

ROBOT HUMANOID PEMAIN BOLA

SOCCER HUMANOID ROBOT

LAPORAN PENELITIAN

Disusun oleh :

Muliady ST., MT. / 220147

Drs. Zaenal Abidin, M. Eng / 220022

Christian Hadinata / 0822017

Mario Kusuma / 0822055



Jurusan Teknik Elektro Fakultas Teknik

Universitas Kristen Maranatha

Bandung

2012

LEMBAR IDENTITAS DAN PENGESAHAN LAPORAN PENELITIAN

1. a. Judul penelitian : Robot Humanoind Pemain Bola
b. Jenis penelitian : Engineering
2. Peneliti
Jumlah peneliti : 4 orang

Ketua peneliti
a) Nama lengkap dan gelar : Muliady, ST., MT.
b) Pangkat/Golongan/NIK : Lektor/IVA/220147
c) Fakultas/Jurusan : Teknik/Teknik Elektro
d) Pusat/Bidang Studi : Robotika

Anggota peneliti
a) Nama lengkap dan gelar : Drs. Zaenal Abidin. M.Eng
Pangkat/Golongan/NIK : AA/IVB/220022
b) Nama lengkap dan gelar : Christian Hadinata
Pangkat/Golongan/NRP : 0822017
c) Nama lengkap dan gelar : Mario Kusuma
Pangkat/Golongan/NRP : 0822055
3. Lokasi penelitian : Laboratorium Robotika, Teknik Elektro, UKM
5. Sumber dana penelitian : Universitas Kristen Maranatha
6. Biaya penelitian : Rp. 21.670.000
7. Lama Penelitian : Agustus 2011 sampai dengan Juli 2012

Menyetujui,
Dekan Fakultas teknik
UK Maranatha

Bandung, 26 Juli 2012
Ketua Peneliti,

Ir. Aan Darmawan, MT.

Muliady, ST., MT.

Mengetahui,
Ketua LPPM UK Maranatha,

Prof. Dr. Ir. Benjamin Soenarko, MSME.

ROBOT HUMANOID PEMAIN BOLA

Disusun oleh :

Muliady, Zainal Abidin, Christian Hadinata, Mario Kusuma

Jurusan Teknik Elektro, Fakultas Teknik, Universitas Kristen Maranatha,
Jl.Prof.Drg.Suria Sumantri, MPH No. 65, Bandung, Indonesia.

Email : mld_ang@yahoo.com

ABSTRAK

Penelitian di bidang robotika khususnya dalam robot humanoid semakin cepat berkembang. Robot humanoid pemain bola umumnya hanya memiliki 3 gerakan dasar dalam berjalan yaitu gerak lurus, gerakan samping dan gerakan memutar. Konsep *omnidirectional* mengkombinasikan ketiga gerakan dasar tersebut, posisi dan orientasi dari setiap langkah kaki dikontrol agar robot dapat mengubah arah dan jarak langkah berdasarkan letak target yang dituju.

Pada Penelitian ini, telah dibuat robot humanoid yang mempunyai 20 sendi dengan sensor CMUCam3 digunakan untuk sistem penglihatan robot. Sensor percepatan DE-ACCM3D digunakan untuk mengetahui kondisi robot ketika terjatuh. Otak dari robot menggunakan pengontrol mikro ATMEGA128 dan pengontrol servo SSC-32 untuk mengatur pergerakan servo.

Setiap gerakan pada robot humanoid diatur secara manual dengan memperhatikan posisi COG (*Center of Gravity*) yang harus selalu berada pada *support polygon*. Dengan gerakan *omnidirectional*, robot humanoid dapat mengatur arah dari setiap langkah sehingga pada saat akan menendang bola, letak bola dapat searah dengan letak gawang. Robot humanoid ini gagal menendang bola ke gawang ketika posisi bola dipindahkan ke area yang berbeda terhadap gawang.

Kata Kunci : Robot Humanoid, Sensor CMUCam3, Sensor Percepatan DE-ACCM3D, Pengontrol Mikro Atmega 128, pengontrol servo SSC-32.

SOCCER HUMANOID ROBOT

Composed by :

Muliady, Zainal Abidin, Christian Hadinata, Mario Kusuma

Electrical Engineering Department, Maranatha Christian University,

Jl.Prof.Drg.Suria Sumantri, MPH No. 65, Bandung, Indonesia.

Email : mld_ang@yahoo.com

ABSTRACT

Research in robotics, especially for humanoid robot has been increased rapidly. Soccer humanoid robot generally have three basic movements, which on straight motion, sideway motion, and rotate motion. Omnidirectional concepts combines the three basic movements, positions and rotations of each steps is controlled so that the robot can change the direction and distance based on the desire location.

On this research, has been made a humanoid robot which has 20 joint with sensor CMUCam3 used for robot vision system. Accelerometers are used to determine the condition of the robot when it falls. The main controller of the robot is microcontroller ATMEGA128 and use servo controller SSC-32 to manage the servo movements.

Every movement of the humanoid robot is set manually by observing the position of the COG (Center of Gravity) which must always be on the support polygon. With this omnidirectional movement, humanoid robot can adjust the direction of each step so that when the robot kick the ball, the position of the ball can be in the direction of the goal location. The robot fail to kick the ball to the goal if the ball position moved to the different area of the goal.

Keywords: Humanoid Robot, Camera Sensor CMUCam3, Accelerometer DE-ACCM3D, Micro Controller Atmega 128, Servo Controller SSC-32.

KATA PENGANTAR

Segala puji syukur kami panjatkan ke hadirat Tuhan Yang Maha Esa atas segala rahmat dan karunia-Nya sehingga penulis mampu menyelesaikan penelitian ini dengan baik dan tepat pada waktunya di Laboratorium Robotika. Laporan Penelitian yang berjudul “ROBOT HUMANOID PEMAIN BOLA” ini disusun untuk menjawab tantangan teknologi robot humanoid yang semakin berkembang.

Pada pelaksanaan Penelitian ini, penulis banyak mendapat bantuan dari berbagai pihak. Pada kesempatan ini, penulis menyampaikan terima kasih kepada:

1. Bapak Prof. Dr. Ir. Benjamin Soenarko, selaku Ketua LPPM Universitas Kristen Maranatha
2. Bapak Ir. Aan Darmawan, MT., selaku Dekan Fakultas Teknik Universitas Kristen Maranatha
3. Bapak Dr. Ir. Daniel Setiadikarunia, MT., selaku Ketua Jurusan Teknik Elektro Universitas Kristen Maranatha.
4. Bapak Ir. Yusak Gunadi S., MM., selaku Ketua LPPM Universitas Kristen Maranatha periode 2008-2012
5. Keluarga tercinta yang telah memberikan perhatian, semangat, serta bantuan doa dalam pelaksanaan penelitian sehingga dapat diselesaikan dengan baik.
6. Seluruh Tim robot KRCI dan KRSI yang telah memberi saran dan masukannya.
7. Seluruh karyawan dan civitas akademika Universitas Kristen Maranatha yang telah membantu dalam menyelesaikan penelitian ini.

Penulis menyadari sepenuhnya bahwa masih banyak kekurangan dan kesalahan dalam penelitian ini, walaupun penulis telah berusaha sebaik mungkin dengan segala kemampuan yang ada. Oleh karena itu, dengan segala kerendahan hati, penulis mengharapkan saran dan kritik yang membangun yang dapat menyempurnakan penelitian ini.

Bandung, 25 Juni 2012

(Muliady, ST., MT.)

DAFTAR ISI

	Halaman
ABSTRAK	i
ABSTRACT	ii
KATA PENGANTAR DAN UCAPAN TERIMA KASIH	iii
DAFTAR ISI	iv
DAFTAR TABEL	vi
DAFTAR GAMBAR	viii
DAFTAR LAMPIRAN	xi
BAB I PENDAHULUAN	
I.1 Latar Belakang Masalah	1
I.2 Identifikasi Masalah	1
I.3 Pembatasan Masalah	2
I.4 Perumusan Masalah	2
I.5 Tujuan dan Manfaat Penelitian	2
I.6 Sistematika Penulisan	3
I.7 Metodologi Penelitian	4
BAB II TINJAUAN PUSTAKA	
II.1 Teori Robot Humanoid	5
II.2 Gerakan <i>Omnidirectional</i>	6
II.3 Pengontrol Servo	10
II.3.1 Komponen – komponen SSC-32.....	10
II.3.2 Cara Komunikasi SSC-32 dengan ATMEGA128.....	12
II.3.3 Spesifikasi SSC-32.....	12
II.4 Sensor Percepatan DE-ACCM3D	13
II.5 Sensor Kamera CMUCam3	14
II.5.1 Arsitektur Perangkat Keras.....	14
II.5.1.1 Koneksi Perangkat Keras.....	17
II.5.2 Arsitektur Perangkat Lunak.....	18
II.5.3 Metoda Pelacakan Warna CMUCam3.....	20

BAB III PERANCANGAN DAN REALISASI	
III.1 Perancangan Sistem Robot Humanoid Pemain Bola.....	22
III.2 Realisasi Sistem Robot Humanoid Pemain Bola	24
III.2.1 Sistem Mekanika Robot Humanoid Pemain Bola.....	24
III.2.2 Sistem Elektronika Robot Humanoid Pemain Bola.....	26
III.2.2.1. Sensor.....	26
III.2.2.1.1. Sensor Percepatan DE-ACCM3D.....	26
III.2.2.1.2. Sensor Kamera CMUCam3.....	27
III.2.2.2. Pengontrol.....	27
III.2.2.1.1. Pengontrol Servo SSC-32.....	27
III.2.2.1.2. Pengontrol Mikro ATMEGA128.....	29
III.2.3 Algoritma Pemrograman Robot Humanoid Pemain Bola.....	30
BAB IV DATA PENGAMATAN DAN ANALISIS	
IV.1. Sensor Kamera CMUCam3.....	49
IV.1.1. Pengujian Sensor dalam Melacak Nilai RGB Tertentu....	49
IV.1.2. Pengujian Posisi Servo Leher Terhadap Posisi Bola.....	52
IV.1.3. Pengujian Posisi Servo Leher Terhadap Posisi Gawang...	58
IV.1.4. Pengujian Pelacakan pada Bola yang Bergerak.....	60
IV.2. Robot Humanoid.....	67
IV.2.1. Pengamatan Terhadap Gerakan Robot.....	67
IV.2.2. Pengujian Gerakan Berjalan pada Robot.....	74
IV.2.3. Gerakan Omnidirectional.....	77
BAB V KESIMPULAN DAN SARAN	
V.1. Kesimpulan.....	85
V.2. Saran.....	86
DAFTAR PUSTAKA.....	87
LAMPIRAN	

DAFTAR TABEL

		Halaman
Tabel 2.1	Keterangan Koordinat – koordinat Frame.....	8
Tabel 3.1	Hubungan Port – port SSC-32 dengan Sendi pada Robot	28
Tabel 3.2a	Pergerakan Servo pada saat Gerakan Berjalan(1).....	38
Tabel 3.2b	Pergerakan Servo pada saat Gerakan Berjalan(2).....	39
Tabel 4.1	Rentang Nilai yang Diprogram pada CMUCam3 untuk Dapat Melacak Warna Bola.....	49
Tabel 4.2	Rentang Nilai yang Diprogram pada CMUCam3 untuk Dapat Melacak Warna Gawang.....	50
Tabel 4.3	Hasil Pengujian Keberhasilan Sensor Terhadap Rentang Nilai RGB yang Diprogram untuk Dapat Melacak Warna Bola.....	50
Tabel 4.4	Hasil Pengujian Keberhasilan Sensor Terhadap Rentang Nilai RGB yang Diprogram untuk Dapat Melacak Warna Gawang.....	51
Tabel 4.5a	Hasil Pengujian Servo Leher Terhadap Posisi Bola di Sumbu $y = 20$ cm.....	53
Tabel 4.5b	Hasil Pengujian Servo Leher Terhadap Posisi Bola di Sumbu $y = 40$ cm.....	53
Tabel 4.5c	Hasil Pengujian Servo Leher Terhadap Posisi Bola di Sumbu $y = 60$ cm.....	54
Tabel 4.5d	Hasil Pengujian Servo Leher Terhadap Posisi Bola di Sumbu $y = 80$ cm.....	55
Tabel 4.5e	Hasil Pengujian Servo Leher Terhadap Posisi Bola di Sumbu $y = 100$ cm.....	56
Tabel 4.5f	Hasil Pengujian Servo Leher Terhadap Posisi Bola di Sumbu $y = 120$ cm.....	57
Tabel 4.6	Hasil Pengujian Posisi Gawang Terhadap Posisi Robot.....	58
Tabel 4.7	Gerakan Berdiri.....	68
Tabel 4.8	Gerakan Persiapan Berjalan.....	68
Tabel 4.9	Gerakan Berjalan.....	68
Tabel 4.10	Gerakan Menendang.....	70
Tabel 4.11	Gerakan Bangkit Berdiri.....	73

Tabel 4.12	Data Kelancaran Robot pada saat Berjalan Lurus.....	75
Tabel 4.13	Data Kelancaran Robot pada saat Berjalan Arah Kanan	76
Tabel 4.14	Data Kelancaran Robot pada saat Berjalan Arah Kiri	77
Tabel 4.15	Nilai Posisi Servo pada Saat robot Melakukan Arah Gerak ke Kiri	84

DAFTAR GAMBAR

	Halaman
Gambar 2.1 <i>Support Polygon</i>	6
Gambar 2.2 <i>Support Polygon</i> dengan Warna Abu – abu : (a) <i>Double Support Polygon</i> , (b) <i>Double Support Polygon (Pre-Swing)</i> , (c) <i>Single Support Polygon</i>	6
Gambar 2.3 Pola Langkah Kaki dari Gerakan <i>Omnidirectional</i>	7
Gambar 2.4 Hubungan Koordinat – koordinat Frame pada Teknik <i>Preview Controller</i>	8
Gambar 2.5 Koordinat – koordinat Frame pada saat Melangkah.....	9
Gambar 2.6 Pergerakan Sudut Langkah Kaki pada saat Tertumpu pada <i>Single Support Polygon</i>	10
Gambar 2.7 Bentuk dan Letak Komponen SSC-32	10
Gambar 2.8 Pin SSC-32 untuk Komunikasi dengan ATMEGA128	12
Gambar 2.9 Sensor Percepatan DE-ACCM3D	13
Gambar 2.10 Nilai Tegangan Tiap Axis pada Posisi – posisi Tertentu ..	14
Gambar 2.11 CMUCam3	14
Gambar 2.12 Arsitektur Perangkat Keras CMUCam3.....	16
Gambar 2.13 Koneksi Perangkat Keras CMUCam3.....	17
Gambar 3.1 Struktur Robot Humanoid Pemain Bola.....	23
Gambar 3.2 Diagram Blok Sistem Elektronika Robot Humanoid Pemain Bola	23
Gambar 3.3 Diagram Blok Sistem Kontrol	24
Gambar 3.4 Dimensi Robot pada saat Berdiri.....	25
Gambar 3.5 Sistem Gerak dan Peletakan Servo pada Robot	26
Gambar 3.6 Skematik Rangkaian Pengontrol Mikro ATMEGA128	29
Gambar 3.7 Papan Rangkaian Khusus untuk ATMEGA128.....	30
Gambar 3.8 Arah <i>Scanning</i> pada Kamera	31
Gambar 3.9a Diagram Alir Program Utama Kamera CMUCam3.....	32
Gambar 3.9b Diagram Alir Subrutin <i>Scanning</i>	33
Gambar 3.9c Diagram Alir Subrutin Cari Titik Tengah Objek	34

Gambar 3.9d Diagram Alir Subrutin Simpan Posisi Terakhir	35
Gambar 3.10 Arah Gerak Robot pada saat Bola di Sebelah Kiri Gawang	36
Gambar 3.11 Arah Gerak Robot pada saat Bola di Sebelah Kanan Gawang	36
Gambar 3.12a Diagram Alir Program Utama Pengontrol Mikro ATMEGA128	41
Gambar 3.12b Diagram Alir Subrutin Ambil Data Kamera	42
Gambar 3.12c Diagram Alir Subrutin Menentukan Arah Gerak	43
Gambar 3.12d Diagram Alir Subrutin Menentukan Arah jalan untuk Menyearahkan	43
Gambar 3.12e Diagram Alir Subrutin Menentukan Arah Jalan untuk Mendekati Bola.....	44
Gambar 3.12f Diagram Alir Subrutin Berjalan	45
Gambar 3.12g Diagram Alir Subrutin Persiapan Berjalan.....	46
Gambar 3.11h Diagram Alir Subrutin Berdiri.....	46
Gambar 3.12i Diagram Alir Subrutin Menendang	47
Gambar 3.12j Diagram Alir Subrutin Bangkit Berdiri.....	48
Gambar 4.1 Kemungkinan Posisi Gawang yang Dapat Dilacak Kamera Ketika Robot Berada pada Koordinat -60cm sampai 60cm.....	59
Gambar 4.2a Grafik Pelacakan Pergeseran Bola dengan Kecepatan 2 cm/det di Sumbu x.....	60
Gambar 4.2b Grafik Pelacakan Pergeseran Bola dengan Kecepatan 6 cm/det di Sumbu x.....	61
Gambar 4.2c Grafik Pelacakan Pergeseran Bola dengan Kecepatan 10 cm/det di Sumbu x.....	62
Gambar 4.2d Grafik Pelacakan Pergeseran Bola dengan Kecepatan 20 cm/det di Sumbu x.....	63
Gambar 4.2e Grafik Pelacakan Pergeseran Bola dengan Kecepatan 2 cm/det di Sumbu y.....	64
Gambar 4.2f Grafik Pelacakan Pergeseran Bola dengan Kecepatan 6 cm/det di Sumbu y.....	65

Gambar 4.2g	Grafik Pelacakan Pergeseran Bola dengan Kecepatan 10 cm/det di Sumbu y.....	66
Gambar 4.2h	Grafik Pelacakan Pergeseran Bola dengan Kecepatan 20 cm/det di Sumbu y.....	67
Gambar 4.3	Lintasan Robot Menuju Bola yang Berada Searah Gawang	78
Gambar 4.4	Lintasan Robot Menuju Bola yang Berada di Sebelah Kiri Gawang	79
Gambar 4.5	Lintasan Robot Menuju Bola yang Berada di Sebelah Kanan Gawang.....	80
Gambar 4.6	Lintasan Robot Terhadap Pemindahan Bola dari Sebelah Kiri Gawang Menuju ke Kiri	81
Gambar 4.7	Lintasan Robot Terhadap Pemindahan Bola dari Sebelah Kanan Gawang Menuju ke Kanan	82
Gambar 4.8	Lintasan Robot Terhadap Pemindahan Bola dari Sebelah Kanan Gawang Menuju ke Kiri	82
Gambar 4.9	Lintasan Robot Terhadap Pemindahan Bola dari Sebelah Kiri Gawang Menuju ke Kanan	83

DAFTAR LAMPIRAN

	Halaman
LAMPIRAN A Foto Robot Humanoid Pemain Bola.....	A-1
LAMPIRAN B Program Pengontrol Mikro ATmega128 dan CMUCam3...	B-1
LAMPIRAN C Data Sheet	C-1

BAB I

PENDAHULUAN

Bab ini membahas tentang latar belakang masalah, perumusan masalah, identifikasi masalah, tujuan, pembatasan masalah, serta sistematika penulisan laporan Penelitian.

I.1 Latar Belakang Masalah

Sekarang ini, penelitian di bidang robotika khususnya dalam robot humanoid semakin cepat berkembang. Teknik kontrol dan metoda - metoda banyak dilakukan untuk dapat menyempurnakan kemampuan robot humanoid. Kebanyakan penelitian sekarang ini fokus kepada teknik berjalan untuk dapat menyerupai pergerakan manusia sehingga membutuhkan konsentrasi pada kestabilan dan cara melangkah. Namun, hal tersebut tidak cocok untuk pertandingan sepak bola.

Dalam pertandingan sepak bola, robot humanoid diharapkan dapat berjalan lebih cepat menuju bola, mengganti arah jalan, bangkit berdiri bila terjatuh dan dapat menendang bola ke gawang lawan. Robot pemain bola juga diharapkan memiliki kecerdasan lebih dalam mendeteksi letak bola dan arah gawang. Robot humanoid pemain bola umumnya hanya memiliki 3 gerakan dasar dalam berjalan yaitu gerak lurus, gerakan samping dan gerakan memutar tetapi tidak dapat mengkombinasikan ketiga gerakan tersebut.

Konsep *omnidirectional* mengkombinasikan ketiga gerakan dasar tersebut, posisi dan orientasi dari setiap langkah kaki dikontrol agar robot dapat mengubah arah dan jarak langkah berdasarkan letak target yang dituju sehingga robot dapat lebih cepat mencapai bola atau posisi yang diinginkan.

I.2 Identifikasi Masalah

Terdapat kebutuhan robot humanoid yang dapat mendeteksi bola, berjalan menuju posisi yang diinginkan dengan konsep *omnidirectional* dan menendang bola ke gawang lawan serta bangkit berdiri ketika terjatuh.

I.3 Pembatasan Masalah

Pada Penelitian ini pembatasan masalah mengacu pada aturan perlombaan robot *humanoid soccer* kategori *Kidsize* yang meliputi :

1. Lapangan permainan terbuat dari kayu multipleks yang dilapisi karpet *polyester* halus berwarna hijau tua polos
2. Garis-garis pembatas lapangan berukuran 3 sampai dengan 5 cm yang dibuat dengan cara dicat putih
3. Gawang terbuat dari pipa silinder dengan diameter 10 cm yang diberi warna kuning. Gawang berukuran tinggi 80 cm dan lebar 150 cm. Latar belakang gawang berwarna hitam atau abu-abu
4. Di atas lapangan terdapat lampu-lampu berwarna putih lembut yang cahayanya merata ke seluruh permukaan lapangan dengan intensitas cahaya antara 300 sampai 400 lux
5. Bola yang digunakan adalah bola tenis dengan warna oranye
6. Robot *humanoid* harus menyerupai manusia yang memiliki sistem tangan, sistem kaki, sistem kepala dan dapat berdiri dengan kedua telapak kaki
7. Robot *humanoid* memiliki tinggi (H) antara 30 cm sampai dengan 60 cm
8. Telapak kaki robot berupa persegi panjang dengan luas tidak lebih dari $H^2/28$
9. Ukuran rentang tangan tidak melebihi $1,2 \times H$
10. Ukuran panjang kaki (H_{leg}) di antara $0,35 \times H$ sampai dengan $0,7 \times H$ diukur dari lantai
11. Ukuran tinggi kepala (H_{head}) di antara $0,05 \times H$ sampai dengan $0,25 \times H$ diukur dari pangkal leher

I.4 Perumusan Masalah

Perumusan yang akan dibahas pada Penelitian ini adalah:

1. Bagaimana merealisasikan robot *humanoid* yang dapat mendeteksi bola dan gawang lawan?
2. Bagaimana merealisasikan robot *humanoid* yang dapat berjalan menuju bola atau posisi yang diinginkan dengan konsep *omnidirectional*?

3. Bagaimana merealisasikan robot humanoid yang dapat menendang bola ke gawang lawan?
4. Bagaimana merealisasikan robot humanoid yang dapat bangkit berdiri ketika terjatuh?

I.5 Tujuan dan Manfaat Penelitian

Penelitian ini bertujuan untuk merealisasikan robot humanoid yang dapat mendeteksi bola, berjalan menuju posisi yang diinginkan dengan konsep *omnidirectional* dan menendang bola ke gawang lawan serta dapat bangkit berdiri ketika terjatuh. Hasil Penelitian bermanfaat untuk menunjang keberhasilan dalam merealisasikan robot jenis humanoid di masa depan, sehingga dapat membantu kehidupan manusia.

I.6 Sistematika Penulisan

Sistematika penulisan untuk Penelitian ini adalah sebagai berikut:

BAB I. PENDAHULUAN

Bab ini berisi tentang latar belakang masalah, identifikasi masalah, perumusan masalah, tujuan, pembatasan masalah, dan sistematika penulisan laporan Penelitian.

BAB II. TINJAUAN PUSTAKA

Pada bab ini dijelaskan teori-teori penunjang yang diperlukan dalam merancang dan merealisasikan robot humanoid yaitu berupa teori tentang robot humanoid, gerakan *omnidirectional*, pengontrol servo, sensor percepatan, dan sensor kamera.

BAB III. PERANCANGAN DAN REALISASI

Pada bab ini dijelaskan tentang perancangan dan realisasi sistem robot humanoid pemain bola, perancangan dan realisasi rangkaian sensor dan pengontrol, serta algoritma pemrograman robot humanoid pemain bola.

BAB IV. DATA PENGAMATAN DAN ANALISIS DATA

Pada bab ini ditampilkan data-data hasil pengamatan kinerja robot, pengujian pergerakan robot, pengujian kemampuan robot untuk dapat menendang bola ke gawang lawan, pengujian kemampuan robot yang dapat berdiri ketika terjatuh, pengujian sensor kamera dalam mendeteksi warna.

BAB V. KESIMPULAN DAN SARAN

Bab ini berisi tentang simpulan-simpulan yang didapat dari keseluruhan perancangan dan realisasi robot humanoid pemain bola dari awal sampai akhir. Lalu bab ini juga berisi saran yang diberikan untuk penelitian lebih lanjut oleh pihak lain.

I.7 Metodologi Penelitian

Penelitian dilakukan dengan melakukan studi literatur terhadap berbagai konstruksi robot humanoid yang pernah ada, manuver yang dapat dilakukan, pendeteksian bola dengan menggunakan kamera. Pada tahap berikutnya lalu dirancang robot humanoid sesuai dengan spesifikasi untuk lomba KRCI Humanoid Robot Soccer League. Robot yang telah dibuat lalu diuji coba dengan berbagai kondisi yang mungkin untuk melihat performansi robot.

BAB II

TINJAUAN PUSTAKA

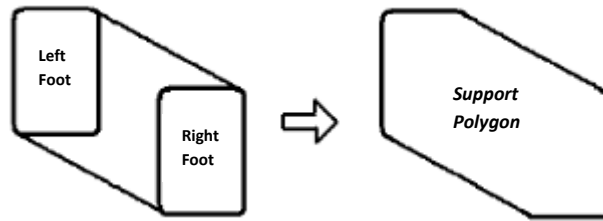
Pada bab ini dijelaskan teori-teori penunjang yang diperlukan dalam merancang dan merealisasikan robot humanoid yaitu berupa teori tentang robot humanoid, gerakan *omnidirectional*, pengontrol mikro, pengontrol servo, sensor percepatan, dan kamera CMUcam3.

II.1 Teori Robot Humanoid^{[1][2]}

Sebelum merancang dan merealisasikan robot humanoid, pada sub bab ini akan dibahas teori dasar mengenai robot humanoid. Dilihat dari unsur pembentuk katanya, robot humanoid terdiri dari kata robot dan humanoid. Definisi robot adalah perangkat cerdas mekanik atau virtual yang dapat melakukan tugas secara otomatis atau dengan bimbingan. Sedangkan definisi humanoid adalah segala sesuatu yang memiliki struktur menyerupai manusia. Maka robot humanoid dapat didefinisikan sebagai sebuah perangkat cerdas mekanik atau virtual yang memiliki bentuk dan sejumlah karakteristik menyerupai manusia baik secara keseluruhan struktur maupun pergerakan yang dapat melakukan tugas secara otomatis atau dengan bimbingan.

Faktor penting dalam merancang robot humanoid adalah faktor keseimbangan. Secara sederhana kestabilan dapat dicapai dengan menyeimbangkan (membuat jadi nol) semua gaya - gaya yang bekerja. Titik pada posisi jumlah semua gaya - gaya yang bekerja menjadi nol disebut titik keseimbangan atau *center of gravity*. Keseimbangan dicapai dengan merancang postur stabil dari setiap gerakan robot humanoid. Kestabilan robot paling banyak dipengaruhi oleh bagian kaki. Salah satu teknik yang baik untuk membuat robot seimbang ketika berjalan adalah teknik *support polygon*. *Support polygon* adalah daerah berbentuk segi banyak yang merupakan daerah di antara kedua kaki dengan bantuan garis lurus yang ditarik dari siku luar masing-masing kaki. Prinsip dari teknik ini adalah menempatkan proyeksi vertikal dari titik keseimbangan dari robot

humanoid untuk selalu berada di dalam *support polygon* ditunjukkan pada Gambar 2.1.



Gambar 2.1 *Support Polygon*

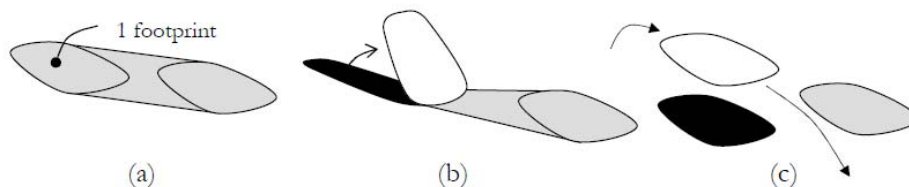
Terdapat 2 macam *support polygon* yang dapat terbentuk pada robot humanoid, yaitu :

- *Double Support Polygon*

Double Support Polygon adalah kondisi pada saat robot bertumpu pada kedua kaki nya tetapi tidak harus kedua permukaan kaki nya menempel penuh pada dasar. Gambaran *Double Support Polygon* ini dapat dilihat pada Gambar 2.2a dan 2.2b.

- *Single Support Polygon*

Single Support Polygon adalah kondisi pada saat robot hanya bertumpu pada salah satu telapak kaki seperti pada Gambar 2.2(c).

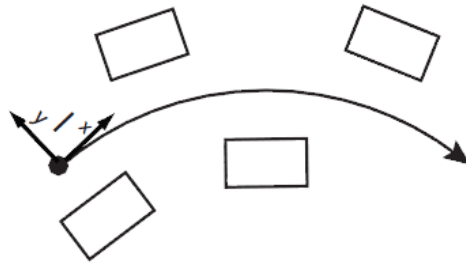


Gambar 2.2 *Support Polygon dengan Warna Abu-abu* : (a) *Double Support Polygon*, (b) *Double Support Polygon (Pre-Swing)*, (c) *Single Support Polygon*

II.2 Gerakan *Omnidirectional*^{[3][4]}

Gerakan *omnidirectional* pada pemain bola sangatlah penting karena pemain bola selalu mengejar letak bola yang senantiasa berubah posisi. Pola gerakan yang pada umumnya terdapat pada robot humanoid pemain bola adalah gerakan berjalan lurus, bergerak ke samping dan memutar tetapi tidak dapat mengkombinasikan ketiga gerakan tersebut.

Gerakan *omnidirectional* adalah bentuk gerakan ke segala arah yang mengkombinasikan ketiga gerakan tersebut. Gerakan *omnidirectional* dalam robot humanoid ini ada pada gerakan kaki sehingga robot dapat menentukan arah dan jarak langkah ketika berjalan. Pola langkah kaki dari gerakan *omnidirectional* ini dapat dilihat pada Gambar 2.3.



Gambar 2.3 Pola Langkah Kaki dari Gerakan *Omnidirectional*

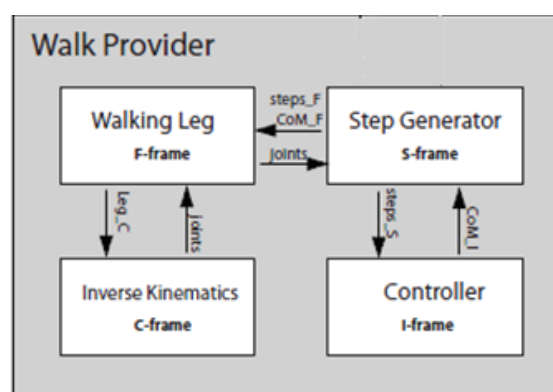
Terdapat banyak teknik untuk dapat melakukan gerakan *omnidirectional* dengan mempertimbangkan keseimbangan dari robot humanoid seperti teknik kontrol Fuzzy dan *Preview Controller* dengan kriteria kestabilan ZMP (*Zero Moment Point*). ZMP adalah titik yang memiliki keseimbangan antara momentum yang bekerja pada robot dengan momentum yang dilawan oleh dasar tumpuan. Jika titik ini berada pada *support polygon* robot maka dapat dipastikan robot tidak akan jatuh.^[5] Semua teknik kontrol yang digunakan selalu mengutamakan pada keseimbangan robot agar dapat bertumpu pada *single support polygon* sehingga rotasi pada kaki lain dapat dilakukan.

Implementasi gerakan *omnidirectional* ini tidak mudah, dibutuhkan perhitungan dan transformasi matriks yang rumit untuk dapat menentukan sudut pergerakan dari setiap persendian. Pada teknik kontrol menggunakan *Preview Controller*, intinya adalah membagi perencanaan pola langkah pada beberapa koordinat *frame* sehingga dapat membuat sistem menjadi lebih sederhana. Setiap koordinat *frame* mengatur perencanaan langkah, pengambilan langkah dan pengontrolan sendi pada kaki yang didasarkan pada *frame* referensi (*I-frame*). Ini memastikan sistem tetap terjaga karena setiap komponen hanya memiliki informasi yang terbatas terhadap koordinat *frame* nya. Gambaran dan hubungan tiap

– tiap *frame* ini dapat dilihat pada Gambar 2.4. Informasi – informasi dari satu koordinat *frame* dapat diolah oleh koordinat *frame* lain dengan menggunakan transformasi matriks. Meskipun memungkinkan adanya perhitungan yang cukup sulit dengan metoda ini tetapi metoda ini lebih mengurangi kesalahan dan mengurangi kerumitan sistem, dan ukuran matriks dengan cara ini menjadi lebih kecil (3x3). Nilai – nilai dari setiap koordinat *frame* ini diperbaharui setiap pergantian langkah atau setiap waktu tertentu. Keterangan – keterangan koordinat *frame* ini dapat dilihat pada Tabel 2.1.

Tabel 2.1 Keterangan Koordinat – Koordinat Frame

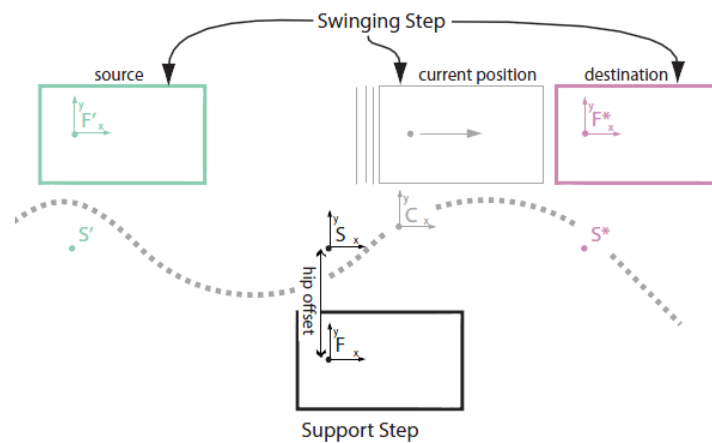
Koordinat Frame	Titik Pusat	Waktu Diperbaharui
C (<i>center of mass</i>)	CoM	Setiap waktu pergerakan
F (<i>Foot</i>)	<i>Support Foot</i>	Pada saat pergantian dari <i>single</i> ke <i>double support polygon</i>
S (<i>Step</i>)	$F \pm Ho$	Pada saat pergantian dari <i>single</i> ke <i>double support polygon</i>
I (<i>Initial</i>)	<i>World</i>	Tidak diperbaharui



Gambar 2.4 Hubungan Koordinat – Koordinat Frame Pada Teknik *Preview Controller*

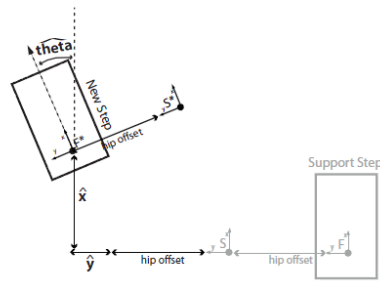
Koordinat *S-frame* adalah sebuah *step generator* yang berisi pola – pola langkah robot yang diinginkan. Setelah setiap langkah terlewati maka

koordinat *S-frame* akan memberikan koordinat untuk langkah selanjutnya. Koordinat dari *S-frame* ini dipindahkan ke *F-frame* dan *I-frame* dengan perhitungan transformasi matriks. Pada koordinat *I-frame* yang merupakan sebuah kontroler, koordinat – koordinat tersebut digunakan untuk mengubah koordinat dari CoM(*Center of Mass*) sehingga keseimbangan robot tetap dapat terjaga. Pada saat tertumpu pada *single support polygon*, koordinat – koordinat pada *S-frame* akan berpindah ke koordinat *F-frame* yang bertumpu pada *single support polygon*. Pergerakan pada setiap koordinat ini dapat dilihat pada Gambar 2.5.



Gambar 2.5 Koordinat – Koordinat Frame Pada Saat Melangkah

Pada kondisi ini, sudut rotasi dari kaki yang digerakan dapat diatur sehingga menuju arah yang diinginkan. Gambar 2.6 menunjukkan perputaran sudut pada sendi kaki yang digerakkan. Setelah semua pola langkah didapatkan maka koordinat – koordinat itu dikirimkan ke koordinat *C-frame*, di frame ini koordinat digunakan untuk menggerakkan sudut dan mengatur kecepatan dari setiap persendian menggunakan metoda *invers kinematics*.



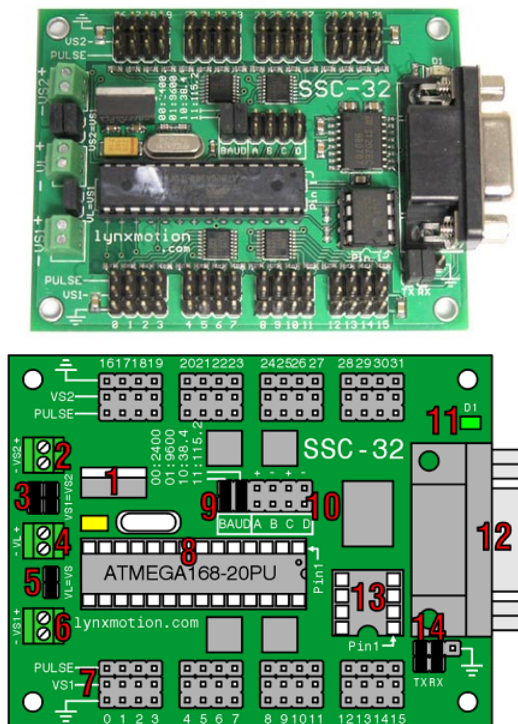
Gambar 2.6 Pergerakan Sudut Langkah Kaki Pada Saat Tertumpu Pada *Single Support Polygon*

II.3 Pengontrol Servo^[6]

Pengontrol Servo SSC-32 adalah perangkat keras yang memiliki IC pengontrol tersendiri yang berfungsi untuk mengatur secara serempak servo-servo dengan jumlah servo maksimal 32 buah.

II.3.1 Komponen-komponen SSC-32^[6]

SSC-32 terdiri dari beberapa komponen penyusun yang memiliki fungsi - fungsi tersendiri. Bentuk dan letak komponen - komponen SSC-32 ditunjukkan pada Gambar 2.7.



Gambar 2.7 Bentuk dan letak komponen-komponen SSC-32

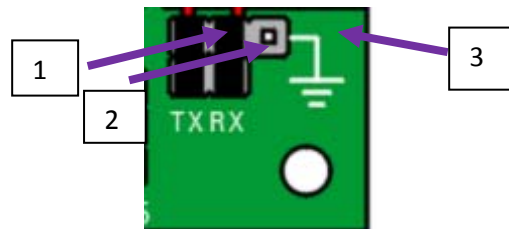
Penjelasan *board* pada SSC-32:

1. Regulator VDC untuk menurunkan tegangan dari maksimal tegangan 9 VDC menjadi 5 VDC. Nilai minimal tegangan yang masuk ke regulator adalah 5,5 VDC
2. Terminal untuk menghubungkan sumber tegangan dengan pin tegangan untuk servo 16 sampai 31.
3. Jumpers untuk menghubungkan VS1 dengan VS2. Hal ini digunakan ketika baterai yang dipakai hanya satu untuk semua servo.
4. VL atau *logic voltage* biasa memakai tegangan 9 VDC untuk keperluan IC. Ketika servo diberi tegangan terpisah dari VL maka *jumper* VS1=VL dilepas.
5. *Jumper* untuk membuat sumber tegangan dialokasikan untuk IC pengontrol dan servo. Tetapi ketika terlalu banyak servo yang digunakan lebih baik untuk memisahkan sumber tegangan pengontrol mikro dengan sumber tegangan servo-servo.
6. Terminal untuk menghubungkan sumber tegangan dengan pin sumber untuk servo 0 sampai 15.
7. Tempat untuk memasang kabel dari servo-servo. Kabel berwarna kuning untuk pin pulsa, kabel berwarna merah untuk pin VS, dan kabel berwarna hitam atau putih untuk pin ground.
8. Tempat Atmel IC berada yaitu ATMEGA 168. Dengan peletakan pin 1 pada bagian kanan atas.
9. Tempat konfigurasi nilai *baud rate* dengan cara mengubah posisi *jumpers*.
10. *Normally open switch* untuk menghubungkan *input* dengan ground.
11. LED sebagai indikator. Ketika SSC-32 mendapat tegangan maka LED akan menyala. Ketika SSC-32 menerima perintah serial maka LED akan kedip selama menerima perintah.
12. Tempat DB9 dipasang untuk komunikasi serial dengan PC untuk keperluan mengatur posisi servo.
13. 8-pin EEPROM yang didukung oleh 2.01GP firmware.

14. Tempat untuk memilih untuk mengaktifkan *TTL serial port* atau *DB9 serial port*.

II.3.2 Pengkabelan untuk Komunikasi SSC-32 dengan ATMEGA 128

Pengontrol servo SSC-32 berkomunikasi dengan ATMEGA 128 dengan menerima *input* data serial dari ATMEGA 128 melalui port serial (Tx,Rx). Data serial tersebut kemudian diterjemahkan oleh ATMEGA 168 yang berfungsi sebagai otak dari rangkaian SSC-32 menjadi perintah untuk menggerakkan servo sesuai dengan parameter yang diinginkan. Pada Gambar 2.8, pin no. 1 dihubungkan dengan pin Rx dari ATMEGA 128, pin no. 2 dihubungkan dengan pin Tx dari ATMEGA 128, pin no 3 (*ground*) dihubungkan dengan pin *Ground* dari ATMEGA 128.



Gambar 2.8 Pin SSC-32 Untuk Komunikasi Dengan ATMEGA128

II.3.3 Spesifikasi SSC-32^[6]

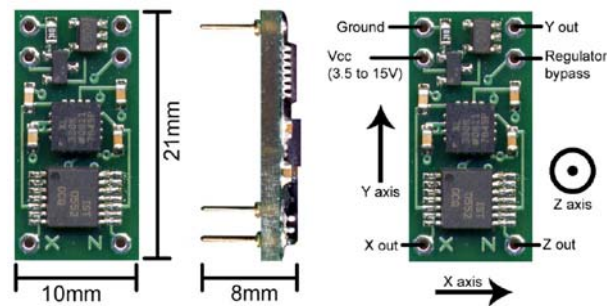
Pengontrol servo SSC-32 memiliki spesifikasi sebagai berikut:

1. Pengontrol mikro: ATMEGA168-20PU
2. EEPROM: 24LC32P
3. Osilator Crystal: 14.75 MHz
4. *Input* serial: RS-232 atau TTL
5. Baudrate serial: 2400, 9600, 38.4k, 115.2k
6. Arus *input*: 31 mA
7. PC interface: DB9F
8. Kendali motor servo: 32 servo maximal
9. Tipe servo: Futaba / Hitec
10. Jangkauan sudut servo: 180°
11. Resolusi servo: 1us, 0.09°
12. Ukuran pengontrol: 3 x 2,3 inchi

Kapasitas arus servo maximal: 30 A

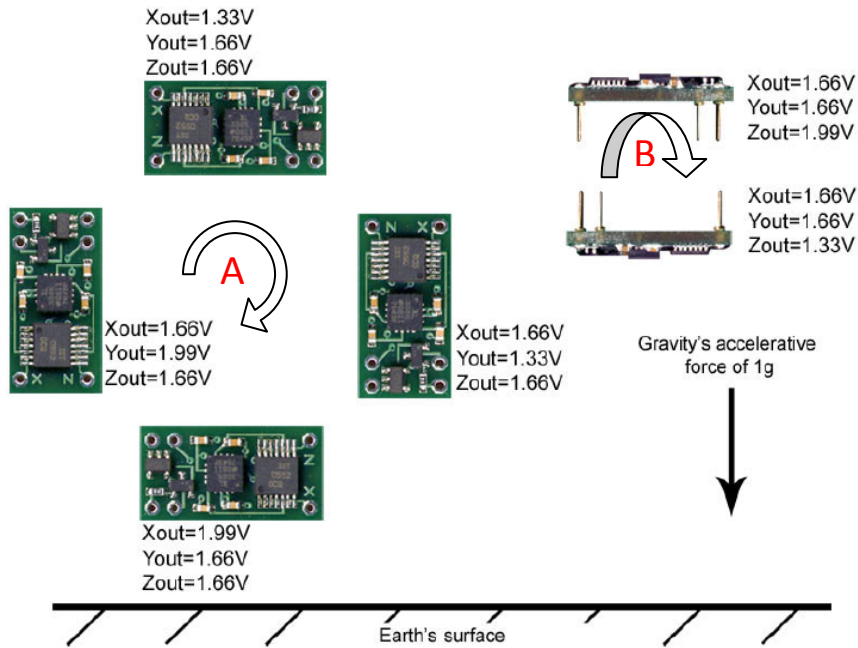
II.4 Sensor Percepatan DE-ACCM3D^[7]

DE-ACCM3D adalah sensor percepatan dengan tiga keluaran analog yang dapat digunakan untuk mendeteksi kemiringan atau percepatan pada tiga *axis*. Sensor ini dilengkapi dengan *operational amplifier buffer* yang terintegrasi sehingga dapat dikoneksikan secara langsung dengan masukan analog pengontrol mikro. Sensor ini juga memiliki regulator di dalamnya yang berfungsi untuk meregulasi catu daya yang masuk menjadi 3,3 volt. Untuk dapat bekerja dengan baik, sensor ini membutuhkan catu daya 3,5 sampai 15 volt. Gambar 2.9 memperlihatkan sensor percepatan DE-ACCM3D beserta dimensi dan arah tiap *axis*.



Gambar 2.9 Sensor Percepatan DE-ACCM3D

Dengan catu daya sebesar 3,3 volt, sensor ini memberikan keluaran analog pada tiap *axis* minimal 1,33 volt dan maksimal 1,99 volt. Nilai – nilai tersebut merepresentasikan besarnya kemiringan / percepatan pada tiap *axis* yang dapat dilihat pada Gambar 2.10. Pada Gambar 2.10 perputaran sensor dalam arah A akan mengubah besarnya nilai tegangan pada x *axis* dan y *axis*, sedangkan perputaran sensor dalam arah B akan mengubah besarnya nilai tegangan pada z *axis*.



Gambar 2.10 Nilai Tegangan Tiap Axis Pada Posisi – Posisi Tertentu

II.5 Sensor Kamera CMUCam3

CMUCam3 adalah sensor khusus penglihatan yang didesain agar murah, dapat diprogram secara penuh, dan sesuai untuk proses yang *realtime*. CMUCam3 adalah kamera CMOS generasi ketiga yang dibuat bersama oleh *Carnegie Mellon University* dan *Acroname*. Berikut ini tampilan CMUCam3 pada Gambar 2.11.



Gambar 2.11 CMUCam3

II.5.1 Arsitektur Perangkat Keras^[8]

Arsitektur perangkat keras CMUCam3 ini terdiri dari 3 komponen utama yaitu, CMOS camera chip, frame buffer, dan pengontrol mikro.

Pengontrol mikro mengkonfigurasi sensor CMOS menggunakan protokol serial dua kabel (*Two wire serial protocol*), kemudian pengontrol mikro menginisialisasi pengiriman gambar secara langsung dari camera CMOS ke *frame buffer*. Pengontrol mikro ini harus menunggu awal dari sebuah *frame* baru yang masuk pada saat itu, mengkonfigurasi sistem supaya dapat memuat data gambar ke tempat penyimpanan secara *asynchronous*. Setelah sensor CMOS ini telah memenuhi paling tidak dua blok memori *frame buffer* (128 bytes), prosesor utama dapat mulai mewaktu data 8 bits secara *asynchronous* saat waktu penyimpanan gambar habis. Akhir dari *frame* ini memicu interupsi perangkat keras yang pada saat itu prosesor utama mematikan jalur kontrol penulisan *frame buffer* sampai *frame buffer* selanjutnya dibutuhkan.

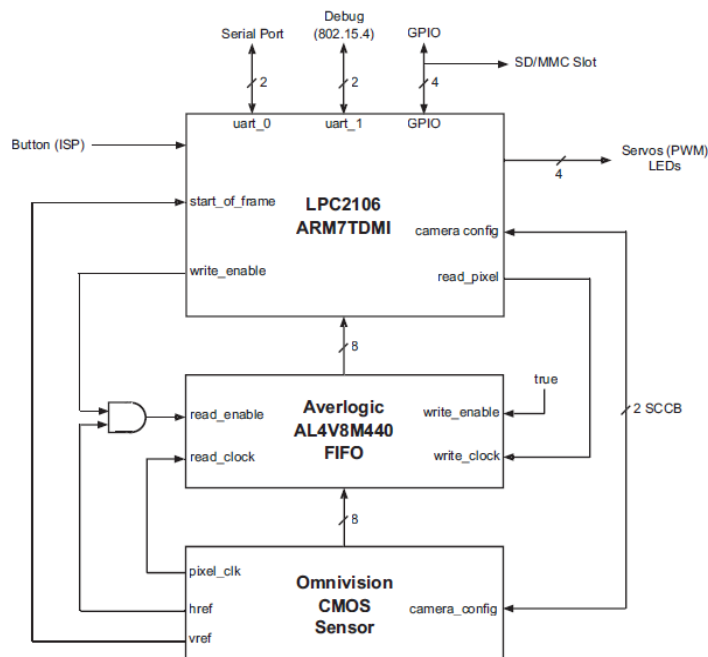
CMUCam3 ini memiliki dua sambungan serial (*one level shifted*), I²C, SPI, empat keluaran servo standar, tiga buah lampu yang dapat dikontrol secara perangkat lunak, tombol, dan slot MMC. Skenario operasi umum yang terjadi adalah pengontrol mikro berkomunikasi dengan CMUCam3 menggunakan koneksi serial. Tidak seperti sistem CMUCam yang sebelumnya, semua peripheral ini dikontrol oleh perangkat keras *processor* sehingga tidak mengurangi waktu pemrosesan.

Masukan gambar ke sistem dapat dihasilkan baik dari *Omnivision* OV6620 atau OV7620 *CMOS Camera*. OV6620 ini mendukung resolusi maksimum 352x288 dengan 50 *frame* per detik. Kamera ini akan mengeluarkan data 8 bits *color pixel* RGB atau YCrCb. Parameter kamera seperti *color saturation*, *brightness*, *contrast*, *white balance gains*, *exposure time*, dan *output mode* dikontrol menggunakan protokol dua kabel SCCB (*Serial Camera Control Bus*).

Perbedaan utama antara CMUCam2 dengan CMUCam3 adalah dari penggunaan pengontrol mikro NXP LPC2106. LPC2106 adalah 32-bit 60MHz ARM7TDMI dengan 64KiB RAM dan 128KiB memori flash dibangun di dalamnya. Prosesor ini mampu mengontrol skala frekuensi secara perangkat lunak dan mempunyai *memory acceleration module* (MAM) yang menyediakan prosesor ini untuk dapat mengambil data dari

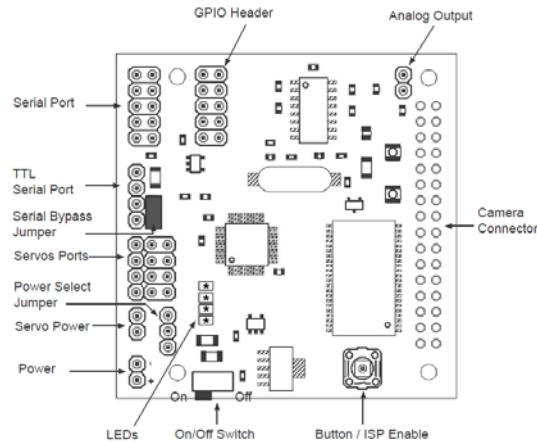
FLASH dalam 1 siklus pendek. *Bootloader* yang berada di dalamnya memungkinkan untuk mengunduh program yang telah tereksekusi menggunakan sambungan serial tanpa perangkat pemrograman luar seperti *downloader*. Selama prosesor menggunakan kumpulan instruksi ARM maka kode-kode dapat dikompilasi menggunakan kompilasi GNU GCC. Kode C yang umum dapat dikembangkan untuk CMUCam3 menggunakan port dari *GNU Toolchain* bersama dengan kumpulan *library* yang *open source* dan contoh program.

Frame buffer dalam CMUCam3 adalah 50MHz, 1MB AL4V8M440 *video FIFO* (*First In, First Out*) yang diproduksi oleh Averlogic. *Video FIFO* ini sangat penting karena memungkinkan kamera untuk beroperasi pada kecepatan penuh dan memisahkan pemrosesan dalam CPU dari pewaktu piksel kamera. Meskipun piksel tidak dapat diakses dalam *random access fashion*, FIFO memungkinkan untuk mengulang *pointer* baca yang memungkinkan untuk meneruskan beberapa proses gambar. Satu kerugian dari LPC2106 adalah I/O yang relatif lambat. Untuk membaca satu piksel dibutuhkan 14 putaran waktu, yang 12 nya digunakan untuk menunggu adanya I/O. Arsitektur perangkat keras CMUCam3 ini dapat dilihat pada Gambar 2.12.



Gambar 2.12 Arsitektur Perangkat Keras CMUCam3

II.5.1.1 Koneksi Perangkat Keras



Gambar 2.13 Koneksi Perangkat Keras CMUCam3

Koneksi perangkat keras CMUCam3 dapat dilihat pada Gambar 2.13 dengan penjelasan sebagai berikut :

Power

Catu daya masukan ke CMUCam3 yang terlebih dahulu melalui *regulator* 5 volt. Catu daya yang diperbolehkan untuk CMUCam3 agar dapat bekerja dengan baik adalah 6 sampai 15 volt dan dapat memberikan arus 150 mA. Catu daya untuk servo dapat secara langsung diambil dari catu daya utama atau dapat juga menggunakan catu daya luar melalui masukan *servo power*. Pada saat menggunakan catu daya luar maka *power select jumper* harus dilepas.

Serial Port

CMUCam3 mempunyai jalur serial *level shifted* standar untuk berkomunikasi dengan komputer dan serial TTL untuk berkomunikasi dengan pengontrol mikro tanpa harus menggunakan *level shifted IC*. Pada saat menggunakan jalur serial TTL, *serial jumper* harus dilepas.

Camera Bus

Camera bus adalah jalur yang untuk berkomunikasi dengan kamera CMOS.

Servo Port

Jalur keluaran yang digunakan untuk mengendalikan servo. Jalur keluaran ini dapat memuat 4 buah servo.

Expansion Port GPIO

Jalur GPIO (*General Purpose Input Output*) dapat digunakan untuk mengakses UART kedua, bermacam – macam kontrol catu daya dan jalur SPI.

Analog Output Port

Dengan menggunakan modul kamera OV6620 maka sinyal video PAL dapat diambil dari jalur keluaran analog ini.

LED

LED 0 terhubung dengan pin MOSI pada port GPIO dan akan menyala pada saat MMC terpasang.

LED 1 terhubung dengan servo nomor 2. Pada saat menggunakan jalur servo nomor 2 maka LED 1 akan menyala.

LED 2 terhubung dengan servo nomor 3. Pada saat menggunakan jalur servo nomor 3 maka LED 2 akan menyala.

II.5.2 Arsitektur Perangkat Lunak

Sistem standar pencitraan mengasumsikan ketersediaan perangkat keras sekelas PC. Sistem seperti OpenCV, LTI-Lib, MATLAB membutuhkan ruang untuk alamat memori yang sangat besar (*megabytes*) dan ditulis dalam bahasa yang berat seperti C++ atau java. CMUCam3 hanya memiliki 64KiB RAM dan oleh sebab itu tidak bisa menggunakan *library* pencitraan standar. Untuk memecahkan masalah ini, CMUCam3 didesain dan diimplementasikan sistem pencitraan cc3 sebagai perangkat lunak utama untuk CMUCam3.

Sistem cc3 adalah C API (*Application Programming Interface*) untuk melakukan pencitraan dan kontrol, dioptimalkan untuk CMUCam3. Fitur-fitur yang dimiliki oleh sistem cc3 ini adalah :

- Lapisan abstraksi untuk berinteraksi dengan sistem perangkat keras di masa depan
- Aturan modern C99 dengan nama tipe dan nama fungsi yang konsisten
- Mendukung sejumlah format gambar untuk kesederhanaan
- Dokumentasi yang disediakan melalui Doxygen

- Berversi API untuk dapat diperluas di masa depan
- Modul *virtual-cam* untuk *testing* dan *debugging*

Sistem cc3 ini memiliki *library* yang berisi fungsi – fungsi yang mendukung pengolahan citra. Fungsi – fungsi yang dipakai dalam Penelitian ini adalah sebagai berikut :

cc3_uart_init

menginisialisasi koneksi *serial* UART seperti *baud rate*, *control bits*, *stop bit*, *mode*.

cc3_camera_init

menginisialisasi perangkat keras kamera. Fungsi ini akan me *reset* kamera dan menghilangkan parameter – parameter *pixel buffer*.

cc3_camera_set_resolution

mengatur resolusi dari perangkat keras kamera.

cc3_gpio_set_mode

mengkonfigurasi gpio untuk menjadi masukan, keluaran atau untuk mengendalikan servo.

cc3_gpio_set_servo_position

mengatur pergerakan servo ke posisi tertentu.

cc3_pixbuf_load

mengambil gambar dari kamera dan memuat nya di *internal pixel buffer*.

cc3_pixbuf_read_rows

melakukan duplikasi baris demi baris dari *pixel buffer* ke dalam blok memori.

cc3_malloc_rows

mengalokasikan jumlah baris dari ukuran gambar yang didapat dari parameter – parameter *pixel buffer*.

cc3_track_color_scanline_start

menginisialisasi paket – paket posisi ke kondisi awal untuk proses pencarian warna.

cc3_track_color_scanline

memulai proses pencarian warna, jika ditemukan warna yang sesuai dengan paket – paket warna maka data posisi akan dimasukkan ke dalam paket – paket posisi. Paket – paket posisi ini berisi informasi tentang titik tengah (x,y), lebar dan panjang kotak yang melingkupi, dan banyak nya pixel dari warna yang terdeteksi.

cc3_track_color_scanline_finish

menampilkan paket – paket posisi bila ditemukan dalam proses pencarian, jika tidak ditemukan maka fungsi ini akan mengembalikan paket – paket posisi warna ke kondisi awal.

II.5.3 Metoda Pelacakan Warna CMUCam3^[9]

Pelacakan warna adalah kemampuan untuk mengambil gambar, mengisolasi warna tertentu, dan mengekstrak informasi tentang lokasi suatu daerah gambar yang berisi hanya warna itu. Untuk menentukan warna, diperlukan nilai minimum dan nilai maksimum untuk tiga kanal warna. Setiap warna yang unik diwakili oleh nilai merah, hijau dan biru (RGB) yang mengindikasikan berapa banyak setiap warna pada kanal tersebut dicampur.

Pada CMUCam3, setiap kanal warna tersebut memiliki batasan nilai dari 0 sampai 255 sehingga diperlukan 6 nilai yang mempresentasikan nilai minimum dan maksimum setiap kanal warna yang ingin dilacak. Setelah ditentukan, CMUCam3 mengambil nilai tersebut dan memulai untuk memproses gambar. Banyak metoda yang dapat dipakai untuk melacak warna, CMUCam3 mengambil metoda yang sederhana yaitu memproses setiap *frame* gambar dari kamera secara terpisah. Dimulai dari bagian atas kiri gambar dan secara berurutan meneliti setiap pixel baris demi baris. Jika pixel tersebut memenuhi nilai warna yang ingin dilacak maka posisi tersebut ditandai. Nilai – nilai tersebut kemudian digunakan untuk mengetahui posisi kiri atas, kiri bawah, kanan atas dan kanan bawah dari warna yang terlacak pada gambar sehingga terbentuk kotak yang melingkupi warna yang diinginkan tersebut. Pada akhir gambar, CMUCam3 menjumlahkan nilai pixel yang terdapat pada kotak tersebut

dan mencari titik tengah dari kotak tersebut. Oleh karena itu, informasi yang dikeluarkan oleh pelacakan warna pada CMUCam3 berupa posisi x awal, y awal, x akhir, y akhir, titik tengah x, titik tengah y, dan jumlah pixel.

BAB III

PERANCANGAN DAN REALISASI

Pada bab ini dijelaskan tentang perancangan dan realisasi sistem robot humanoid pemain bola, perancangan dan realisasi rangkaian sensor dan pengontrol, serta algoritma pemrograman robot humanoid pemain bola.

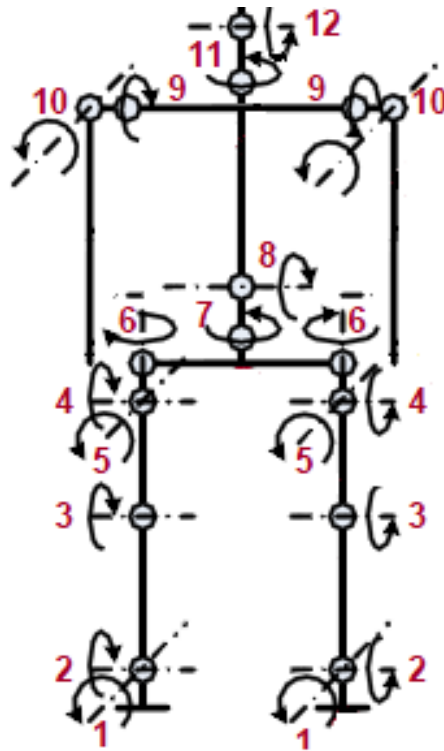
III.1 Perancangan Sistem Robot Humanoid Pemain Bola

Robot humanoid yang dirancang pada Penelitian ini bertujuan agar dapat mendeteksi bola, dapat berjalan menuju bola atau posisi yang diinginkan dengan konsep *omnidirectional*, dapat menendang bola menuju gawang lawan, dapat bangkit berdiri ketika terjatuh sehingga dibutuhkan perancangan yang baik pada sistem.

Sistem mekanika robot humanoid pemain bola ini dirancang agar dapat menyerupai struktur tubuh manusia dengan 20 sendi seperti ditunjukkan pada Gambar 3.1.

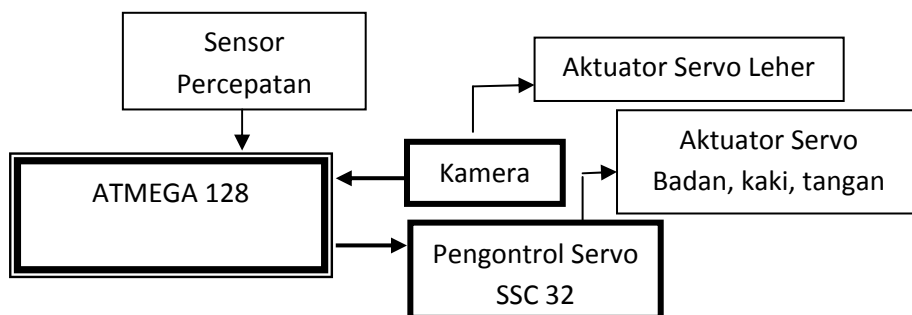
Dengan keterangan – keterangan sebagai berikut :

- | | |
|------------------------------|-----------------------|
| 1. Sendi pangkal betis roll | 7. Sendi perut |
| 2. Sendi pangkal betis pitch | 8. Sendi badan |
| 3. Sendi lutut | 9. Sendi pundak pitch |
| 4. Sendi pangkal paha pitch | 10. Sendi pundak roll |
| 5. Sendi pangkal paha roll | 11. Sendi leher yaw |
| 6. Sendi pinggul | 12. Sendi leher pitch |



Gambar 3.1 Struktur Robot Humanoid Pemain Bola

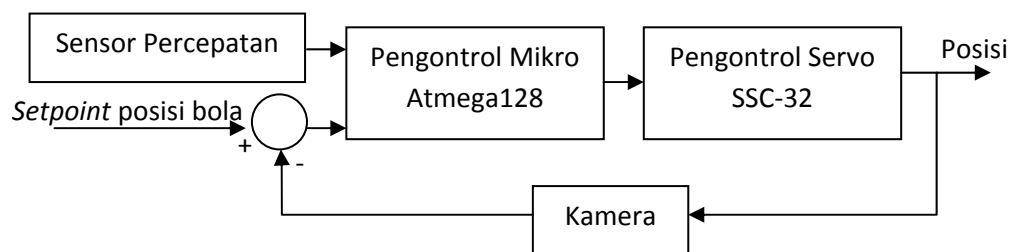
Sistem elektronika robot humanoid pemain bola dirancang untuk dapat mendeteksi warna bola dan gawang menggunakan kamera CMUCam3, menggerakkan sendi – sendi robot yang berupa motor servo menggunakan pengontrol servo SSC-32, dan mendeteksi jatuhnya robot menggunakan sensor percepatan DE-ACCM3D. Keseluruhan proses tersebut dikontrol oleh sebuah pengontrol mikro ATMEGA128. Diagram blok sistem elektronika pada robot ini dapat dilihat pada Gambar 3.2.



Gambar 3.2 Diagram blok sistem elektronika Robot Humanoid Pemain Bola

Kamera CMUCam3 diprogram untuk dapat melacak warna dan menggerakkan motor servo pada leher. Pertama kali, kamera mulai melacak

warna dari gawang. Setelah gawang terlacak, kamera akan secara terus – menerus melacak warna dari bola dan mengirimkan data serial ke pengontrol mikro utama berupa data – data posisi bola yang direpresentasikan oleh posisi servo pada leher. Pengontrol mikro ATMEGA128 mengambil data –data dari kamera yang kemudian digunakan untuk dapat menentukan arah gerak robot menuju posisi yang diinginkan. Sistem gerak pada badan robot digerakkan oleh motor servo yang dikontrol oleh pengontrol servo SSC-32. Pengontrol servo SSC-32 menerima perintah secara serial dari pengontrol mikro ATMEGA128 berupa posisi dan kecepatan tiap servo. Pada saat sensor percepatan mendeteksi robot terjatuh maka pengontrol mikro ATMEGA128 akan mengirimkan perintah ke pengontrol servo SSC-32 untuk melakukan aksi berdiri. Diagram blok sistem kontrol ini dapat dilihat pada Gambar 3.3.



Gambar 3.3 Diagram Blok Sistem Kontrol

III.2 Realisasi Sistem Robot Humanoid Pemain Bola

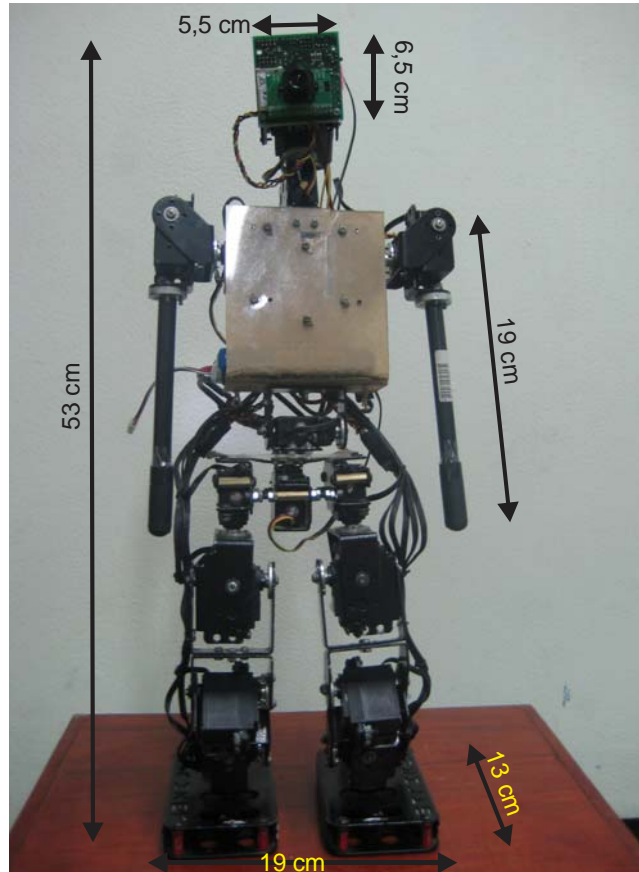
Proses di dalam merealisasikan robot humanoid pemain bola ini dibagi menjadi 3 bagian yaitu pembuatan sistem mekanika, sistem elektronika dan algoritma pemrograman.

III.2.1 Sistem Mekanika Robot Humanoid Pemain Bola

Struktur robot humanoid pemain bola dirancang agar sesuai dengan bentuk tubuh manusia. Rangka yang digunakan terbuat dari bahan alumunium dan akrilik. Bahan alumunium digunakan untuk menyambungkan sendi – sendi robot dan bahan akrilik digunakan untuk badan robot.

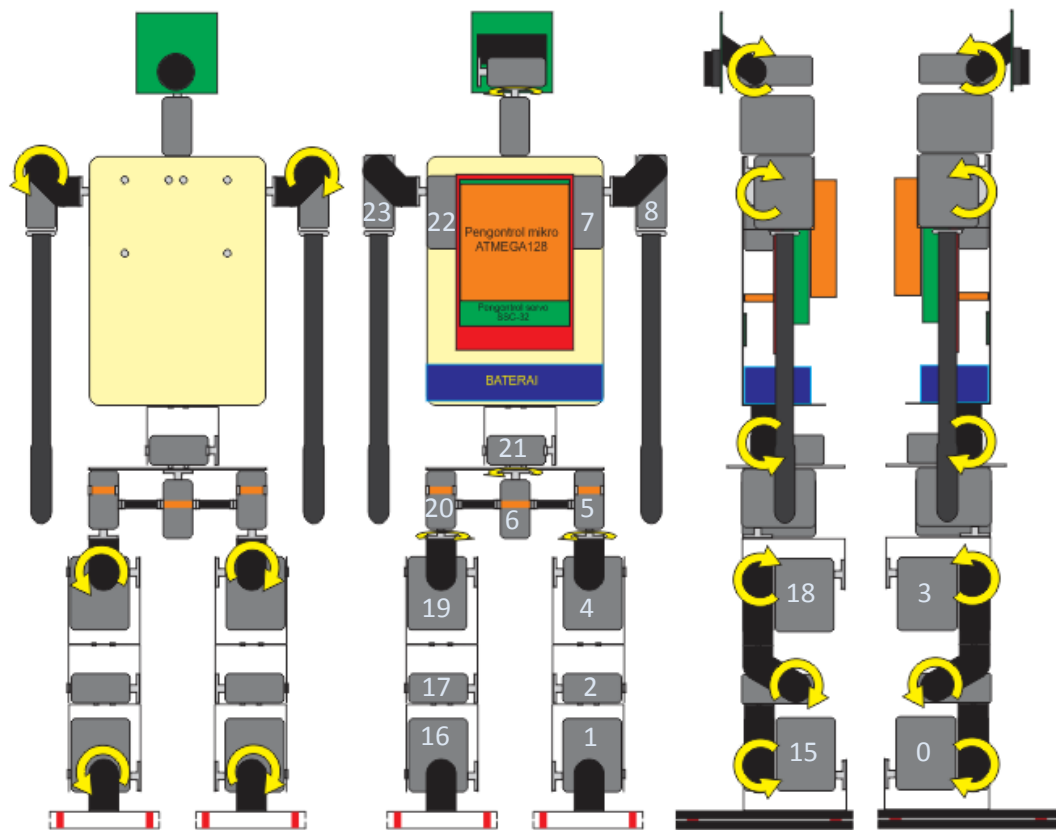
Bentuk robot mengacu pada aturan perlombaan humanoid soccer *kidsize* dengan tinggi(H) 30 sampai 60 cm. Robot humanoid pemain bola ini memiliki panjang 19 cm, lebar 13 cm, dan tinggi maksimal 53cm. Penampang kaki dengan

panjang 12 cm dan lebar 7.5 cm. Memiliki rentang tangan 57cm, rentang kaki 48 cm, bagian kepala memiliki tinggi 6.5 cm dan lebar 5.5 cm. Dimensi robot dapat dilihat pada Gambar 3.4.



Gambar 3.4 Dimensi Robot pada Saat Berdiri

Terdapat 20 sendi pada robot yang digerakan oleh motor servo. 6 sendi pada masing – masing kaki, 2 sendi pada masing-masing tangan, 2 sendi pada perut, dan 2 sendi pada kepala. Sistem gerak dan peletakan servo ini dapat dilihat pada Gambar 3.5. Penomoran pada masing – masing servo berdasarkan peletakan pada pengontrol servo SSC-32.



Gambar 3.5 Sistem Gerak dan Peletakan Servo pada Robot

III.2.2 Sistem Elektronika Robot humanoid Pemain Bola

Sistem elektronika pada robot humanoid pemain bola ini terbagi menjadi 2 yaitu sensor dan pengontrol.

III.2.2.1 Sensor

Sensor yang digunakan pada robot humanoid pemain bola ini meliputi sensor percepatan DE-ACCM3D dan kamera CMUCam3.

III.2.2.1.1 Sensor Percepatan DE-ACCM3D

Sensor percepatan DE-ACCM3D dalam Penelitian ini digunakan untuk memberikan informasi kemiringan 3 sumbu (x,y, dan z) yang terjadi pada robot berupa nilai ADC. Kemiringan tersebut digunakan untuk mengetahui robot terjatuh atau tidak. Sensor membutuhkan catu daya 5 volt yang diambil dari keluaran tegangan pengontrol mikro utama dan memberikan keluaran sensor berupa tegangan antara 1.33 sampai 1.99 volt. Sensor ini diletakkan pada titik keseimbangan robot yang diasumsikan berada di perut.

III.2.2.1.2 Kamera CMUCam3

Kamera CMUCam3 digunakan untuk melacak warna gawang dan bola. Kamera CMUCam3 membutuhkan catu daya 5 volt yang diambil dari keluaran catu daya pengontrol mikro utama dan catu daya untuk menggerakkan servo sebesar 7.4 volt yang diambil langsung dari baterai lithium polymer. Kamera CMUCam3 berada pada bagian kepala dan mengatur pergerakan servo pada leher membentuk 2 derajat kebebasan sehingga dapat melakukan pelacakan dalam 2 sumbu. Keluaran kamera berupa data posisi dari servo leher ketika mendeteksi warna gawang atau bola. Data ini dikirimkan secara serial melalui TTL serial port dan dibaca oleh USART0 pada pengontrol mikro utama ATMEGA128.

III.2.2.2 Pengontrol

Pengontrol yang digunakan pada Penelitian ini adalah pengontrol servo SSC-32 dan pengontrol mikro ATMEGA128.

III.2.2.2.1 Pengontrol Servo SSC-32

Pengontrol servo SSC-32 mengatur pergerakan semua servo pada robot humanoid pemain bola kecuali pada 2 servo leher. Perputaran sudut pada servo dari 0° sampai 180° direpresentasikan dengan nilai 0 sampai 255. Hubungan port keluaran SSC-32 dengan sendi-sendi robot dapat dilihat pada Tabel 3.1.

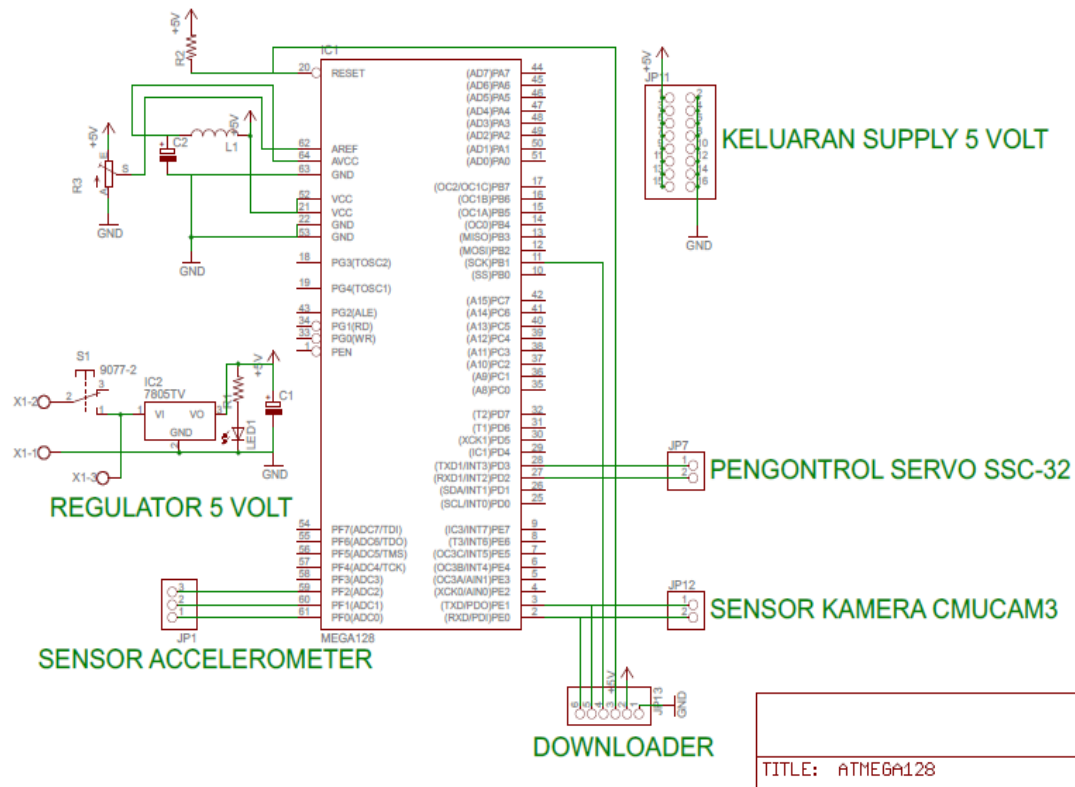
Tabel 3.1 Hubungan Port SSC-32 dengan Sendi pada Robot

Nomor Port	Sendi Robot	Nomor Port	Sendi Robot
0	Sendi pangkal betis <i>roll</i> kanan	16	Sendi pangkal betis <i>roll</i> kiri
1	Sendi pangkal betis <i>pitch</i> kanan	17	Sendi pangkal betis <i>pitch</i> kiri
2	Sendi lutut kanan	18	Sendi lutut kiri
3	Sendi pangkal paha <i>roll</i> kanan	19	Sendi pangkal paha <i>roll</i> kiri
4	Sendi pangkal paha <i>pitch</i> kanan	20	Sendi pangkal paha <i>pitch</i> kiri
5	Sendi pinggul kanan	21	Sendi pinggul kiri
6	Sendi badan	22	Sendi perut
7	Sendi pundak <i>pitch</i> kanan	23	Sendi pundak <i>pitch</i> kiri
8	Sendi pundak <i>roll</i> kanan	24	Sendi pundak <i>roll</i> kiri
9	Tidak dipakai	25	Tidak dipakai
10	Tidak dipakai	26	Tidak dipakai
11	Tidak dipakai	27	Tidak dipakai
12	Tidak dipakai	28	Tidak dipakai
13	Tidak dipakai	29	Tidak dipakai
14	Tidak dipakai	30	Tidak dipakai
15	Tidak dipakai	31	Tidak dipakai

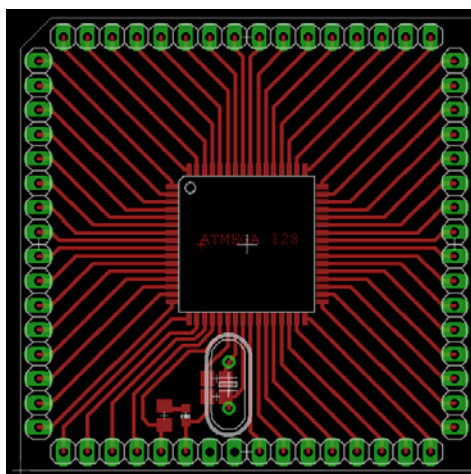
Pengontrol servo SSC-32 membutuhkan catu daya utama 5 volt dan catu daya untuk menggerakkan servo sebesar 7.4 volt. Perintah yang diterima oleh pengontrol servo berupa nilai posisi pergerakan dan kecepatan dari tiap servo yang dikirimkan oleh pengontrol mikro utama ATMEGA128 melalui port serial USART1. Pengontrol servo SSC-32 diletakan pada bagian badan robot humanoid pemain bola.

III.2.2.2.2 Pengontrol Mikro ATMEGA128

Skematik rangkaian pengontrol mikro ATMEGA128 dapat dilihat pada Gambar 3.6. Karena ATMEGA128 hanya tersedia dalam bentuk TQFP, maka didesain sebuah papan rangkaian khusus sehingga mudah untuk dilakukan pergantian jika terjadi kesalahan. Papan rangkaian khusus untuk ATMEGA128 ini dapat dilihat pada Gambar 3.7.



Gambar 3.6 Skematik Rangkaian Pengontrol Mikro ATMEGA128



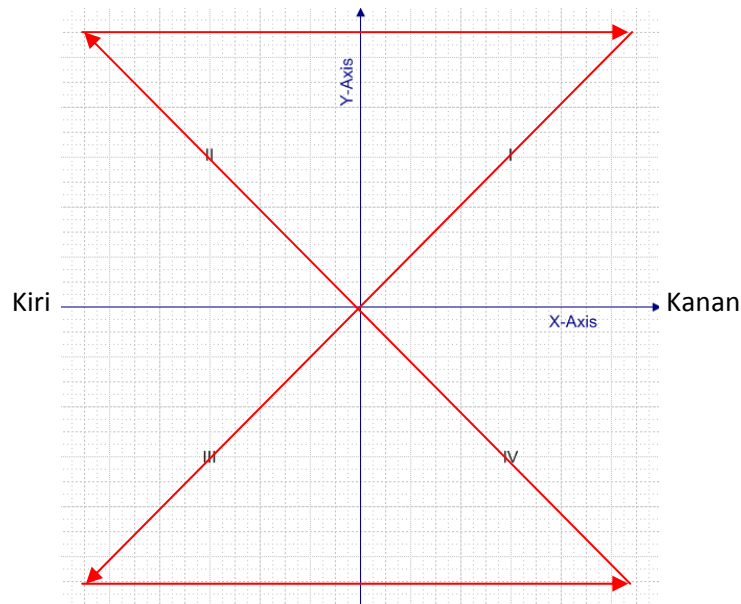
Gambar 3.7 Papan Rangkaian khusus untuk ATMEGA128

Pada rangkaian pengontrol mikro ATMEGA128 terdapat IC LM7805 yang digunakan untuk meregulasi catu daya yang masuk ke ATMEGA128 sebesar 5 volt. *clock* yang digunakan berasal dari crystal sebesar 7.372800MHz dan kapasitor 22pF seperti yang dianjurkan oleh Atmel. Nilai dari crystal ini dianggap cukup untuk dapat menghasilkan kesalahan 0% pada komunikasi serial dengan baudrate 115.200. PORT A.0 sampai PORT A.2 pada ATMEGA128 digunakan untuk masukan nilai ADC dari sensor percepatan. Catu daya referensi yang digunakan pada rangkaian ini diatur oleh potensiometer dan diatur pada nilai 1.99 yang merupakan nilai maksimal dari sensor percepatan. Port D.2 dan Port D.3 yang merupakan *port serial* TX1 dan RX1 digunakan untuk berkomunikasi dengan SSC-32. Kemudian TX0 dan RX0 pada Port E.0 dan E.1 digunakan untuk *downloader* dan berkomunikasi dengan kamera CMUCam3. Pengontrol mikro ATMEGA128 ini terletak pada bagian badan robot.

III.2.3 Algoritma Pemrograman Robot Humanoid Pemain Bola

Dalam robot humanoid pemain bola ini, pemrograman dilakukan pada kamera CMUCam3 dan pada pengontrol mikro utama ATMEGA128. Di awal program, kamera akan mulai melacak warna gawang. Pada saat kamera tidak mendapatkan warna yang ingin dilacak maka kamera akan melakukan proses *scanning*. Kamera memiliki *setpoint* untuk letak target dalam koordinat kamera yaitu pada koordinat $[x,y] = [100,11]$. Setelah posisi gawang berada di *setpoint* kamera, maka kamera akan beralih untuk melacak warna bola. Pergerakan servo pada leher robot dibagi menjadi 4 kuadran sehingga memudahkan untuk proses

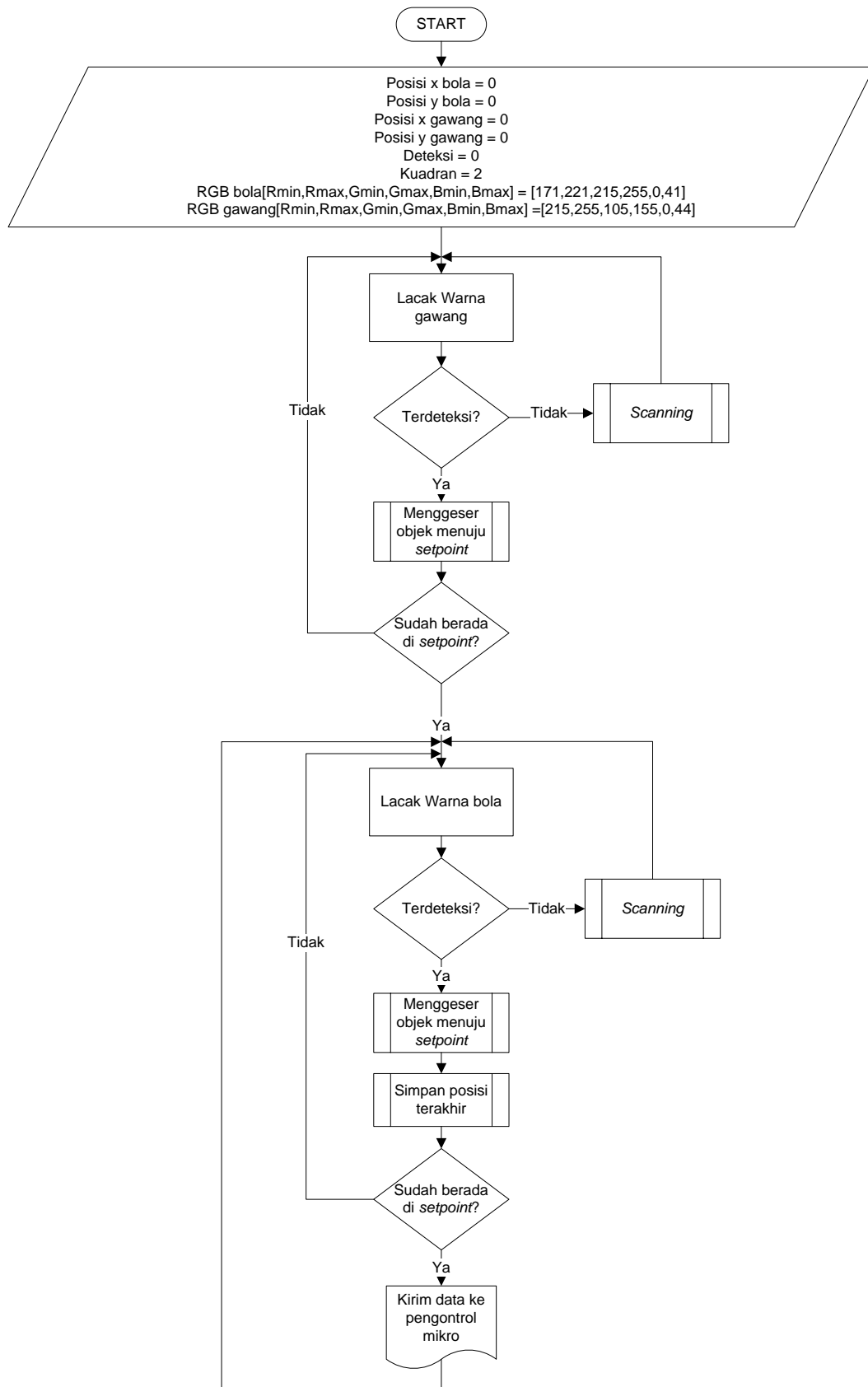
scanning. Karena nilai pergerakan servo pada leher antara 0 sampai 255 maka pembagian kuadran ini terdapat pada nilai(x,y) 128,128. Untuk dapat melacak warna gawang ataupun bola pada saat pertama kali *start*, kamera akan berada pada titik tengah (128,128) kemudian kamera akan melakukan gerakan *scanning* ke kuadran II, I, III, dan IV dengan arah yang digambarkan anak panah berwarna merah pada Gambar 3.8 dengan pandangan kamera masuk ke dalam bidang.



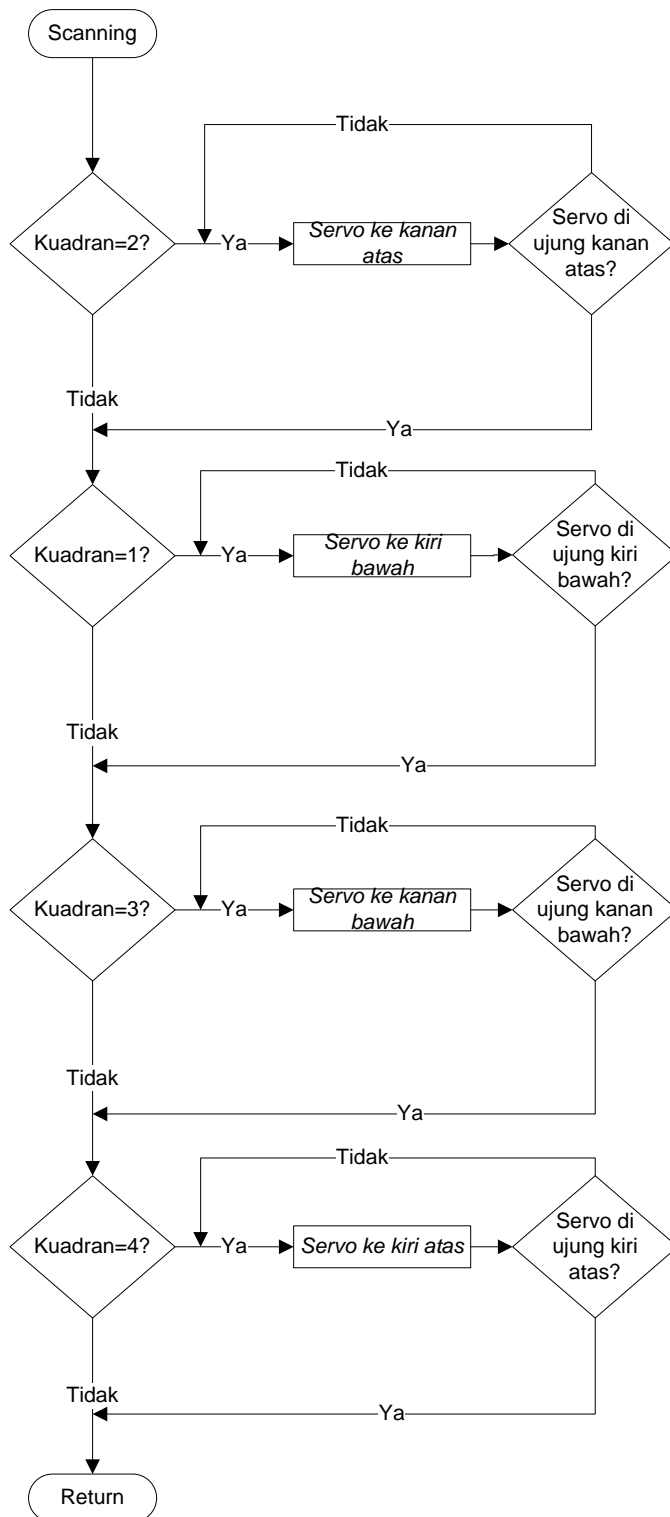
Gambar 3.8 Arah *Scanning* Pada Kamera

Ketika benda yang terlacak oleh kamera lepas, kamera akan membandingkan nilai posisi servo sebelumnya dengan nilai posisi servo saat ini. Jika nilai posisi servo sebelumnya berada di kiri atas nilai posisi servo sekarang maka nilai posisi servo akan dikontrol untuk menuju ke kuadran II, jika nilai posisi servo sebelumnya berada di kanan atas nilai posisi servo sekarang maka nilai posisi servo akan dikontrol untuk menuju ke kuadran I, begitu pula dengan nilai – nilai kiri bawah dan kanan bawah sehingga memungkinkan untuk mendapatkan kembali benda yang dilacak.

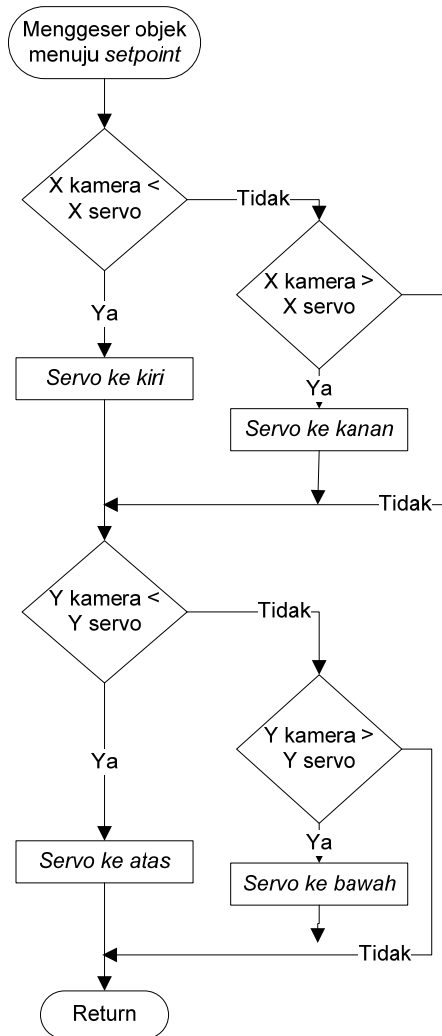
Data – data dari nilai posisi x,y pada gawang dan bola kemudian dikirimkan secara serial ke pengontrol mikro ATMEGA128. Diagram alir dari program pada kamera ini dapat dilihat pada Gambar 3.9a sampai 3.9d.



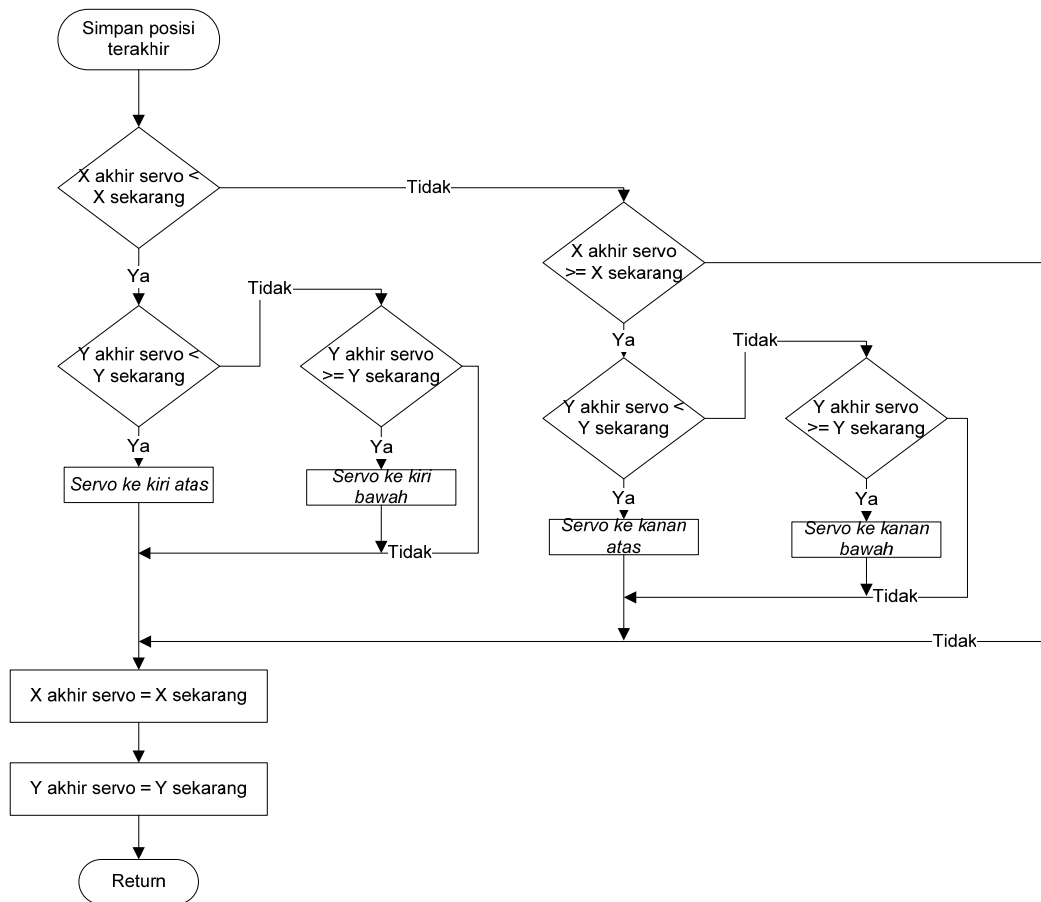
Gambar 3.9a Diagram Alir Program Utama Kamera CMUCam3



Gambar 3.9b Diagram Alir Subrutin *Scanning*

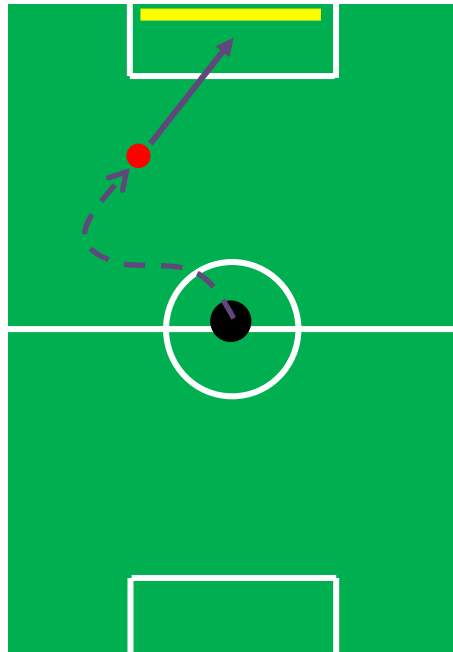


Gambar 3.9c Diagram Alir Subrutin Menggeser Objek Menuju *Setpoint*

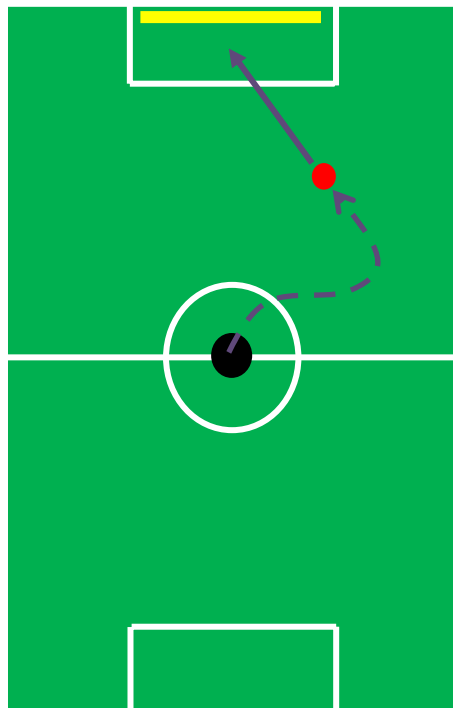


Gambar 3.9d Diagram alir Subrutin Simpan Posisi Terakhir

Sedangkan untuk pengontrol mikro ATMEGA128 pada awal program akan memberikan perintah untuk berdiri tegak dan menunggu kamera mengirimkan data posisi bola. Setelah mendapatkan data posisi bola, pengontrol mikro akan membuat keputusan arah gerak berdasarkan posisi gawang dan posisi bola. Jika bola berada di sebelah kiri gawang maka robot akan memilih arah gerak jauh ke kiri terlebih dahulu, sebaliknya jika bola berada di sebelah kanan gawang maka robot akan memilih arah gerak jauh ke kanan terlebih dahulu, sedangkan jika bola searah dengan gawang maka robot akan melakukan arah gerak lurus. Arah gerak ini digunakan untuk proses gerakan omnidirectional agar gawang berada searah dengan bola pada saat robot menendang bola. Penjelasan arah gerak robot ini dapat dilihat pada Gambar 3.10 dan 3.11.



Gambar 3.10 Arah Gerak Robot pada saat Bola di Sebelah Kanan Gawang



Gambar 3.11 Arah Gerak Robot pada saat Bola di Sebelah Kiri Gawang

Pengontrol mikro mengatur pergerakan – pergerakan pada robot dengan cara mengirimkan perintah serial ke pengontrol servo SSC-32. Perintah ini berupa *channel* servo, posisi servo dan kecepatan servo yang dikehendaki. Pergerakan

pada robot humanoid pemain bola ini semuanya diatur secara *preprogram* dikarenakan perhitungan matematik menggunakan *Inverse Kinematic* untuk menggerakkan 18 buah servo menuju posisi yang diinginkan sangatlah rumit. Robot humanoid pemain bola ini memiliki 5 gerakan yaitu berdiri, persiapan berjalan, menendang bola, dan bangkit berdiri. Gerakan – gerakan tersebut diperoleh dengan cara mengatur pergerakan dari 18 buah servo pada robot. Sudut – sudut servo pada seluruh gerakan kecuali gerakan berjalan dikirimkan secara langsung ke pengontrol servo SSC-32 yang dapat dilihat pada masing - masing subrutin gerakan. Sedangkan untuk gerakan berjalan, sudut – sudut yang telah di peroleh secara *trial* akan dikombinasikan dengan penambahan sudut putar pada bagian pinggul sehingga memungkinkan robot untuk dapat mengubah arah jalan. Tabel pergerakan servo – servo pada robot ketika berjalan dapat dilihat pada Tabel 3.2a dan Tabel 3.2b.

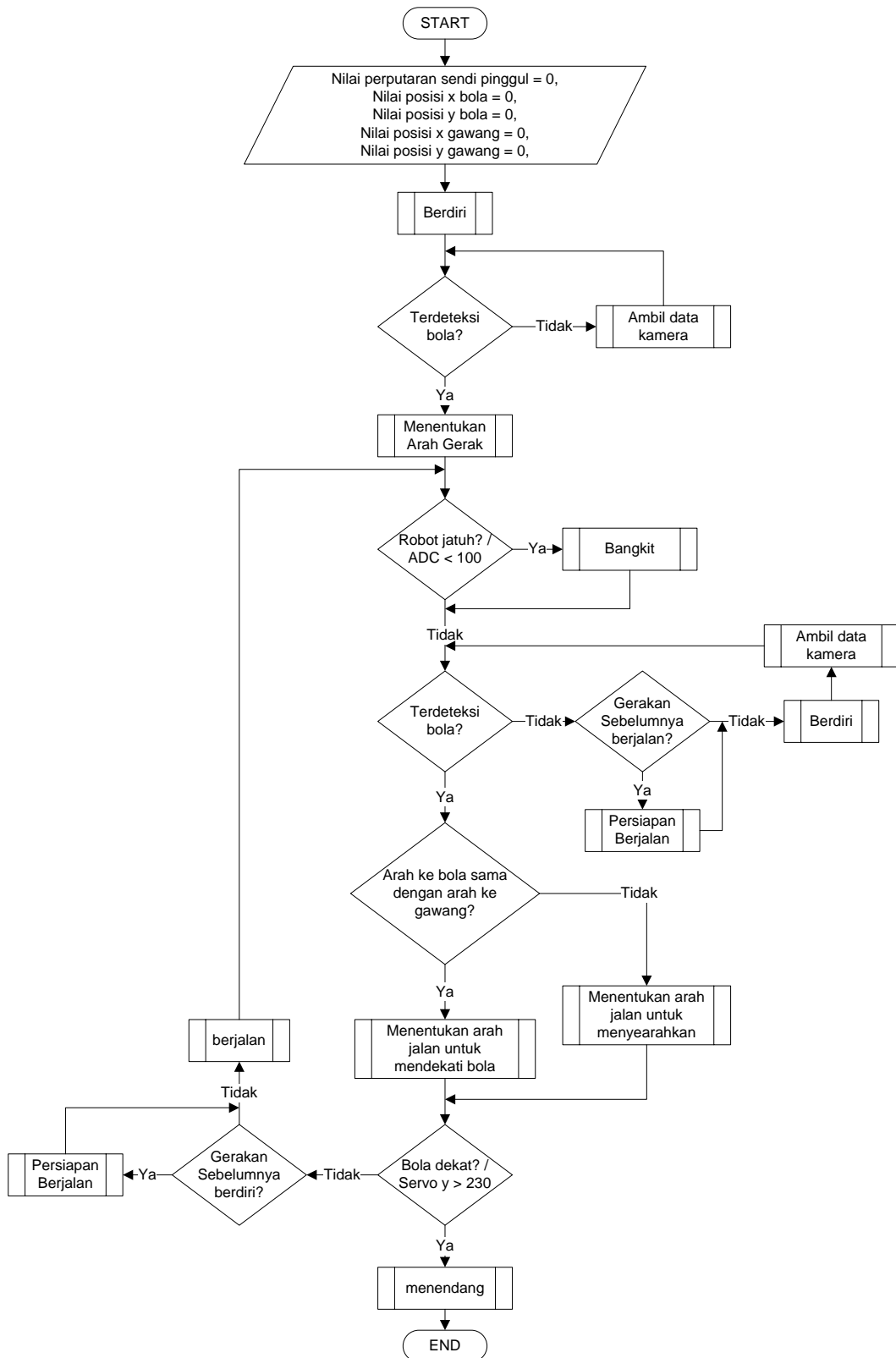
Tabel 3.2a Pergerakan Servo pada Gerakan Berjalan(1)

Servo Channel	Langkah 1		Lankah 2		Langkah 3	
	Posisi	Waktu (detik)	Posisi	Waktu (detik)	Posisi	Waktu (detik)
0	-24.3°	1000	-6.3°	1000	8.19°	1000
1	-29.2°	1000	-36°	1000	-19.9°	1000
2	-28.8°	1000	-25.2°	1000	-45.3°	1000
3	-4.68°	1000	-1.62°	1000	10.6°	1000
4	-60.8°	1000	-60.8°	1000	-70.7°	1000
5	20°	1000	20°	1000	20°	1000
6	73.7°	1000	73.7°	1000	73.7°	1000
7	-2.7°	1000	-2.7°	1000	-2.7°	1000
8	-45°	1000	-45°	1000	-45°	1000
16	-16.3°	1000	-1.8°	1000	-11.9°	1000
17	15.7°	1000	36°	1000	36°	1000
18	58.2°	1000	25.2°	1000	25.2°	1000
19	-8.28°	1000	3.6°	1000	-8.64°	1000
20	79.5°	1000	66.1°	1000	63.5°	1000
21	20°	1000	20°	1000	20°	1000
22	-35°	1000	-35°	1000	-35°	1000
23	4.5°	1000	4.5°	1000	4.5°	1000
24	59.6°	1000	59.6°	1000	59.6°	1000

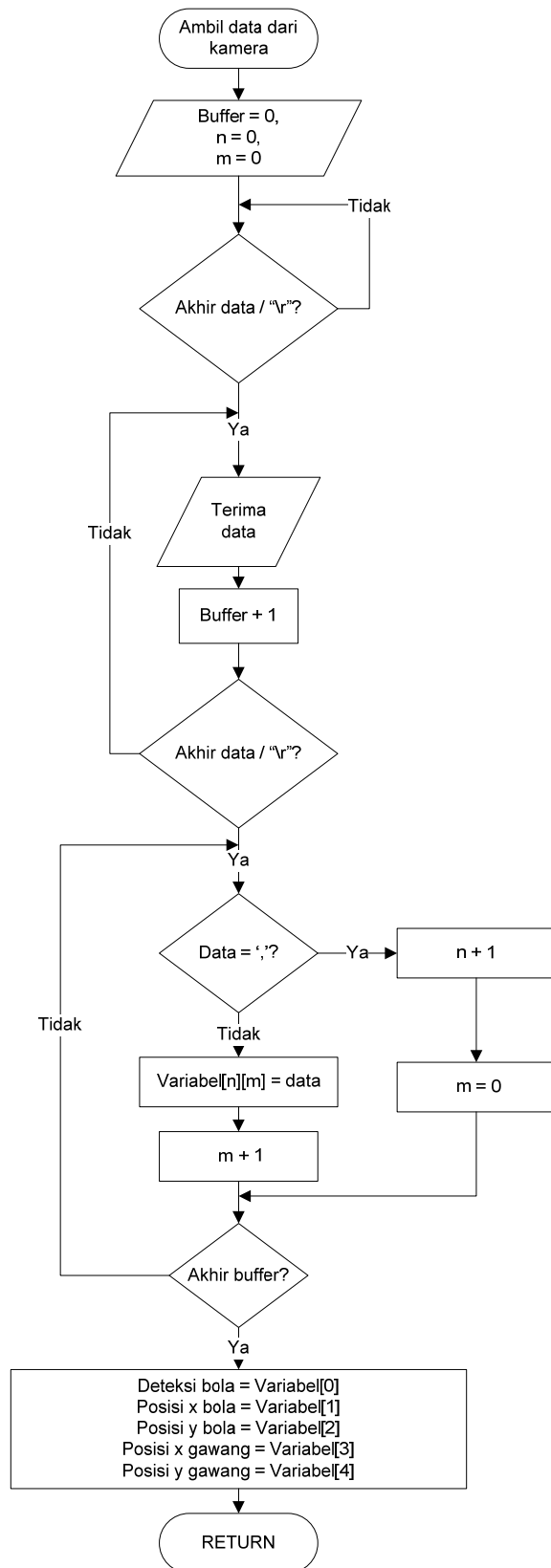
Tabel 3.2b Pergerakan Servo pada Gerakan Berjalan(2)

Servo Channel	Langkah 4		Langkah 5		Langkah 6	
	Posisi	Waktu (detik)	Posisi	Waktu (detik)	Posisi	Waktu (detik)
0	8.19°	1000	-6.3°	1000	-24°	1000
1	-23°	1000	-36°	1000	-36°	1000
2	-45.3°	1000	-25.2°	1000	-25.2°	1000
3	10.6°	1000	1.62°	1000	-4.68°	1000
4	-70.7°	1000	-60.8°	1000	-60.8°	1000
5	-18.1°	1000	-20°	1000	-18.2°	1000
6	73.7°	1000	73.7°	1000	73.7°	1000
7	-2.7°	1000	-2.7°	1000	-2.7°	1000
8	-45°	1000	-45°	1000	-45°	1000
16	12.9°	1000	-1.8°	1000	-16.3°	1000
17	29.2°	1000	36°	1000	12.8°	1000
18	29.6°	1000	25.2°	1000	63.8°	1000
19	8.64°	1000	3.6°	1000	-8.28°	1000
20	63.5°	1000	66.1°	1000	85.5°	1000
21	-20°	1000	-20°	1000	-20°	1000
22	5.04°	1000	5.04°	1000	5.04°	1000
23	4.5°	1000	4.5°	1000	4.5°	1000
24	59.6°	1000	59.6°	1000	59.6°	1000

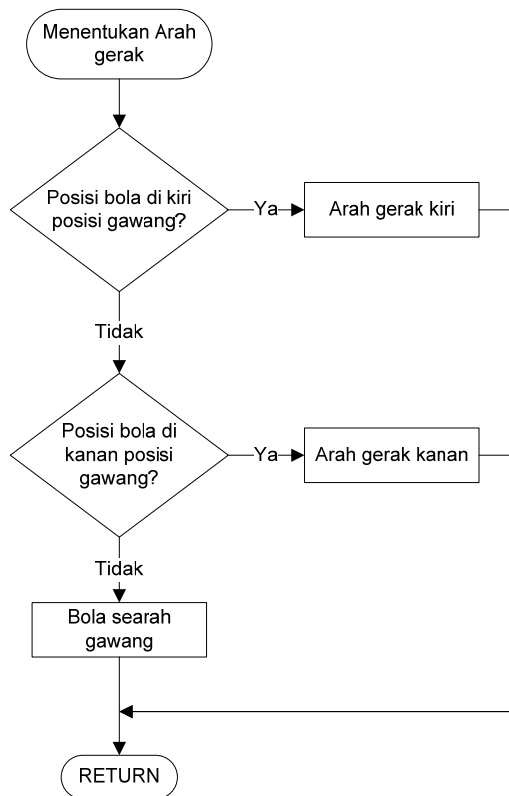
Pengontrol mikro ATMEGA128 juga setiap saat membaca nilai ADC untuk mengetahui kemiringan pada robot. Jika robot pada posisi jatuh maka pengontrol mikro akan mengirimkan perintah untuk bangkit berdiri pada pengontrol servo SSC-32. Diagram alir program pada pengontrol mikro ATMEGA128 ini dapat dilihat pada Gambar 3.12a sampai Gambar 3.12j.



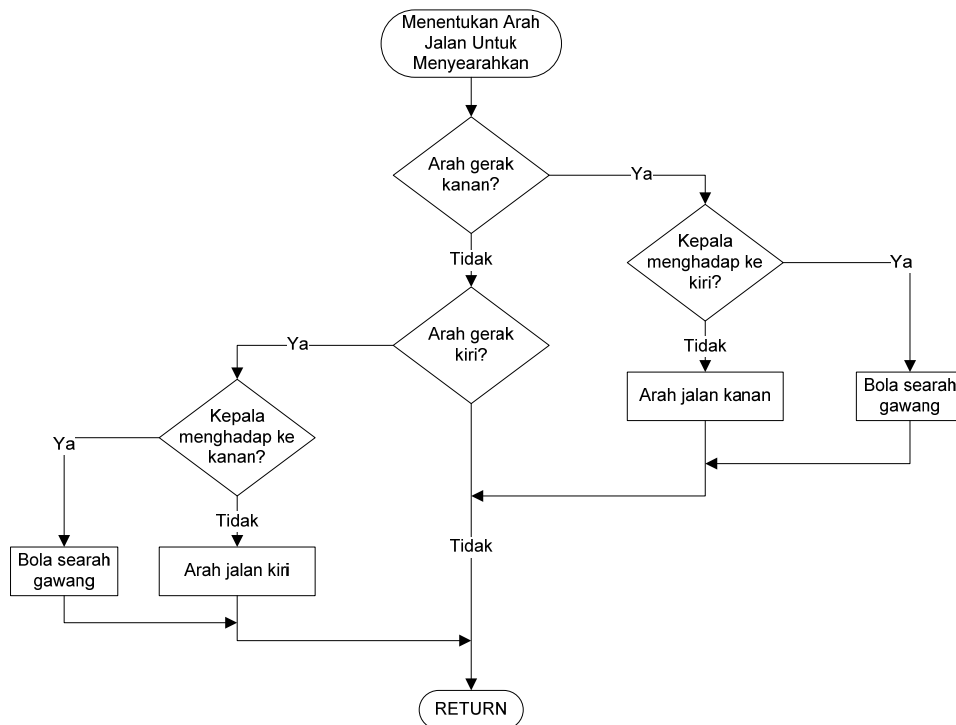
Gambar 3.12a Diagram Alir Program Utama Pengontrol Mikro ATMEGA128



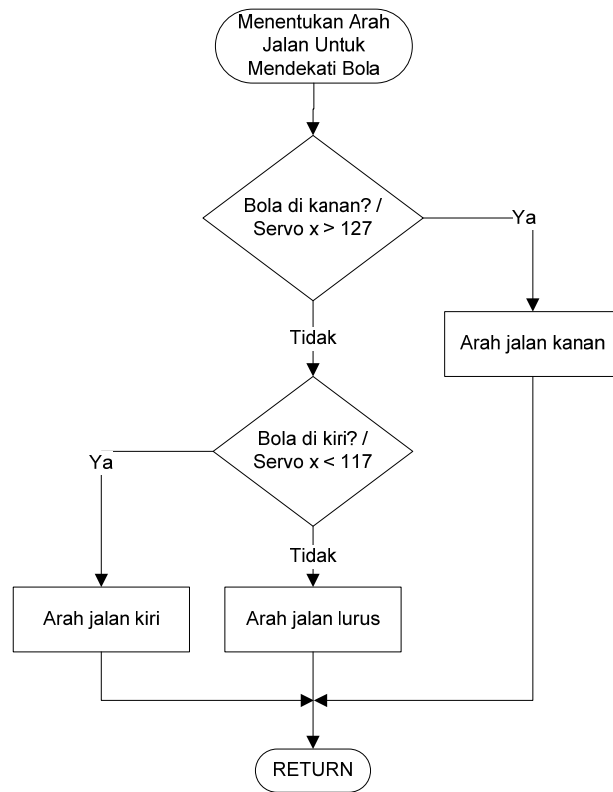
Gambar 3.12b Diagram Alir Subrutin Ambil Data Kamera



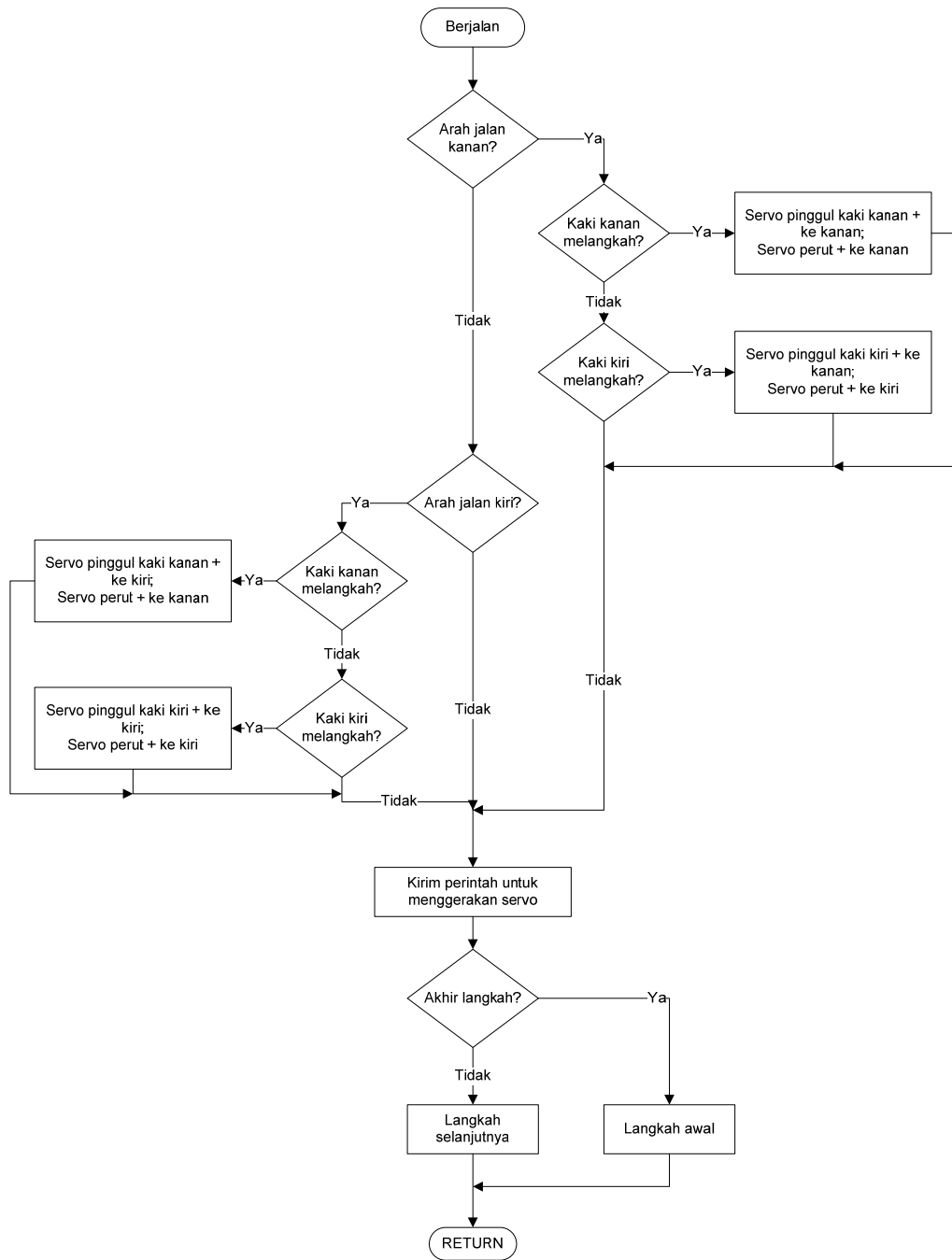
Gambar 3.12c Diagram Alir Subrutin Menentukan Arah Gerak



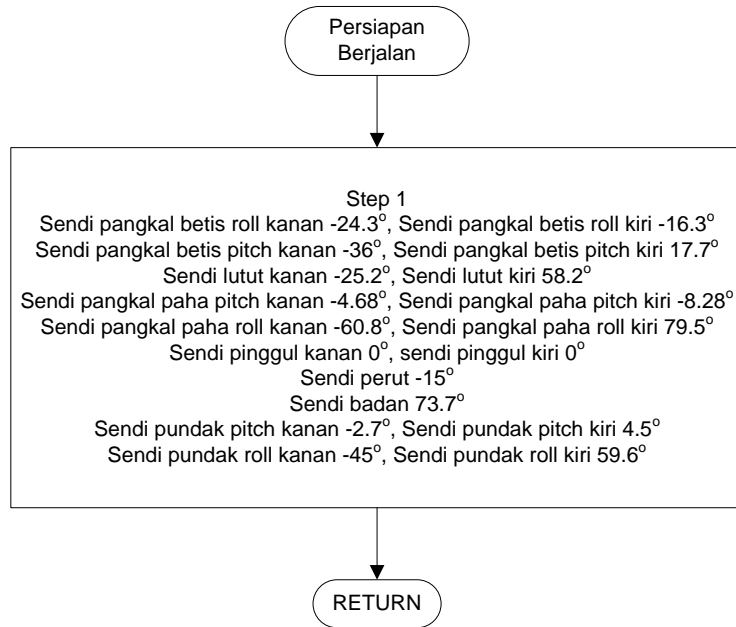
Gambar 3.12d Diagram Alir Subrutin Menentukan Arah Jalan untuk Menyearahkan



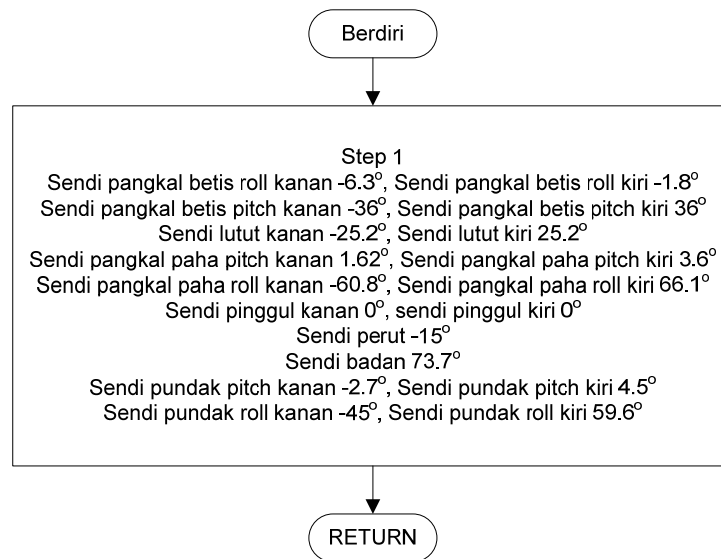
Gambar 3.12e Diagram Alir Subrutin Menentukan Arah Jalan untuk Mendekati Bola



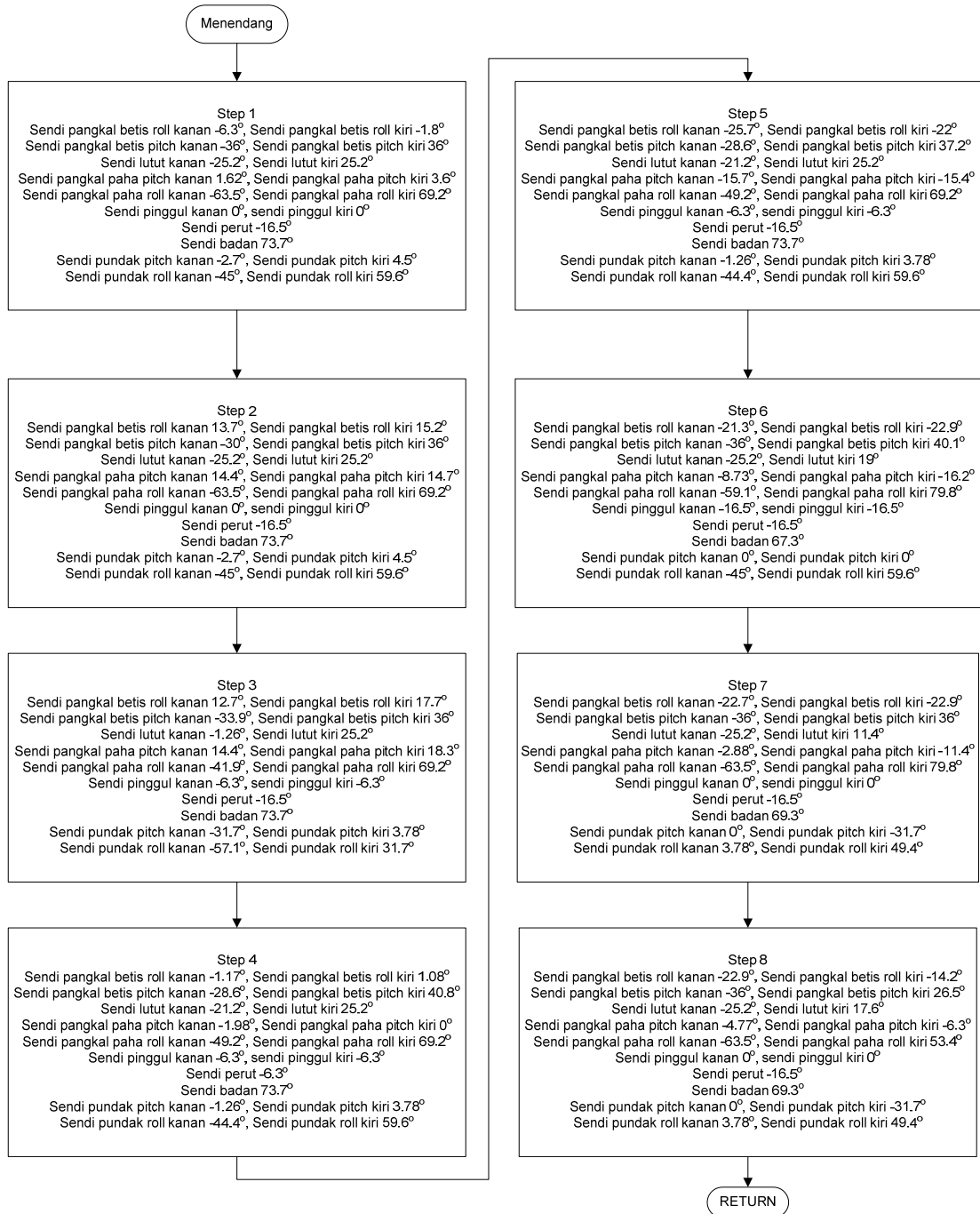
Gambar 3.12f Diagram Alir Subrutin Berjalan



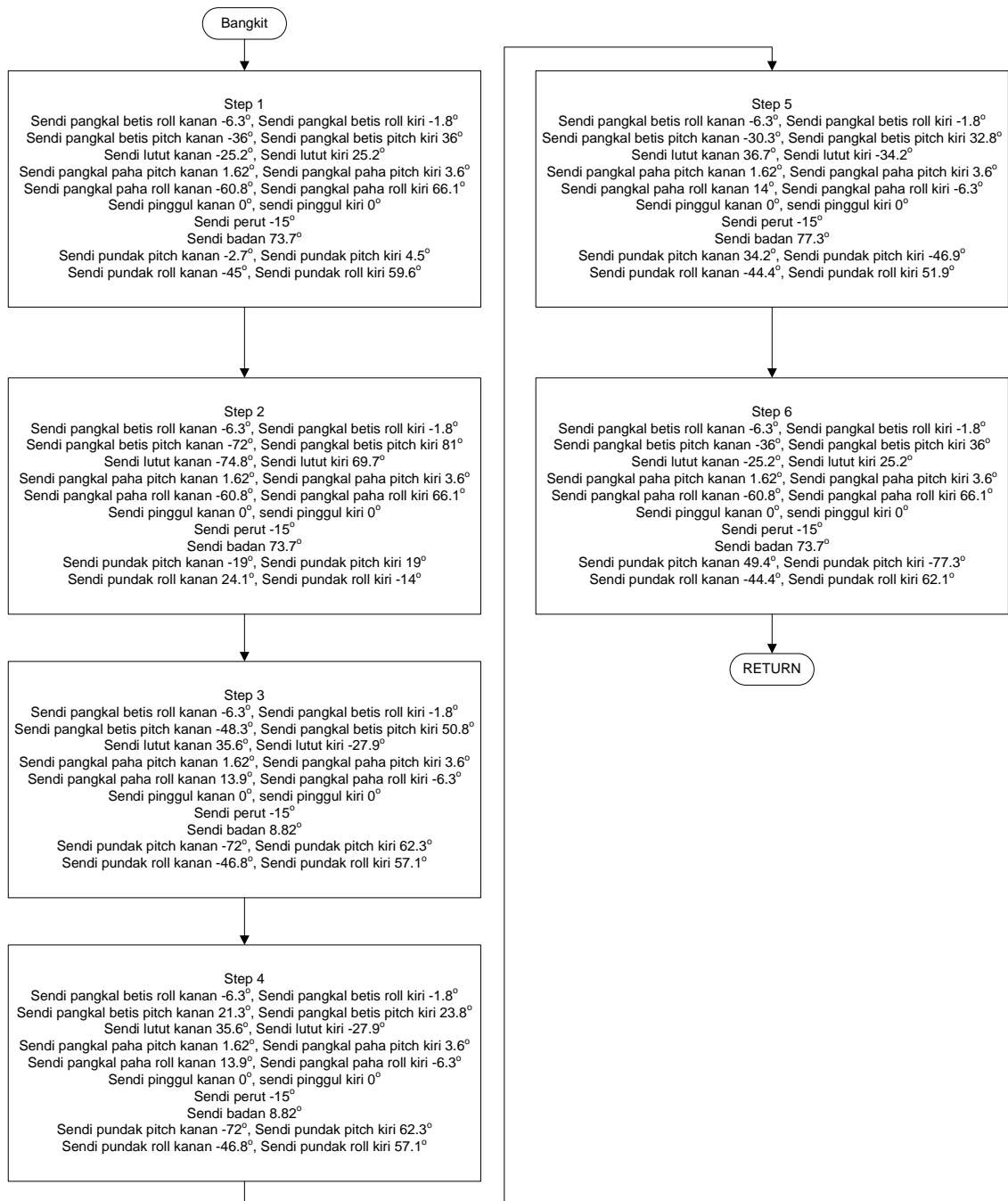
Gambar 3.12g Diagram Alir Subrutin Persiapan Berjalan



Gambar 3.12h Diagram Alir Subrutin Berdiri



Gambar 3.12i Diagram Alir Subrutin Menendang



Gambar 3.12j Diagram Alir Subrutin Bangkit

BAB IV

DATA PENGAMATAN DAN ANALISIS

Pada bab ini ditampilkan data - data hasil pengamatan dan analisa terhadap pengujian kinerja robot, pengujian kemampuan robot yang dapat berdiri ketika terjatuh, dan pengujian kamera dalam melacak warna.

IV.1 Kamera CMUCam3

IV.1.1 Pengujian Pengaturan Rentang Nilai RGB yang Diprogram pada CMUCam3

Rentang nilai RGB yang diprogram pada CMUCam3 untuk dapat melacak warna bola dan warna gawang dapat dilihat pada Tabel 4.1 dan Tabel 4.2.

Dibutuhkan percobaan untuk dapat mengetahui ketepatan sensor kamera dalam melacak warna – warna tertentu dengan rentang nilai tersebut. Percobaan yang dilakukan adalah dengan memberikan objek uji berupa kertas yang dicetak dengan nilai RGB tertentu yang diatur melalui komputer dan mengujikannya ke kamera CMUCam3. Nilai RGB di luar rentang nilai yang diprogram ditandai dengan warna merah. Hasil dari percobaan ini dapat dilihat pada Tabel 4.3 dan Tabel 4.4.

Tabel 4.1 Rentang Nilai RGB yang Diprogram pada CMUCam3 untuk Dapat Melacak Warna Bola

Keterangan	Nilai
Nilai minimal merah [Rmin]	215
Nilai maksimal merah [Rmax]	255
Nilai minimal hijau [Gmin]	105
Nilai maksimal hijau [Gmax]	155
Nilai minimal biru [Bmin]	0
Nilai maksimal biru [Bmax]	44

Tabel 4.2 Rentang Nilai RGB yang Diprogram pada CMUCam3 untuk Dapat Melacak Warna Gawang

Keterangan	Nilai
Nilai minimal merah [Rmin]	171
Nilai maksimal merah [Rmax]	221
Nilai minimal hijau [Gmin]	215
Nilai maksimal hijau [Gmax]	255
Nilai minimal biru [Bmin]	0
Nilai maksimal biru [Bmax]	41

Tabel 4.3 Hasil Pengujian Keberhasilan Sensor Terhadap Rentang Nilai RGB yang Diprogram untuk Dapat Melacak Bola

No	Nilai RGB objek uji [R,G,B]	Terdeteksi
1	[215,105,0]	Ya
2	[215,105,25]	Ya
3	[215,105,50]	Tidak
4	[215,130,0]	Ya
5	[215,130,25]	Ya
6	[215,130,50]	Tidak
7	[215,155,0]	Ya
8	[215,155,25]	Ya
9	[215,155,50]	Tidak
10	[230,105,0]	Ya
11	[230,105,25]	Ya
12	[230,105,50]	Tidak
13	[230,130,0]	Ya
14	[230,130,25]	Ya
15	[230,130,50]	Tidak
16	[230,155,0]	Ya

Lanjutan Tabel 4.3

No	Nilai RGB objek uji [R,G,B]	Terdeteksi
17	[230,155,25]	Ya
18	[230,155,50]	Tidak
19	[255,105,0]	Ya
20	[255,105,25]	Ya
21	[255,105,50]	Tidak
22	[255,130,0]	Ya
23	[255,130,25]	Ya
24	[255,130,50]	Tidak
25	[255,155,0]	Ya
26	[255,155,25]	Ya
27	[255,155,50]	Tidak

Dari data pada Tabel 4.3 dapat diketahui bahwa sensor kamera ini dapat melacak nilai – nilai RGB yang berada dalam rentang nilai yang terprogram dan tidak dapat melacak nilai – nilai RGB diluar rentang nilai tersebut. Tingkat keberhasilan sensor kamera dalam melacak nilai RGB dalam rentang ini adalah 100%.

Tabel 4.4 Hasil Pengujian Keberhasilan Sensor Terhadap Rentang Nilai RGB yang Diprogram untuk Dapat Melacak Gawang

No	Nilai RGB objek uji [R,G,B]	Terdeteksi
1	[171,215,0]	Ya
2	[171,215,25]	Ya
3	[171,215,50]	Tidak
4	[171,230,0]	Ya
5	[171,230,25]	Ya
6	[171,230,50]	Tidak
7	[171,255,0]	Ya
8	[171,255,25]	Ya

Lanjutan Tabel 4.4

No	Nilai RGB objek uji [R,G,B]	Terdeteksi
9	[171,255,50]	Tidak
10	[196,215,0]	Ya
11	[196,215,25]	Ya
12	[196,215,50]	Tidak
13	[196,230,0]	Ya
14	[196,230,25]	Ya
15	[196,230,50]	Tidak
16	[196,255,0]	Ya
17	[196,255,25]	Ya
18	[196,255,50]	Tidak
19	[221,215,0]	Ya
20	[221,215,25]	Ya
21	[221,215,50]	Tidak
22	[221,230,0]	Ya
23	[221,230,25]	Ya
24	[221,230,50]	Tidak
25	[221,255,0]	Ya
26	[221,255,25]	Ya
27	[221,255,50]	Tidak

Dari data pada Tabel 4.4 dapat diketahui bahwa sensor kamera ini dapat melacak nilai – nilai RGB yang berada dalam rentang nilai yang terprogram dan tidak dapat melacak nilai – nilai RGB diluar rentang nilai tersebut. Tingkat keberhasilan sensor kamera dalam melacak nilai RGB dalam rentang ini adalah 100%.

IV.1.2 Pengujian Posisi Servo Leher Terhadap Posisi Bola

Pengujian ini dilakukan dengan maksud untuk mengetahui nilai – nilai posisi pada servo leher terhadap posisi bola yang sebenarnya dan untuk mengetahui batasan

jarak yang masih dapat dilacak oleh sensor kamera. Percobaan yang dilakukan adalah dengan meletakkan robot di koordinat[x,y] [0,0] dalam posisi berdiri dan menggeser bola pada sumbu x dan y dengan interval jarak 20 cm. Hasil pengujian ini dapat dilihat pada Tabel 4.5a sampai Tabel 4.5g.

Tabel 4.5a Hasil Pengujian Posisi Servo Leher Terhadap Posisi Bola di Sumbu y=20 cm

Koordinat	Servo leher	
	x	y
x (cm)		
40	245	137
20	216	168
0	117	187
-20	16	170
-40	15	142
-60	15	119

Dari data pengamatan pada Tabel 4.5a dapat dilihat untuk posisi bola pada sumbu y = 20 cm, batas pergeseran bola pada sumbu x yang masih dapat dilacak oleh sensor kamera adalah dari -60 cm sampai dengan 40 cm. Pada sumbu x = -60 cm, kamera masih dapat melacak warna bola tetapi tidak berada pada titik centroid kamera.

Tabel 4.5b Hasil Pengujian Posisi Servo Leher Terhadap Posisi Bola di Sumbu y=40 cm

Koordinat	Servo leher	
	x	y
x (cm)		
60	245	110
40	228	119
20	178	137
0	121	142
-20	59	133
-40	15	118

Lanjutan Tabel 4.5b

Koordinat	Servo leher	
	x	y
x (cm)	x	y
-60	15	110
-80	15	110

Dari data pengamatan pada Tabel 4.5b dapat dilihat untuk posisi bola pada sumbu $y = 40$ cm, batas pergeseran bola pada sumbu x yang masih dapat dilacak oleh sensor kamera adalah dari -80 cm sampai dengan 60 cm.

Tabel 4.5c Hasil Pengujian Posisi Servo Leher Terhadap Posisi Bola di Sumbu $y=60$ cm

Koordinat	Servo leher	
	x	y
x (cm)	x	y
100	245	110
80	245	110
60	222	110
40	198	110
20	161	110
0	126	110
-20	82	110
-40	40	110
-60	15	110
-80	15	110
-100	15	110
-120	15	110

Dari data pengamatan pada Tabel 4.5c dapat dilihat untuk posisi bola pada sumbu $y = 60$ cm, batas pergeseran bola pada sumbu x yang masih dapat dilacak oleh sensor kamera adalah dari -120 cm sampai dengan 100 cm. Pada sumbu $x = 100$ cm,

$x = -80$ cm sampai $x = -120$ cm, kamera masih dapat melacak warna bola tetapi tidak berada pada titik centroid kamera.

Tabel 4.5d Hasil Pengujian Posisi Servo Leher Terhadap Posisi Bola di Sumbu $y=80$ cm

Koordinat	Servo leher	
	x	y
x (cm)		
120	245	110
100	245	110
80	240	110
60	215	110
40	195	110
20	163	110
0	129	112
-20	92	110
-40	63	110
-60	35	110
-80	15	110
-100	15	110
-120	15	110
-140	15	110

Dari data pengamatan pada Tabel 4.5d dapat dilihat untuk posisi bola pada sumbu $y = 80$ cm, batas pergeseran bola pada sumbu x yang masih dapat dilacak oleh sensor kamera adalah dari -140 cm sampai dengan 120 cm. Pada sumbu $x = 120$ cm, $x = -100$ cm sampai $x = -140$ cm, kamera masih dapat melacak warna bola tetapi tidak berada pada titik centroid kamera.

Tabel 4.5e Hasil Pengujian Posisi Servo Leher Terhadap Posisi Bola di Sumbu $y=100$
cm

Koordinat	Servo leher	
	x	y
x (cm)		
120	245	110
100	245	110
80	225	110
60	194	110
40	171	110
20	152	110
0	120	110
-20	93	110
-40	68	110
-60	47	110
-80	25	110
-100	15	110
-120	15	110
-140	15	110

Dari data pengamatan pada Tabel 4.5e dapat dilihat untuk posisi bola pada sumbu $y = 100$ cm, batas pergeseran bola pada sumbu x yang masih dapat dilacak oleh sensor kamera adalah dari -140 cm sampai dengan 120 cm. Pada sumbu $x = 120$ cm, $x = -120$ cm sampai $x = -140$ cm, kamera masih dapat melacak warna bola tetapi tidak berada pada titik centroid kamera.

Tabel 4.5f Hasil Pengujian Posisi Servo Leher Terhadap Posisi Bola di Sumbu $y=120$ cm

Koordinat	Servo leher	
	x	y
x (cm)		
140	245	110
120	245	110
100	220	110
80	195	110
60	182	110
40	163	110
20	149	110
0	127	110
-20	108	110
-40	88	110
-60	69	110
-80	49	110
-100	35	110
-120	22	110
-140	15	110
-160	15	110

Dari data pengamatan pada Tabel 4.5f dapat dilihat untuk posisi bola pada sumbu $y = 120$ cm, batas pergeseran bola pada sumbu x yang masih dapat dilacak oleh sensor kamera adalah dari -160 cm sampai dengan 140 cm. Pada sumbu $x = 140$ cm, $x = -160$ cm, kamera masih dapat melacak warna bola tetapi tidak berada pada titik centroid kamera.

Dari data keseluruhan posisi servo terhadap posisi bola ini dapat diketahui bahwa bola dianggap berada lurus di depan robot apabila nilai posisi servo antara 117 sampai dengan 127 , bola berada di sebelah kiri robot apabila nilai posisi servo kurang

dari 117 dan bola berada di sebelah kanan robot apabila nilai posisi servo lebih dari 127. Nilai – nilai ini nantinya digunakan dalam program robot untuk menentukan arah jalan. Sensor kamera sudah tidak dapat melacak bola dengan baik jika posisi bola melebihi 120 cm terhadap robot.

IV.1.3 Pengujian Posisi Servo Leher Terhadap Posisi Gawang

Pengujian ini dilakukan dengan maksud untuk mengetahui nilai – nilai posisi pada servo leher terhadap posisi gawang yang sebenarnya dan untuk mengetahui batasan jarak yang masih dapat dilacak oleh sensor kamera. Percobaan yang dilakukan adalah dengan menggeser robot searah sumbu x di sumbu y=0. Gawang berada sejauh 200 cm terhadap robot. Data hasil percobaan ini dapat dilihat pada Tabel 4.6.

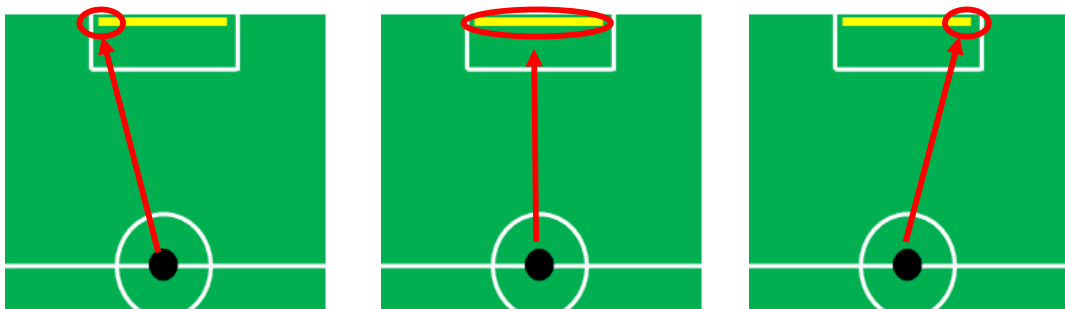
Tabel 4.6 Hasil Pengujian Posisi Gawang Terhadap Posisi Robot

Koordinat	Servo leher	
	x	y
x (cm)		
-200	207	110
-180	182	110
-160	168	110
-140	148	110
-120	136	110
-100	129	110
-80	125	110
-60	131	110
-40	126	110
-20	120	110
0	117	110
20	115	110
40	123	110
60	120	110

Lanjutan Tabel 4.6

Koordinat	Servo leher	
	x	y
x (cm)	x	y
80	119	110
100	104	110
120	93	110
140	85	110
160	77	110
180	64	110
200	59	110

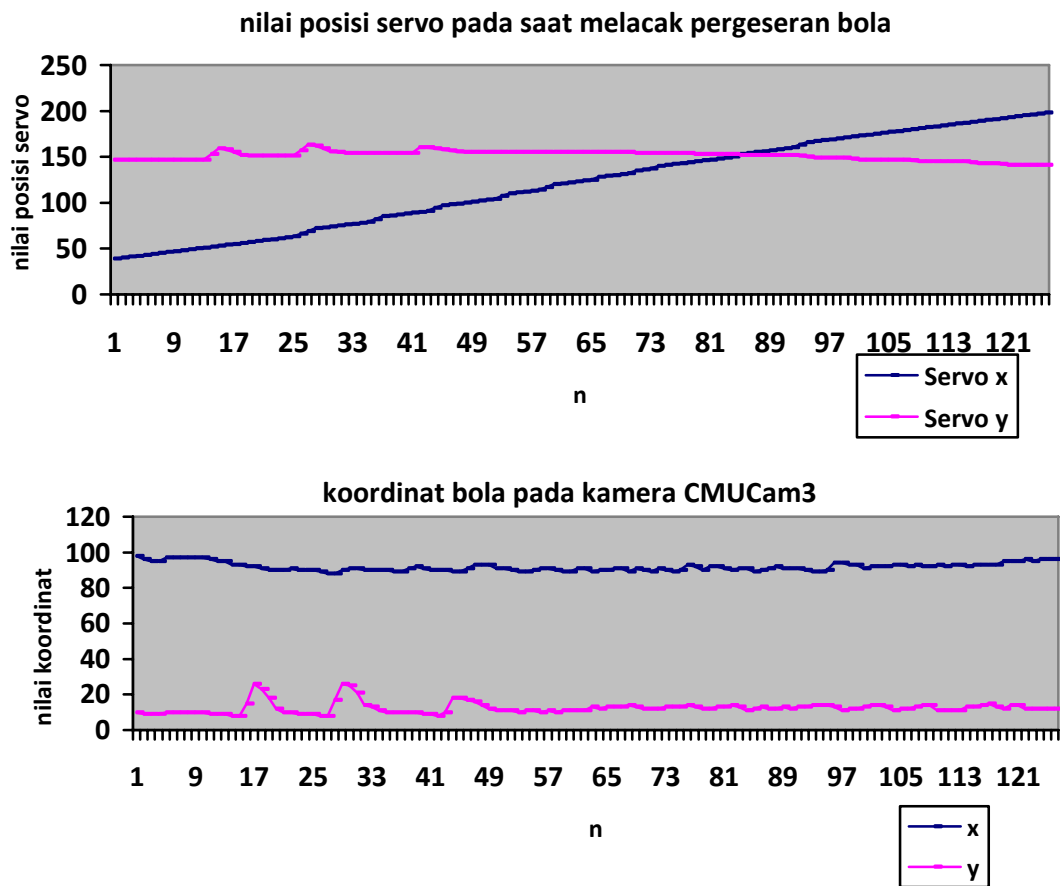
Dari data pengamatan pada Tabel 4.6 diketahui bahwa robot berada pada sisi sebelah kanan gawang bila nilai posisi servo relatif besar, sedangkan robot berada pada sisi sebelah kiri gawang bila nilai posisi servo relatif kecil. Pada saat robot berada di koordinat -60 cm sampai 60 cm, nilai yang didapat tidak pasti dikarenakan gawang memiliki dua tiang di masing – masing tepinya sehingga terdapat tiga kemungkinan posisi yang dilacak oleh sensor kamera yaitu sisi kiri gawang, sisi kanan gawang, atau kedua sisi gawang. Kemungkinan – kemungkinan tersebut dapat dilihat pada Gambar 4.1.



Gambar 4.1 Kemungkinan Posisi Gawang yang Dapat Dilacak Kamera Ketika Robot Berada pada Koordinat -60 cm Sampai 60 cm

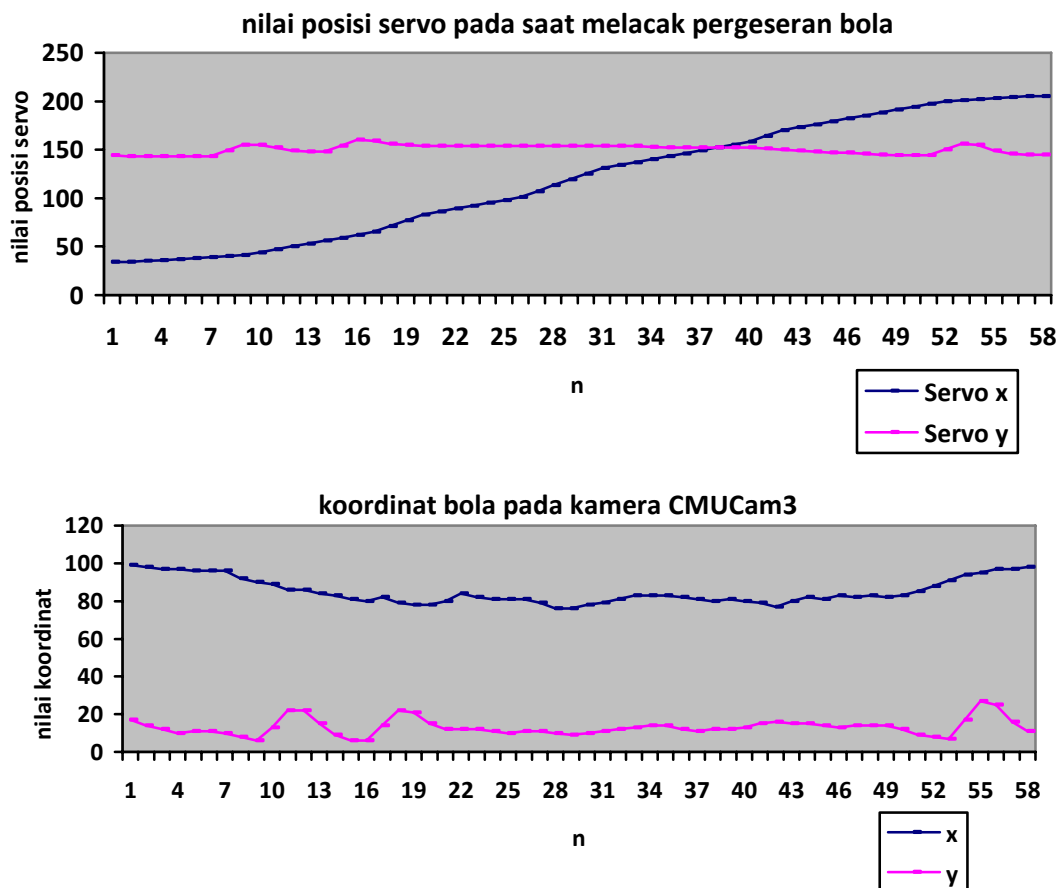
IV.1.4 Pengujian Pelacakan Pada Bola yang Bergerak

Pengujian pelacakan dilakukan pada bola yang bergerak untuk mengetahui batasan kemampuan algoritma pelacakan dalam mengikuti pergeseran dari bola. Percobaan yang dilakukan adalah dengan menggerakkan bola sejauh 60 cm dengan berbagai kecepatan di sumbu x dan sumbu y. *setpoint* letak target dalam koordinat kamera yang diprogram pada CMUCam3 berada pada nilai $[x,y] = [100,11]$. Data yang didapat adalah nilai posisi servo pada saat melacak pergeseran bola dan koordinat bola yang terdeteksi pada kamera CMUCam3. Data - data tersebut dapat dilihat pada Gambar 4.2a sampai Gambar 4.2h.



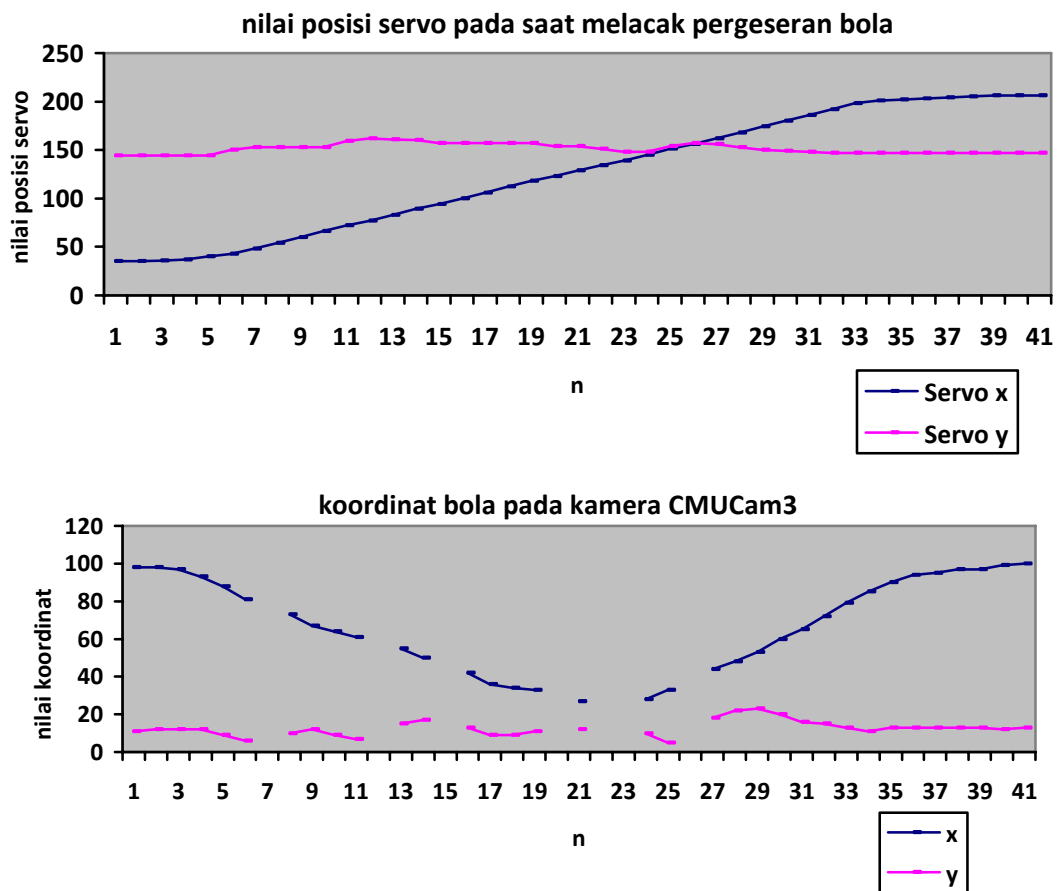
Gambar 4.2a Grafik Pelacakan Pergeseran Bola dengan Kecepatan 2 cm/det di Sumbu x

Dari data pada Gambar 4.2a, dapat dilihat nilai posisi servo x mengikuti pergeseeran bola yang bergerak di sumbu x sedangkan nilai posisi servo y relatif tetap. pergeseeran nilai posisi servo ini menjaga nilai *setpoint* koordinat bola pada kamera agar tetap berada pada nilai [100,11].



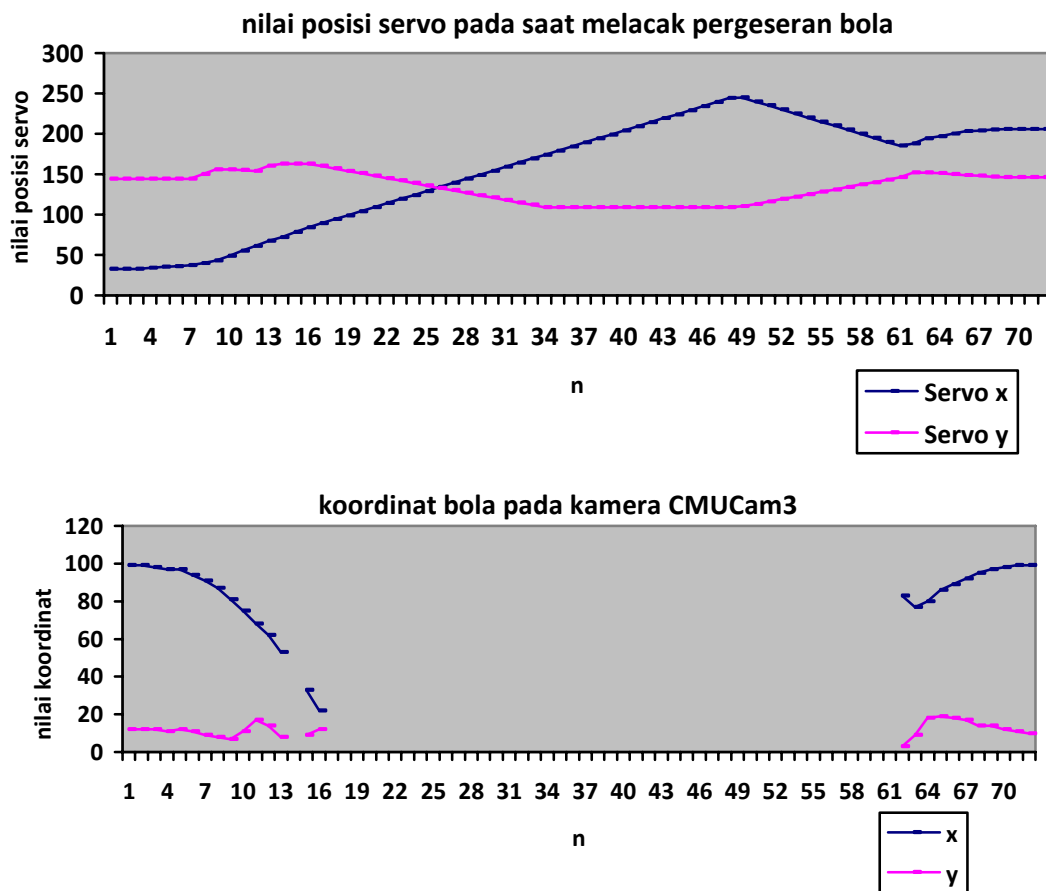
Gambar 4.2b Grafik Pelacakan Pergeseeran Bola dengan Kecepatan 6 cm/det di sumbu x

Pada Gambar 4.2b untuk pergeseeran bola dengan kecepatan 6 cm/det, pergeseeran bola masih dapat diikuti oleh servo tetapi nilai koordinat bola pada kamera tidak dapat dipertahankan pada *setpoint*.



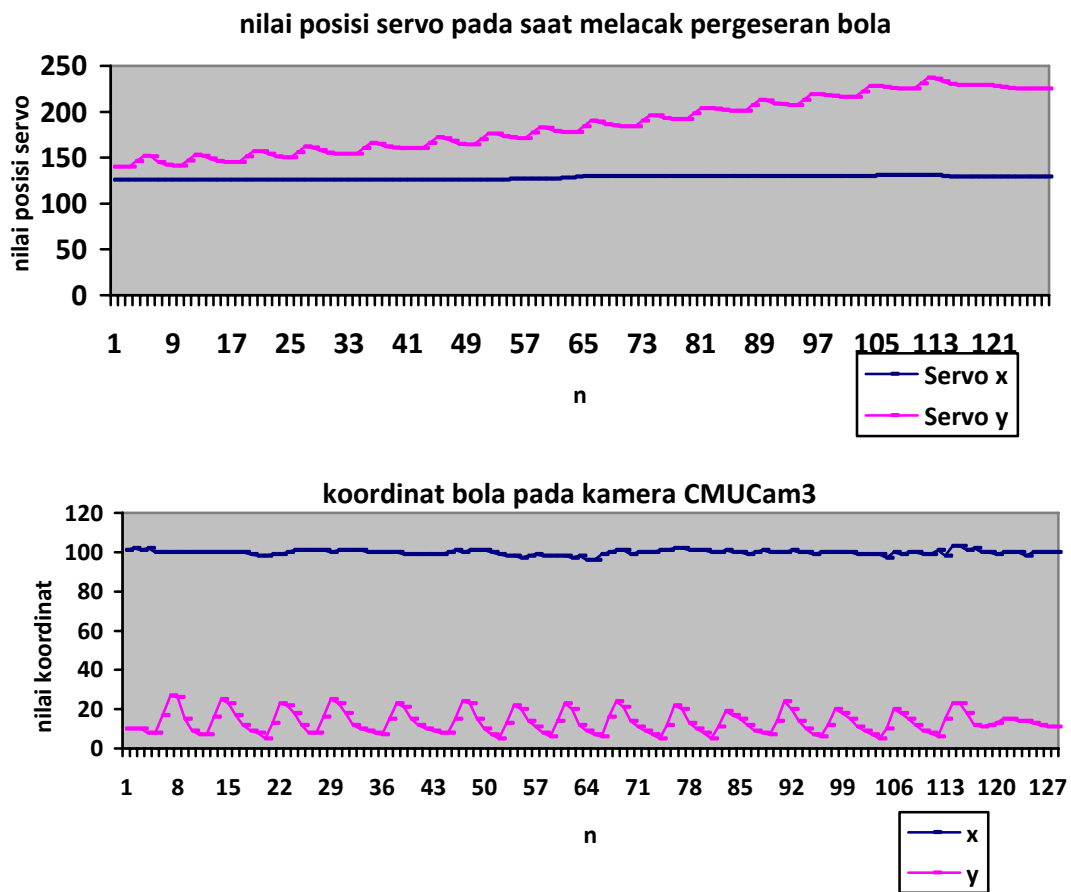
Gambar 4.2c Grafik Pelacakan Pergeseran Bola dengan Kecepatan 10 cm/det di sumbu x

Pada pergeseran bola dengan kecepatan 10 cm/det, bola terkadang lepas dari pandangan kamera tetapi masih dapat dengan cepat didapatkan kembali.



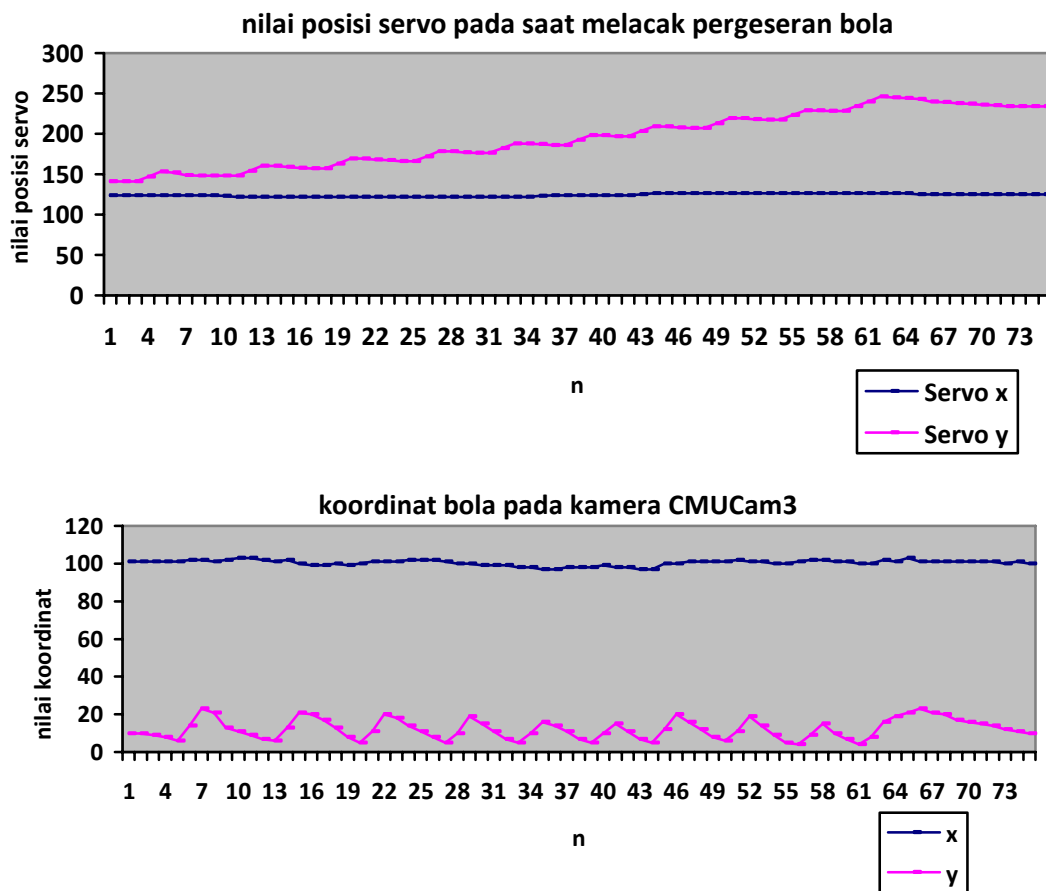
Gambar 4.2d Grafik Pelacakan Pergeseran Bola dengan Kecepatan 20 cm/det di sumbu x

Pada Gambar 4.2d dapat dilihat untuk pergeseran 20 cm/det servo sudah tidak dapat mengikuti pergeseran bola sehingga bola lepas dari pandangan kamera. Pada saat bola lepas dari kamera maka robot akan melakukan proses *scanning* untuk menemukan bola kembali.



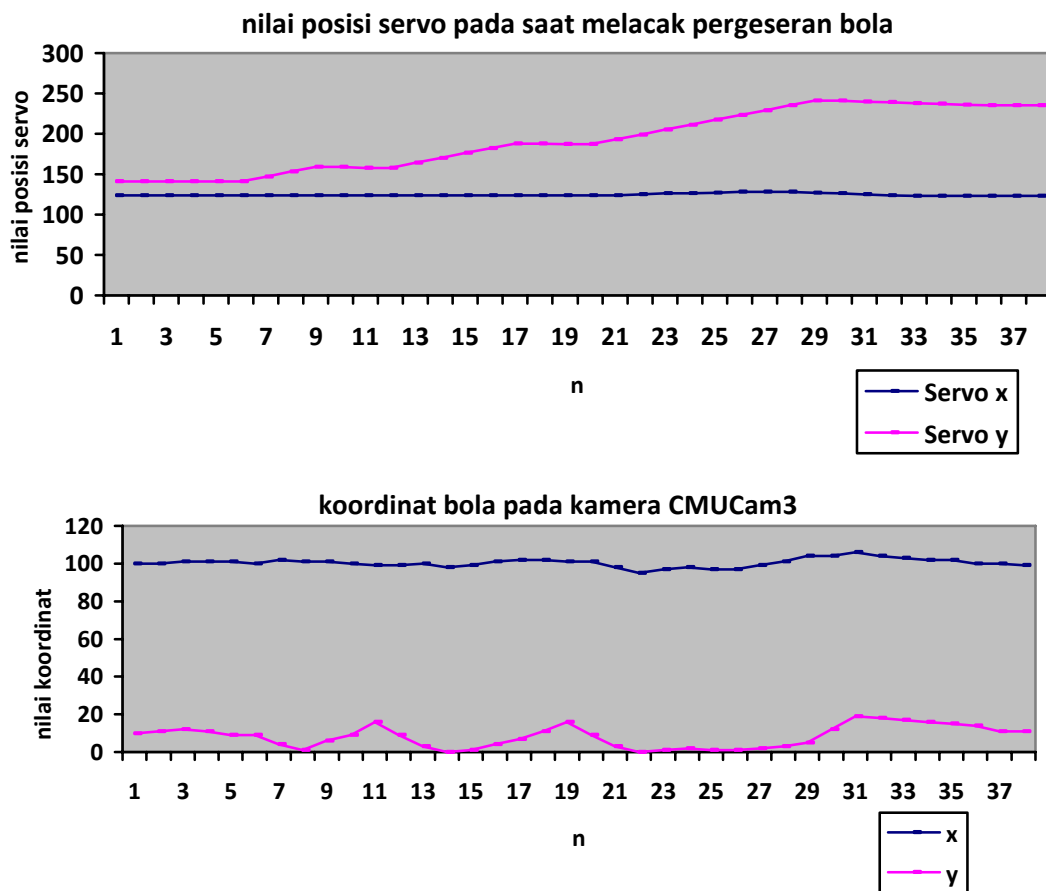
Gambar 4.2e Grafik Pelacakan Pergeseran Bola dengan Kecepatan 2 cm/det di sumbu y

Dari data pada Gambar 4.2e, dapat dilihat nilai posisi servo y mengikuti pergeseran bola yang bergerak di sumbu y sedangkan nilai posisi servo x relatif tetap. pergeseran nilai posisi servo ini menjaga nilai *setpoint* koordinat bola pada kamera agar tetap berada pada nilai [100,11].



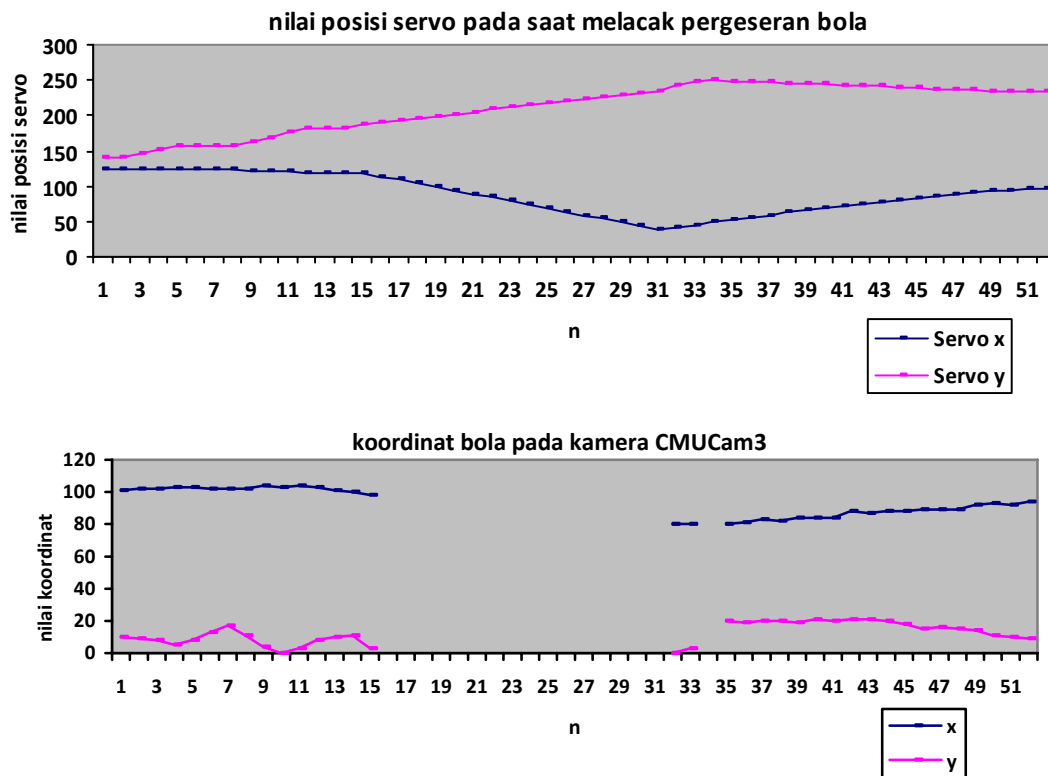
Gambar 4.2f Grafik Pelacakan Pergeseran Bola dengan Kecepatan 6 cm/det di sumbu y

Pada Gambar 4.2f untuk pergeseran bola dengan kecepatan 6 cm/det, pergeseran bola masih dapat diikuti oleh servo.



Gambar 4.2g Grafik Pelacakan Pergeseran Bola dengan Kecepatan 10 cm/det di sumbu y

Pada Gambar 4.2g untuk pergeseran bola dengan kecepatan 10 cm/det, pergeseran bola masih dapat diikuti oleh servo.



Gambar 4.2h Grafik Pelacakan Pergeseran Bola dengan Kecepatan 20 cm/det di sumbu y

Pada Gambar 4.2h dapat dilihat untuk pergeseran 20 cm/det servo sudah tidak dapat mengikuti pergeseran bola sehingga bola lepas dari pandangan kamera. Pada saat bola lepas dari kamera maka robot akan melakukan proses *scanning* untuk menemukan bola kembali.

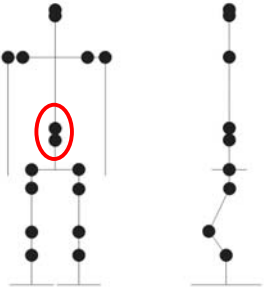
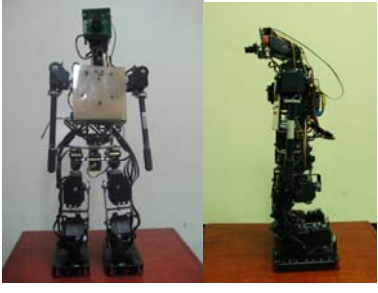
IV.2 Robot Humanoid

IV.2.1 Pengamatan Terhadap Gerakan Robot

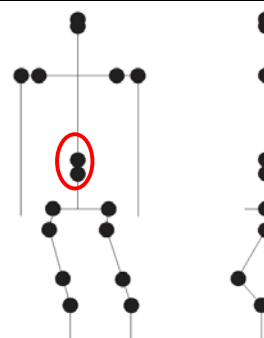
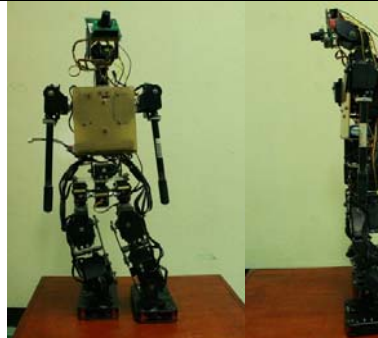
Pengamatan setiap gerakan pada robot dilakukan untuk mengetahui kestabilan dalam setiap gerakan pada robot. Robot humanoid pemain bola ini memiliki 5 gerakan yaitu gerakan berdiri, persiapan jalan, berjalan, menendang dan bangkit berdiri. *Center of Gravity* robot berada pada bagian servo badan yang ditunjukkan

oleh lingkaran berwarna merah. Tabel 4.7 sampai dengan Tabel 4.11 menampilkan gerakan robot setiap langkah nya dalam bentuk sketsa dan gambar nyata.

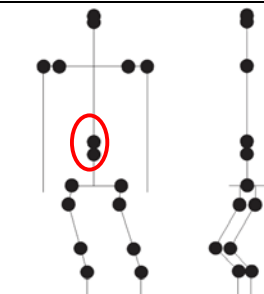
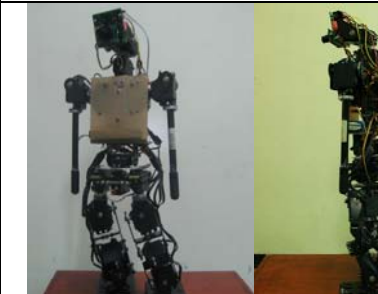
Tabel 4.7 Gerakan Berdiri

Sketsa	Gambar nyata	Keterangan
		Posisi berdiri robot

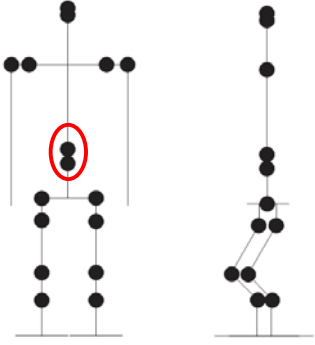
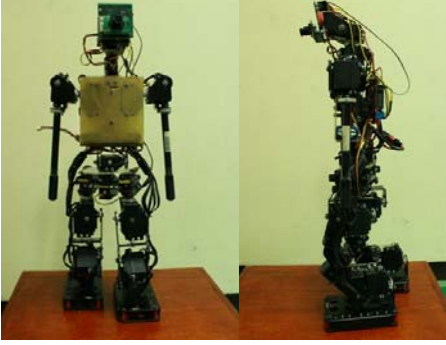
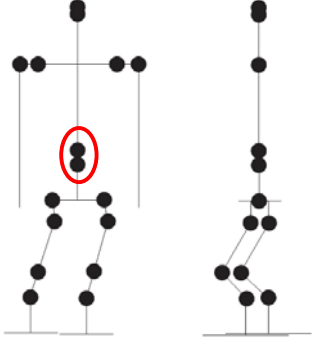
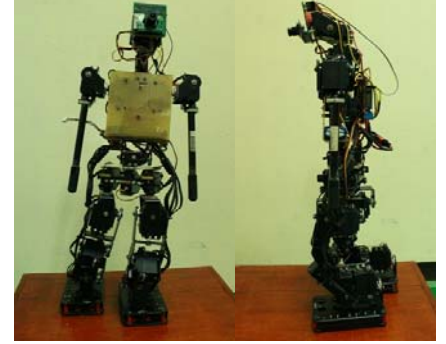
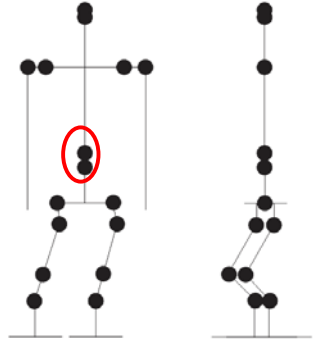
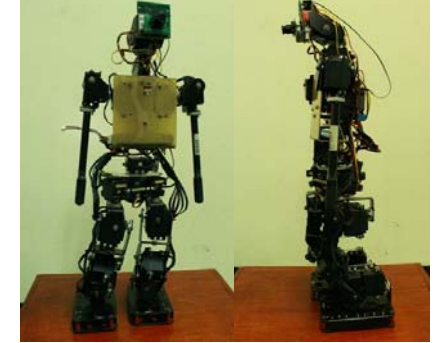
Tabel 4.8 Gerakan Persiapan Berjalan

Sketsa	Gambar nyata	Keterangan
		Gerakan yang dilakukan pada transisi antara robot melakukan gerakan berjalan dan gerakan berdiri

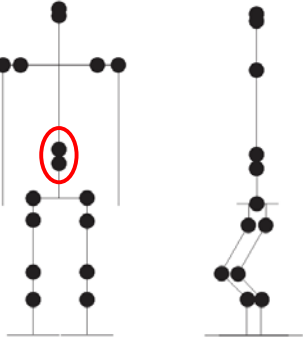
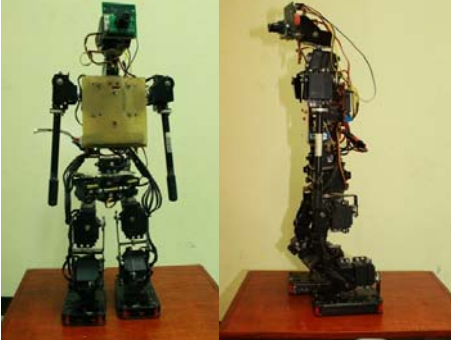
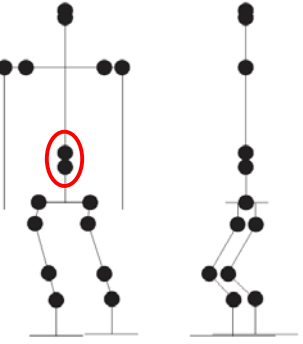
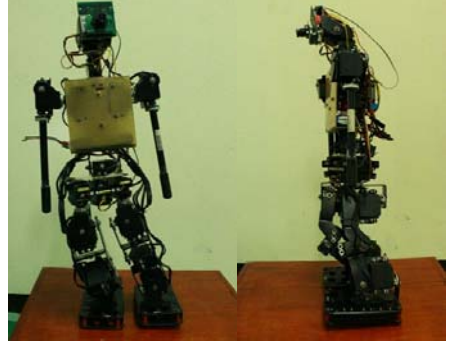
Tabel 4.9 Gerakan Berjalan

Sketsa	Gambar nyata	Keterangan
		COG bertumpu pada kaki kanan, kaki kiri melangkah

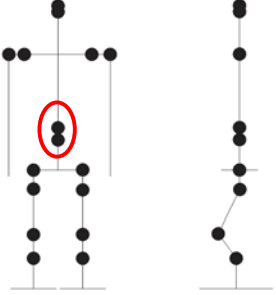
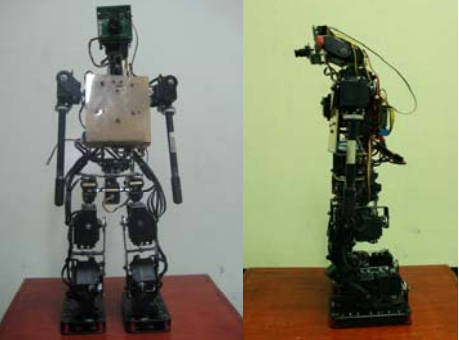
Lanjutan Tabel 4.9

Sketsa	Gambar nyata	Keterangan
		<p>COG bertumpu pada kedua kaki</p>
		<p>COG bertumpu pada kaki kiri</p>
		<p>COG bertumpu pada kaki kiri, kaki kanan melangkah</p>

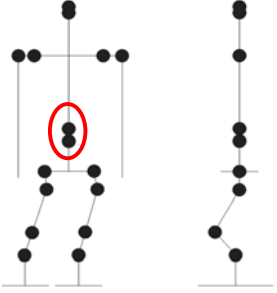
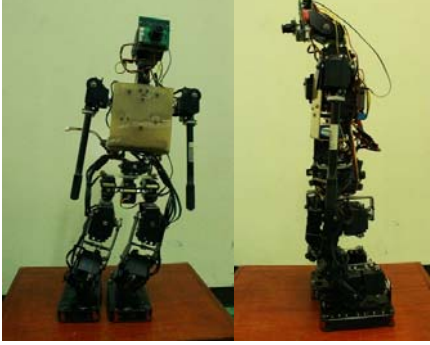
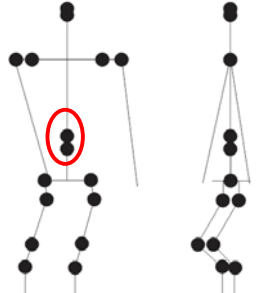
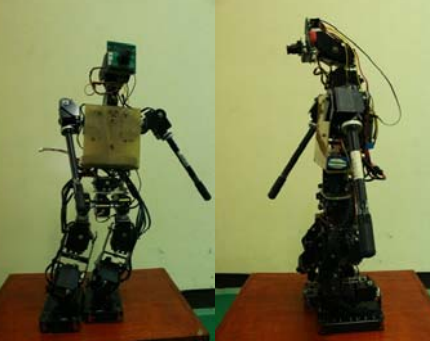
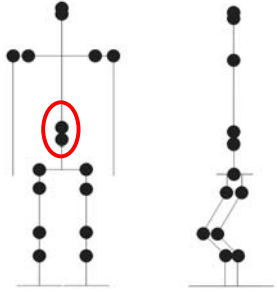
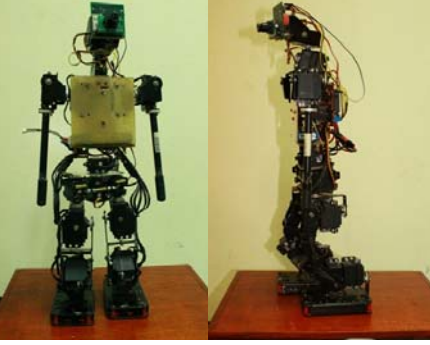
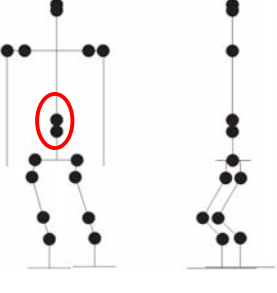
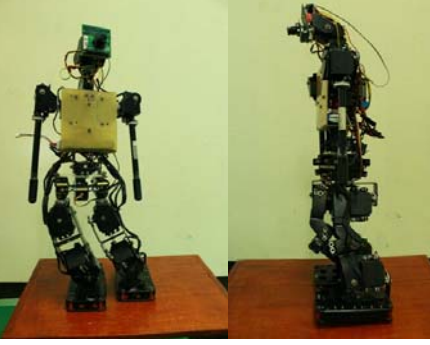
Lanjutan Tabel 4.9

Sketsa	Gambar nyata	Keterangan
		<p>COG bertumpu pada kedua kaki</p>
		<p>COG bertumpu pada kaki kanan</p>

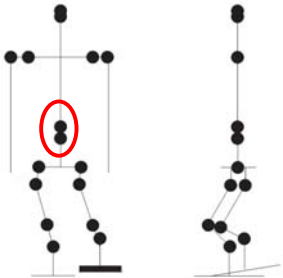
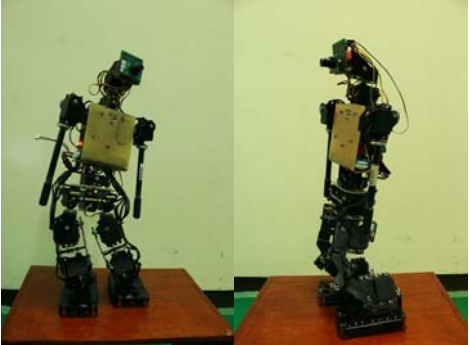
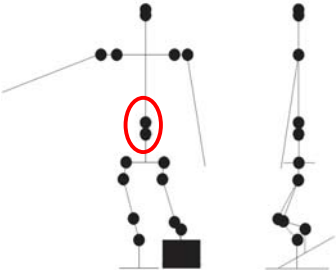
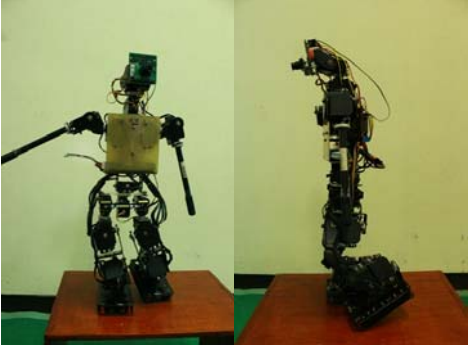
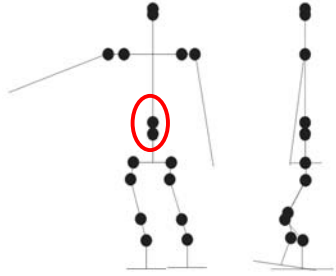
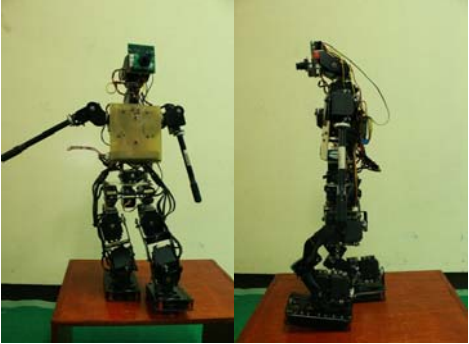
Tabel 4.10 Gerakan Menendang

Sketsa	Gambar nyata	Keterangan
		<p>Posisi berdiri</p>

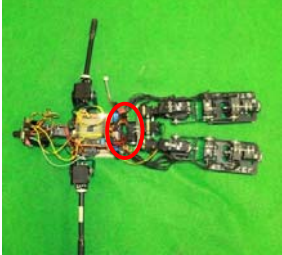
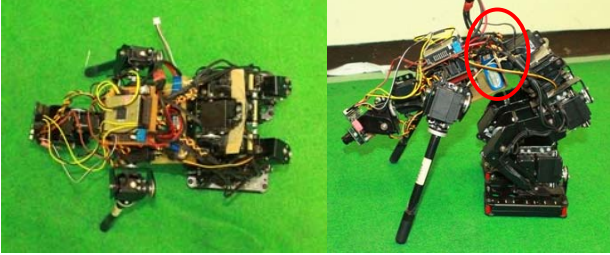
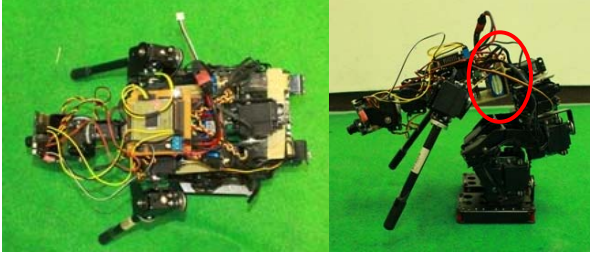
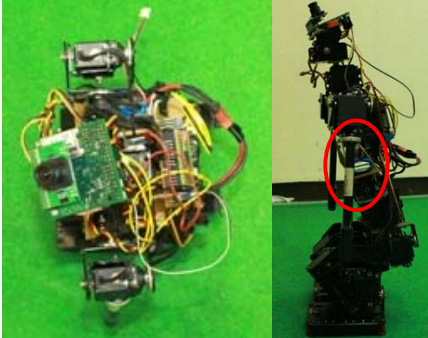
Lanjutan Tabel 4.10

Sketsa	Gambar nyata	Keterangan
		<p>COG bertumpu pada kaki kiri</p>
		<p>COG bertumpu pada kaki kiri, kaki kanan melangkah</p>
		<p>COG bertumpu pada kedua kaki</p>
		<p>COG bertumpu pada kaki kanan</p>

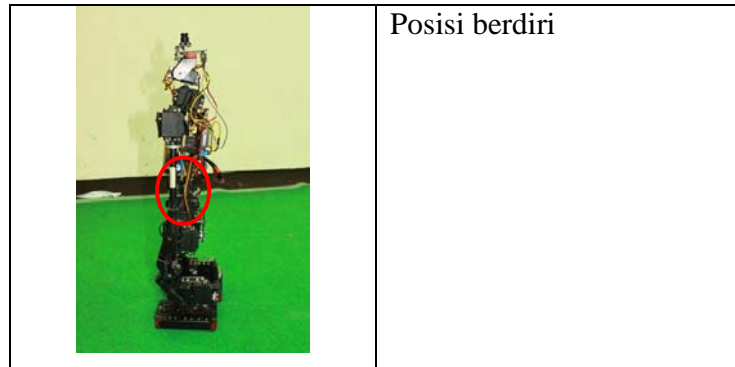
Lanjutan Tabel 4.10

Sketsa	Gambar Nyata	Keterangan
		<p>Persiapan menendang</p>
		<p>Mengangkat kaki kiri untuk persiapan menendang bola</p>
		<p>Mengayunkan kaki kiri untuk menendang bola</p>

Tabel 4.11 Gerakan Bangkit Berdiri

Gambar nyata	Keterangan
	<p>Merentangkan tangan, menapakkan kaki</p>
	<p>Mengangkat badan robot</p>
	<p>Menggeser COG agar bertumpu pada kedua kaki</p>
	<p>Menegakkan posisi badan</p>

Lanjutan Tabel 4.11



Dari hasil pengamatan pada setiap gerakan robot, posisi dari COG (*Center of Gravity*) yang diasumsikan berada pada perut sangatlah penting. Posisi dari COG ini selalu berada dalam *support polygon* sehingga robot tidak terjatuh pada setiap gerakan. Robot akan melakukan gerakan bangkit berdiri jika nilai ADC yang diterima dari sensor percepatan kurang dari 100.

IV.2.2 Pengujian Gerakan Berjalan Pada Robot

Pengujian gerakan berjalan pada robot dimaksudkan untuk mengetahui persentasi kemungkinan robot jatuh pada waktu berjalan. Percobaan ini dilakukan dengan cara menguji robot untuk dapat berjalan sejauh 100 cm dalam arah lurus, arah kanan dan arah kiri. hasil data pengamatan ini dapat dilihat pada Tabel 4.12 sampai Tabel 4.14.

Tabel 4.12 Data Robot pada Saat Berjalan Lurus

Percobaan ke-	Waktu(detik)	Kondisi
1	113	Lancar
2	120	Lancar
3	109	Lancar
4	111	Lancar
5	113	Lancar
6	114	Lancar
7	111	Lancar
8	111	Lancar
9	113	Lancar
10	113	Lancar
Rata - rata	112.8	

Dari data pada Tabel 4.12 dapat diketahui bahwa persentase robot terjatuh pada saat berjalan lurus adalah 0% dan waktu rata – rata robot untuk menempuh jarak 100 cm adalah 112.8 detik.

Tabel 4.13 Data Robot pada Saat Berjalan Arah Kanan

Percobaan ke-	Waktu(detik)	Kondisi
1	135	Lancar
2	144	Lancar
3	144	Lancar
4	130	Lancar
5	135	Lancar
6	144	Lancar
7	144	Lancar
8	144	Lancar
9	137	Lancar
10	137	Lancar
Rata - rata	139.4	

Dari data pada Tabel 4.13 dapat diketahui bahwa persentase robot terjatuh pada saat berjalan arah kanan adalah 0% dan waktu rata – rata robot untuk menempuh jarak 100 cm adalah 139.4 detik.

Tabel 4.14 Data Robot pada Saat Berjalan Arah Kiri

Percobaan ke-	Waktu(detik)	Kondisi
1	137	Lancar
2	135	Lancar
3	137	Lancar
4	137	Lancar
5	139	Lancar
6	137	Lancar
7	-	Jatuh ke depan
8	139	Lancar
9	137	Lancar
10	137	Lancar
Rata - rata	137.2	

Dari data pada Tabel 4.14 dapat diketahui bahwa persentase robot terjatuh pada saat berjalan Arah kiri adalah 10% dan waktu rata – rata robot untuk menempuh jarak 100 cm adalah 137.2 detik.

IV.2.3 Gerakan *Omnidirectional*

Pada keseluruhan sistem, dilakukan pengujian gerakan *omnidirectional* pada robot untuk menuju posisi – posisi tertentu. Pengujian ini dilakukan dengan cara meletakkan bola searah gawang, di sebelah kiri gawang, dan di sebelah kanan gawang. Data – data ini berupa foto dengan kertas putih sebagai indikator lintasan yang dilalui oleh robot dan kertas oranye sebagai indikator lintasan yang dilalui bola. Data – data ini dapat dilihat pada Gambar 4.3 sampai Gambar 4.5.



Gambar 4.3 Lintasan Robot menuju bola yang berada searah gawang

Terdapat rentang nilai dalam menentukan arah gerak ini, jika perbedaan posisi bola dan posisi gawang masih di dalam rentang nilai tersebut maka robot masih menganggap bahwa arah ke bola searah dengan arah ke gawang. Pada saat arah ke bola searah dengan arah ke gawang, robot akan melakukan gerakan berjalan langsung menuju bola dan pada saat bola sudah dekat robot akan melakukan gerakan menendang. Pola langkah lain juga dapat terjadi ketika robot menganggap posisi bola

berada di sebelah kanan atau kiri dari gawang. Anggapan ini dikarenakan gawang tersusun dari 2 tiang, sehingga jika robot hanya mendeteksi posisi dari tiang kanan dan kemudian mendeteksi posisi bola maka robot akan menganggap posisi bola berada di sebelah kiri gawang dan melakukan manuver untuk berjalan jauh ke kiri untuk menyearahkan bola dengan gawang.



Gambar 4.4 Lintasan Robot menuju bola yang berada di sebelah kiri gawang

Pada saat posisi bola yang berada di sebelah kiri gawang, robot akan memutuskan mengambil arah gerak berjalan ke kanan untuk menyearahkan arah ke bola dengan arah ke gawang. Setelah bola dan gawang searah dengan robot maka robot akan melakukan gerakan berjalan untuk mendekati bola dan menendang.



Gambar 4.5 Lintasan Robot menuju bola yang berada di sebelah kanan gawang

Pada saat posisi bola yang berada di sebelah kanan gawang, robot akan memutuskan mengambil arah gerak berjalan ke kiri untuk menyearahkan arah ke bola

dengan arah ke gawang. Setelah bola dan gawang searah dengan robot maka robot akan melakukan gerakan berjalan untuk mendekati bola dan menendang.

Pengujian juga dilakukan dengan memindahkan posisi bola pada saat robot telah berjalan. Terdapat 4 posisi pemindahan yaitu pemindahan bola dari sebelah kanan gawang menuju ke kanan, pemindahan bola dari sebelah kiri gawang menuju ke kiri, pemindahan bola dari sebelah kanan gawang menuju ke kiri, dan pemindahan bola dari sebelah kiri gawang menuju ke kanan. Gambar lintasan robot ini dapat dilihat pada Gambar 4.6 sampai Gambar 4.9.



Gambar 4.6 Lintasan Robot terhadap pemindahan bola dari sebelah kiri gawang menuju ke kiri

Pada Gambar 4.6 dapat dilihat bahwa robot tetap dapat menendang bola ke arah gawang meskipun posisi bola dipindahkan menuju ke kiri. Hal ini dikarenakan posisi awal bola telah berada di sebelah kiri gawang sehingga robot mengambil keputusan untuk melakukan arah gerak ke kanan. Selama pemindahan bola masih berada pada area sebelah kiri gawang maka robot masih dapat menyearahkan arah ke bola dengan arah ke gawang.



Gambar 4.7 Lintasan Robot terhadap pemindahan bola dari sebelah kanan gawang menuju ke kanan

Pada Gambar 4.7 dapat dilihat bahwa robot tetap dapat menendang bola ke arah gawang meskipun posisi bola dipindahkan menuju ke kanan. Hal ini dikarenakan posisi awal bola telah berada di sebelah kanan gawang sehingga robot mengambil keputusan untuk melakukan arah gerak ke kiri. Selama pemindahan bola masih berada pada area sebelah kanan gawang maka robot masih dapat menyearahkan arah ke bola dengan arah ke gawang.



Gambar 4.8 Lintasan Robot terhadap pemindahan bola dari sebelah kanan gawang menuju ke kiri

Pada Gambar 4.8, posisi bola dipindahkan dari yang sebelumnya berada di sebelah kanan gawang menuju ke sebelah kiri gawang. Dalam kondisi ini robot tidak berhasil menendang bola ke arah gawang. Hal ini terjadi karena posisi bola awalnya

berada di sebelah kanan gawang sehingga robot memutuskan untuk mengambil arah gerak ke kiri untuk menyearahkan. Ketika bola dipindahkan menuju ke sebelah kiri gawang maka robot tetap mengambil arah gerak ke kiri sehingga arah robot ke bola berlawanan dengan arah robot ke gawang.



Gambar 4.9 Lintasan Robot terhadap pemindahan bola dari sebelah kiri gawang menuju ke kanan

Kondisi yang sama juga terjadi pada Gambar 4.9 ketika posisi bola yang sebelumnya berada di sebelah kiri gawang dipindahkan menuju ke sebelah kanan gawang. Robot tidak dapat menendang bola ke gawang melainkan arah robot ke bola berlawanan dengan arah robot ke gawang.

Untuk dapat melakukan gerakan *omnidirectional*, robot humanoid selalu melacak posisi bola pada setiap langkah nya. Posisi bola ini direpresentasikan dengan posisi servo pada leher. Untuk mengetahui hubungan posisi servo pada kepala dengan arah jalan robot, dapat dilihat data percobaan pada Tabel 4.15. Data – data ini diambil ketika robot melakukan arah gerak ke kiri untuk menyerahkan, yaitu pada saat posisi bola berada di sebelah kiri gawang.

Tabel 4.15 Nilai Posisi Servo Pada Saat Robot Melakukan Arah Gerak Ke Kiri

Langkah ke-	Nilai Posisi Servo		Langkah ke-	Nilai Posisi Servo	
	X	Y		X	Y
0	53	110	11	240	169
1	72	110	12	221	171
2	94	110	13	195	175
3	117	110	14	158	179
4	126	110	15	136	184
5	148	110	16	118	186
6	163	125	17	130	194
7	183	128	18	105	206
8	208	143	19	128	216
9	232	152	20	94	220
10	245	166			

Dari Tabel 4.15 dapat diketahui bahwa pada langkah ke-0 nilai posisi servo x bernilai 53 yang menandakan posisi bola berada di sebelah kiri robot. Pada percobaan ini, robot mengambil keputusan melakukan arah gerak ke kiri untuk menyearahkan, sehingga robot akan terus berjalan ke kiri sampai posisi servo x menandakan bahwa posisi bola sudah berada di sebelah kanan robot yaitu pada langkah ke-10. Setelah bola berada di sebelah kanan maka robot akan melakukan gerakan berjalan untuk mendekati bola dengan bergerak ke kanan sampai bola berada di posisi tengah dan dekat dengan kaki robot yaitu pada langkah ke-20.

BAB V

KESIMPULAN DAN SARAN

Pada bab ini berisi kesimpulan dari Penelitian dan saran-saran yang perlu dilakukan untuk perbaikan di masa mendatang.

V.1 Kesimpulan

Dengan memperhatikan data pengamatan dan analisis pada bab sebelumnya, dapat disimpulkan bahwa:

1. Sensor kamera CMUCam3 dapat digunakan untuk melacak warna pada bola dan warna pada gawang dengan rentang nilai yang diprogram.
2. Kelancaran robot humanoid pada saat berjalan lurus adalah 100% dengan waktu rata – rata pada saat menempuh jarak 100 cm adalah 112.8 detik, kelancaran robot pada saat berjalan arah kanan sejauh 100 cm adalah 100% dengan waktu rata – rata 139.4 detik, kelancaran robot pada saat berjalan arah kiri sejauh 100 cm adalah 90% dengan waktu rata – rata 137.2 detik.
3. Robot humanoid dapat melakukan gerakan bangkit berdiri ketika terjatuh yang dideteksi menggunakan sensor percepatan.
4. Secara keseluruhan, robot humanoid pemain bola ini berhasil menendang bola ke gawang jika posisi bola tetap berada pada area yang sama terhadap gawang, robot humanoid pemain bola akan gagal menendang bola ke gawang jika posisi bola dipindahkan ke area yang berbeda terhadap gawang ketika robot sudah bergerak.

V.2 Saran

Saran-saran yang dapat diberikan untuk perbaikan dan pengembangan dari Penelitian ini adalah sebagai berikut:

1. Struktur robot yang lebih ringan dan kuat agar robot *humanoid* pemain bola dapat berjalan lebih cepat.

2. Dibutuhkan metoda untuk mendeteksi posisi tengah dari gawang sehingga robot dapat lebih tepat menentukan arah gerak misalnya dengan bantuan sensor kompas.
3. Dibutuhkan algoritma untuk mendeteksi posisi gawang dan posisi bola secara terus menerus dengan proses yang cepat.
4. Robot harus melakukan aksi jalan dan *scanning* ketika tidak mendeteksi bola, yaitu pada saat bola jauh di depan robot.

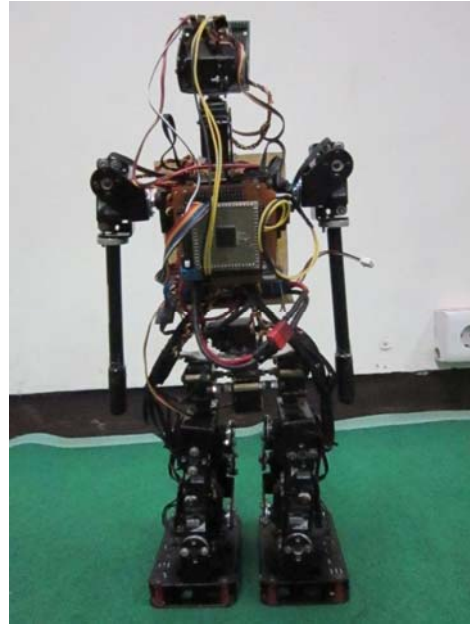
DAFTAR PUSTAKA

1. Erick V. Cuevas, Daniel Zaldivar, Raul Rojas, *Bipedal Robot Description*, 2005.
2. Kenji Kaneko, et.al, *Design of Prototype Humanoid Robotics Platform for HRP*”Proceedings of the 2002 IEEE/RSJ, Intl. Conference on Intelligent Robots and Systems EPFL, Lausanne, Switzerland, 2002.
3. Sven Behnke. *Online Trajectory Generation for Omnidirectional Biped Walking*. 2006 IEEE *International Conference on Robotics and Automation*, 2006.
4. Shuuji Kajita, Fumio Kanehiro, Kenjo Kaneko, Kiyoshi Fujiwara, Kensuke Harada, Kazuhito Yokoi, dan Hirohisa Hirukawa. *Biped Walking Pattern Generation Using Preview Control of Zero-Moment Point*. In *International Conference on Robotics and Automation; Proceedings of the 2003 IEEE*, 2003.
5. Miomir Vukobratovic dan Branislav Borovac. *Zaro moment point – thirty five years of its life*. *International Journal of Humanoid Robotics*, 1(1):127-173, 2004.
6. http://www.robot-italy.net/downloads/ssc32_manual.pdf diakses 15 April 2012
7. <http://www.dimensionengineering.com/datasheets/DE-ACCM3D.pdf> diakses 15 April 2012
8. http://www.superrobotica.com/download/cmucam3/CMUcam3_datasheet.pdf diakses 15 April 2012
9. http://www.cmucam.org/projects/cmucam4/wiki/Color-tracking_Explanation diakses 15 April 2012

LAMPIRAN A
FOTO ROBOT HUMANOID PEMAIN BOLA



TAMPAK DEPAN



TAMPAK BELAKANG



TAMPAK KIRI



TAMPAK KANAN

LAMPIRAN B

PROGRAM PADA PENGONTROL MIKRO

ATMEGA 128 DAN SENSOR KAMERA CMUCAM3

PROGRAM UTAMA

/*****

This program was produced by the
CodeWizardAVR V1.25.3 Standard
Automatic Program Generator
© Copyright 1998-2007 Pavel Haiduc, HP InfoTech s.r.l.
<http://www.hpinfotech.com>

Project : Humanoid Soccer 2012
Version : 1.0
Date : 6/28/2011
Author : Christian
Company : Christian Hadinata
Comments:

Chip type : ATmega128L
Program type : Application
Clock frequency : 7.372800 MHz
Memory model : Small
External SRAM size : 0
Data Stack size : 1024

*****/

```
#include <mega128.h>
```

```
// Alphanumeric LCD Module functions
```

```
#asm  
.equ __lcd_port=0x15 ;PORTC  
#endasm
```

```
#include <lcd.h>  
#include <stdio.h>  
#include <math.h>  
#include <string.h>  
#include <delay.h>  
#include <stdlib.h>
```

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

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

```

#define X_PIN read_adc(0)
#define Y_PIN read_adc(1)
#define Z_PIN read_adc(2)

// Get a character from the USART1 Receiver
#pragma used+
char getchar1(void)
{
char status,data;
while (1)
{
while (((status=UCSR1A) & RX_COMPLETE)==0);
data=UDR1;
if ((status & (FRAMING_ERROR | PARITY_ERROR | DATA_OVERRUN))==0)
return data;
};
}
#pragma used-

// Write a character to the USART1 Transmitter
#pragma used+
void putchar1(char c)
{
while ((UCSR1A & DATA_REGISTER_EMPTY)==0);
UDR1=c;
}
#pragma used-

#define ADC_VREF_TYPE 0x00

// Read the AD conversion result
unsigned int read_adc(unsigned char adc_input)
{
ADMUX=adc_input | (ADC_VREF_TYPE & 0xff);
// Start the AD conversion
ADCSRA|=0x40;
// Wait for the AD conversion to complete
while ((ADCSRA & 0x10)==0);
ADCSRA|=0x10;
return ADCW;
}

int detect=0,servox=0,servoy=0,servox_goal=0,servoy_goal=0;
char process=0;
int LastServoxGoal=0,LastServoyGoal=0;

#include "Servo Controller.c"

void CameraIn(){
char Buffer[32];
char Camera[5][4] = {{0,0,0,0},
{0,0,0,0},
{0,0,0,0},
{0,0,0,0},
{0,0,0,0}};
unsigned char counter = 0, i = 0;
unsigned char step = 0, j = 0;

while(getchar() != '\n');
do{
Buffer[counter] = getchar();
counter++;
}

```

```

while(Buffer[counter-1] != '\n');
Buffer[counter-1] = 0;

for(i=0;i<counter;i++){
    if(Buffer[i] == ','){
        step++;
        j=0;
    }
    else{
        Camera[step][j] = Buffer[i];
        j++;
    }
}
step = 0;
j = 0;
counter = 0;

detect = atoi(Camera[0]);
servox = atoi(Camera[1]);
servoy = atoi(Camera[2]);
servox_goal = atoi(Camera[3]);
servoy_goal = atoi(Camera[4]);
}
// Declare your global variables here

void main(void)
{
    unsigned char i=0, Turn_Condition = 0;
    unsigned char Turn_Move = 0, Done = 0;
    int n;
    /*
        Turn Condition =
        0 = straight
        1 = turn right
        2 = turn left
    */

    // Declare your local variables here

    // Input/Output Ports initialization
    // Port A initialization
    // Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
    // State7=P State6=P State5=P State4=P State3=P State2=P State1=P State0=P
    PORTA=0xFF;
    DDRA=0x00;

    // Port B initialization
    // Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
    // State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
    PORTB=0x00;
    DDRB=0x00;

    // Port C initialization
    // Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
    // State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
    PORTC=0x00;
    DDRC=0x00;

    // Port D initialization
    // Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
    // State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
    PORTD=0x00;
    DDRD=0x00;

```



```

// Port E initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTE=0x00;
DDRE=0x00;

// Port F initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTF=0x00;
DDRF=0x00;

// Port G initialization
// Func4=In Func3=In Func2=In Func1=In Func0=In
// State4=T State3=T State2=T State1=T State0=T
PORTG=0x00;
DDRG=0x00;

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

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: Timer 1 Stopped
// Mode: Normal top=FFFFh
// OC1A output: Discon.
// OC1B output: Discon.
// OC1C output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer 1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
// Compare C Match Interrupt: Off
TCCR1A=0x00;
TCCR1B=0x00;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;
OCR1CH=0x00;
OCR1CL=0x00;

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

```

```

// Timer/Counter 3 initialization
// Clock source: System Clock
// Clock value: Timer 3 Stopped
// Mode: Normal top=FFFFh
// Noise Canceler: Off
// Input Capture on Falling Edge
// OC3A output: Discon.
// OC3B output: Discon.
// OC3C output: Discon.
// Timer 3 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
// Compare C Match Interrupt: Off
TCCR3A=0x00;
TCCR3B=0x00;
TCNT3H=0x00;
TCNT3L=0x00;
ICR3H=0x00;
ICR3L=0x00;
OCR3AH=0x00;
OCR3AL=0x00;
OCR3BH=0x00;
OCR3BL=0x00;
OCR3CH=0x00;
OCR3CL=0x00;

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
// INT2: Off
// INT3: Off
// INT4: Off
// INT5: Off
// INT6: Off
// INT7: Off
EICRA=0x00;
EICRB=0x00;
EIMSK=0x00;

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

// USART0 initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART0 Receiver: On
// USART0 Transmitter: Off
// USART0 Mode: Asynchronous
// USART0 Baud rate: 115200
UCSR0A=0x00;
UCSR0B=0x10;
UCSR0C=0x06;
UBRR0H=0x00;
UBRR0L=0x03;

// USART1 initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART1 Receiver: On
// USART1 Transmitter: On
// USART1 Mode: Asynchronous
// USART1 Baud rate: 115200
UCSR1A=0x00;
UCSR1B=0x18;

```

```

UCSR1C=0x06;
UBRR1H=0x00;
UBRR1L=0x03;

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

// ADC initialization
// ADC Clock frequency: 921.600 kHz
// ADC Voltage Reference: AREF pin
ADMUX=ADC_VREF_TYPE & 0xff;
ADCSRA=0x83;

// LCD module initialization
lcd_init(16);

Walking_Step[n][22] = 1334;
SendMovement(HP_Pos,Time);
delay_ms(3000);

n = 0;
Behaviour = 1;
Increased_Pos = 10;
Turn_Pos = 60;
Turn_Condition = 2;
CameraIn();

while (1){

    if(Z_PIN < 200){
        for(i=0;i<6;i++){
            if(i == 5)    Time = 3000;
            else        Time = 1000;
            SendMovement(Kick_Step[i],Time);
        }
    }

    while((servox == 0) && (servoy == 0)){
        CameraIn();
        delay_ms(10);
    };

    if(n==1 || n==4 || Behaviour == 0){
        CameraIn();
        delay_ms(10);
    }

    if(process == 0){
        if(servox_goal > (servox - 20)){
            Turn_Move = 2; //ke kiri
        }
        else if(servox_goal < (servox + 20)){
            Turn_Move = 1; //ke kanan
        }
        else{
            Turn_Move = 0; //lurus
        }
        process = 1;
    }
}

```

```

if((detect == 0) && (n==0 || n==5)){
    Time = 1000;
    if(Last_Behaviour == 1){
        Time = 1000;
        SendMovement(Preparation,Time);
    }
    Behaviour = 0;
}
else{
    if(Last_Behaviour == 0){
        Time = 1000;
        SendMovement(Preparation,Time);
    }
    Behaviour = 1;
    if(Done == 1){
        if(Turn_Move == 2){
            if(servox < 180){
                Walking_Step[n][22] = 1334;
                Walking_Step[2][16] = 1630;
                Walking_Step[3][16] = 1630;
            }
        }
        else if(Turn_Move == 1){
            if(servox > 70){
                Walking_Step[n][22] = 1334;
                Walking_Step[0][0] = 1250;
                Walking_Step[5][0] = 1250;
            }
        }
        if(servox > 90){
            Turn_Condition = 1;
        }
        else if(servox < 70){
            Turn_Condition = 2;
        }
        else{
            Turn_Condition = 0;
        }
    }

    if((servoy > 235) && (n==0 || n==5)){
        Time = 1000;
        SendMovement(Preparation,Time);
        SendMovement(HP_Pos,Time);
        delay_ms(1000);
        for(i=0;i<8;i++){
            if(i == 7) Time = 100;
            else Time = 1000;
            SendMovement(Kick_Step[i],Time);
        }
        Time = 1000;
        SendMovement(HP_Pos,Time);
        delay_ms(1000);
        while(1);
        process = 0;
        Done = 0;
    }
}
else{
    if(Turn_Move == 2){
        if(servox <180){
            Walking_Step[n][22] = (1334 + 100);
            Walking_Step[2][16] = (1630 - 20);
            Walking_Step[3][16] = (1630 - 20);
            Walking_Step[0][0] = (1250 + 50);
        }
    }
}

```

```

        Walking_Step[5][0] = (1250 + 50);
    }
    if(servox < 220){
        Turn_Condition = 2;
    }
    else Done = 1;
}
else if(Turn_Move == 1){
    if(servox > 70){
        Walking_Step[n][22] = (1334 - 100);
        Walking_Step[2][16] = (1630 - 20);
        Walking_Step[3][16] = (1630 - 20);
        Walking_Step[0][0] = (1250 + 50);
        Walking_Step[5][0] = (1250 + 50);
    }
    if(servox > 30){
        Turn_Condition = 1;
    }
    else Done = 1;
}
else Done = 1;
}
}

switch(Behaviour){
    case 0 :
    {
        for(i=0;i<32;i++){
            Destination_Pos[i] = HP_Pos[i];
        }
    };
    break;
    case 1 :
    {
        for(i=0;i<32;i++){
            Destination_Pos[i] = Walking_Step[n][i];
            if(Turn_Condition == 1){
                if(n==3 || n==4 || n==5){
                    Destination_Pos[5] = Walking_Step[n][5] - Turn_Pos;
                    Destination_Pos[22] = Walking_Step[n][22] + 221;
                }
                else if(n==0 || n==1 || n==2){
                    Destination_Pos[21] = Walking_Step[n][21] - (Turn_Pos+20);
                    Destination_Pos[22] = Walking_Step[n][22] - 222;
                }
            }
            else if(Turn_Condition == 2){
                if(n==0 || n==1 || n==2){
                    Destination_Pos[21] = Walking_Step[n][21] + (Turn_Pos+20);
                    Destination_Pos[22] = Walking_Step[n][22] - 222;
                }
                else if(n==3 || n==4 || n==5){
                    Destination_Pos[5] = Walking_Step[n][5] + Turn_Pos;
                    Destination_Pos[22] = Walking_Step[n][22] + 221;
                }
            }
        }
    };
    break;
};
if(n==2 || n==5){
    Time = 1000;
}
else{

```

```

        Time = 800;
    }
    SendMovement(Destination_Pos,Time);

    n = n+1;
    if(n > (Step_Number[Behaviour]-1))    n = 0;

    Last_Behaviour = Behaviour;

    LastServoxGoal = servox_goal;
    LastServoyGoal = servoy_goal;

};
}

```

SUBPROGRAM PENGIRIMAN PERINTAH KE SSC-32

```

Eeprom int HP_Pos[64] =
{1430,1100,1220,1518,824,1500,2318,2048,1000,1500,1500,1500,1500,1500,1500,1500,1481,1900,1780,1539,2
233,1500,1334,800,2162,1500,1500,1500,1500,1500,1500,1500,1000,1000,1000,1000,1000,1000,1000,1000,100
0,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000
,1000,1000,1000};

```

```

eeprom int Preparation[64] =
{1250,1100,1220,1449,824,1500,2318,2048,1000,1500,1500,1500,1500,1500,1500,1500,1319,1696,2146,1409,2
383,1500,1334,800,2162,1500,1500,1500,1500,1500,1500,1500,1000,1000,1000,1000,1000,1000,1000,1000,100
0,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000
,1000,1000,1000};

```

```

int Time = 1000;

```

```

eeprom int Walking_Step[6][64] = {
{1250,1177,1180,1449,824,1721,2300,2048,1000,1500,1500,1500,1500,1500,1500,1500,1319,1674,2147,1409,2
383,1700,1112,800,2162,1500,1500,1500,1500,1500,1500,1500,1000,1000,1000,1000,1000,1000,1000,1000,100
0,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000
,1000,1000,1000},
{1430,1100,1220,1518,780,1721,2300,2048,1000,1500,1500,1500,1500,1500,1500,1500,1481,1900,1780,1539,2
233,1700,1112,800,2162,1500,1500,1500,1500,1500,1500,1500,1000,1000,1000,1000,1000,1000,1000,1000,1000,100
0,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000
,1000,1000,1000},
{1590,1280,997,1617,715,1721,2300,2048,1000,1500,1500,1500,1500,1500,1500,1500,1630,1900,1780,1596,22
05,1700,1112,800,2162,1500,1500,1500,1500,1500,1500,1500,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000
,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000
,1000,1000,1000},
{1590,1245,997,1617,715,1299,2300,2048,1000,1500,1500,1500,1500,1500,1500,1500,1630,1823,1829,1596,22
05,1279,1555,800,2162,1500,1500,1500,1500,1500,1500,1500,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000
,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000
,1000,1000,1000},

```



```

3000,3000,3000},
{1430,1263,1895,1518,1654,1500,1598,700,980,1500,1500,1500,1500,1500,1500,1500,1481,1764,1190,1539,14
30,1500,1334,2192,2134,1500,1500,1500,1500,1500,1500,1500,3000,3000,3000,3000,3000,3000,3000,3000,300
0,3000,3000,3000,3000,3000,3000,3000,3000,3000,3000,3000,3000,3000,3000,3000,3000,3000,3000,3000
,3000,3000,3000},
{1430,1163,1908,1518,1654,1500,2358,1880,1008,1500,1500,1500,1500,1500,1500,1500,1481,1864,1120,1539,
1430,1500,1334,980,2077,1500,1500,1500,1500,1500,1500,1500,1000,1000,1000,1000,1000,1000,1000,1000,100
0,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,100
0,1000,1000,1000},
{1430,1100,1220,1518,824,1500,2300,2048,1008,1500,1500,1500,1500,1500,1500,1500,1481,1900,1780,1539,2
233,1500,1334,750,2189,1500,1500,1500,1500,1500,1500,1500,1000,1000,1000,1000,1000,1000,1000,1000,1000
,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000
,1000,1000,1000}
};

```

```

unsigned char Step_Number[4] = {1,6,8,6}; //HP_Pos,Walking_Step,Kick_Step,Wake_Step

```

```

int Current_Pos[32], Increased_Pos, Turn_Pos;
eeprom int Destination_Pos[32];

```

```

unsigned char Behaviour = 0, Last_Behaviour = 0;

```

```

/*

```

```

0 = Home Position

```

```

1 = Walking Position

```

```

*/

```

```

void SendUSART1(char Text[])

```

```

{
    unsigned char i;
    for(i=0;i<strlen(Text);i++)
    {
        putchar1(Text[i]);
    }
}

```

```

void SendMovement(eeprom int Position[], int Time)

```

```

{
    unsigned char Pin;
    char Text[33];
    for(Pin=0;Pin<32;Pin++)
    {
        sprintf(Text, "%d P%d T%d \r", Pin, Position[Pin], Time);
        SendUSART1(Text);
    }
}

```



```

        Current_Pos[Pin] = Position[Pin];
    }
    delay_ms(Time);
}

void GoServoPosition(int ServoChannel, int ServoPosition, int ServoTime)
{
    char Text[33];
    sprintf(Text, "%d P%d T%d \r", ServoChannel, ServoPosition, ServoTime);
    SendUSART1(Text);
}

```

PROGRAM UTAMA SENSOR KAMERA CMUCAM3

```

#include <stdio.h>
#include <stdlib.h>
#include <cc3.h>
#include <cc3_ilp.h>
#include <time.h>
#include <cc3_color_track.h>
#include <math.h>

void simple_track_color(cc3_track_pkt_t * t_pkt);

int main(void) {
    cc3_track_pkt_t t_pkt;

    cc3_uart_init (0,
                  CC3_UART_RATE_115200,
                  CC3_UART_MODE_8N1,
                  CC3_UART_BINMODE_TEXT);

    cc3_camera_init ();

    t_pkt.track_invert = 0;
    cc3_g_pixbuf_frame.coi = CC3_CHANNEL_ALL;

    cc3_gpio_set_mode(2,CC3_GPIO_MODE_SERVO);
    cc3_gpio_set_mode(3,CC3_GPIO_MODE_SERVO);
    cc3_camera_set_resolution(CC3_CAMERA_RESOLUTION_LOW);

    // init pixbuf with width and height
    cc3_pixbuf_load ();

    uint8_t servo_updown = 128;
    uint8_t servo_leftright = 128;
    uint8_t servo_updown_ball=0 , servo_leftright_ball=0 , servo_updown_goal=0 , servo_leftright_goal=0
;
    uint8_t last_servo_updown, last_servo_leftright;
    uint8_t LastServoUpdownBall, LastServoLeftrightBall;
    uint8_t range_updown=0, range_leftright=0, FirstSight = 0, FirstDetect = 0;
    uint8_t mid_x = 88;
    uint8_t mid_y = 11;
    uint8_t kuadran = 2;
    uint8_t detect_goal = 0, detect_ball = 0, tolerance = 2, condition = 0;

```

```

char Buffer;

cc3_gpio_set_servo_position(2,servo_leftright);
cc3_gpio_set_servo_position(3,servo_updown);

while(true) {
if(condition == 0){ //gawang
    t_pkt.lower_bound.channel[CC3_CHANNEL_RED] = 171;
    t_pkt.upper_bound.channel[CC3_CHANNEL_RED] = 221;
    t_pkt.lower_bound.channel[CC3_CHANNEL_GREEN] = 215;
    t_pkt.upper_bound.channel[CC3_CHANNEL_GREEN] = 255;
    t_pkt.lower_bound.channel[CC3_CHANNEL_BLUE] = 0;
    t_pkt.upper_bound.channel[CC3_CHANNEL_BLUE] = 41;
    t_pkt.noise_filter = 2;
}
else{ //bola
    t_pkt.lower_bound.channel[CC3_CHANNEL_RED] = 215;
    t_pkt.upper_bound.channel[CC3_CHANNEL_RED] = 255;
    t_pkt.lower_bound.channel[CC3_CHANNEL_GREEN] = 105;
    t_pkt.upper_bound.channel[CC3_CHANNEL_GREEN] = 155;
    t_pkt.lower_bound.channel[CC3_CHANNEL_BLUE] = 0;
    t_pkt.upper_bound.channel[CC3_CHANNEL_BLUE] = 44;
    t_pkt.noise_filter = 2;
}

simple_track_color(&t_pkt);

if(t_pkt.int_density == 0){
    detect_goal = 0;
    detect_ball = 0;
    if(kuadran == 2){
        if(servo_leftright < 245){
            servo_leftright = servo_leftright + 5;
            //cc3_timer_wait_ms (5);
        }
        if(servo_updown > 110){
            servo_updown = servo_updown - 3;
            //cc3_timer_wait_ms (5);
        }
        if(servo_leftright >= 245 && servo_updown <= 110){
            servo_leftright = 245;
            servo_updown = 110;
            kuadran = 1;
        }
        cc3_gpio_set_servo_position(2,servo_leftright);
        cc3_gpio_set_servo_position(3,servo_updown);
    }
    else if(kuadran == 1){
        if(servo_leftright > 15){
            servo_leftright = servo_leftright - 5;
            //cc3_timer_wait_ms (5);
        }
        if(servo_updown < 250){
            servo_updown = servo_updown + 3;
            //cc3_timer_wait_ms (5);
        }
        if(servo_leftright <= 15 && servo_updown >= 250){
            servo_leftright = 15;
            servo_updown = 250;
            kuadran = 3;
        }
        cc3_gpio_set_servo_position(2,servo_leftright);
        cc3_gpio_set_servo_position(3,servo_updown);
    }
}

```

```

}
else if(kuadran == 3){
    if(servo_leftright < 245){
        servo_leftright = servo_leftright + 5;
        //cc3_timer_wait_ms (5);
    }
    if(servo_updown < 250){
        servo_updown = servo_updown + 3;
        //cc3_timer_wait_ms (5);
    }
    if(servo_leftright >= 245 && servo_updown >= 250){
        servo_leftright = 245;
        servo_updown = 250;
        kuadran = 4;
    }
    cc3_gpio_set_servo_position(2,servo_leftright);
    cc3_gpio_set_servo_position(3,servo_updown);
}
else if(kuadran == 4){
    if(servo_leftright > 15){
        servo_leftright = servo_leftright - 5;
        //cc3_timer_wait_ms (5);
    }
    if(servo_updown > 110){
        servo_updown = servo_updown - 3;
        //cc3_timer_wait_ms (5);
    }
    if(servo_leftright <= 15 && servo_updown <= 110){
        servo_leftright = 15;
        servo_updown = 110;
        kuadran = 2;
    }
    cc3_gpio_set_servo_position(2,servo_leftright);
    cc3_gpio_set_servo_position(3,servo_updown);
}
}
else{
    if(condition == 0){
        detect_goal = 1;
    }
    else{
        detect_ball = 1;
        FirstDetect = 1;
    }
}

if(t_pkt.centroid_y > (mid_y + tolerance)){
    if(servo_updown > 110){
        if(t_pkt.centroid_y > (mid_y + 15)){
            servo_updown -= 6;
        }
        else if(t_pkt.centroid_y > (mid_y + 10)){
            servo_updown -= 3;
        }
        else{
            servo_updown -= 1;
        }
    }
    else if(servo_updown < 110){
        servo_updown = 110;
    }
}
else if(t_pkt.centroid_y < (mid_y - tolerance)){
    if(servo_updown < 250){
        if(t_pkt.centroid_y < (mid_y - 15)){

```

```

        servo_updown += 6;
    }
    else if(t_pkt.centroid_y < (mid_y - 10)){
        servo_updown += 3;
    }
    else{
        servo_updown += 1;
    }
}
else if(servo_updown > 250){
    servo_updown = 250;
}
}
if(t_pkt.centroid_x > (mid_x + tolerance)){
    if(servo_leftright > 15){
        if(t_pkt.centroid_x > (mid_x + 20)){
            servo_leftright -= 6;
        }
        else if(t_pkt.centroid_x > (mid_x + 10)){
            servo_leftright -= 3;
        }
        else{
            servo_leftright -= 1;
        }
    }
    else if(servo_leftright < 15){
        servo_leftright = 15;
    }
}
else if(t_pkt.centroid_x < (mid_x - tolerance)){
    if(servo_leftright < 245){
        if(t_pkt.centroid_x < (mid_x - 20)){
            servo_leftright += 6;
        }
        else if(t_pkt.centroid_x < (mid_x - 10)){
            servo_leftright += 3;
        }
        else{
            servo_leftright += 1;
        }
    }
    else if(servo_leftright > 245){
        servo_leftright = 245;
    }
}

cc3_gpio_set_servo_position(2,servo_leftright);
cc3_gpio_set_servo_position(3,servo_updown);

if(condition == 0){
    servo_leftright_goal = servo_leftright;
    servo_updown_goal = servo_updown;
}
else{
    servo_leftright_ball = servo_leftright;
    servo_updown_ball = servo_updown;
}

if(FirstSight != 0){
    if(last_servo_leftright < servo_leftright){
        if(last_servo_updown < servo_updown){
            kuadran = 3;
        }
        else if(last_servo_updown >= servo_updown){

```

```

        kuadran = 2;
    }
}
else if(last_servo_leftright >= servo_leftright){
    if(last_servo_updown < servo_updown){
        kuadran = 1;
    }
    else if(last_servo_updown >= servo_updown){
        kuadran = 4;
    }
}
}

if(FirstSight != 0){
    range_updown = abs>LastServoUpdownBall-servo_updown_ball);
    range_leftright = abs>LastServoLeftrightBall-servo_leftright_ball);
}
FirstSight = 1;

if((condition==0) && (last_servo_updown==servo_updown) &&
(last_servo_leftright==servo_leftright)){
    condition = 1;
    FirstSight = 0;
    cc3_timer_wait_ms (1000);
}

last_servo_updown = servo_updown;
last_servo_leftright = servo_leftright;

LastServoUpdownBall = servo_updown_ball;
LastServoLeftrightBall = servo_leftright_ball;

}
printf( "%d,%d,%d,%d,%d\n", detect_ball, servo_leftright_ball, servo_updown_ball,
servo_leftright_goal, servo_updown_goal);
}

}

void simple_track_color(cc3_track_pkt_t * t_pkt)
{
    cc3_image_t img;

    img.channels = 3;
    img.width = cc3_g_pixbuf_frame.width;
    img.height = 1; // image will hold just 1 row for scanline processing
    img.pix = cc3_malloc_rows (1);
    if (img.pix == NULL) {
        return;
    }

    cc3_pixbuf_load ();
    if (cc3_track_color_scanline_start (t_pkt) != 0) {
        while (cc3_pixbuf_read_rows (img.pix, 1)) {
            cc3_track_color_scanline (&img, t_pkt);
        }
    }
    cc3_track_color_scanline_finish (t_pkt);

    free (img.pix);
    return;
}

```

LAMPIRAN C
DATASHEET

SSC-32 MANUAL



SSC-32 Ver 2.0

Manual written for f
irmware version SSC32-
1.06XE Range is
0.50mS to 2.50mS



Things that go Boom!



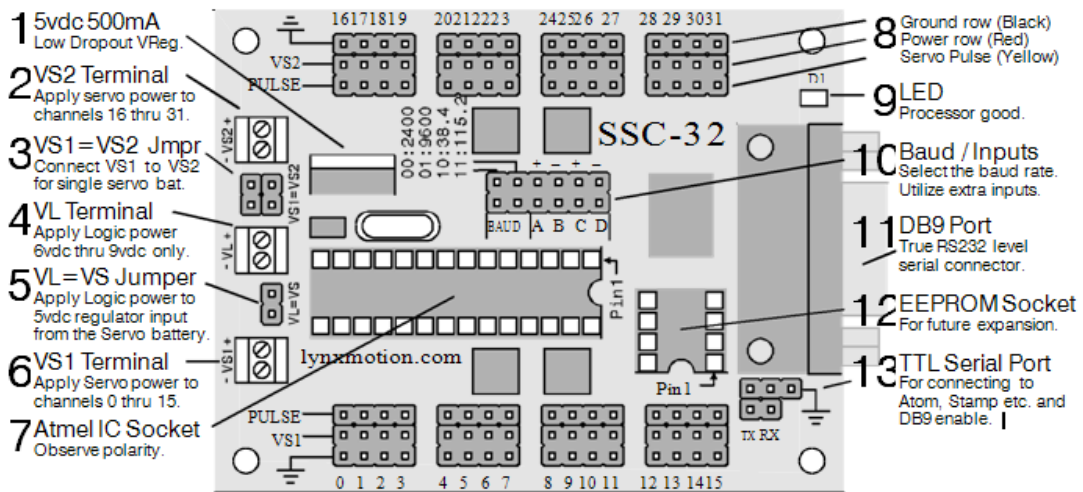
Caution! Read this quick start guide completely before wiring and applying power to the board! Errors in wiring can damage the SSC-32 board, Atmel or EEPROM Chip, and any attached servos or peripherals.



Caution! Never reverse the power coming in to the board. Make sure the black wire goes to (-) ground, and the red wire goes to (+) Vlogic, or Vservo. Never connect peripherals when the board is powered on.



Caution! The onboard regulator can provide 250mA total. This includes the microcontroller chip, the onboard LEDs, and any attached peripherals. Drawing too much current can cause the regulator to overheat.



1 The Low Dropout regulator will provide 5vdc out with as little as 5.5vdc coming in. This is important when operating your robot from a battery. It can accept a maximum of 9vdc in. The regulator is rated for 500mA, but we are de-rating it to 250mA to prevent the regulator from getting too hot.

2 This terminal connects power to servo channels 16 thru 31. Apply 4.8vdc to 7.2vdc for normal servos. Apply 4.8vdc to 6.0vdc when using micro servos. Do not exceed 7.4vdc (measure it) when using HSR-5995TG servos!

3 These jumpers are used to connect VS1 to VS2. Use this option when you are powering all servos from the same battery. Use both jumpers.

4 This is the Electronics Power Input. It is also referred to as the Logic Voltage, or VL. This input is normally used with a 9vdc battery connector to provide power to the ICs and anything connected to the 5vdc lines on the board. This input is used to isolate the logic from the Servo Power Input.

5 This jumper allows powering the microcontroller and support circuitry from the servo power supply. This requires at least 6vdc to operate correctly. If the microcontroller resets when many servos are moving it may be necessary to power the microcontroller separately using the VL input. A 9vdc battery works nicely for this.

6 This terminal connects power to servo channels 0 thru 15. Apply 4.8vdc to 7.2vdc for normal servos. Apply 4.8vdc to 6.0vdc when using micro servos. Do not exceed 7.4vdc (measure it) when using HSR-5995TG servos!

7 This is where the Atmel IC chip goes. Be careful to insert it with Pin 1 in the upper right corner as pictured. Take care not to bend the pins.

Board	Input
VS2+	RED
VS2-	BLACK

Board	Input
VL+	RED
VL-	BLACK

Board	Input
VL1+	RED
VL1-	BLACK

8 This is where you connect the servos or other output devices. Use caution and remove power when connecting anything to the I/O bus.

9 This is the Processor Good LED. It will light steady when power is applied and will remain lit until the processor has received a valid serial command. It will then go out and then blink whenever it is receiving serial data. Note, this feature may not be used on user-submitted firmware for the SSC-32.

10 The two BAUD inputs allow configuring the baud rate. Please see the examples below. The ABCD inputs have both static and latching support. The inputs have internal weak (50k) pullups that are used when a Read Digital Input command is used. A normally open switch connected from the input to ground will work fine. These features may not be used on user-submitted firmware for the SSC-32.

Jumpers	Baud Rate
0 0	2400
0 1	9600
1 0	38.4k
1 1	115.2k

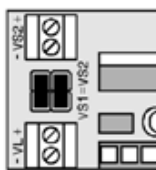
11 Simply plug a straight-through M/F DB9 cable from this plug to a free 9 pin serial port on your PC for receiving servo positioning data. Alternately a USB-to-serial adaptor will work well.

12 This is an 8 pin EEPROM socket. It is not used in this version of the firmware, although it will be used in future versions.

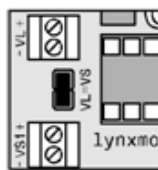
13 This is the TTL serial port or DB9 serial port enable. Install two jumpers as illustrated below to enable the DB9 port. Install wire connectors to utilize TTL serial communication from a host microcontroller.

Shorting Bar Jumpers and Connectors at a glance

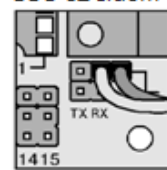
Applies VS1 to VS2.



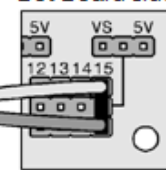
Applies VS to VL.



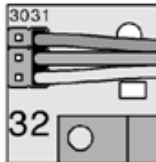
TTL Serial com. SSC-32 side...



TTL Serial com. Bot Board side...

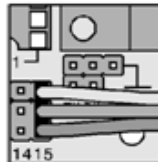


Example servo connection 16-31.



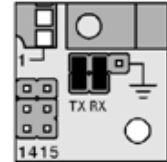
Black
Red
Yellow

Example servo connection 0-15.

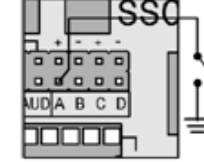


Yellow
Red
Black

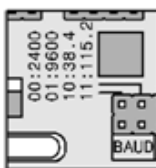
DB9 enable for PC use.



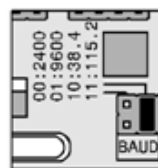
ABCD auxiliary inputs.



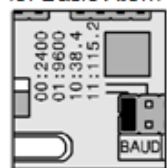
Baud rate 2400.



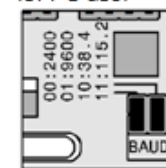
Baud rate 9600.



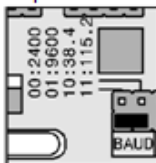
Baud rate 38.4k for Basic Atom use.



Baud rate 115.2k for PC use.



Update the Atmel chip firmware.



Caution! Don't do this if you don't know what you're doing. Connecting this jumper can overwrite the Atmel chip's firmware.



ACCELEROMETER DE-ACCM3D

DE-ACCM3D Buffered $\pm 3g$ Tri-axis Accelerometer

General Description

The DE-ACCM3D is a complete 3D $\pm 3g$ analog accelerometer solution.

It features integrated op amp buffers for direct connection to a microcontroller's analog inputs, or for driving heavier loads.

The onboard 3.3V regulator and decoupling capacitor give you great flexibility when powering the device, and can also be bypassed for operation down to 2.0V.

The DE-ACCM3D is designed to fit the DIP-16 form factor, making it suitable for breadboarding, perfboarding, and insertion into standard chip sockets.

It is based on the Analog Devices ADXL330 for superior sensitivity and tighter accuracy tolerances.

Features

Triple axis $\pm 3g$ sense range

Up to 360mV/g sensitivity

500Hz bandwidth

Operating voltage 3.5V to 15V (onboard regulator) Operating voltage 2.0V to 3.6V (without regulator)

3.3V regulator can power external microcontroller

Reverse voltage protection Output short protected

Standard DIP-16 form factor

Integrated power supply decoupling

Draws 0.9mA

Can accurately drive 500 loads

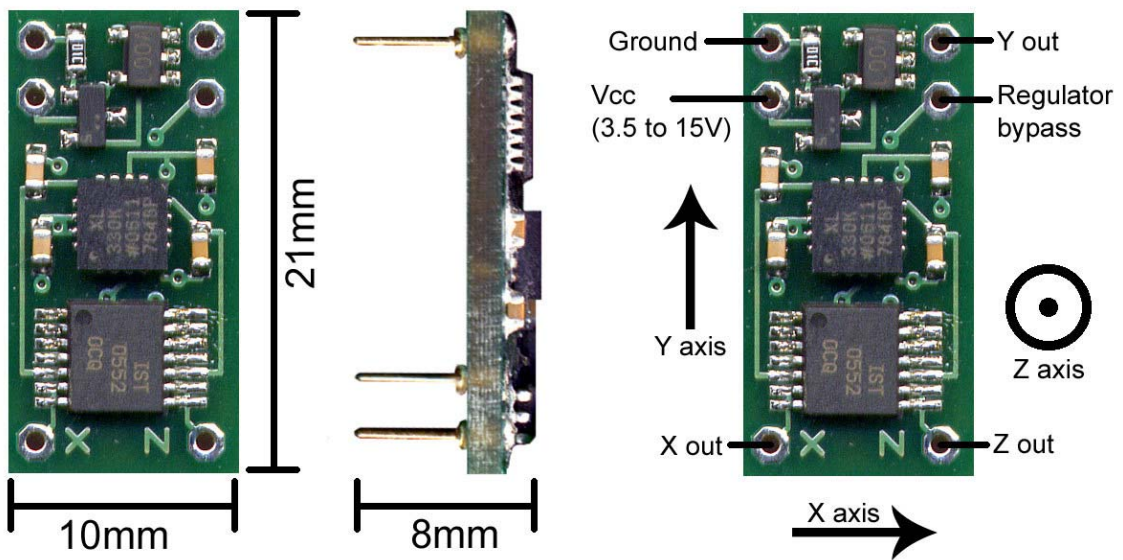
Applications

Motion, tilt and slope measurement

Device positioning

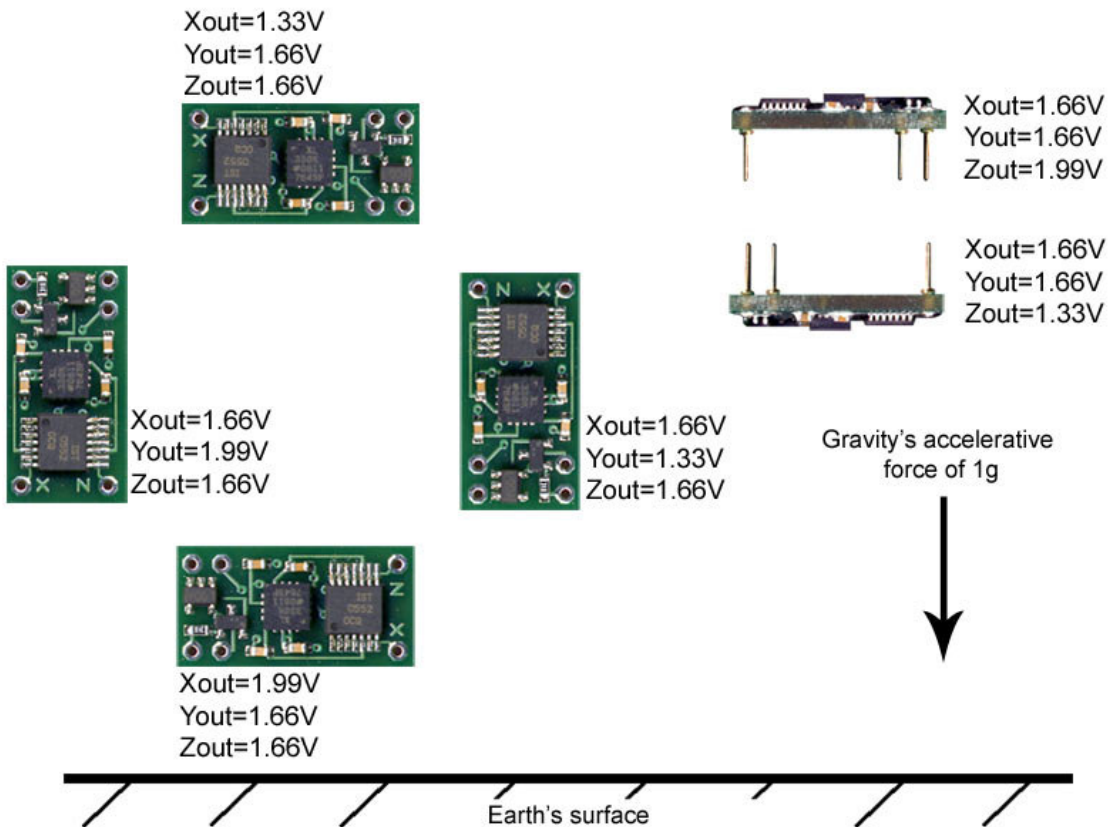
Shock sensing

Vehicle acceleration logging



Measuring acceleration and tilt

(Using onboard 3.3V supply)



The voltage outputs on the DE-ACCM3D correspond to acceleration being experienced in the X, Y and Z directions. The output is ratiometric, so the output sensitivity (in mV/g) will depend on the supply voltage.

Sensitivity and accuracy

Here are some typical sensitivity values for common operating voltages:

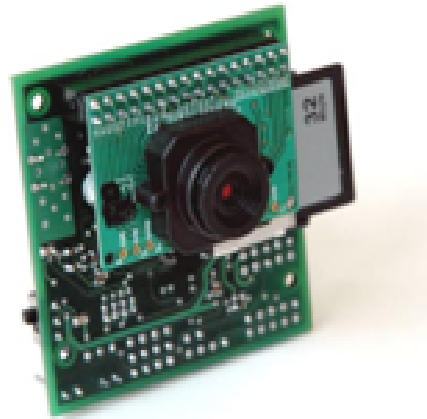
Operating voltage	Sensitivity
3.6V	360 mV/g
3.33V (default when using onboard regulator)	333 mV/g
3.0V	300 mV/g
2.0V	195 mV/g

Due to manufacturing variances when Analog Devices makes their accelerometer chips, these values aren't always set in stone. Sensitivity can vary by up to 10% in extreme cases, and the 0g bias point can vary up to 5% on the X and Y axes, and 10% on the Z axis. For projects that require a very high degree of accuracy, we recommend that you incorporate measured calibrations into your hardware/software.

CMUCAM3

Embedded Vision Processor

CMUcam3



CMUcam3 Datasheet

1. INTRODUCTION

The CMUcam3 is an ARM7TDMI based fully programmable embedded computer vision sensor. The main processor is the Philips LPC2106 connected to an Omnivision CMOS camera sensor module. Custom C code can be developed for the CMUcam3 using a port of the GNU toolchain along with a set of open source libraries and example programs. Executables can be flashed onto the board using the serial port with no external downloading hardware required. Make sure to check www.cmu.cam.org for the most up to date documentaton and Quick-Start guides.

Features

- § CIF Resolution (352x288) RGB color sensor
- § Open Source Development Environment for Windows and Linux
- § MMC Flash Slot with FAT16 driver support
- § Four-port Servo Controller
- § Load Images into Memory at 26 Frames Per Second
- § LUA light-weight language interpreter allows for rapid prototyping
- § Software JPEG compression
- § Basic Image Manipulation Library
 - Arbitrary Image Clipping
 - Image Downsampling
 - Mutable camera image properties
 - Threshold and Convolution Functions
 - RGB, YCrCb and HSV Color Space
- § CMUcam2 Emulation
 - User defined color blobs
 - Frame differencing
 - Mean and variance data collection
 - Raw images dumps over serial
 - Histogram Generation
- § B/W Analog video output (PAL or NTSC)*
- § FIFO image buffer for multiple pass hi-res image processing
- § Compatible Connector with Wireless Motes (Tmote Sky, FireFly, 802.15.4)

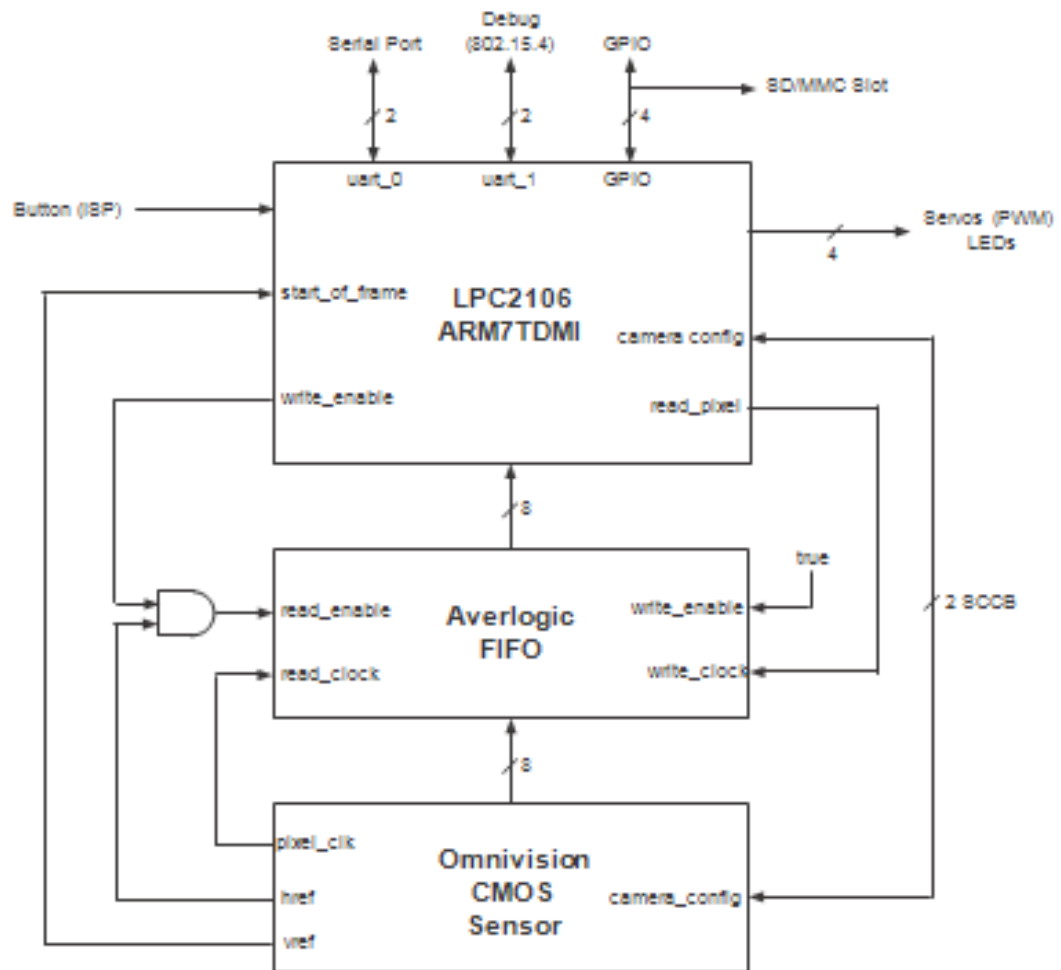
Applications

The CMUcam3 is a hardware platform coupled with an open source development environment. It is targeted toward users that are already familiar with basic image processing and who are comfortable with microcontroller programming. For users who want basic image processing accessible through a simple serial interface and do not wish to implement their own algorithms, you can install a flash utility and download CMUcam2 emulation firmware. This allows the CMUcam3 to mimic the interface provided by the CMUcam2. Refer to the CMUcam2 user manual available on the CMUcam website for more information about CMUcam2 firmware functionality. The CMUcam3 can be used for a variety of applications including: While CMUcam2 functionality can be duplicated using CMUcam3, the major benefit of CMUcam3 stems from the ability of advanced users to directly program the system with their specific algorithms, even starting from the openly available firmware for CMUcam3 that emulates a majority of the CMUcam2 functionality.

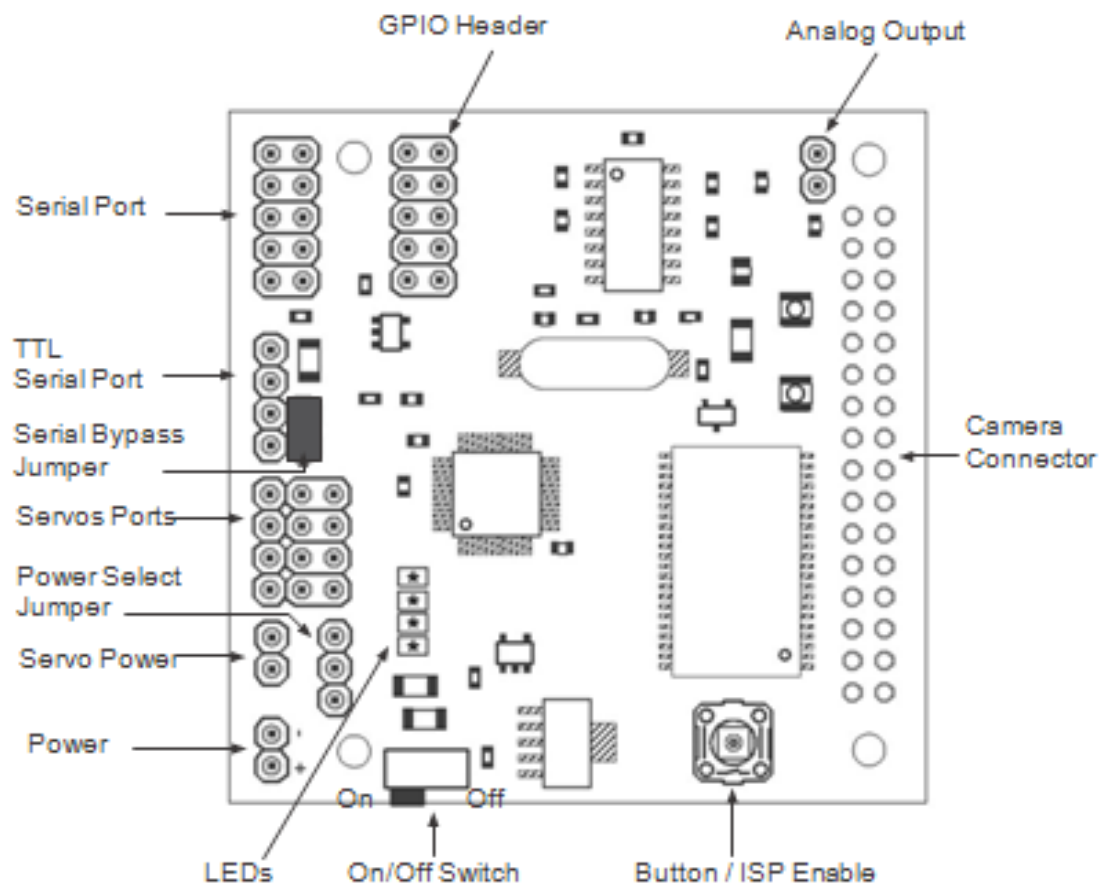
- ↳ Robotics
- ↳ Surveillance
- ↳ Sensor Networks
- ↳ Education
- ↳ Interactive Toys
- ↳ Object recognition and tracking
- ↳ Programmable Servo Control
- ↳ Serial MMC Flash Data Logging

2. BLOCK DIAGRAM

Below is a high level block diagram of the major components found on the CMUcam3.



3. HARDWARE CONNECTIONS

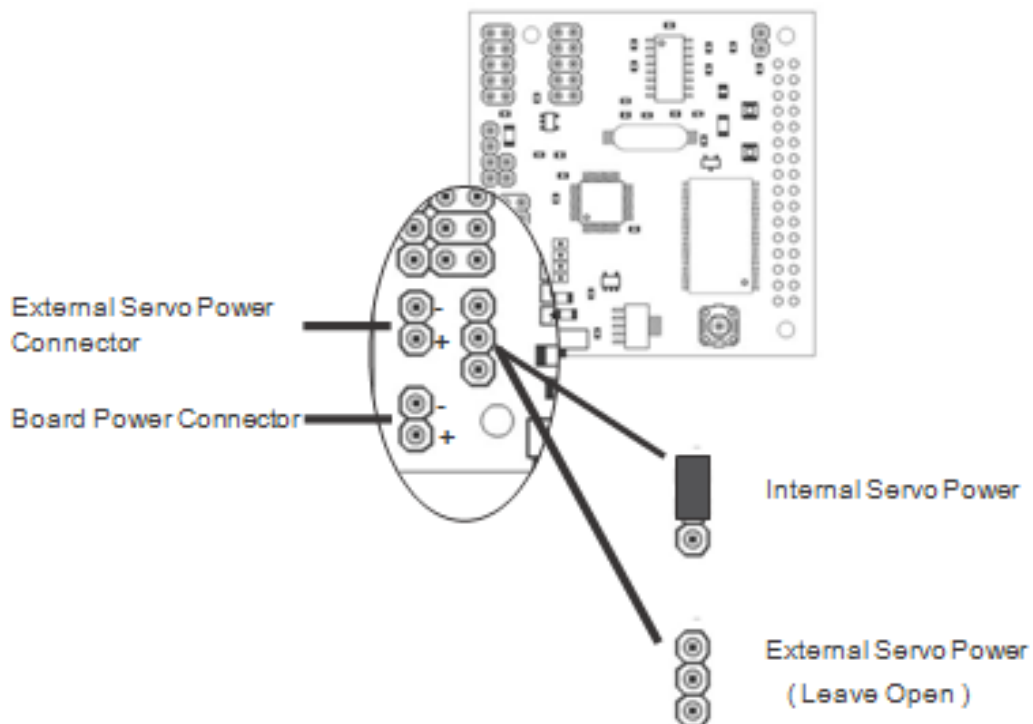


Power

The input power to the board goes through a 5 volt regulator. It is ideal to supply the board with between 8 and 15 volts of DC power that is capable of supplying at least 150 milliamperes of current.

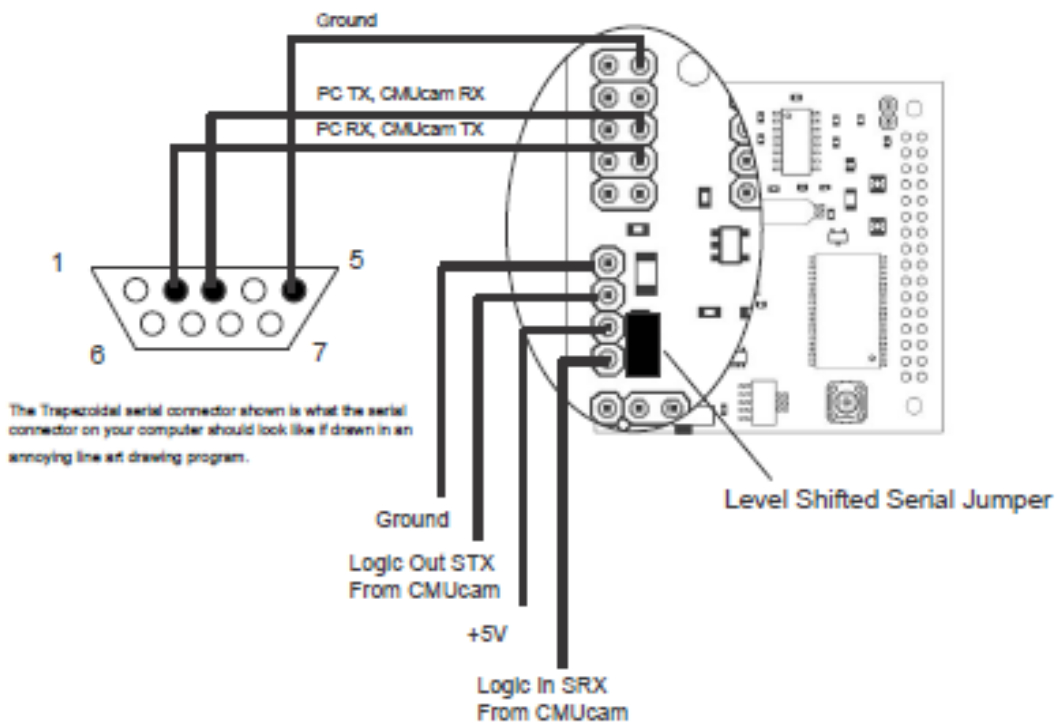
The servos can either be powered by internal power, or by the external servo power connector. To run them off of external power, remove the internal servo power jumper like shown below. Then connect a second power supply to the "External Servo Power Connector". To run off of internal power, connect the "Internal Servo Power" jumper and disconnect any external servo power.

Warning: powering the servos from internal power means that if the servos require more current than is available to power both servos and the processor, then the processor may reset or fail to operate at all. Running three or more servos off of internal power will likely not succeed.



Serial Port

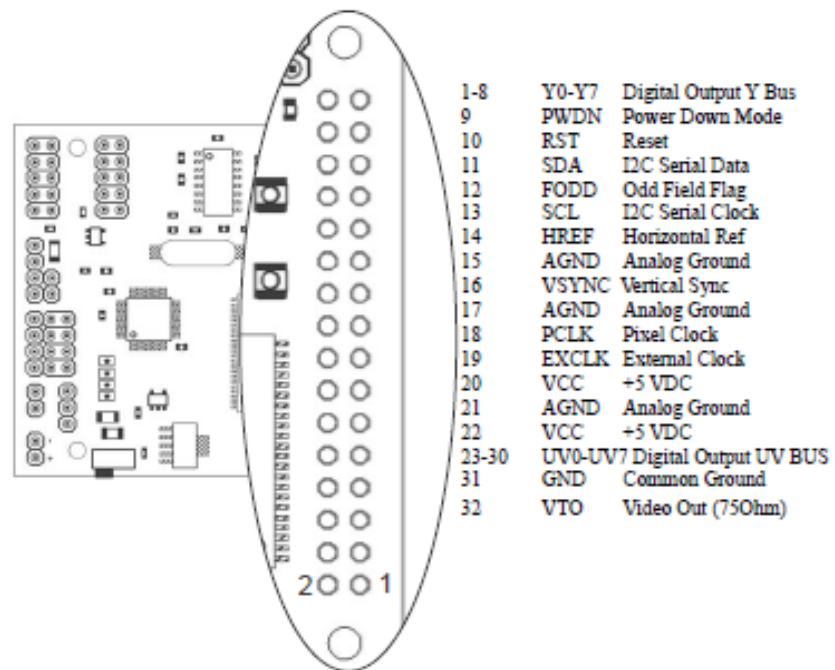
The CMUcam3 has a standard level shifted serial port to talk to a computer as well as a TTL serial port for talking to a microcontroller. The level shifted serial port only uses 3 of the 10 pins. It is in a 2x5 pin configuration that fits a standard 9 pin ribbon cable clip-on serial sockets and 10 pin female clip on serial headers that can both attach to a 10 wire ribbon cable. If this initially does not work, try flipping the direction that the ribbon cable connects to the CMUcam2 board. Make sure the serial jumper is in place when you use this mode. The TTL connector can be used to talk to a microcontroller without the use of a level shifting chip. The TTL pins output between 0 and 3.3volts, but are 5 volt tolerant for input. Remove the Serial Jumper when you use this mode.



Camera Bus

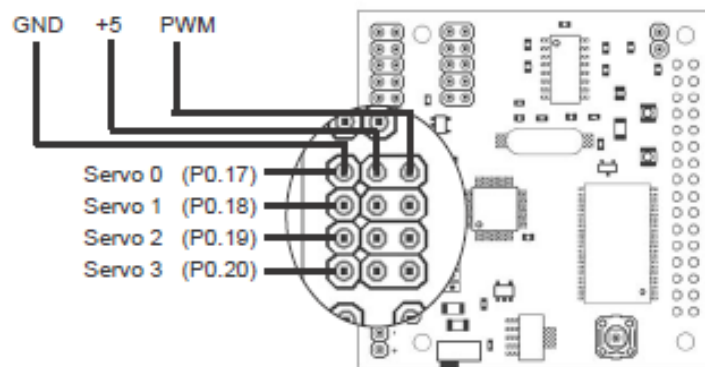
This bus interfaces with the CMOS camera chip. The CMOS camera board is mounted parallel to the processing part of the board and connects starting at pin 1. The female camera header should be soldered on the back of the main CMUcam3 board.

The CMUcam3 currently works with the OV6620 and OV7630 camera modules.



Servo Port

The CMUcam3 has the ability to control 4 servos. This can be useful if you do not wish to use a separate servo controller. The servo port can also be used as a general purpose digital outputs.

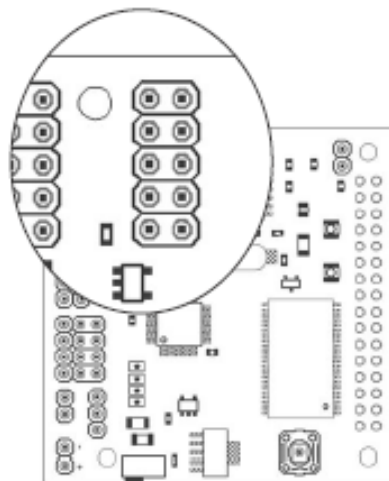
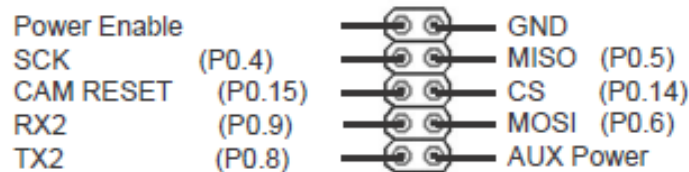


Expansion Port GPIO

The general purpose I/O header allows access to the second UART, various power control pins and the SPI pins.

Power Enable - When pulled low, the main CMUcam3 regulator is disabled causing the board to draw less than 0.01uA. All devices are shutoff and lose all active state information. When the line is released or pulled high, the board will reboot. By default, the pin is internally pulled high.

AUX Power - This pin can be configured to either externally power the board, or power an expansion board. By default, the pin is connected to the 3.3volt internal supply. By removing resistor R11 and adding a jumper resistor in place of R6, the pin is connected to the main power before the 5 volt regulator.



CAM RESET - This pin can be used as external I/O if the camera state is not required. Normally this pin resets the camera module and should not be used.

TX2 - The transmit pin on UART2 is not level shifted and hence can not be directly connected to a PC or none TTL external device. This pin can also be used as GPIO.

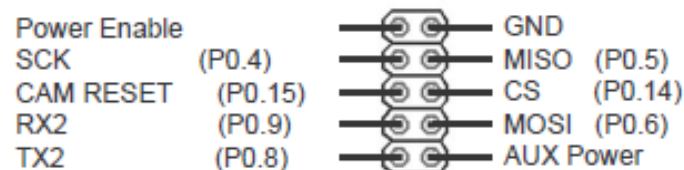
RX2 - The receive pin on UART2 is not level shifted and hence can not be directly connected to a PC or none TTL external device. This pin can also be used as GPIO.

CS - On reboot, if this pin is held low, the LPC2106 will enter bootstrap mode. Reboot can be externally induced by pulsing the power enable pin. Normally this pin will be controlled by the MMC driver. This pin can also be used as GPIO or as the SPI chip select when an MMC card is not inserted.

MOSI - Normally this pin is controlled by the MMC driver. This pin can also be used as GPIO or as an SPI output when an MMC card is not inserted.

MISO - Normally this pin is controlled by the MMC driver. This pin can also be used as GPIO or as an SPI input when an MMC card is not inserted.

SCK - Normally this pin is controlled by the MMC driver. This pin can also be used as GPIO or as the SPI clock pin when an MMC card is not inserted.



LEDs

LED 0 - This pin is shared with the MOSI pin and should be used only when CS is disabled while an MMC card is inserted.

LED 1 - This pin is shared with the Servo 2 pin. When using Servo 2, the LED will flash.

LED 2 - This pin is shared with the Servo 3 pin. When using Servo 3, the LED will flash.

