table 7*.

JP1 is configured according to the maximum current consumption of the board when powered by USB (U5V). JP1 jumper can be set in case the board is powered by USB and maximum current consumption on U5V does not exceed 100 mA (including an eventual extension board or Arduino shield). In such condition USB enumeration will always succeed since no more than 100mA is requested to the PC. Possible configurations of JP1 are summarized in *Table 5*.

Table 5. JP1 configuration table

Jumper state	Power supply	Allowed current
JP1 jumper OFF	USB power through CN1	300 mA max
JP1 jumper ON		100 mA max

Warning: **If the maximum current consumption of the NUCLEO and its extension boards exceeds 300 mA, it is mandatory to power the NUCLEO using an external power supply connected to E5V or VIN.**

Note: *In case the board is powered by an USB charger, there is no USB enumeration, so the led LD3 remains set to OFF permanently and the target MCU is not powered. In this specific case the jumper JP1 needs to be set to ON, to allow target MCU to be powered anyway.*

6.3.2 External power supply inputs: VIN and E5V

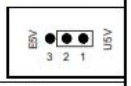
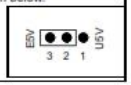
The external power sources VIN and E5V are summarized in the *Table 6*. When the board is power supplied by VIN or E5V, the jumpers configuration must be the following:

- Jumper on JP5 pin 2 and pin 3
- Jumper removed on JP1

Table 6. External power sources

Input power name	Connectors pins	Voltage range	Max current	Limitation
VIN	CN6 pin 8 CN7 pin 24	7 V to 12 V	800 mA	From 7 V to 12 V only and input current capability is linked to input voltage: 800 mA input current when Vin=7 V 450 mA input current when 7 V<Vin (< or =) 9 V 250 mA input current when 9 V<Vin (< or =) 12 V
E5V	CN7 pin 6	4.75 V to 5.25 V	500 mA	-

Table 7. Power-related jumper

Jumper	Description
JP5	U5V (ST-LINK VBUS) is used as power source when JP5 is set as shown below (Default setting)  VIN or E5V is used as power source when JP5 is set as shown below. 

Using VIN or E5V as external power supply

VIN or E5V can be used as external power supply in case the current consumption of NUCLEO and extensions boards exceeds the allowed current on USB. In this condition it is still possible to use the USB for communication, for programming or debugging only, but it is mandatory to power supply the board first using VIN or E5V then connect the USB cable to the PC. Proceeding this way ensures that the enumeration occurs thanks to the external power source.

The following power sequence procedure must be respected:

Figure 17. NUCLEO-F401RE

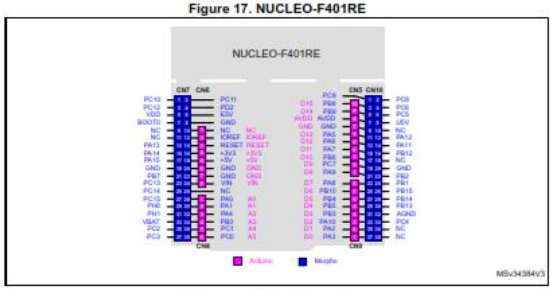


Figure 18. NUCLEO-F411RE

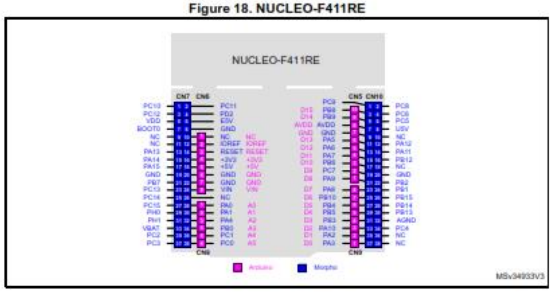


Table 14. Arduino connectors on NUCLEO-F334R8 (continued)

Connector	Pin	Pin name	MCU pin	Function
CN5 digital	2	D9	PC7	TIM3_CH2
	1	D8	PA9	-
CN9 digital	8	D7	PA8	-
	7	D6	PB10	TIM2_CH3
	6	D5	PB4	TIM3_CH1
	5	D4	PB5	-
	4	D3	PB3	TIM2_CH2
	3	D2	PA10	-
	2	D1	PA2	USART2_TX
	1	D0	PA3	USART2_RX

1. Refer to *Table 9: Solder bridges* for details.

Table 15. Arduino connectors on NUCLEO-F401RE and NUCLEO-F411RE

Connector	Pin	Pin name	MCU pin	Function
Left connectors				
CN6 power	1	NC	-	-
	2	IOREF	-	3.3V Ref
	3	RESET	NRST	RESET
	4	+3V3	-	3.3V input/output
	5	+5V	-	5V output
	6	GND	-	Ground
	7	GND	-	Ground
	8	VIN	-	Power input
CN8 analog	1	A0	PA0	ADC1_0
	2	A1	PA1	ADC1_1
	3	A2	PA4	ADC1_4
	4	A3	PB0	ADC1_8
	5	A4	PC1 or PB9(1)	ADC1_11 (PC1) or I2C1_SDA (PB9)
	6	A5	PC0 or PB8(1)	ADC1_10 (PC0) or I2C1_SCL (PB8)
Right connectors				
CN5 digital	10	D15	PB8	I2C1_SCL
	10	D15	PB8	I2C1_SCL
	9	D14	PB9	I2C1_SDA
	8	AREF	-	AVDD
	7	GND	-	Ground

LM2576xx Series SIMPLE SWITCHER® 3-A Step-Down Voltage Regulator

1 Features

- 3.3-V, 5-V, 12-V, 15-V, and Adjustable Output Versions
- Adjustable Version Output Voltage Range, 1.23 V to 37 V (57 V for HV Version) ±4% Maximum Over Line and Load Conditions
- Specified 3-A Output Current
- Wide Input Voltage Range: 40 V Up to 60 V for HV Version
- Requires Only 4 External Components
- 52-kHz Fixed-Frequency Internal Oscillator
- TTL-Shutdown Capability, Low-Power Standby Mode
- High Efficiency
- Uses Readily Available Standard Inductors
- Thermal Shutdown and Current Limit Protection

2 Applications

- Simple High-Efficiency Step-Down (Buck) Regulator
- Efficient Preregulator for Linear Regulators
- On-Card Switching Regulators
- Positive-to-Negative Converter (Buck-Boost)

3 Description

The LM2576 series of regulators are monolithic integrated circuits that provide all the active functions for a step-down (buck) switching regulator, capable of driving 3-A load with excellent line and load regulation. These devices are available in fixed output voltages of 3.3 V, 5 V, 12 V, 15 V, and an adjustable output version.

Requiring a minimum number of external components, these regulators are simple to use and include fault protection and a fixed-frequency oscillator.

The LM2576 series offers a high-efficiency replacement for popular three-terminal linear regulators. It substantially reduces the size of the heat sink, and in some cases no heat sink is required.

A standard series of inductors optimized for use with the LM2576 are available from several different manufacturers. This feature greatly simplifies the design of switch-mode power supplies.

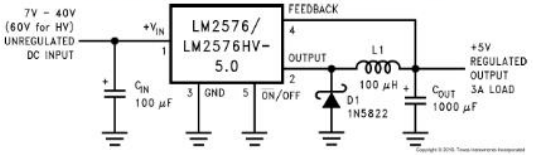
Other features include a ±4% tolerance on output voltage within specified input voltages and output load conditions, and ±10% on the oscillator frequency. External shutdown is included, featuring 50-µA (typical) standby current. The output switch includes cycle-by-cycle current limiting, as well as thermal shutdown for full protection under fault conditions.

Device Information[1]

PART NUMBER	PACKAGE	BODY SIZE (NOM)
LM2576	TO-220 (5)	10.16 mm × 8.51 mm
LM2576HV	DDPAK/TO-263 (5)	10.16 mm × 8.42 mm

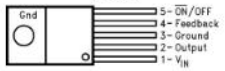
(1) For all available packages, see the orderable addendum at the end of the data sheet.

Fixed Output Voltage Version Typical Application Diagram

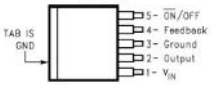


5 Pin Configuration and Functions

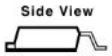
KC Package
5-Pin TO-220
Top View



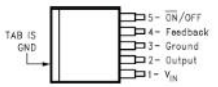
KTT Package
5-PIN DDPAK/TO-263
Top View



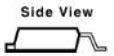
Side View



DDPAK/TO-263 (S) Package
5-Lead Surface-Mount Package
Top View



Side View



Pin Functions

PIN NO.	PIN NAME	I/O[1]	DESCRIPTION
1	V_{IN}	I	Supply input pin to collector pin of high-side transistor. Connect to power supply and input bypass capacitors CIN. Path from V_{IN} pin to high frequency bypass C_{IN} and GND must be as short as possible.
2	OUTPUT	O	Emitter pin of the power transistor. This is a switching node. Attached this pin to an inductor and the cathode of the external diode.
3	GROUND	—	Ground pin. Path to C_{IN} must be as short as possible.
4	FEEDBACK	I	Feedback sense input pin. Connect to the midpoint of feedback divider to set VOUT for ADJ version or connect this pin directly to the output capacitor for a fixed output version.
5	$\overline{ON}$/OFF	I	Enable input to the voltage regulator. High = OFF and low = ON. Connect to GND to enable the voltage regulator. Do not leave this pin float.
—	TAB	—	Connected to GND. Attached to heatsink for thermal relief for TO-220 package or put a copper plane connected to this pin as a thermal relief for DDPAK package.

(1) I = INPUT, O = OUTPUT

6 Specifications

6.1 Absolute Maximum Ratings

over the recommended operating junction temperature range of -40°C to 125°C (unless otherwise noted)[1][2]

		MIN	MAX	UNIT
Maximum supply voltage	LM2576		45	V
	LM2576HV		63	
$\overline{ON}$ /OFF pin input voltage		−0.3V ≤ V ≤ +V_{IN}		V
Output voltage to ground	(Steady-state)	−1		V
Power dissipation		Internally Limited		
Maximum junction temperature, T_J			150	°C
Storage temperature, T_{stg}		−65	150	°C

(1) Stresses beyond those listed under *Absolute Maximum Ratings* may cause permanent damage to the device. These are stress ratings only, which do not imply functional operation of the device at these or any other conditions beyond those indicated under *Recommended Operating Conditions*. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.
(2) If Military/Aerospace specified devices are required, please contact the TI Sales Office/ Distributors for availability and specifications.

6.2 ESD Ratings

			VALUE	UNIT
$V_{(ESD)}$	Electrostatic discharge	Human-body model (HBM), per ANSI/ESDA/JEDEC JS-001[1]	±2000	V

(1) JEDEC document JEP155 states that 500-V HBM allows safe manufacturing with a standard ESD control process.

6.3 Recommended Operating Conditions

over the recommended operating junction temperature range of -40°C to 125°C (unless otherwise noted)

		MIN	MAX	UNIT
Temperature	LM2576, LM2576HV	−40	125	°C
Supply voltage	LM2576		40	V
	LM2576HV		60	

6.4 Thermal Information

THERMAL METRIC[1][2][3]		LM2576, LM2576HV		UNIT
		KTT (TO-263)	KC (TO-220)	
		5 PINS	5 PINS	
$R_{\theta JA}$	Junction-to-ambient thermal resistance	42.6	32.4	°C/W
$R_{\theta JC(top)}$	Junction-to-case (top) thermal resistance	43.3	41.2	°C/W
$R_{\theta JB}$	Junction-to-board thermal resistance	22.4	17.6	°C/W
ψ_{JT}	Junction-to-top characterization parameter	10.7	7.8	°C/W
ψ_{JB}	Junction-to-board characterization parameter	21.3	17	°C/W
$R_{\theta JC(bot)}$	Junction-to-case (bottom) thermal resistance	0.4	0.4	°C/W

(1) For more information about traditional and new thermal metrics, see the *Semiconductor and IC Package Thermal Metrics* application report, SPRA953 and the *Using New Thermal Metrics* applications report, SBVA025.
(2) The package thermal impedance is calculated in accordance with JESD 51-7
(3) Thermal Resistances were simulated on a 4-layer, JEDEC board.

6.5 Electrical Characteristics: 3.3 V

Specifications are for T_J = 25°C (unless otherwise noted).

PARAMETER		TEST CONDITIONS		MIN	TYP	MAX	UNIT
SYSTEM PARAMETERS TEST CIRCUIT Figure 26 and Figure 32[1]							
V_{OUT}	Output Voltage	V_{IN} = 12 V, I_{LOAD} = 0.5 A Circuit of Figure 26 and Figure 32		3.234	3.3	3.366	V
	Output Voltage: LM2576	6 V ≤ V_{IN} ≤ 40 V, 0.5 A ≤ I_{LOAD} ≤ 3 A Circuit of Figure 26 and Figure 32	T_J = 25°C	3.168	3.3	3.432	V
			Applies over full operating temperature range	3.135		3.465	
	Output Voltage: LM2576HV	6 V ≤ V_{IN} ≤ 60 V, 0.5 A ≤ I_{LOAD} ≤ 3 A Circuit of Figure 26 and Figure 32	T_J = 25°C	3.168	3.3	3.45	V
			Applies over full operating temperature range	3.135		3.482	
η	Efficiency	V_{IN} = 12 V, I_{LOAD} = 3 A			75%		

(1) External components such as the catch diode, inductor, input and output capacitors can affect switching regulator system performance. When the LM2576/LM2576HV is used as shown in Figure 26 and Figure 32, system performance is as shown in *Electrical Characteristics: All Output Voltage Versions*.

6.6 Electrical Characteristics: 5 V

Specifications are for T_J = 25°C for the Figure 26 and Figure 32 (unless otherwise noted).

PARAMETER		TEST CONDITIONS		MIN	TYP	MAX	UNIT
SYSTEM PARAMETERS TEST CIRCUIT Figure 26 and Figure 32[1]							
V_{OUT}	Output Voltage	V_{IN} = 12 V, I_{LOAD} = 0.5 A Circuit of Figure 26 and Figure 32		4.9	5	5.1	V
V_{OUT}	Output Voltage LM2576	0.5 A ≤ I_{LOAD} ≤ 3 A, 8 V ≤ V_{IN} ≤ 40 V Circuit of Figure 26 and Figure 32	T_J = 25°C	4.8	5	5.2	V
			Applies over full operating temperature range	4.75		5.25	
V_{OUT}	Output Voltage LM2576HV	0.5 A ≤ I_{LOAD} ≤ 3 A, 8 V ≤ V_{IN} ≤ 60 V Circuit of Figure 26 and Figure 32	T_J = 25°C	4.8	5	4.75	V
			Applies over full operating temperature range	5.225		5.275	
η	Efficiency	V_{IN} = 12 V, I_{LOAD} = 3 A			77%		

(1) External components such as the catch diode, inductor, input and output capacitors can affect switching regulator system performance. When the LM2576/LM2576HV is used as shown in Figure 26 and Figure 32, system performance is as shown in *Electrical Characteristics: All Output Voltage Versions*.

6.7 Electrical Characteristics: 12 V

Specifications are for T_J = 25°C (unless otherwise noted).

PARAMETER		TEST CONDITIONS		MIN	TYP	MAX	UNIT
SYSTEM PARAMETERS TEST CIRCUIT Figure 26 and Figure 32[1]							
V_{OUT}	Output Voltage	V_{IN} = 25 V, I_{LOAD} = 0.5 A Circuit of Figure 26 and Figure 32		11.76	12	12.24	V
V_{OUT}	Output Voltage LM2576	0.5 A ≤ I_{LOAD} ≤ 3 A, 15 V ≤ V_{IN} ≤ 40 V Circuit of Figure 26 and Figure 32 and	T_J = 25°C	11.52	12	12.48	V
			Applies over full operating temperature range	11.4		12.6	
V_{OUT}	Output Voltage LM2576HV	0.5 A ≤ I_{LOAD} ≤ 3 A, 15 V ≤ V_{IN} ≤ 60 V Circuit of Figure 26 and Figure 32	T_J = 25°C	11.52	12	12.54	V
			Applies over full operating temperature range	11.4		12.66	
η	Efficiency	V_{IN} = 15 V, I_{LOAD} = 3 A			88%		

(1) External components such as the catch diode, inductor, input and output capacitors can affect switching regulator system performance. When the LM2576/LM2576HV is used as shown in Figure 26 and Figure 32, system performance is as shown in *Electrical Characteristics: All Output Voltage Versions*.

6.8 Electrical Characteristics: 15 V

over operating free-air temperature range (unless otherwise noted).

PARAMETER		TEST CONDITIONS		MIN	TYP	MAX	UNIT
SYSTEM PARAMETERS TEST CIRCUIT Figure 26 and Figure 32[1]							
V_{OUT}	Output Voltage	V_{IN} = 25 V, I_{LOAD} = 0.5 A Circuit of Figure 26 and Figure 32		14.7	15	15.3	V
V_{OUT}	Output Voltage LM2576	0.5 A ≤ I_{LOAD} ≤ 3 A, 18 V ≤ V_{IN} ≤ 40 V Circuit of Figure 26 and Figure 32	T_J = 25°C	14.4	15	15.6	V
			Applies over full operating temperature range	14.25		15.75	
V_{OUT}	Output Voltage LM2576HV	0.5 A ≤ I_{LOAD} ≤ 3 A, 18 V ≤ V_{IN} ≤ 60 V Circuit of Figure 26 and Figure 32	T_J = 25°C	14.4	15	14.25	V
			Applies over full operating temperature range	15.66		15.83	
η	Efficiency	V_{IN} = 15 V, I_{LOAD} = 3 A			88%		

(1) External components such as the catch diode, inductor, input and output capacitors can affect switching regulator system performance. When the LM2576/LM2576HV is used as shown in Figure 26 and Figure 32, system performance is as shown in *Electrical Characteristics: All Output Voltage Versions*.

6.9 Electrical Characteristics: Adjustable Output Voltage

over operating free-air temperature range (unless otherwise noted).

PARAMETER		TEST CONDITIONS		MIN	TYP	MAX	UNIT
SYSTEM PARAMETERS TEST CIRCUIT Figure 26 and Figure 32[1]							
V_{OUT}	Feedback voltage	V_{IN} = 12 V, I_{LOAD} = 0.5 A V_{OUT} = 5 V, Circuit of Figure 26 and Figure 32		1.217	1.23	1.243	V
V_{OUT}	Feedback Voltage LM2576	0.5 A ≤ I_{LOAD} ≤ 3 A, 8 V ≤ V_{IN} ≤ 40 V V_{OUT} = 5 V, Circuit of Figure 26 and Figure 32	T_J = 25°C	1.193	1.23	1.267	V
			Applies over full operating temperature range	1.18		1.26	
V_{OUT}	Feedback Voltage LM2576HV	0.5 A ≤ I_{LOAD} ≤ 3 A, 8 V ≤ V_{IN} ≤ 60 V V_{OUT} = 5 V, Circuit of Figure 26 and Figure 32	T_J = 25°C	1.193	1.23	1.273	V
			Applies over full operating temperature range	1.18		1.286	
η	Efficiency	V_{IN} = 12 V, I_{LOAD} = 3 A, V_{OUT} = 5 V			77%		

(1) External components such as the catch diode, inductor, input and output capacitors can affect switching regulator system performance. When the LM2576/LM2576HV is used as shown in Figure 26 and Figure 32, system performance is as shown in *Electrical Characteristics: All Output Voltage Versions*.

6.10 Electrical Characteristics: All Output Voltage Versions

over operating free-air temperature range (unless otherwise noted)

PARAMETER		TEST CONDITIONS		MIN	TYP[1]	MAX	UNIT
SYSTEM PARAMETERS TEST CIRCUIT Figure 26 and Figure 32[2]							
I_b	Feedback Bias Current	V_{OUT} = 5 V (Adjustable Version Only)	T_J = 25°C	100	50		nA
			Applies over full operating temperature range	500			
f_O	Oscillator Frequency[3]	T_J = 25°C		47	52	58	kHz
		Applies over full operating temperature range		42		63	

(1) All limits specified at room temperature (25°C) unless otherwise noted. All room temperature limits are 100% production tested. All limits at temperature extremes are specified through correlation using standard Statistical Quality Control (SQC) methods.
(2) External components such as the catch diode, inductor, input and output capacitors can affect switching regulator system performance. When the LM2576/LM2576HV is used as shown in Figure 26 and Figure 32, system performance is as shown in *Electrical Characteristics: All Output Voltage Versions*.
(3) The oscillator frequency reduces to approximately 11 kHz in the event of an output short or an overload which causes the regulated output voltage to drop approximately 40% from the nominal output voltage. This self protection feature lowers the average power dissipation of the IC by lowering the minimum duty cycle from 5% down to approximately 2%.

Electrical Characteristics: All Output Voltage Versions (continued)

over operating free-air temperature range (unless otherwise noted)

PARAMETER		TEST CONDITIONS		MIN	TYP[1]	MAX	UNIT
V_{SAT}	Saturation Voltage	I_{OUT} = 3 A [4]	T_J = 25°C		1.4	1.8	V
			Applies over full operating temperature range			2	
DC	Max Duty Cycle (ON)[5]			93%	95%		
I_{CL}	Current Limit[4][3]		T_J = 25°C	4.2	5.8	6.9	A
			Applies over full operating temperature range	3.5		7.5	
I_L	Output Leakage Current	Output = 0 V Output = −1 V Output = −1 V [6][7]		2	7.5	30	mA
I_Q	Quiescent Current[6]				5	10	mA
I_{STBY}	Standby Quiescent Current	$\overline{ON}$ /OFF Pin = 5 V (OFF)			50	200	µA
$\overline{ON}$ /OFF CONTROL TEST CIRCUIT Figure 26 and Figure 32							
V_{IH}	$\overline{ON}$ /OFF Pin Logic Input Level	V_{OUT} = 0 V	T_J = 25°C	2.2	1.4		V
			Applies over full operating temperature range	2.4			
V_{IL}		V_{OUT} = Nominal Output Voltage	T_J = 25°C		1.2	1	V
			Applies over full operating temperature range			0.8	
I_{IH}	$\overline{ON}$ /OFF Pin Input Current	$\overline{ON}$ /OFF Pin = 5 V (OFF)			12	30	µA
I_{IL}		$\overline{ON}$ /OFF Pin = 0 V (ON)			0	10	µA

(4) Output pin sourcing current. No diode, inductor or capacitor connected to output.
(5) Feedback pin removed from output and connected to 0V.
(6) Feedback pin removed from output and connected to +12 V for the Adjustable, 3.3-V, and 5-V versions, and +25 V for the 12-V and 15-V versions, to force the output transistor OFF.
(7) V_{IN} = 40 V (60 V for high voltage version).

L298

DUAL FULL-BRIDGE DRIVER

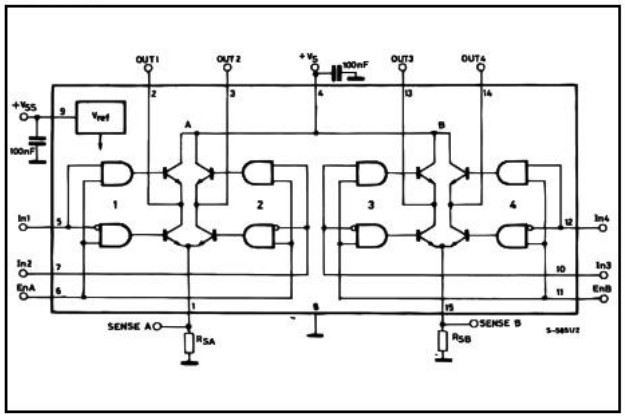
- OPERATING SUPPLY VOLTAGE UP TO 46 V
- TOTAL DC CURRENT UP TO 4 A
- LOW SATURATION VOLTAGE
- OVERTEMPERATURE PROTECTION
- LOGICAL "0" INPUT VOLTAGE UP TO 1.5 V (HIGH NOISE IMMUNITY)

ORDERING NUMBERS : L298N (Multiwatt Vert.)
L298HN (Multiwatt Horiz.)
L298P (PowerSO20)

DESCRIPTION

The L298 is an integrated monolithic circuit in a 15-lead Multiwatt and PowerSO20 packages. It is a high voltage, high current dual full-bridge driver designed to accept standard TTL logic levels and drive inductive loads such as relays, solenoids, DC and stepping motors. Two enable inputs are provided to enable or disable the device independently of the input signals. The emitters of the lower transistors of each bridge are connected together and the corresponding external terminal can be used for the connection of an external sensing resistor. An additional supply input is provided so that the logic works at a lower voltage.

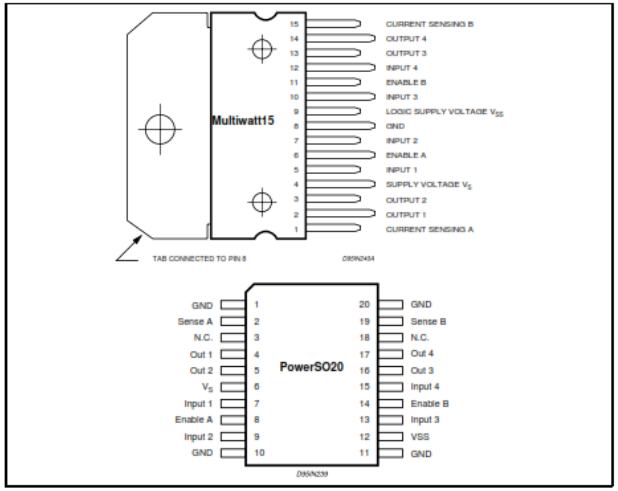
BLOCK DIAGRAM



ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_S	Power Supply	50	V
V_{SS}	Logic Supply Voltage	7	V
V_i, V_{en}	Input and Enable Voltage	–0.3 to 7	V
I_O	Peak Output Current (each Channel) – Non Repetitive (t = 100µs) –Repetitive (80% on –20% off; t_{on} = 10ms) –DC Operation	 3 2.5 2	 A A A
V_{sens}	Sensing Voltage	–1 to 2.3	V
P_{tot}	Total Power Dissipation (T_{case} = 75°C)	25	W
T_{op}	Junction Operating Temperature	–25 to 130	°C
T_{stg}, T_j	Storage and Junction Temperature	–40 to 150	°C

PIN CONNECTIONS (top view)



THERMAL DATA

Symbol	Parameter		PowerSO20	Multiwatt15	Unit
$R_{th\ j\text{-case}}$	Thermal Resistance Junction-case	Max.	–	3	°C/W
$R_{th\ j\text{-amb}}$	Thermal Resistance Junction-ambient	Max.	13 (*)	35	°C/W

(*) Mounted on aluminum substrate

ELECTRICAL CHARACTERISTICS (continued)

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
T_1 (V_i)	Source Current Turn-off Delay	0.5 V, to 0.9 I_L (2); (4)		1.5		µs
T_2 (V_i)	Source Current Fall Time	0.9 I_L to 0.1 I_L (2); (4)		0.2		µs
T_3 (V_i)	Source Current Turn-on Delay	0.5 V, to 0.1 I_L (2); (4)		2		µs
T_4 (V_i)	Source Current Rise Time	0.1 I_L to 0.9 I_L (2); (4)		0.7		µs
T_5 (V_i)	Sink Current Turn-off Delay	0.5 V, to 0.9 I_L (3); (4)		0.7		µs
T_6 (V_i)	Sink Current Fall Time	0.9 I_L to 0.1 I_L (3); (4)		0.25		µs
T_7 (V_i)	Sink Current Turn-on Delay	0.5 V, to 0.9 I_L (3); (4)		1.6		µs
T_8 (V_i)	Sink Current Rise Time	0.1 I_L to 0.9 I_L (3); (4)		0.2		µs
fc (V_i)	Commutation Frequency	I_L = 2A		25	40	KHz
T_1 (V_{en})	Source Current Turn-off Delay	0.5 V_{en} to 0.9 I_L (2); (4)		3		µs
T_2 (V_{en})	Source Current Fall Time	0.9 I_L to 0.1 I_L (2); (4)		1		µs
T_3 (V_{en})	Source Current Turn-on Delay	0.5 V_{en} to 0.1 I_L (2); (4)		0.3		µs
T_4 (V_{en})	Source Current Rise Time	0.1 I_L to 0.9 I_L (2); (4)		0.4		µs
T_5 (V_{en})	Sink Current Turn-off Delay	0.5 V_{en} to 0.9 I_L (3); (4)		2.2		µs
T_6 (V_{en})	Sink Current Fall Time	0.9 I_L to 0.1 I_L (3); (4)		0.35		µs
T_7 (V_{en})	Sink Current Turn-on Delay	0.5 V_{en} to 0.9 I_L (3); (4)		0.25		µs
T_8 (V_{en})	Sink Current Rise Time	0.1 I_L to 0.9 I_L (3); (4)		0.1		µs

1) 1)Sensing voltage can be –1 V for t ≤ 50 µsec; in steady state V_{sens} min ≥ – 0.5 V.
2) See fig. 2.
3) See fig. 4.
4) The load must be a pure resistor.

Figure 1 : Typical Saturation Voltage vs. Output Current.

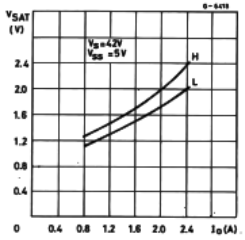
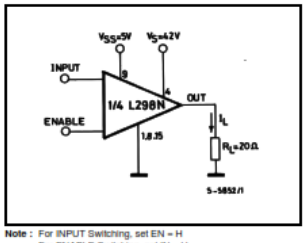


Figure 2 : Switching Times Test Circuits.



Note : For INPUT Switching, set EN = H
For ENABLE Switching, set IN = H

ELECTRICAL CHARACTERISTICS (V_S = 42V; V_{SS} = 5V, T_j = 25°C; unless otherwise specified)

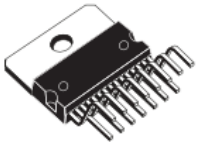
Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
V_S	Supply Voltage (pin 4)	Operative Condition	V_{IH} +2.5		46	V
V_{SS}	Logic Supply Voltage (pin 9)		4.5	5	7	V
I_S	Quiescent Supply Current (pin 4)	V_{en} = H; I_L = 0 V_i = L V_i = H		13 50	22 70	mA mA
		V_{en} = L V_i = X			4	mA
I_{SS}	Quiescent Current from V_{SS} (pin 9)	V_{en} = H; I_L = 0 V_i = L V_i = H		24 7	36 12	mA mA
		V_{en} = L V_i = X			6	mA
V_{IL}	Input Low Voltage (pins 5, 7, 10, 12)		–0.3		1.5	V
V_{IH}	Input High Voltage (pins 5, 7, 10, 12)		2.3		VSS	V
I_{IL}	Low Voltage Input Current (pins 5, 7, 10, 12)	V_i = L			–10	µA
I_{IH}	High Voltage Input Current (pins 5, 7, 10, 12)	V_i = H ≤ V_{SS} –0.6V		30	100	µA
V_{en} = L	Enable Low Voltage (pins 6, 11)		–0.3		1.5	V
V_{en} = H	Enable High Voltage (pins 6, 11)		2.3		V_{SS}	V
I_{en} = L	Low Voltage Enable Current (pins 6, 11)	V_{en} = L			–10	µA
I_{en} = H	High Voltage Enable Current (pins 6, 11)	V_{en} = H ≤ V_{SS} –0.6V		30	100	µA
V_{CEsat} (H)	Source Saturation Voltage	I_L = 1A I_L = 2A	0.95	1.35 2	1.7 2.7	V V
V_{CEsat} (L)	Sink Saturation Voltage	I_L = 1A (5) I_L = 2A (5)	0.85	1.2 1.7	1.6 2.3	V V
V_{CEsat}	Total Drop	I_L = 1A (5) I_L = 2A (5)	1.80		3.2 4.9	V V
V_{sens}	Sensing Voltage (pins 1, 15)		–1 (1)		2	V

PIN FUNCTIONS (refer to the block diagram)

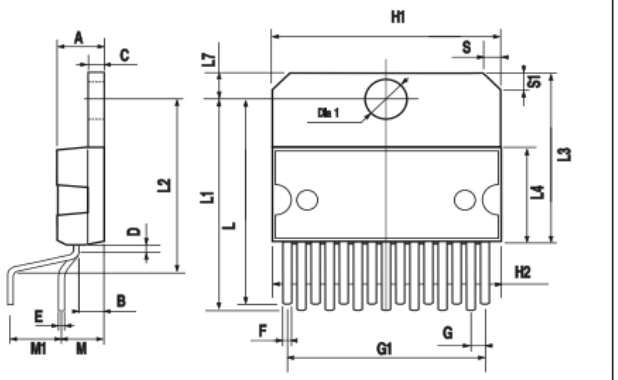
MW.15	PowerSO	Name	Function
1;15	2;19	Sense A; Sense B	Between this pin and ground is connected the sense resistor to control the current of the load.
2;3	4;5	Out 1; Out 2	Outputs of the Bridge A; the current that flows through the load connected between these two pins is monitored at pin 1.
4	6	V_S	Supply Voltage for the Power Output Stages. A non-inductive 100nF capacitor must be connected between this pin and ground.
5;7	7;9	Input 1; Input 2	TTL Compatible Inputs of the Bridge A.
6;11	8;14	Enable A; Enable B	TTL Compatible Enable Input: the L state disables the bridge A (enable A) and/or the bridge B (enable B).
8	1;10;11;20	GND	Ground.
9	12	VSS	Supply Voltage for the Logic Blocks. A100nF capacitor must be connected between this pin and ground.
10; 12	13;15	Input 3; Input 4	TTL Compatible Inputs of the Bridge B.
13; 14	16;17	Out 3; Out 4	Outputs of the Bridge B. The current that flows through the load connected between these two pins is monitored at pin 15.
–	3;18	N.C.	Not Connected

DIM.	mm			inch		
	MIN.	TYP.	MAX.	MIN.	TYP.	MAX.
A			5			0.197
B			2.65			0.104
C			1.6			0.063
D		1			0.039	
E	0.49		0.55	0.019		0.022
F	0.66		0.75	0.026		0.030
G	1.02	1.27	1.52	0.040	0.050	0.060
G1	17.53	17.78	18.03	0.690	0.700	0.710
H1	19.6			0.772		
H2			20.2			0.795
L	21.9	22.2	22.5	0.862	0.874	0.886
L1	21.7	22.1	22.5	0.854	0.870	0.886
L2	17.65		18.1	0.695		0.713
L3	17.25	17.5	17.75	0.679	0.689	0.699
L4	10.3	10.7	10.9	0.406	0.421	0.429
L7	2.65		2.9	0.104		0.114
M	4.25	4.55	4.85	0.167	0.179	0.191
M1	4.63	5.08	5.53	0.182	0.200	0.218
S	1.9		2.6	0.075		0.102
S1	1.9		2.6	0.075		0.102
Dia1	3.65		3.85	0.144		0.152

OUTLINE AND MECHANICAL DATA

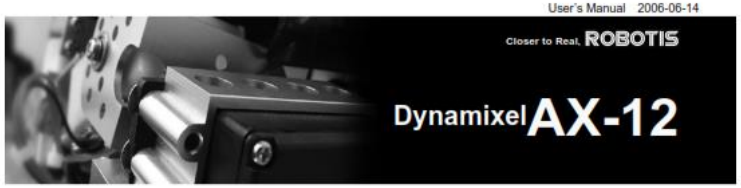


Multiwatt15 V



4. Datasheet *Servo Serial Dynamixel* AX-12

User's Manual 2006-06-14



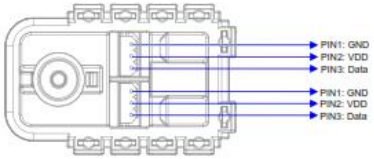
1-2. Main Specifications

	AX-12	
Weight (g)	55	
Gear Reduction Ratio	1/254	
Input Voltage (V)	at 7V	at 10V
Final Max Holding Torque(kgf.cm)	12	16.5
Sec/60degree	0.269	0.196

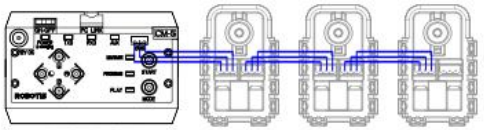
Resolution	0.35°
Operating Angle	300°, Endless Turn
Voltage	7V~10V (Recommended voltage: 9.6V)
Max. Current	900mA
Operate Temperature	-5℃ ~ +85℃
Command Signal	Digital Packet
Protocol Type	Half duplex Asynchronous Serial Communication (8bit,1stop,No Parity)
Link (Physical)	TTL Level Multi Drop (daisy chain type Connector)
ID	254 ID (0~253)
Communication Speed	7343bps ~ 1 Mbps
Feedback	Position, Temperature, Load, Input Voltage, etc.
Material	Engineering Plastic

2-3. Dynamixel Wiring

Pin Assignment The connector pin assignments are as the following. The two connectors on the Dynamixel are connected pin to pin, thus the AX-12 can be operated with only one connector attached.



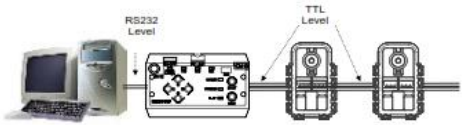
Wiring Connect the AX-2 actuators pin to pin as shown below. Many AX-12 actuators can be controlled with a single bus in this manner.



Control Box "CM-5"

Main Controller To operate the Dynamixel actuators, the main controller must support TTL level half duplex UART. A proprietary controller can be used, but the use of the Dynamixel controller CM-5 is recommended.

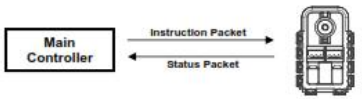
PC LINK A PC can be used to control the Dynamixel via the CM-5 controller.



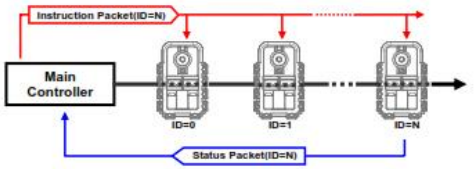
3. Communication Protocol

3-1. Communication Overview

Packet The main controller communicates with the Dynamixel units by sending and receiving data packets. There are two types of packets; the "Instruction Packet" (sent from the main controller to the Dynamixel actuators) and the "Status Packet" (sent from the Dynamixel actuators to the main controller.)



Communication For the system connection below, if the main controller sends an instruction packet with the ID set to N, only the Dynamixel unit with this ID value will return its respective status packet and perform the required instruction.



Unique ID If multiple Dynamixel units have the same ID value, multiple packets sent simultaneously collide, resulting in communication problems. Thus, it is imperative that no Dynamixel units share the same ID in a network node.

Protocol The Dynamixel actuators communicate through asynchronous serial communication with 8 bit, 1 stop bit and no parity.

3-2. Instruction Packet

The Instruction Packet is the packet sent by the main controller to the Dynamixel units to send commands. The structure of the Instruction Packet is as the following.

Instruction Packet | 0XFF | 0XFF | ID | LENGTH | INSTRUCTION | PARAMETER1 ... PARAMETER N | CHECK SUM |

The meanings of each packet byte definition are as the following.

0XFF 0XFF The two 0XFF bytes indicate the start of an incoming packet.

ID The unique ID of a Dynamixel unit. There are 254 available ID values, ranging from 0X00 to 0XFD.

Broadcasting ID ID 0XFE is the Broadcasting ID which indicates all of the connected Dynamixel units. Packets sent with this ID apply to all Dynamixel units on the network. Thus packets sent with a broadcasting ID will not return any status packets.

LENGTH The length of the packet where its value is "Number of parameters (N) + 2"

INSTRUCTION The instruction for the Dynamixel actuator to perform.

PARAMETER0...N Used if there is additional information needed to be sent other than the instruction itself.

CHECK SUM The computation method for the 'Check Sum' is as the following.

Check Sum = ~ (ID + Length + Instruction + Parameter1 + ... Parameter N)
If the calculated value is larger than 255, the lower byte is defined as the checksum value.
~ represents the NOT logic operation.

3-3. Status Packet(Return Packet)

The Status Packet is the response packet from the Dynamixel units to the Main Controller after receiving an instruction packet. The structure of the status packet is as the following.

| 0XFF | 0XFF | ID | LENGTH | ERROR | PARAMETER1 | PARAMETER2 ... PARAMETER N | CHECK SUM |

The meanings of each packet byte definition are as the following.

0XFF 0XFF The two 0XFF bytes indicate the start of the packet.

ID The unique ID of the Dynamixel unit returning the packet. The initial value is set to 1.

LENGTH The length of the packet where its value is "Number of parameters (N) + 2"

ERROR The byte representing errors sent from the Dynamixel unit. The meaning of each bit is as the following.

Bit	Name	Details
Bit 7	0	-
Bit 6	Instruction Error	Set to 1 if an undefined instruction is sent or an action instruction is sent without a Reg_Write instruction.
Bit 5	Overload Error	Set to 1 if the specified maximum torque can't control the applied load.
Bit 4	Checksum Error	Set to 1 if the checksum of the instruction packet is incorrect.
Bit 3	Range Error	Set to 1 if the instruction sent is out of the defined range.
Bit 2	Overheating Error	Set to 1 if the internal temperature of the Dynamixel unit is above the operating temperature range as defined in the control table.
Bit 1	Angle Limit Error	Set as 1 if the Goal Position is set outside of the range between CW Angle Limit and CCW Angle Limit.
Bit 0	Input Voltage Error	Set to 1 if the voltage is out of the operating voltage range as defined in the control table.

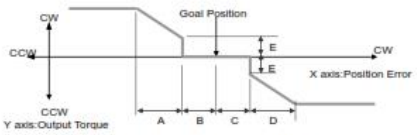
PARAMETER0...N Used if additional information is needed.

CHECK SUM The computation method for the 'Check Sum' is as the following.
Check Sum = ~ (ID + Length + Instruction + Parameter1 + ... Parameter N)
If the calculated value is larger than 255, the lower byte is defined as the checksum value. ~ represents the NOT logic operation.

Address 0x18 **Torque Enable**. When the power is first turned on, the Dynamixel actuator enters the Torque Free Run condition (zero torque). Setting the value in Address 0x18 to 1 enables the torque.

Address 0x19 **LED**. The LED turns on when set to 1 and turns off if set to 0.

Address 0x1A~0x1D **Compliance Margin and Slope**. The compliance of the Dynamixel actuator is defined by setting the compliance Margin and Slope. This feature can be utilized for absorbing shocks at the output shaft. The following graph shows how each compliance value (length of A, B, C & D) is defined by the Position Error and applied torque.



A : CCW Compliance Slope(Address0x1D)
B : CCW Compliance Margin(Address0x1B)
C : CW Compliance Margin(Address0x1A)
D : CW Compliance Slope (Address0x1C)
E : Punch(Address0x30,31)

Address 0X1E,0x1F **Goal Position** Requested angular position for the Dynamixel actuator output to move to. Setting this value to 0x3ff moves the output shaft to the position at 300°.