Implementasi Kamera OV7670 Sebagai Pendeteksi Garis Pada Robot *Line Follower*

Diajukan untuk Memenuhi Persyaratan

Memperoleh Gelar Sarjana Teknik

Disusun oleh :

MUHAMMAD RIZAL

NIM. 0910630081 - 63

KEMENTERIAN PENDIDIKAN DAN KEBUDAYAAN

JURUSAN TEKNIK ELEKTRO

FAKULTAS TEKNIK

UNIVERSITAS BRAWIJAYA

MALANG

2013

LEMBAR PERSETUJUAN

Implementasi Kamera OV7670 Sebagai Pendeteksi Garis pada Robot *Line Follower*

SKRIPSI

Diajukan untuk Memenuhi Persyaratan
Memperoleh Gelar Sarjana Teknik

Disusun oleh:

MUHAMMAD RIZAL

NIM. 0910630081 - 63

Telah diperiksa dan disetujui oleh :

Dosen Pembimbing I	Dosen Pembimbing II
Mochammad Rif'an,ST., MT.	**Waru Djuriatno,ST., MT.**
NIP. 19710301 200012 1 001	**NIP. 19690725 199702 1 001**

LEMBAR PENGESAHAN

IMPLEMENTASI KAMERA OV7670 SEBAGAI PENDETEKSI GARIS PADA ROBOT *LINE FOLLOWER*

Disusun oleh:
MUHAMMAD RIZAL
NIM. 0910630081-63

Skripsi ini telah diuji dan dinyatakan lulus pada
tanggal 17 Desember 2013

DOSEN PENGUJI

Ir. M. Julius St., MS.	**Ir. Nanang Sulistyanto, MT.**
NIP. 19540720 198203 1 002	**NIP. 19700113 199403 1 002**

Ir. Ponco Siwindarto, M.Eng.Sc.
NIP. 19590304 198903 1 001

Mengetahui,
Ketua Jurusan Teknik Elektro

M. Aziz Muslim,ST., MS., Ph.D.
NIP. 19741203 200012 1 001

PENGANTAR

Alhamdulillah, puji dan syukur penulis panjatkan kehadirat Allah SWT yang telah memberikan rahmat dan hidayah-Nya, sehingga penulis dapat menyelesaikan skripsi yang berjudul "Implementasi Kamera OV7670 Sebagai Pendeteksi Garis pada Robot *Line Follower*" dengan baik. Skripsi ini disusun sebagai salah satu syarat untuk mencapai gelar Sarjana Teknik dari jurusan Teknik Elektro Fakultas Teknik Universitas Brawijaya.

Penulis menyadari bahwa tanpa bantuan, bimbingan serta dorongan dari semua pihak, penyelesaian skripsi ini tidak mungkin bisa terwujud. Pada kesempatan ini penulis menyampaikan rasa terima kasih yang sebesar-besarnya kepada:

- Orangtua dan kakak penulis yang selalu memberikan kasih sayang, dukungan dan semangat,
- Bapak Aziz Muslim, ST,.MT.,Ph.D selaku Ketua Jurusan Teknik Elektro Universitas Brawijaya,
- Bapak Moch.Rif'an.,ST.,MT selaku Ketua Prodi Strata Satu Jurusan Teknik Elektro Universitas Brawijaya dan juga Dosen Pembimbing skripsi atas segala bimbingan, ide dan motivasi yang telah diberikan,
- Bapak Waru Djuriatno,ST.,MT sebagai Dosen Pembimbing skripsi atas segala bimbingan, pengarahan, ide, saran serta motivasi yang telah diberikan selama pengerjaan skripsi,
- Ibu Ir. Nurussa adah.,MT sebagai Ketua Kelompok Dosen Keahlian Elektronika Jurusan Teknik Elektro Universitas Brawijaya,
- Bapak . Ir. Sholeh Hadi Pramono, MS selaku Dosen Penasehat Akademik,
- Staff Recording Jurusan Teknik Elektro,
- Teman-teman Ampere angkatan 2009,
- Teman-teman Laboratorium SisDig dan Laboratorium Elka,
- Seluruh Keluarga Besar Tim Robot UB Jurusan Teknik Elektro periode 2011 - 2013 atas segala bantuan, dan pelajaran yang telah di berikan,

- Seluruh teman-teman serta semua pihak yang tidak mungkin untuk dicantumkan namanya satu-persatu, terima kasih banyak atas segala bentuk bantuan dan dukungannya.

Dalam penyusunan skripsi ini, penulis menyadari bahwa skripsi ini belumlah sempurna, karena keterbatasan ilmu dan kendala-kendala lain yang terjadi selama pengerjaan skripsi ini. Semoga tulisan ini dapat bermanfaat dan dapat digunakan untuk pengembangan lebih lanjut.

Malang, Desember 2013

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

ABSTRAK

Muhammad Rizal, Jurusan Teknik Elektro Fakultas Teknik Universitas Brawijaya, November 2013, *Implementasi Kamera OV7670 Sebagai Pendeteksi Garis pada Robot Line Follower*, Dosen Pembimbing: Mochammad Rif'an, ST, MT dan Waru Djuriatno, ST, MT.

Navigasi *line following* merupakan salah satu sistem navigasi *autonomous mobile robot* , dimana robot tipe *line follower* ini bergerak mengikuti lintasan dalam bentuk garis tebal. Tugas akhir ini merancang dan mengimplementasikan kamera OV7670 pada robot *line follower* beroda tipe *differential drive* untuk mendapat gambar dari lintasan. Dari gambar yang diperoleh dapat diketahui jarak antara robot dengan tengah lintasan (*l*) dan juga sudut hadap robot dengan lintasan (α). Digunakan Kontroler logika *fuzzy* untuk mengendalikan robot dengan masukan α dan *l*. Dengan bantuan kontroler Logika *fuzzy* robot *line follower* mampu bernavigasi dengan baik. Dengan menggunakan kamera OV7670 robot line follower dapat mendeteksi jarak robot (l) dan sudut simpangan robot (α) dengan kesalahan rata rata pembacaan (*l*) 6.954 % dan kesalahan rata rata pembacaan (α) 0.18%.

Kata Kunci : robot *line follower,* kontroler logika fuzzy, kamera OV7670, jarak robot dengan lintasan (*l*), sudut robot dengan lintasan (α)

BAB I

PENDAHULUAN

1.1 LATAR BELAKANG

Line follower merupakan salah satu jenis *autonomus mobile robot* atau robot yang mampu bergerak ke tujuan secara otomatis. *Mobile robot line follower* bergerak dengan cara mengikuti garis untuk mencapai tujuan (garis finish). Robot *line follower* dibedakan menjadi dua macam yaitu robot *line follower* dengan mikrokontroller dan tanpa mikrokontroller. Kelebihan dari robot *line tracer* dengan mikrokontroller yaitu robot dapat diprogram pergerakannya sesuai dengan keinginan.

Pada robot *line tracer* ini pada umumnya digunakan sensor cahaya yang menangkap besar nya intesitas cahaya yang di terima dari sebuah lampu LED yang di pantulkan ke alas. Perbedaan intesitas terjadi bila lampu LED di pantulkan ke alas dengan warna terang dan warna gelap. Alas dengan warna terang lebih banyak memantulkan cahaya dibanding dengan alas berwarna gelap.

Kelemahan dari sensor ini yaitu sensor hanya dapat mengerti jarak antara robot dengan garis, itupun dengan ketelitian yang terbatas, tergantung dari jumlah sensor yang dipakai dan peletakannya.. Hal ini menyebabkan robot line follower tidak mengetahui besar simpangan antara robot dengan garis dan juga kurangnya ketelitian dalam membaca posisi garis. Salah satu solusi untuk mengatasi masalah ini yaitu dengan mengganti sensor pendeteksi garis yang dapat mengetahui apa yang ada di depan robot. Salah satu sensor yang dapat menunjang hal tersebut yaitu dengan menggunakan sensor kamera.

Maka dalam skripsi ini dirancanglah implementasi kamera OV7670 sebagai pendeteksi garis pada robot *line follower*. Sistem ini dirancang supaya dapat mengatasi masalah sebelumnya, sehingga robot *line follower* dapat mengetahui simpangan robot dengan garis dan jarak dari robot ke garis secara

teliti. Kendali logika *fuzzy* diperlukan untuk mengontrol pergerakan robot sehingga robot dapat stabil mengikuti garis.

1.2 RUMUSAN MASALAH

Berdasarkan latar belakang yang telah diuraikan di atas, maka pembahasan skripsi ini ditekankan pada :

1) Bagaimana merancang sistem akses kamera OV7670 dengan mikrokontroller untuk melakukan pengaturan kamera dan untuk memperoleh data gambar.
2) Bagaimana pengaruh tata letak kamera terhadap gambar yang akan didapatkan.
3) Bagaimana merancang sistem pengolahan data gambar yang diterima oleh mikrokontroller yang akan digunakan sebagai masukkan sistem kendali robot *line follower*.
4) Bagaimana merancang dan membuat suatu sistem pengendalian menggunakan kontrol logika *fuzzy* sebagai pengontrol kestabilan robot dalam mengikuti garis.
5) Bagaimana mengaplikasikan kontroler tersebut pada pergerakan robot *line follower*.

1.3 BATASAN MASALAH

Dengan mengacu pada permasalahan yang telah dirumuskan, maka hal-hal yang berkaitan dengan alat diberi batasan sebagai berikut :

1) Motor yang dipakai adalah motor DC magnet permanen.
2) Robot menggunakan kamera OV7670.
3) Robot menggunakan *development board* STM32F4 Discovery sebagai pengolah data dan pengendali robot.
4) Robot bersistem gerak 2WD atau torsi dari motor disalurkan ke 2 roda belakang sebelah kanan dan kiri.
5) Lapangan pengujian hanya memiliki dua warna, yaitu hitam dan putih. Warna putih sebagai alas dan warna hitam sebagai jalur yang akan diikuti oleh robot *line follower*.

1.4 TUJUAN

Tujuan skripsi ini adalah untuk merancang dan membuat robot line follower yang dalam mendeteksi garisnya digunakan kamera OV7670, sehingga diharapkan dapat menghasilkan navigasi *line following* yang lebih akurat.

1.5 SISTEMATIKA PENULISAN

Sistematika penulisan dalam penelitian ini sebagai berikut:

BAB I Pendahuluan

Memuat latar belakang, rumusan masalah, ruang lingkup, tujuan, dan sistematika pembahasan.

BAB II Tinjauan Pustaka

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

BAB III Metodologi

Berisi tentang metode-metode yang dipakai dalam melakukan perancangan, pengujian, dan analisis data.

BAB IV Perancangan

Perancangan dan perealisasian alat yang meliputi spesifikasi, perencanaan blok diagram, prinsip kerja dan realisasi alat.

BAB V Pengujian dan Analisis

Memuat aspek pengujian meliputi penjelasan tentang cara pengujian dan hasil pengujian. Aspek analisis meliputi penilaian atau komentar terhadap hasil-hasil pengujian. Pengujian dan analisis ini terhadap alat yang telah direalisasikan berdasarkan masing-masing blok dan sistem secara keseluruhan.

BAB VI Kesimpulan dan Saran

Memuat intisari hasil pengujian dan menjawab rumusan masalah serta memberikan rekomendasi untuk perbaikan kualitas penelitian dimasa yang akan datang.

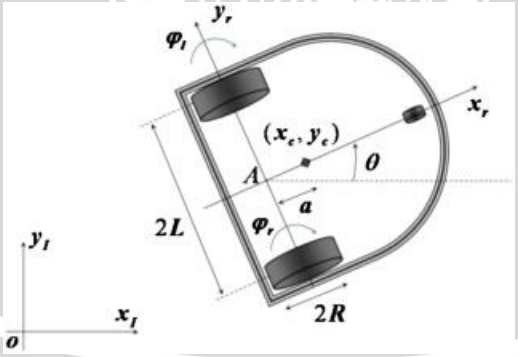
BAB II

TINJAUAN PUSTAKA

Beberapa teori pendukung yang perlu dibahas dalam pembuatan sistem ini meliputi literatur mengenai *mobile robot* sistem *diferential drive*, *motor dc brushed*, kamera OV7670, komunikasi SCCB, STM32F4 Discovery, kontrol logika *fuzzy*.

2.1 *Mobile Robot* sistem *diferential drive*

Mobile Robot sistem diferensial merupakan robot mobil yang bergerak dengan menggunakan dua buah roda yang dapat digerakkan secara terpisah. Roda diletakkan pada kedua sisi robot. Robot dapat merubah arah pergerakannya dengan mengatur putaran pada kedua roda. Kedua roda ini berfungsi sebagai penggerak sekaligus sebagai kemudi robot mobil. Oleh karena itu pada robot ini tidak diperlukan pengendali arah gerakan tambahan. Sehingga dalam sistem *differential drive* dibutuhkan minimal dua buah motor sebagai penggerak roda-roda tersebut. Dengan mengkombinasikan kecepatan putaran motor-motornya, robot dapat dikemudikan lurus, membentuk lengkungan (*curve*), dan juga berputar (*rotation*) di tempat. *Mobile robot* sistem *differntial drive* dan koordinat referensi ditunjukkan dalam Gambar 2.1.



Gambar 2.1 *Mobile robot* Tipe *Differntial Drive* dan Koordinat Referensi.
Sumber : http://www.sciencedirect.com

Untuk menganalisis kinematika *mobile robot* dengan sistem *differential drive*, maka robot dimodelkan menjadi suatu rangka statis di atas roda, yang mana robot akan beroperasi pada bidang horizontal. Pada bidang tersebut rangka robot memiliki tiga dimensi yaitu dua posisi dalam bidang dan satu untuk orientasi sepanjang sumbu vertikal yang ortogonal terhadap bidang. Kenyataannya tentu akan ditemui banyak derajat kebebasan dan fleksibilitas karena as roda, *wheel steering joints*, serta *wheel castor joints*. Namun dalam hal ini rangka robot yang dimodelkan hanya mengacu pada rangka statis robot itu sendiri, sehingga dapat dikatakan mengabaikan sendi (*joints*) dan derajat kebebasan (*degree of freedom*) internal terhadap robot dan roda-rodanya.

2.2 Motor DC *Brushed*

Prinsip kerja motor DC *brushed* sesuai dengan hukum kemagnetan Lorenz, yaitu membangkitkan fluksi magnet pada suatu konduktor berarus dalam medan magnet sehingga timbul ggl induksi.

Setiap arus yang mengalir melalui sebuah konduktor akan menimbulkan medan magnet. Arah medan magnet dapat ditentukan dengan kaidah tangan kiri.

Kaidah tangan kiri untuk motor menunjukkan arah arus yang mengalir didalam sebuah konduktor yang berada dalam medan magnet. Jari tengah menunjukkan arah arus yang mengalir pada konduktor, jari telunjuk menunjukkan arah medan magnet dan ibu jari menunjukkan arah gaya putar. Adapun besarnya gaya yang bekerja pada konduktor tersebut dapat dirumuskan dengan :

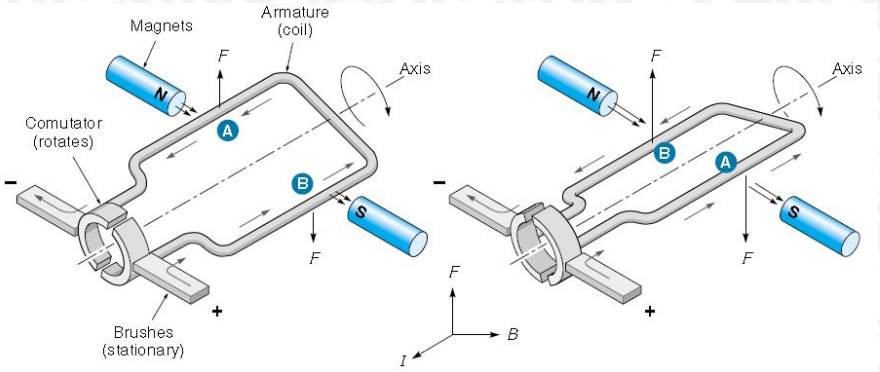
$F = B.I.L$ (Newton) (2.1)

Dimana :

B = kerapatan fluks magnet (weber)

L = panjang konduktor (meter)

I = arus listrik (ampere)



Gambar 2.2. Ilustrasi Motor DC Brushed
Sumber : Kilian, 2002

Ilustrasi cara kerja motor DC yang mempunyai satu lilit kawat a–b berada di dalam medan magnet ditunjukkan pada Gambar 2.2. Lilitan ini dapat berputar dengan bebas, lilitan ini biasa disebut dengan jangkar (*armature*).

Pada jangkar diberikan arus yang berasal dari sumber yang terhubung dengan sikat (*brushes*). Sikat-sikat ini terpasang pada sebuah cincin yang terbelah dua, yang disebut cincin belah (*comutator*). Adapun tujuan dari konstruksi ini adalah agar lilitan kawat dapat berputar apabila ada arus listrik yang melewatinya.

Pada kawat yang berada di kanan arus mengalir dari depan ke belakang. Pada kawat yang berada di bagian kiri, arus mengalir dari belakang ke depan kawat a dan b secara bergantian berada di kiri dan kanan. Karena itu arah arus di a dan arah arus di b selalu bersifat bolak-balik. Pembalikan arah arus itu terjadi pada saat lilitan kawat melintasi posisi vertikal.

Bagian *comutator* berfungsi sebagai penyearah mekanik. Fluksi magnet yang ditimbulkan magnet permanen disebut medan magnet motor. Dalam gambar 2.2. arah fluk magnetik adalah dari kiri ke kanan. Adapun gaya yang bekerja pada penghantar b adalah ke atas, sementara gaya yang bekerja pada penghantar a adalah ke bawah . Gaya-gaya yang bekerja sama kuatnya, sehingga terdapat kopel yang bekerja pada kawat sehingga lilitan jangkar dapat berputar. Setelah berputar 180° arah arus berbalik, pada saat itu penghantar a dan penghantar b bertukar tempat. Akibatnya arah gerak putaran tidak berubah.

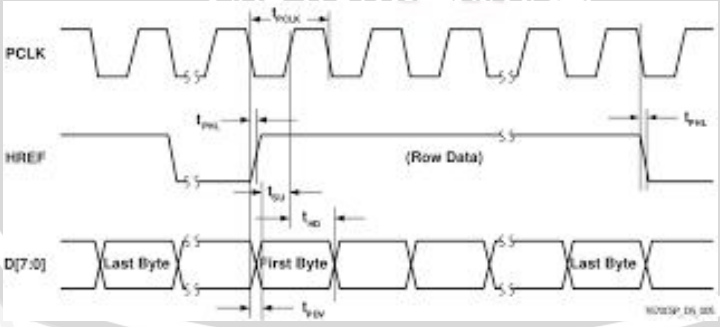
2.3 Kamera OV7670

Sensor kamera OV7670 yang dipakai dalam skripsi ini merupakan produksi dari OmniVision. Modul kamera OV7670 terdiri atas chip tunggal VGA dan juga pemroses gambar. Kamera ini menyediakan format gambar *full frame* dan juga *windowed*. Kamera ini dapat beroperasi sampai 30 fps (*frame per second*) dan pengguna dapat mengatur format dan kualitas gambar melalui antarmuka SCCB (*Serial Camera Control Bus*).

Gambar 2.3. Modul kamera OV7670

Sumber : OmniVision, 2006

Kamera OV7670 mengirim data secara *parallel synchronous*. Untuk mendapat data dari modul kamera, pin XCLK di pin modul ini harus di berikan masukkan *clock* antara 10 – 48 MHz. Setelah pin XCLK diberi masukkan *clock* maka modul kamera OV7670 akan mulai menjalankan VSYNC, HREF dan D0-D7.

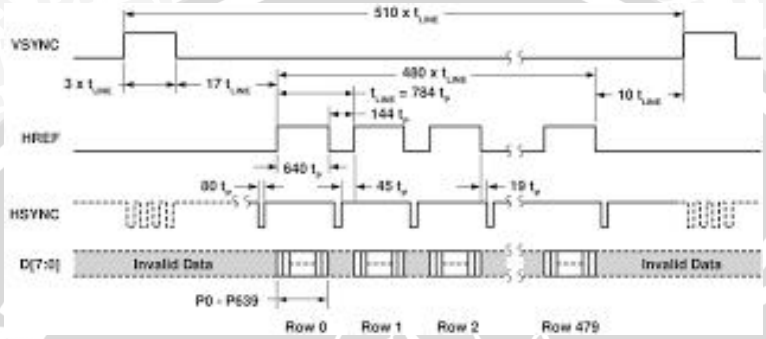


Gambar 2.4. Horizontal Timing kamera OV7670

Sumber : OmniVision, 2006

Data D0 – D7 diambil oleh mikrokontroller saat tepi naik dari sinyal PCLK, dan saat sinyal dari pin HREF berlogika *high* yang ditunjukkan Gamber

2.4. Tepi naik dari sinyal HREF menandai mulainya suatu baris, dan tepi turun dari sinyal HREF menandai akhir dari baris. Semua byte data yang tersampel saat HREF berlogika *high* merupakan data semua piksel dalam satu baris, satu byte data bukan berarti satu piksel, semua tergantung dari format yang telah diatur.



Gambar 2.5. VGA frame timing kamera OV7670
Sumber : OmniVision, 2006

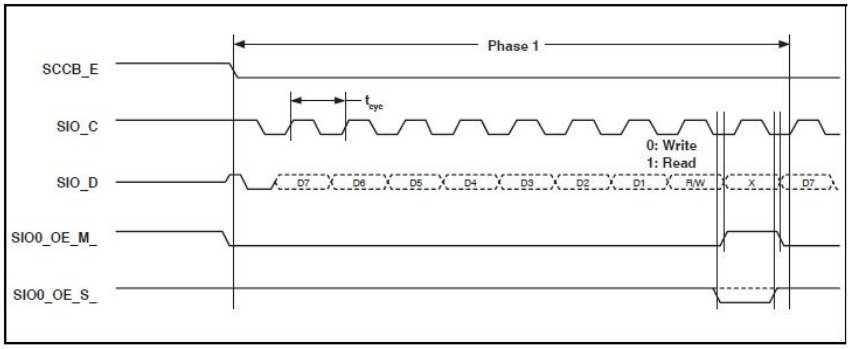
Gambar 2.5 menunjukkan *timing diagram* dari OV7670 dengan format VGA. Yang berarti saat HREF berlogika *high* terdapat 640 piksel yang harus didapat. Satu frame didapat saat VSYNC berlogika *low*. Tepi turun dari VSYNC menunjukkan awal dari frame dan tepi naik dari VSYNC menunjukkan akhir dari sebuah frame.

2.4 SCCB (*Serial Camera Control Bus*)

SCCB merupakan komunikasi *serial synchronous* dari OmniVision yang dipakai di modul kamera OV7670. Terdapat tiga jalur untuk komunikasi SCCB, tapi di dalam OV7670 hanya memakai dua jalur saja yaitu SIO_D dan SIO_C. yang membedakan komunikasi tiga jalur dan dua jalur adalah adanya sinyal *enable* yang harus diaktifkan ketika *master* akan berkomunikasi dengan *slave*. Sedangkan untuk dua jalur *slave* selalu siap untuk berkomunikasi dengan *master*.

Komunikasi SCCB terdiri atas 3 fasa, fasa pertama yaitu pengenalan alamat, dimulai dengan *master* melakukan kondisi *start* supaya *slave* tahu akan ada transmisi data. Lalu *master* mengirim 8 bit data, 7 bit pertama adalah alamat dari *slave*, dan bit terakhir merupakan perintah yang akan dilakukan yaitu baca

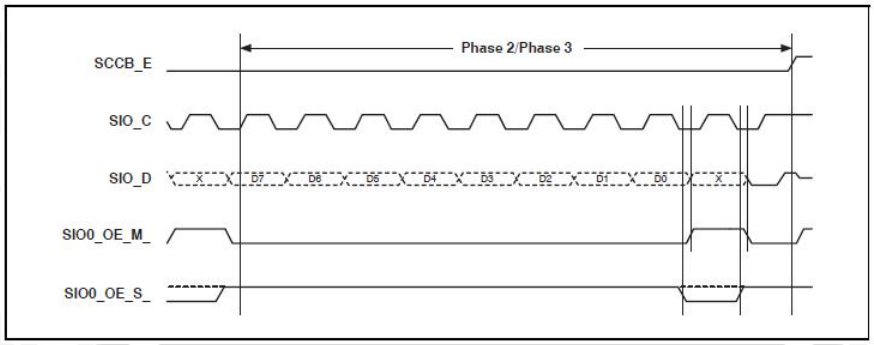
atau tulis seperti yang ditunjukkan dalam Gambar 2.6. Jika alamat *slave* benar maka *slave* akan memberikan penanda dengan memberikan logika *low* pada SIO_D saat bit ke-9 dan komunikasi akan dilanjutkan ke tahap dua, sedangkan jika alamat *slave* salah maka *slave* akan memberi penanda logika *high* pada SIO_D saat bit ke-9. Lalu *master* melakukan kondisi stop untuk mengakhiri komunikasi.



Gambar 2.6. Timing diagram pengenalan alamat
Sumber : OmniVision, 2003

Saat komunikasi memasuki fasa kedua, *master* tidak lagi memulai dengan melakukan kondisi *start*, tapi akan langsung menstransmisikan 8 bit data, yang berupa alamat register yang akan dihubungi. Jika alamat register itu ada maka pada bit ke-9 *slave* akan memberikan logika *low* di SIO_D, dan begitu pula sebaliknya.

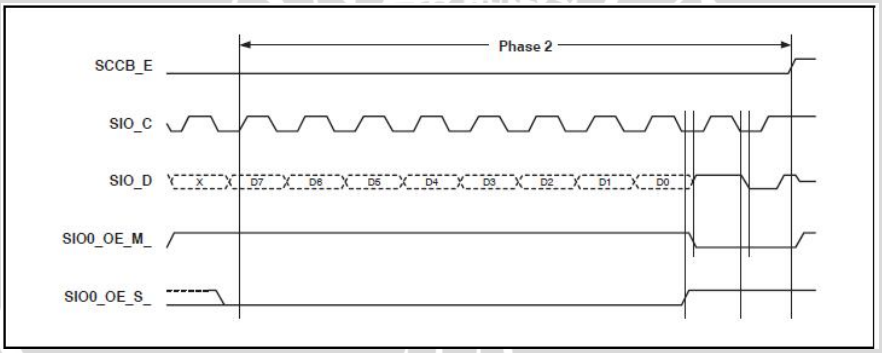
Untuk proses tulis, dilanjutkan dengan mengirim 8 bit data yang akan diisikan ke register yang telah dihubungi pada saat fasa kedua. Setelah 8 bit terkirim maka akan ada penanda dari *slave* pada bit ke-9. Setelah itu komunikasi diakhiri dengan *master* memberikan kondisi stop.



Gambar 2.7. Timing diagram pengiriman data

Sumber : OmniVision, 2003

Untuk proses baca, maka *master* akan memulai lagi mengirim 8 bit data sebagai alamat dan perintah untuk membaca, jika bit ke-9 bernilai 0 maka akan dilanjutkan ke pengambilan data. Saat pengambilan data, *slave* akan mengirim 8 bit data yang tersimpan di register yang telah ditulis di fase kedua. Setelah itu *slave* memberi sinyal *high* pada SIO_D saat bit ke-9, lalu master melakukan kondisi stop untuk menandai selesainya komunikasi.



Gambar 2.8. Timing diagram penerimaan Data

Sumber : OmniVision, 2003

2.5 STM32F4 Discovery

STM32F4 Discovery merupakan suatu *development board* buatan STMicroelectronics yang dapat membantu untuk mempelajari fitur fitur dari mikrokontroller dan untuk pengembangan aplikasi. STM32F4 Discovery memilik beberapa fitur sebagai berikut :

- Mikrokontroler STM32F407VGT6 yang memiliki 1 MB memori *flash*, 192 KB RAM yang dikemas dengan kemasan LQFP100.
- ST-LINK/V2 yang bisa juga digunakan sebagai alat debug yang mandiri.
- *Board* catu daya melalui kabel USB atau catu 5V dari luar.
- Catu daya 3V dan 5V untuk aplikasi lain.
- LIS30DL, ST MEMS *motion sensor*, 3-axis *accelerometer* digital.
- MIP45DT02, ST MEMS *audio sensor,* mikrofon digital *omnidirectional.*
- CS43L22, audio DAC yang terintegrasi *speaker driver* kelas D.
- 8 buah LED.
- 2 tombol.
- USB OTG dengan konekor micro-AB.
- Header sambungan untuk LQFP100 I/O.

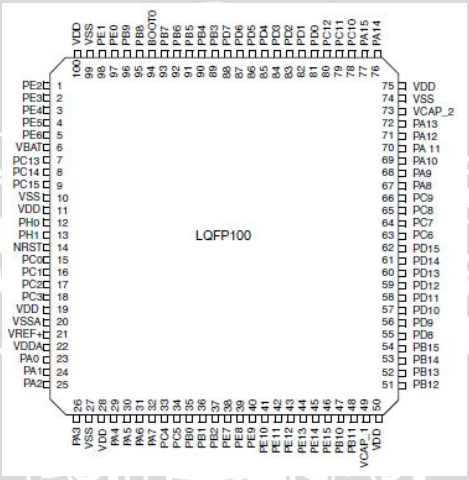
Gambar 2.9. *Development Board* STM32F4 Discovery

Sumber : STMicroelectronics, 2012

2.5.1 Mikrokontroller STM32F407VGT6

Mikrokontroller merupakan sebuah permrosess data yang di dalamnya sudah terdapat mikroprosessor, memori dan perangkat - perangkat lain yang menjadi satu kesatuan dalam satu chip.

Mikrokontroller STM32F407VGT6 adalah mikrokontroller CMOS 32-bit dengan arsitektur RISC buatan STMicroelectronics keluarga ARM Cortex-M4 dengan *clock* sampai dengan 168 MHz, mampu mengeksekusi perintah sampai dengan 210 MIPS (*Million Instruction per Second*). Mikrokontroller ini memiliki 1 MByte Flash PEROM (*Flash Programmable and Eraseble Read Only Memory*), 192 Kbyte SRAM, 100 pin, 5 buah port I/O yang mana setiap pin dalam masing masing port dapat diprogram tersendiri, memiliki dua belas buah *timer/counter* 16 bit dan dua buah *timer/counter* 32 bit, memiliki antarmuka kamera secara parallel 8-14 bit dengan kecepatan sampai 54 MByte/s, memiliki 24 channel ADC (*Analog Digital Converter*).



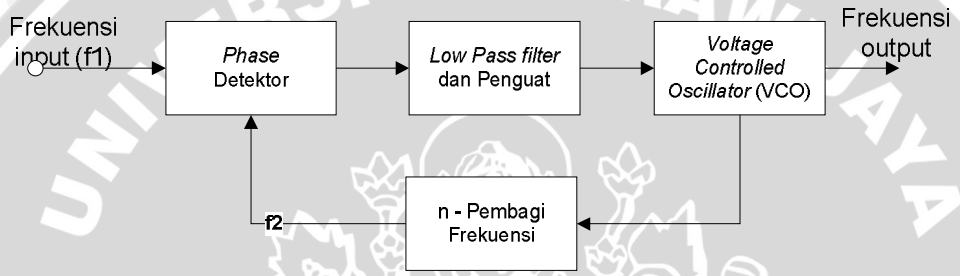
Gambar 2.10. Konfigurasi pin STM32F407VGT6

Sumber: STMicroelectronics, 2012

2.5.1.1 *Phase Locked Loop* (PLL)

Blok diagram *Phase Locked Loop* (PLL) ditunjukkan pada Gambar 2.11. Sebuah sinyal input yang memiliki frekuensi sebesar f1 masuk ke *phase* detektor. Untuk dibandingkan dengan frekuensi f2 dari pembagi frekuensi. dimana sinyal ini diperoleh dari sinyal keluaran VCO yang frekuensinya telah dibagi dengan *n*. Pada kondisi awal f1 ≠ f2 karena frekuensi dari VCO = 0 Hz. Karena ada perbedaan frekuensi antara f1 dan f2, maka rangkaian *phase* detektor akan menghasilkan sinyal yang nantinya akan mencatu VCO setelah melewati filter dan penguat. Tegangan ini menyebabkan rangkaian VCO berosilasi dan menghasilkan

sebuah frekuensi f2. Rangkaian VCO akan terus berosilasi dan menghasilkan frekuensi f2 sampai f2 = f1. Ketika f2 = f1, maka tegangan masukan dari VCO = 0 dan ini menyebabkan rangkaian VCO berhenti berosilasi (*locked*). Karena rangkaian loop ini akan mengunci (*locked*) saat frekuensi dan fase dari kedua sinyal sama, maka rangkaian ini disebut dengan *Phase Locked Loop*. Karena besarnya f2 = f1 dan f2 merupakan sinyal dari VCO yang di bagi dengan *n*. Maka keluaran dari rangkaian PLL atau keluaran dari VCO memiliki frekuensi *n* kali frekuensi input.



Gambar 2.11. Blok diagram PLL
Sumber: Harsono, Nonot 1993

2.5.1.2 *General Purpose Input/Output* (GPIO)

Masing – masing pin dari GPIO bisa dijadikan input, output dan fungsi alternatif. Bila pin dijadikan sebagai output ada beberapa konfigurasi yang dapat dilakukan secara software, yaitu jenis output *push pull* atau *open drain*, dengan atau tanpa *pull-up/pull-down*. Jika digunakan sebagai input maka dapat dikonfigurasi sebagai *floating* , *pull-up* dan *pull –down*. Untuk fungsi alternatif setiap pin GPIO memiliki fungsi tersendiri, salah satunya yaitu sebagai pin untuk ADC. Kecepatan I/O untuk berosilasi maksimum sampai 84 Mhz.

2.5.1.3 Pengendali DMA

Direct Memory Acces (DMA) merupakan suatu teknik perpindahan data dari suatu alamat memori ke alamat memori yang lain tanpa mengganggu kerja dari mikroprosessor. Pada mikrokontroller STM32F407VGT6 terdapat 2 port DMA, yang masing masing port memiliki 8 aliran data yang sanggup menangani

perpindahan dari memori ke memori, *peripheral* ke memori dan memori ke *peripheral*. DMA dapat diintegrasikan dengan *peripheral* :

- SPI (*Serial Peripheral Interface*)
- I2C (*Inter Integrated Circuit*)
- USART (*Universal Synchronous Asynchronous Receiver Transmitter*)
- *Timer*
- DAC (*Digital to Analog Converter*)
- SDIO (*Secure Digital Input/Output*)
- Antarmuka Kamera (DCMI)
- ADC (*Analog to Digital Converter*)

2.5.1.4 Antarmuka Kamera (DCMI)

Mikrokontroller STM32F407VGT6 memiliki antarmuka kamera yang dapat terhubung dengan kamera melalui 8 bit sampai 14 bit antarmuka parallel. Untuk menerima data video kecepatan transfer data bisa sampai 54 MByte/s. fitur yang terdapat pada antarmuka kamera antara lain :

- Polaritas *clock* piksel input dan sinyal sinkronisasi yang dapat diprogram.
- Komunikasi parallel data dapat diatur 8, 10, 12 atau 14 bit.
- Mendukung format Monokrom, YcbCr 4:2:2, RGB 565 dan data yang terkompres (seperti JPEG).
- Dapat diatur mode *snapshot* dan *continous*.
- Kemampuan untuk memotong gambar otomatis.

2.6 Kendali Logika *Fuzzy*

Fuzzy secara harfiah berarti samar, sedangkan kebalikannya dalam hal ini adalah C*risp* yang secara harfiah berarti tegas. Dalam kehidupan sehari-hari nilai samar lebih akrab daripada nilai tegas. Temperatur/suhu tertentu biasa dinyatakan sebagai panas, agak panas, atau sangat dingin daripada dinyatakan dalam nilai terukur tertentu.

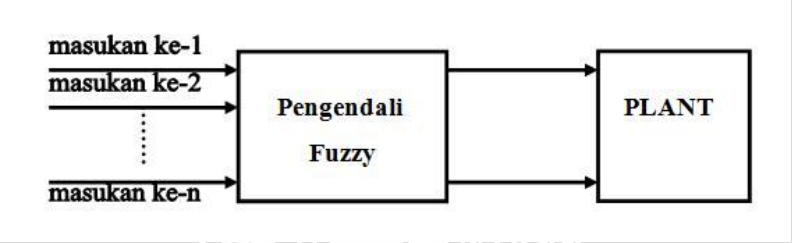
Tahun 1965 L.A. Zadeh memodifikasi teori himpunan yang disebut himpunan kabur (*Fuzzy Set*). Himpunan *fuzzy* didasarkan pada gagasan untuk

memperluas jangkauan fungsi karakteristik sehingga fungsi tersebut akan mencakup bilangan real pada interval [0,1]. Nilai keanggotaannya menunjukkan bahwa suatu nilai dalam semesta pembicaraan tidak hanya berada pada 0 atau 1, namun juga nilai yang terletak diantaranya. Dengan kata lain nilai kebenaran suatu hal tidak hanya bernilai benar atau salah. Nilai 0 menunjukkan salah, nilai 1 menunjukkan benar dan masih ada nilai-nilai yang terletak diantaranya.

Sejak tahun 1985 pengendalian berbasis logika *fuzzy* mengalami perkembangan pesat, terutama dalam hubungannya dengan penyelesaian masalah kendali yang bersifat tak linier, sulit dimodelkan, berubah karakteristiknya terhadap waktu (*time varying*) dan kompleks.

2.6.1 Struktur Dasar Kontrol Logika Fuzzy

Dalam sistem pengendalian dengan logika *fuzzy* dilibatkan suatu blok pengendali yang menerima satu atau lebih masukan dan mengumpankan satu atau lebih keluaran ke plant atau blok lain sebagaimana ditunjukan dalam Gambar 2.12.



Gambar 2.12. Pengendali Fuzzy
Sumber : *Donald Coughanowr,1991*

Komponen utama penyusun *Fuzzy Logic Controller* adalah unit fuzzifikasi, *fuzzy inference*, basis pengetahuan dan unit defuzzifikasi.

Basis pengetahuan terdiri dari dua jenis (Yan : 94) :

- Basis data

  Mendefinisikan parameter *fuzzy* sebagai bagian dari himpunan *fuzzy* dengan menentukan batas-batas fungsi keanggotaan pada semesta pembicaraan untuk tiap-tiap variabel.

- Basis aturan

Memetakan nilai masukan *fuzzy* menjadi nilai keluaran *fuzzy*.

2.6.2 Fungsi Keanggotaan

Fungsi keanggotaan menotasikan nilai kebenaran anggota-anggota himpunan *fuzzy*. Interval nilai yang digunakan untuk menentukan fungsi keanggotaan, yaitu nol dan satu. Tiap fungsi keanggotaan memetakan elemen himpunan *crisp* ke semesta himpunan *fuzzy*.

Suatu himpunan *fuzzy* A dalam semesta pembicaraan U dinyatakan dengan fungsi keanggotaan, μ_A yang harganya berada dalam interval [0,1] (Kuswadi, 2000:27). Secara matematika hal ini dinyatakan dengan :

$$\mu_A : U \rightarrow [0,1] \tag{2.2}$$

Berikut ini beberapa macam keanggotaan yang sering digunakan antara lain fungsi keanggotaan S, π, T (*triangular*).

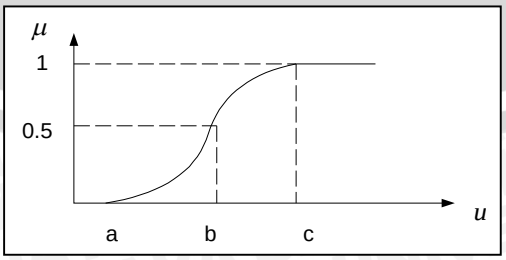
1. Fungsi keanggotaan bentuk S

   Definisi fungsi-S adalah sebagai berikut :

$$S(u,a,b,c) = \begin{cases} 0 & u < a \\ 2\left(\dfrac{u-a}{c-a}\right)^2 & a \le u \le b \\ 1-2\left(\dfrac{u-a}{c-a}\right)^2 & b \le u \le c \\ 1 & u > c \end{cases} \tag{2.3}$$

$$b = \frac{(a+c)}{2}$$

Bentuk fungsi keanggotaan bentuk S ditunjukkan dalam Gambar 2.13.



Gambar 2.13. Fungsi Keanggotaan Bentuk S
***Sumber** : Jun Yan,1994 : 18*

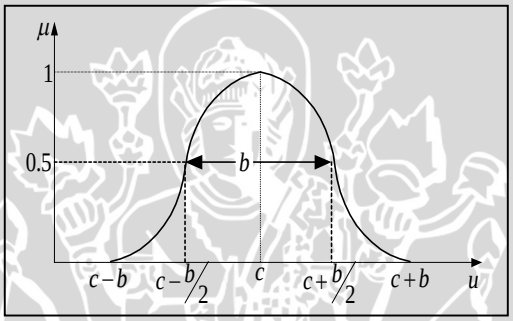
Fungsi keanggotaan bentuk S ini digunakan bila diinginkan himpunan *fuzzy* mempunyai nilai kebenaran mendekati nol dan satu lebih banyak.

2. Fungsi keanggotaan bentuk π

Definisi fungsi-π adalah sebagai berikut:

$$\pi(u;a,b,c) = \begin{cases} S(u;c-b,c-\frac{b}{2},c) & u \le c \\ 1-S(u;c,c+\frac{b}{2},c+b) & u \ge c \end{cases} \quad (2.4)$$

Bentuk fungsi keanggotaan bentuk π ditunjukkan dalam Gambar 2.14.



Gambar 2.14. Fungsi Keanggotaan Bentuk π
***Sumber** : Jun Yan, 1994 : 19*

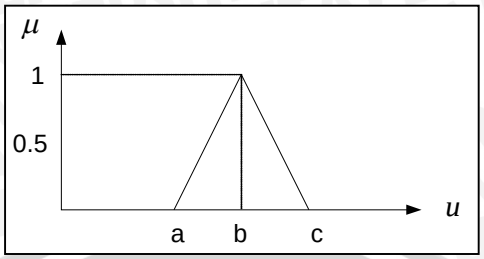
Fungsi keanggotaan bentuk π digunakan jika diinginkan elemen himpunan *fuzzy* memiliki nilai kebenaran mendekati nol lebih banyak.

3. Fungsi keanggotaan bentuk Triangular

Definisi fungsi triangular sebagai berikut:

$$T(u;a,b,c) = \begin{cases} 0 & u < a \\ \frac{u-a}{b-a} & a \le u \le b \\ \frac{c-u}{c-b} & b \le u \le c \\ 0 & u > c \end{cases} \quad (2.5)$$

Fungsi keanggotaan bentuk Tringular ditunjukkan dalam Gambar 2.15



Gambar 2.15. Fungsi Keanggotaan Bentuk Triangular (T)
***Sumber** : Jun Yan, 1994 : 19*

Fungsi keanggotaan bentuk triangular ini digunakan bila diinginkan himpunan *fuzzy* mempunyai nilai proporsional terhadap nol maupun satu.

2.6.3 Kendali Logika *Fuzzy*

Kendali logika *fuzzy* adalah sistem berbasis aturan (*rule based system*) yang didalamnya terdapat himpunan aturan *fuzzy* yang mempresentasikan mekanisme pengambilan keputusan. Aturan yang dibuat digunakan untuk memetakan variabel input ke variabel output dengan pernyataan *If - Then.*

Kontroler ini akan menggunakan data tertentu *(crisp)* dari sejumlah sensor kemudian mengubahnya menjadi bentuk linguistik atau fungsi keanggotaan melalui proses fuzzifikasi. Lalu dengan aturan *fuzzy*, *inference engine* yang akan menentukan hasil keluaran *fuzzy*. Setelah itu hasil ini akan diubah kembali menjadi bentuk numerik melalui proses defuzzifikasi.

2.6.3.1 Fuzzifikasi

Proses fuzzifikasi merupakan proses untuk mengubah variabel non *fuzzy* (variabel numerik) menjadi variabel *fuzzy* (variabel linguistik). Nilai masukan-masukan yang masih dalam bentuk variabel numerik yang telah dikuantisasi sebelum diolah oleh pengendali logika *fuzzy* harus diubah terlebih dahulu ke dalam variabel *fuzzy*. Melalui fungsi keanggotaan yang telah disusun, maka dari nilai-nilai masukan tersebut menjadi informasi *fuzzy* yang berguna nantinya untuk proses pengolahan secara *fuzzy* pula. Proses ini disebut fuzzifikasi (Yan,1994:49). Proses fuzzifikasi diekspresikan sebagai berikut:

x = *fuzzifier* (x_0)

dengan:

x_0 = nilai *crisp* variabel masukan

x = himpunan *fuzzy* variabel yang terdefinisi

fuzzifier = operator fuzzifikasi yang memetakan himpunan *crisp* ke himpunan *fuzzy*

Pedoman memilih fungsi keanggotaan untuk proses fuzzifikasi, menurut Jun Yan, menggunakan :

1. Himpunan *fuzzy* dengan distribusi simetris.
2. Gunakan himpunan *fuzzy* dengan jumlah ganjil, berkaitan erat dengan jumlah kaidah (*rules*).
3. Mengatur himpunan *fuzzy* agar saling menumpuk.
4. Menggunakan fungsi keanggotaan bentuk segitiga atau trapesium.

2.6.3.2 Kaidah Atur *Fuzzy* (*Fuzzy Rule*)

Fuzzy rule adalah bagian yang menggambarkan dinamika suatu sistem terhadap masukan yang dikarakteristikan oleh sekumpulan variabel-variabel linguistik dan berbasis pengetahuan seorang operator ahli. Pernyataan tersebut umumnya dinyatakan oleh suatu pernyataan bersyarat.

Dalam pengendali berbasis *fuzzy*, aturan pengendalian *fuzzy* berbentuk aturan " IF – THEN". Untuk sebuah sistem MISO *(Multi Input Single Output)* basis aturan pengendalian fuzzy berbentuk seperti berikut ini.

Rule 1 IF X_1 is A_{11} AND … AND x_m is A_{1m} THEN Y is B_1

Rule 2 IF X_1 is A_{21} AND … AND x_m is A_{2m} THEN Y is B_2

. .

. .

. .

Rule n IF X_n is A_{n1} AND … AND x_m is A_{nm} THEN Y is B_n

Dengan X_j merupakan variabel masukan sistem , A_{ij} merupakan *fuzzy set* untuk X_j , Y merupakan variabel keluaran sistem, B_i merupakan *fuzzy set* untuk Y , AND adalah operator *fuzzy*.

2.6.3.3 Metode Inferensi

Metode inferensi merupakan proses untuk mendapatkan keluaran dari suatu kondisi masukan dengan mengikuti aturan-aturan yang telah ditetapkan. Keputusan yang didapatkan pada proses ini masih dalam bentuk *fuzzy* yaitu derajat keanggotaan keluaran.

Ada beberapa cara untuk mengidentifikasi aturan mana yang akan dipakai dengan menggunakan nilai keanggotaan masukan. Menurut Jun Yan, diantara bermacam-macam metode inferensi *fuzzy* ada dua metode yang paling sering digunakan pada kendali logika *fuzzy* yaitu :

1. Metode Inferensi MAX – MIN

   Pada metode Max – Min aturan operasi minimum Mamdani digunakan untuk implikasi *fuzzy*. Persamaan aturan minimum adalah

   $$\mu_{C'} = \bigcup_{1}^{n} \alpha_i \wedge \mu ci \qquad (2.6)$$

   dengan $\alpha_i = \mu_{Ai}(x_0) \wedge \mu_{Bi}(y_0)$

   Sebagai contoh , terdapat dua basis kaidah atur *fuzzy*, yaitu :

   R_1 : Jika x adalah A_1 dan y adalah B_1 maka z adalah C_1

   R_2 : Jika x adalah A_2 dan y adalah B_2 maka z adalah C_2

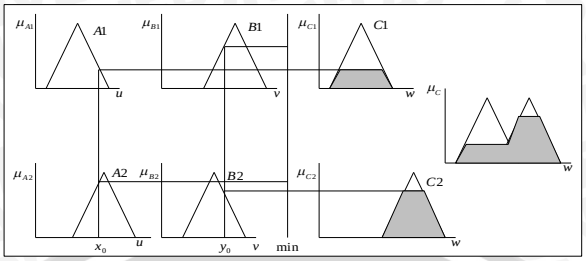
   Pada metode penalaran MAX-MIN fungsi keanggotaan konsekuen dinyatakan dengan

   $$\mu c_{1'}(W) = \mu_{c'1} \vee \mu_{c'2} = [\alpha_1 \wedge \mu_{c1}(w)] \vee [\alpha_2 \wedge \mu_{c2}(w)] \qquad (2.7)$$

   dimana $\alpha_1 = \mu_{A1}(x_0) \wedge \mu_{B1}(y_0)$ (2.8)

   $$\alpha_2 = \mu_{A2}(x_0) \wedge \mu_{B2}(y_0) \qquad (2.9)$$

   Lebih jelas metode ini dideskripsikan dalam Gambar 2.16.



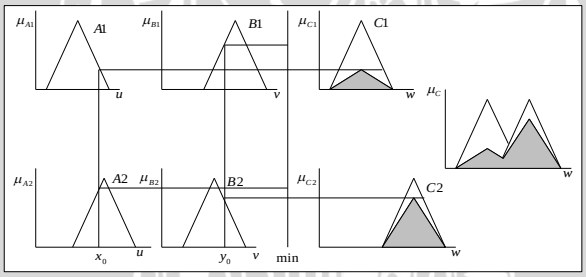
Gambar 2.16. Inferensi Fuzzy dengan Metode MAX-MIN
***Sumber** : Jun Yan, 1994 : 55*

2. Metode INFERENSI MAX – DOT

Pada metode inferensi Max – Dot, aturan operasi Larsen digunakan untuk fungsi implikasi *fuzzy*. Metode MAX-DOT fungsi keanggotaan konsekuen C dinyatakan dengan:

$$\mu_c(\omega) = (\alpha_1 \bullet \mu_{c1}(\varpi)) \vee (\alpha_2 \bullet \mu_{c2}(\varpi)) \qquad (2.10)$$

Metode penalaran MAX-DOT ditunjukan dalam Gambar 2.17.



Gambar 2.17. Inferensi Fuzzy dengan Metode MAX-DOT
***Sumber** : Jun Yan, 1994 : 55*

2.6.3.4 Defuzzifikasi

Defuzzifikasi adalah proses untuk mendapatkan nilai numerik dari data *fuzzy* yang dihasilkan dari proses inferensi (Yan, 1994 : 55). Proses defuzzifikasi dinyatakan sebagai berikut :

$$y_0 = defuzzifier(y) \qquad (2.11)$$

dengan:

y : aksi kontrol *fuzzy*.

y_0 : aksi kontrol *crisp*.

defuzzifier : operator *defuzzifikasi*

Dua metode defuzzifikasi yang umum digunakan adalah :

1. Metode (*Center Of Area*)

   Metode ini didefinisikan sebagai berikut:

$$U = \frac{\sum_{i=1}^{n} w_i u_i}{\sum_{i=1}^{n} w_i} \qquad (2.12)$$

   dengan:

   U = Keluaran

   w_i = Bobot nilai benar w_i

   u_i = Nilai linguistik pada fungsi keanggotaan keluaran

   n = Banyak derajat keanggotaan

2. Metode (*Mean of Maximum*).

   Metode ini didefinisikan sebagai berikut:

$$Z = \frac{\sum_{i=1}^{n} z_i}{n} \qquad (2.13)$$

   dengan:

   n = Jumlah proposisi / domain

   Z = Keluaran

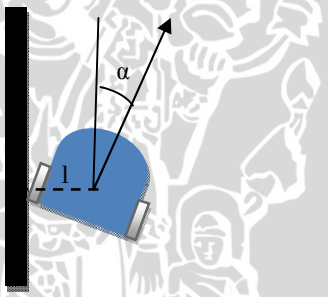
   z_i = Domain yang memiliki derajat keanggotaan terbesar / maksimum.

BAB III
METODOLOGI PENELITIAN

Penyusunan proposal ini didasarkan pada masalah yang bersifat aplikatif, yaitu perencanaan dan perealisasian alat agar dapat bekerja sesuai dengan yang direncanakan dengan mengacu pada rumusan masalah. Langkah-langkah yang perlu dilakukan untuk merealisasikan alat yang dirancang adalah penentuan spesifikasi alat, studi literatur, perancangan sistem, pengujian alat, dan pengambilan kesimpulan.

3.1. Penentuan Spesifikasi Alat

Spesifikasi alat secara global ditetapkan terlebih dahulu sebagai acuan dalam perancangan selanjutnya.



Gambar 3.1. Ilustrasi keadaan robot terhadap garis

Gambar 3.1. menunjukkan ilustrasi dari robot saat berada di lintasan garis dimana terdapat sudut simpangan antara arah hadap robot terhadap garis yang diwakilkan dengan nilai α. Simpangan inilah yang akan diatur sehingga nilainya mendekati 0. Jarak antara robot terhadap garis atau *l* juga diatur supaya nilainya mendekati 0. spesifikasi alat yang direncanakan adalah sebagai berikut :

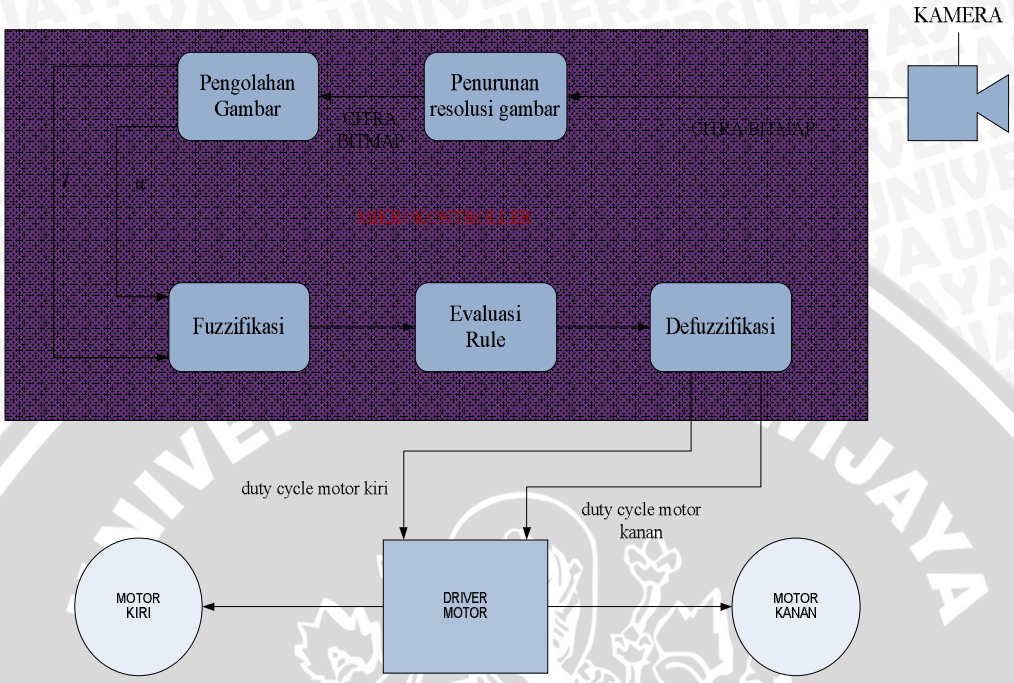
1) Menggunakan dua buah motor DC 250 rpm sebagai aktuator.
2) Jari-jari roda *mobile* robot sebesar 3 cm dan tebal 1 cm.
3) Dimensi robot berukuran panjang 18 cm dan lebar 16 cm
4) Kamera memiliki sudut tangkap vertikal 25° dan sudut tangkap horizontal 33°.

5) Kamera diletakkan didepan, mengarah ke bawah dengan sudut 25° dari sumbu vertikal dengan tinggi 14.5 cm dari alas.
6) Mikrokontroler sebagai pengendali utama *mobile* robot.
7) Mobile robot menggunakan sistem gerak diferensial.
8) Gambar yang diambil oleh modul kamera beresolusi QCIF (176x144) piksel.
9) Mobile robot bernavigasi dengan kecepatan 30 cm/s dan memiliki ketelitian 2°.
10) Sudut simpangan (α) yang dapat dideteksi robot minimal 2° dan jarak antara garis dan robot yang dapat dideteksi minimal 3mm.

3.2. Studi Literatur

Studi literatur dilakukan untuk mempelajari teori penunjang sistem yang dibutuhkan dalam perencanaan dan pembuatan alat. Teori yang diperlukan antara lain berkaitan dengan *mobile robot* sistem *diferential drive,* motor dc *brushed*, kamera OV7670, komunikasi SCCB, STM32F4 Discovery, kontrol logika *fuzzy*.

3.3. Perancangan Sistem

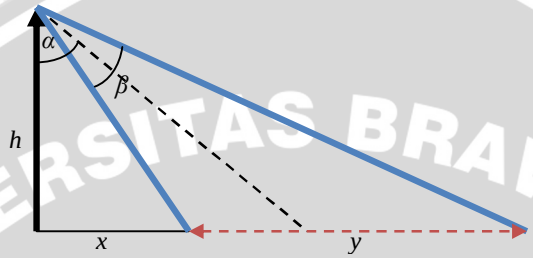


Gambar 3.2. Diagram blok sistem secara keseluruhan

Prinsip kerja sistem ditunjukkan pada Gambar 3.2. Awalnya kamera akan menangkap gambar berupa citra bitmap yang kemudian dikirim ke mirokontroller. Karena citra bitmap yang diterima memiliki resolusi yang besar, maka dilakukanlah penurunan resolusi gambar. Citra bitmap yang telah diturunkan resolusinya akan diolah supaya didapatkan nilai dari sudut simpangan (α) dan juga jarak garis terhadap robot (*l*). Kedua nilai ini menjadi masukkan dari kontroller logika *fuzzy*. Kedua masukkan berupa α dan *l* diproses dengan fuzzifikasi untuk memperoleh variabel linguistik. Dari variabel linguistik yang diperoleh, didapatkan aturan yang berlaku. Lalu dilakukan proses defuzzifikasi untuk menentukan presentase kecepatan sudut. Dari presentasi tersebut didapat *duty cycle* untuk motor kanan dan kiri. Sinyal dari mikrokontroller kemudian dikirim ke *driver* motor sebagai antarmuka mikrokontroller dengan motor.

3.3.1. Penempatan Kamera

Penempatan kamera sangat berpengaruh terhadap navigasi *line following*, karena penempatan kamera menentukan bidang tangkap kamera yang diproyeksikan dalam bentuk gambar.



Gambar 3.3. Bidang tangkap vertikal kamera

Gambar 3.3. menunjukkan bidang tangkap kamera dengan kamera yang menghadap kebawah dengan sudut α. dimana :

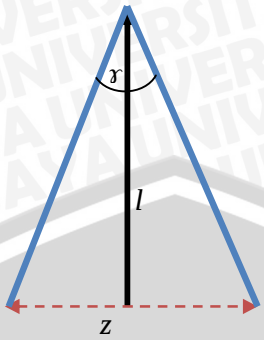
h	: Tinggi kamera
x	: Jarak antara kamera dengan bidang tangkap kamera
y	: Panjang bidang tangkap vertikal dari kamera
α	: Sudut kamera terhadap normal.
β	: Sudut tangkap vertikal dari kamera.

Dari Gambar 3.3 diatas dapat dihitung nilai *x* dan *y* jika *h, α,* dan *β* sudah ditentukan dimana:

$$x = \mathrm{h} \times \tan\left(\alpha - \left(\frac{1}{2} \times \beta\right)\right)$$

$$y = \left(\mathrm{h} \times \tan\left(\alpha + \left(\frac{1}{2} \times \beta\right)\right)\right) - x$$

Untuk bidang tankap horizontal :



Gambar 3.4. Bidang tangkap horizontal kamera

Gambar 3.4 menunjukkan bidang tangkap horizontal kamera dimana :

l = jarak tegak lurus dari kamera ke alas

$ɤ$ = sudut tangkap horizontal dari kamera

z = panjang bidang tangkap horizontal dari kamera

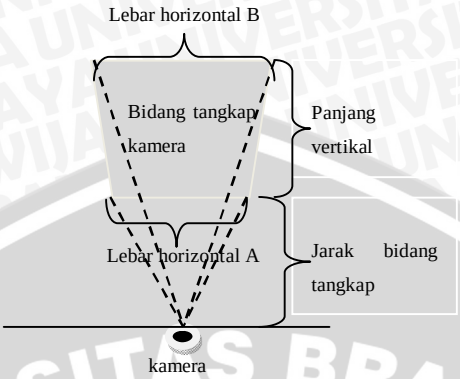
Untuk menentukan panjang bidang tangkap horizontal dari kamera (z) dapat dihitung :

$$z = 2 \times \left(l \times \tan\left(\frac{1}{2} ɤ\right)\right)$$

Sedangkan nilai l didapatkan bila diketahui tinggi kamera (h) dan sudut kamera terhadap sumbu vertikal (α) dengan persamaan :

$$l = \frac{h}{\cos \alpha}$$

Berdasarkan spesifikasi alat kamera akan dipasang dengan ketinggian 14.5 cm dengan sudut kemiringan (α) 25°. Pada spesifikasi alat ditentukan sudut tangkap vertikal kamera sebesar 25° dan sudut tangkap horizontal kamera sebesar 33°. Gambar 3.5. menunjukkan ilustrasi bidang tangkap dari kamera.



Gambar 3.5. Bidang tangkap kamera dari atas

Dengan sudut kemiringan sebesar 25° didapat bidang tangkap kamera yang berbentuk trapesium dimana letak sisi yang lebih pendek berada paling dekat dengan robot dan sisi yang paling panjang berada yang paling jauh dari robot. Dengan ketentuan tersebut maka bidang tangkap kamera berada cm dari 3.21 cm di depan kamera, memiliki panjang vertikal 7.91 cm, lebar horizontal a yaitu 8.85 cm dan lebar horizontal b yaitu 10.82 cm.

3.3.2. Pengolahan gambar

Gambar yang ditangkap kamera akan diisimpan di dalam memori dari mikrokontroller. Ukuran gambar sesuai dengan resolusi dari kamera. Karena kamera di atur memiliki resolusi QCIF (176x144), dan gambar yang diambil memiliki format YCbCr 4:2:2 maka satu piksel diwakili oleh dua byte data. Satu byte mewakili data keabuan (Y) dan satu byte mewakili data Cb atau Cr nya. Ukuran satu gambar yang di tampung oleh mikrokontroller sebesar 176 x 144 x 2 byte, atau 50688 byte. Data tersebut disimpan ke dalam sebuah *array* dari integer 8 bit yang berarti nilai dari tiap anggotanya bisa bernilai 0 – 255. Karena dalam pengolahannya yang di pakai hanya data keabuan nya saja. Ada 25344 byte data yang merupakan data keabuan setiap piksel.

$$x(n) = \begin{pmatrix} x_1 & x_3 & \cdot & \cdot & x_{351} \\ x_{353} & \cdot & \cdot & \cdot & x_{703} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ x_{50337} & \cdot & \cdot & \cdot & x_{50687} \end{pmatrix}$$

Susunan data tersebut menunjukkan bahwa nilai dari x_1 merupakan nilai keabuan dari piksel pertama, x_3 merupakan keabuan piksel kedua dan x_{351} merupakan keabuan dari piksel ke 176. Nilai keabuan dari piksel ke-n yaitu pada $x_{(n\,x\,2)-1}$. Nilai x_1 sampai x_{351} menunjukkan data keabuan pada baris pertama. Gambar yang beresolusi QCIF(176 x 144) piksel ini akan diturunkan resolusinya sesuai dengan ketelitian yang diinginkan. Gambar lintasan yang akan diambil memiliki ukuran panjang 7.91 cm, lebar horizontal A (Bagian paling dekat dengan robot) 8.85 cm dan lebar horizontal B (Bagian paling jauh dengan robot) 10.82 cm. yang diproyeksikan dalam gambar beresolusi QCIF. Karena di inginkan kemiringan minimal garis yang dapat dideteksi oleh robot 2°, maka resolusi diturunkan supaya dalam satu piksel pada baris pertama memiliki lebar 2.76 mm. Ukuran lebar tersebut didapat dengan perhitungan :

$$lebar = \tan(2°) \times panjang$$

panjang merupakan panjang gambar yang ditangkap yaitu 7.91 cm. karena ukuran piksel bukanlah dalam bentuk *real*, maka supaya mendekati nilai lebar yang diinginkan resolusi diturunkan menjadi 44 x 24. Sehingga dalam satu piksel baris pertama memiliki ukuran panjang 3.59 mm dan lebar 2.45 mm. Dalam proses penurunan resolusi setiap 4 kolom dan 6 baris menjadi satu piksel atau setiap 24 piksel menjadi 1 piksel.

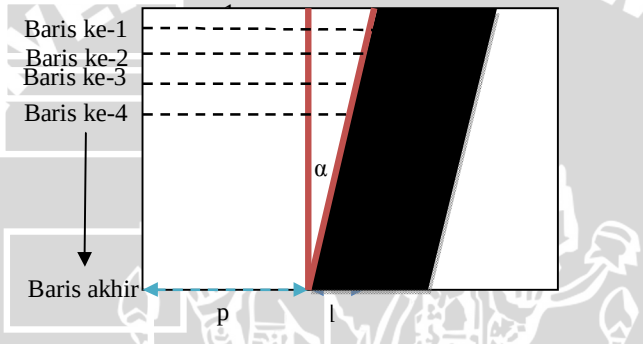
$$\begin{pmatrix} x_0 & \cdot & x_3 \\ x_{176} & \cdot & x_{179} \\ \cdot & \cdot & \cdot \\ x_{880} & \cdot & x_{883} \end{pmatrix} menjadi\ (y_0)$$

Untuk memperoleh nilai dari y_0 ditentukan dulu nilai nilai dari matriks pembentuknya di klasifikasi kedalam 2 jenis saja yaitu gelap dan terang.

$$x_n = \begin{cases} Gelap, & x_n < batas \\ Terang, & x_n \geq batas \end{cases}$$

Jika dalam matriks lebih banyak terdapat gelap daripada terang maka nilai y_0 bernilai gelap, dan sebaliknya jika dalam matriks pembentuk lebih banyak terdapat nilai terang.

Setelah didapat gambar dengan resolusi yang lebih kecil, maka gambar akan diolah untuk memperoleh sudut simpangan dan jarak robot terhadap garis.



Gambar 3.6. Ilustrasi hasil tangkapan kamera

Gambar 3.6. merupakan ilustrasi dari lintasan yang tertangkap oleh kamera, dimana:

p = jarak dari tepi ke sumbu pusat pada baris paling akhir

k = sumbu pusat

a(n) = jarak dari tepi baris ke-n ke koordinat tepi garis pada baris ke-n

b(n) = jarak dari baris ke-n ke baris paling akhir

α = sudut simpangan terhadap sumbu pusat (k)

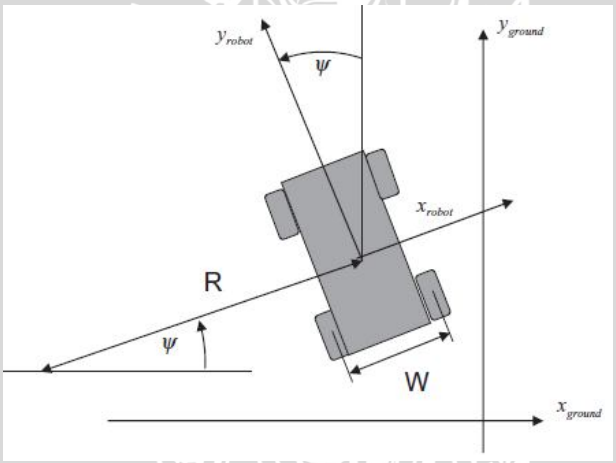
l = jarak sumbu pusat ke tengah garis pada baris paling akhir

Sumbu pusat (k) merupakan garis vertikal yang terletak di tengah bidang gambar. Jika satu baris terdiri dari 44 piksel maka letak sumbu pusat yaitu pada piksel ke 22. Masukkan pertama untuk kontroller logika fuzzy yaitu jarak antara

pusat robot dengan tengah garis (l). Untuk mendapatkan nilai dari jarak ini digunakan data pada baris terakhir. jarak (l) merupakan jarak sumbu pusat dengan tengah garis pada baris terakhir. Masukkan lain dari kontroller logika *fuzzy* yaitu berupa sudut simpangan (α). Setiap baris memiliki sudut simpangan terhadap sumbu pusatnya (k), karena bidang tangkap kamera berupa trapesium maka setiap baris memiliki lebar piksel yang berbeda. Untuk mempercepat proses pengolahan data, yang digunakan adalah sudut simpangan dari salah satu baris. Sudut dari baris ke-n bisa diketahui :

$$Sudut\ baris\ ke-n = archtan\left(\frac{a(n) - p}{b(n)}\right)$$

3.3.3. Kinematika robot



Gambar 3.7. Robot dengan sistem *differential drive*

Menurut Gerald Cook pada bukunya yang berjudul "*Mobile Robot, Navigation, Control and Remote Sensing*" pada robot dengan sistem penggerak *differential drive*, ω merupakan kecepatan sudut yang dikarenakan perbedaan kecepatan roda kanan dan kiri, dan W merupakan jarak antar roda. Berdasarkan Gambar 3.7 jika nilai ω positif maka robot berbelok ke kiri dan jika nilai ω negatif maka robot berbelok ke arah kanan. Hubungannya yaitu:

$$\omega = \frac{V_{kanan} - V_{kiri}}{W}$$

R merupakan jari – jari pada gerak melingkar robot yang disebabkan oleh perbedaan kecepatan.

$$R = \frac{V_{kiri}}{\omega} + \frac{W}{2}$$

Dan kecepatan bidang y merupakan perkalian antara ω dan R. Pada spesifikasi alat sudah ditentukan bahwa robot dapat bernavigasi pada kecepatan 30 cm/s. nilai ini merupakan nilai dari V_y yang dianggap konstan. Didapatkan hubungan antara kecepatan roda kiri dengan perubahan kecepatan.

$$V_y = \omega \times R$$

$$\frac{V_y}{\omega} = \frac{V_{kiri}}{\omega} + \frac{W}{2}$$

$$V_{kiri} = \frac{2V_y - \omega W}{2}$$

Ditetapkan bahwa V_y bernilai konstan yaitu 30 cm/s dan W yaitu jarak antara roda bernilai 16 cm. maka di dapatkan :

$$V_{kiri} = \frac{60 - (\omega)16}{2}$$

$$V_{kiri} = 30 - 8\omega$$

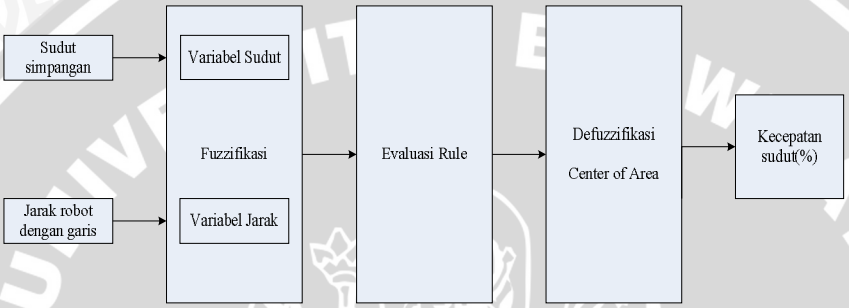
$$V_{kanan} = 30 + 8\omega$$

Jika diketahui kecepatan yang diinginkan maka dapat diketahui besar *duty cycle* yang diinginkan. Konversi *duty cycle* menjadi kecepatan pada robot yaitu :

$$duty\ cycle\ (\%) = \frac{V(cm/s)}{kecepatan\ motor\ (rps) \times keliling\ roda\ (cm)} \times 100\%$$

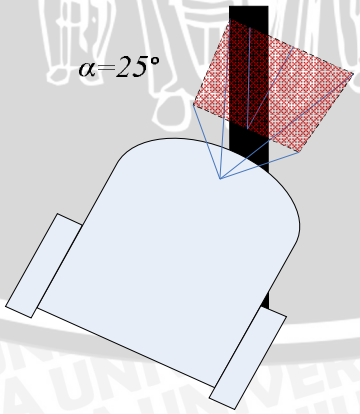
3.3.4. Perancangan Logika *Fuzzy*

Kontroller logika *fuzzy* digunakan sebagai perhitungan untuk mengambil keputusan Presentase kecepatan sudut robot. Terdapat dua masukkan untuk kontroller logika *fuzzy* yaitu sudut simpangan (α) dan jarak robot ke garis (*l*). Blok diagram kontroller logika *fuzzy* yang akan dirancang ditunjukkan oleh Gambar 3.8.

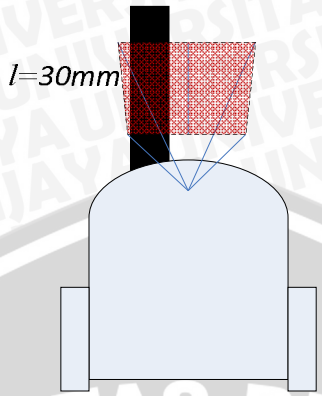


Gambar 3.8. Blok diagram kontroller logika *fuzzy*

Penentuan nilai batas pada masing-masing fungsi keanggotaan dilakukan dengan memperhatikan sudut simpangan terhadap garis (α) pada saat robot berada di tengah garis dan juga jarak posisi robot dengan garis (*l*) saat arah robot sejajar dengan garis. Ilustrasi posisi robot dalam lintasan ditunjukkan dalam Gambar 3.9 dan Gambar 3.10.

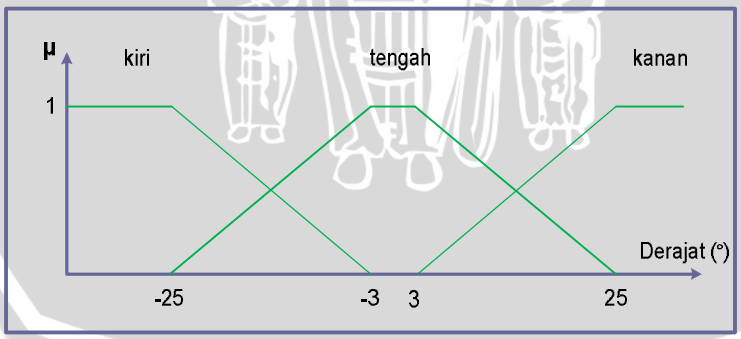


Gambar 3.9. Ilustrasi arah robot menyimpang dan berada di tengah garis



Gambar 3.10. Ilustrasi arah robot sejajar dan posisi robot tidak ditengah garis

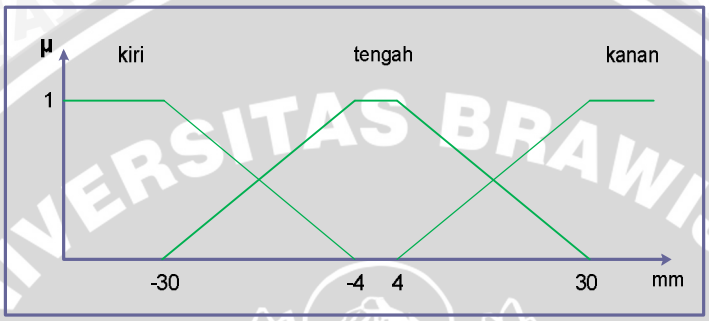
Setiap variabel masukkan dari logika *fuzzy* yaitu sudut simpangan (α) dan jarak antara robot dan garis (*l*) memiliki fungsi keanggotaan. Untuk fungsi keanggotaan dari sudut simpangan (α) terdapat tiga label yaitu tengah, kanan dan kiri. Label tengah yaitu kondisi dimana nilai sudut simpangan mendekati 0. Jika sudut bernilai -3° sampai 3° maka nilai keanggotaanya 1. Untuk label kiri yang artinya arah hadap robot condong ke kiri. Nilai batas derajat keanggotaan 1 untuk label kiri yaitu -25° yang berbentuk trapezoidal. Jadi jika sudut simpangan ≤ -25° maka nilai keanggotaan bernilai 1. Untuk label kanan nilai batas keanggotaan 1 yaitu 25° yang berbentuk trapezoidal. Gambar 3.11 menunjukkan fungsi masukkan untuk sudut simpangan (α).



Gambar 3.11. Fungsi Keanggotaan Masukan Sudut Simpangan

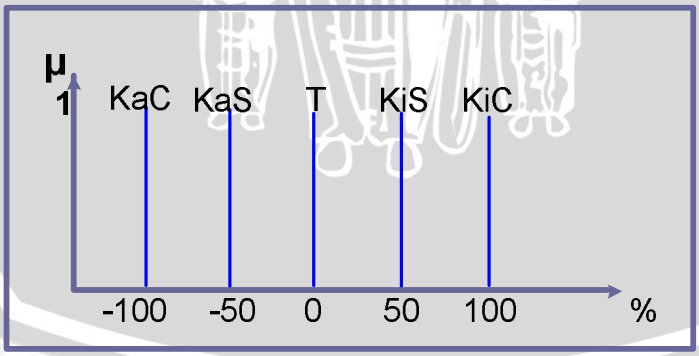
Untuk masukkan lainnya yaitu jarak robot terhadap garis (*l*) juga dibagi atas tiga label yaitu tengah, kanan dan kiri. Label tengah berarti garis berada ditengah robot. Nilai batas derajat keanggotaan 1 untuk label ini yaitu -4mm

sampai 4mm. Fungsi keanggotaan untuk label tengah ini berbentuk trapezoidal. Untuk label kiri nilai batas keanggotaan 1 yaitu jika lebih kecil dari -30mm berbentuk trapezoidal dan untuk nilai batas keanggotaan 1 label kanan yaitu lebih dari 30mm berbentuk trapezoidal. Gambar 3.12 menunjukkan fungsi keanggotaan untuk jarak robot dengan garis (*l*).



Gambar 3.12. Fungsi keanggotaan masukan jarak robot

Fungsi keanggotaan presentase kecepatan sudut merupakan presentase dari kecepatan sudut maksimum yang disebabkan karena perbedaan kecepatan pada roda kanan dan kiri.. Fungsi keanggotaan berbentuk *singleton*. Fungsi keluaran motor DC terdiri dari 5 label yaitu T (tengah), KaS (kanan sedang), KaC (kanan cepat), KiS (kiri sedang), KiC (kiri cepat). Fungsi keanggotaan presentase kecepatan sudut ditunjukkan pada Gambar 3.13.



Gambar 3.13. Fungsi Keanggotaan Presentase kecepatan sudut

Berdasarkan jurnal Thiang, Resmana, Wahyudi disimpulkan bahwa bentuk *membership function* antara segitiga dan trapezoid tidak memberikan pengaruh yang cukup besar dalam menentukan respon sistem. Semakin banyak jumlah label

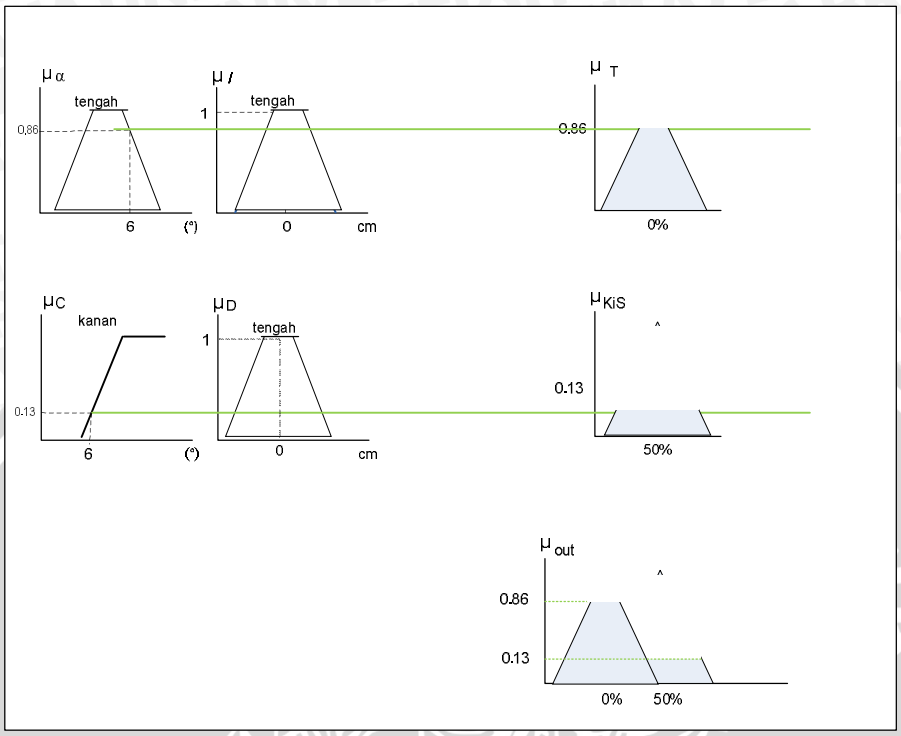
dalam *membership function* memungkinkan untuk menghasilkan kendali yang lebih baik terutama untuk metode defuzzifikasi mean of maxima. Pada skripsi ini alasan pemilihan bentuk m*embership function* berdasar pada jurnal tersebut.

Kaidah atur (*rule*) dalam logika *fuzzy* didasarkan pada model perilaku bergerak mengikuti garis dan disusun dalam bentuk jika – maka. Setelah masukkan diubah ke dalam variabel linguistik selanjutnya diolah sesuai dengan kaidah aturannya. Tabel 3.1 menunjukkan kaidah atur logika *fuzzy*.

Tabel 3.1. Kaidah atur logika *fuzzy*

Sudut simpangan	Jarak robot dengan garis	Kec. Sudut
Kiri	Kiri	KiC
Tengah		KiS
Kanan		T
Kiri	Tengah	KiS
Tengah		T
Kanan		KaS
Kiri	Kanan	T
Tengah		KaS
Kanan		KaC

Visualisasi proses pengolahan masukan ketika sudut simpangan (α) sebesar 2° dan jarak garis terhadap robot (*l*) sebesar 0 mm dengan metode MAX-MIN menggunakan kaidah atur utama yang telah ditetapkan ditunjukkan dalam Gambar 3.14.



Gambar 3.14. Ilustrasi Metode MAX-MIN

Setelah didapatkan aturan yang berlaku selanjutnya dilakukan defuzzifikasi. Defuzzifikasi adalah proses mengubah keluaran *fuzzy* menjadi keluaran *crips*. Hasil defuzzifikasi inilah yang digunakan untuk mengatur kecepatan sudut robot . Metode defuzzifikasi yang digunakan adalah *Center of Area* (COA).

Presentase kecepatan sudut :

$$U = \frac{\sum_{i=1}^{n} w_i u_i}{\sum_{i=1}^{n} w_i}$$

$$= \frac{0.86 \times T + 0.13 \times KiS}{0.86 + 0.13}$$

$$= \frac{0.86 \times 0 + 0.13 \times 50}{0.99} = 6.5\%$$

Jadi ketika sudut simpangan (α) sebesar 6° dan jarak garis terhadap robot (l) sebesar 0mm maka presentase kecepatan sudut yaitu 6.5%. Jika kecepatan sudut maksimal yaitu 4.9 rad/s. maka perbedaan kecepatan roda kanan dan kiri yaitu 5.1 cm/s. dengan rumus yang telah didapat maka kecepatan roda kanan dan kiri yaitu 42.6 cm/s dan 37.4cm/s. Dengan demikian jika menggunakan pwm dengan 8 bit maka duty cycle motor kanan dan duty cycle motor kiri yaitu 138 dan 121 dengan nilai maksimum duty cycle yaitu 255. Hubungan antara duty cycle dan presentase kecepatan sudut yaitu:

$$duty\ cycle_{kiri} = \frac{30 - 8(presentase\ kecepatan\ sudut(\%) \times \omega_{max})}{kecepatan\ roda_{max}}$$

$$duty\ cycle_{kanan} = \frac{30 + 8(presentase\ kecepatan\ sudut(\%) \times \omega_{max})}{kecepatan\ roda_{max}}$$

Dengan demikian robot dapat mengikuti garis dengan kecepatan konstan yang sudah ditunjukkan pada spesifikasi alat yaitu 30 cm/s.

BAB IV
PERANCANGAN DAN PEMBUATAN ALAT

Perancangan robot dilakukan secara bertahap sehingga akan memudahkan dalam analisis pada setiap bloknya maupun secara keseluruhan. Perancangan ini terdiri dari:

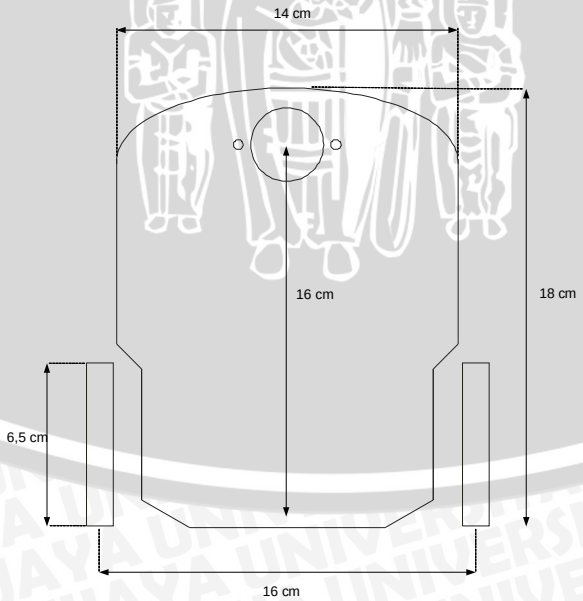
- Perancangan perangkat keras terdiri dari dua bagian yaitu sistem mekanik robot dan desain sistem elektronik.
- Perancangan algoritma mikrokontroller.

4.1 Perancangan Perangkat Keras

4.1.1. Perancangan Mekanik Robot

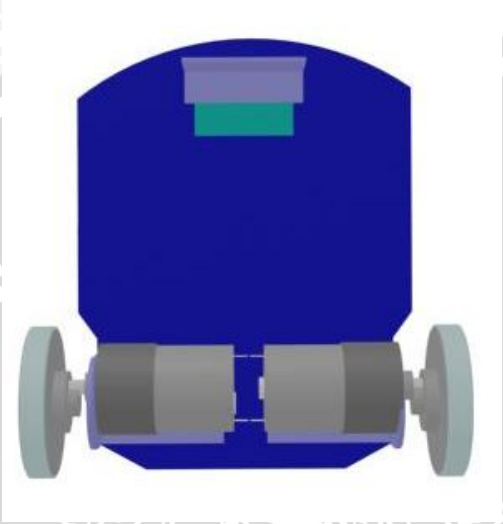
Sistem mekanik yang baik, mendukung pergerakan robot menjadi lebih baik, oleh karena itu perancangan mekanik dalam hal ini bodi dan rangka robot haruslah proporsional dengan panjang dan lebar serta tinggi dari robot. Dimensi robot adalah sebagai berikut :

- panjang = 18 cm
- lebar = 14 cm
- tinggi = 21.5 cm



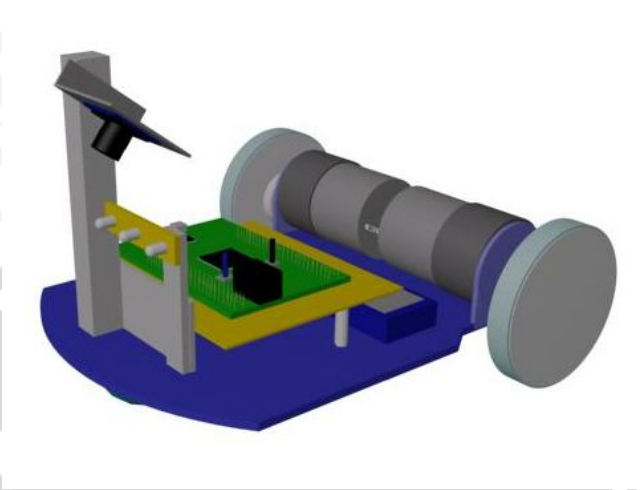
Gambar 4.1. Rancangan ukuran robot

Gambar perspektif Robot beroda ditunjukkan dalam Gambar 4.2. dan Gambar 4.3.



Gambar 4.2. Robot tampak atas

Badan robot terbuat dari bahan mika *acrylic* dengan ketebalan 3 mm, ke dua buah roda belakang berbahan nilon dengan tebal 1 cm dan berdiameter 6,5 cm. Roda depan menggunakan roda *castor* (roda bebas). Bentuk dan dimensi robot dirancang sesuai secara proporsional dengan harapan robot dapat bermanuver dengan baik salah satunya adalah gerakan pivot pada satu titik.



Gambar 4.3. Robot tampak samping

4.1.2 Perancangan Desain Sistem Elektronik

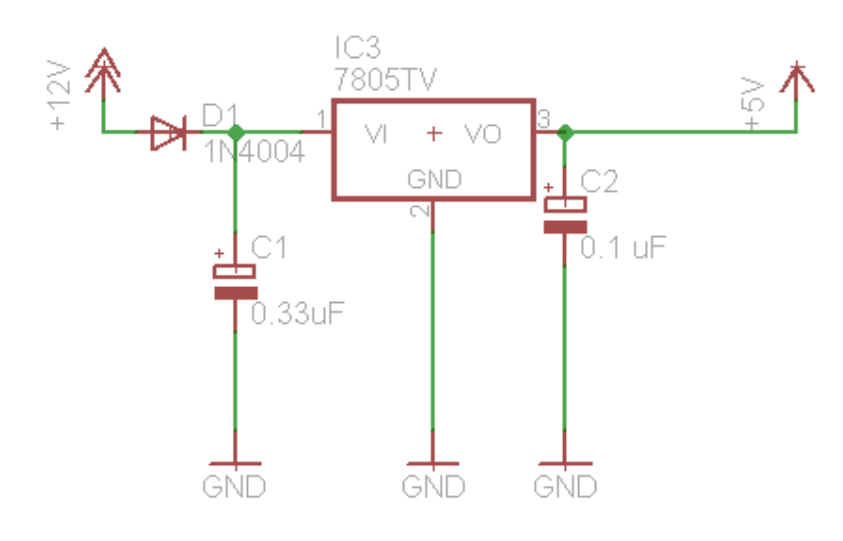
Dalam perancangan desain sistem elektronik terdapat empat bagian perancangan. Yaitu perancangan sistem catu daya sebagai penyuplai tegangan. Perancangan *driver* motor L298 sebagai antarmuka antara mikrokontroller dan motor. Perancangan rangkaian lampu LED sebagai penerang, dan perancangan antarmuka dari *development board* STM32F4 Discovery terhadap kamera OV7670 dan *driver* motor.

4.1.2.1 Perancangan Catu Daya Sistem

Robot ini Menggunakan satu jenis catu daya, yaitu catu daya 5 V untuk rangkaian *development board* STM32F4Discovery, rangkaian *driver* motor L298 dan rangkaian lampu LED. Sumber catu daya yang dipakai adalah satu buah baterai lipo (*lithium polimer*) 11,1 V.

Mikrokontroller STM32F4 bekerja pada tegangan 3 V hingga 3.3 V. Di dalam *development board* STM32F4Discovery sudah terdapat regulator untuk 3V sehingga hanya diperlukan satu jenis tegangan untuk menyuplai semua rangkaian. Pada perancangan digunakan catu daya sebesar 5V yang diperoleh dari rangkaian *Fixed Output Regulator* pada datasheet LM7805. Pada rangkaian digunakan regulator LM7805 agar diperoleh tegangan keluaran yang bisa diatur supaya nilai

keluaran regulator bisa mendekati 5V. Skema rangkaian catu daya ditunjukkan dalam Gambar 4.4.



Gambar 4.4 Rangkaian Catu 5V

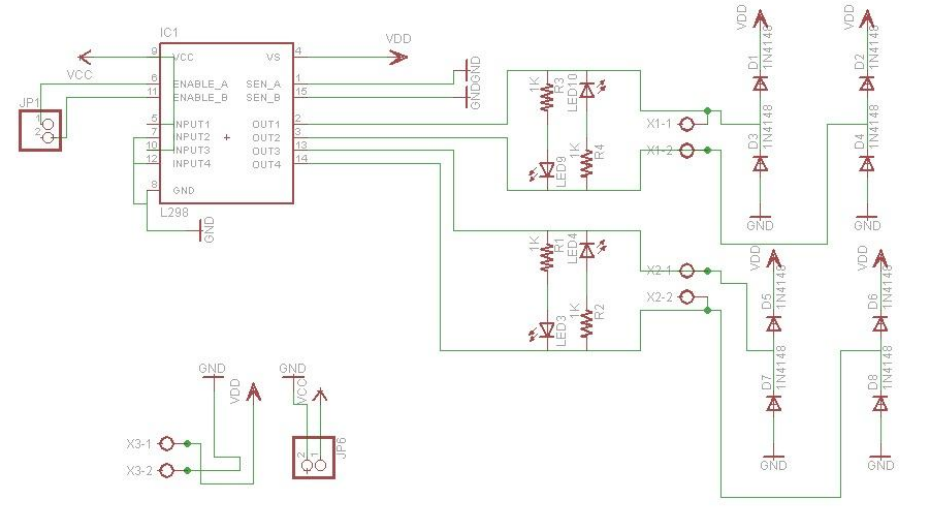
4.1.2.2 Perancangan Rangkaian *Driver* Pengendali Motor DC

Driver pengendali menggunakan *driver* L298N yaitu sebuah perangkat keras berupa rangkaian yang berfungsi untuk menggerakkan motor DC. IC L298N terdiri dari transistor - transistor logik (TTL) dengan gerbang nand yang memudahkan dalam menentukkan arah putaran suatu motor DC. *Driver* motor yang diperlukan oleh mobile robot cukup satu buah, karena IC L298N dapat mengendalikan dua buah motor DC sekaligus karena IC ini memiliki dua buah rangkaian *H-Bridge* di dalamnya.Spesifikasi IC L298N ini meliputi:

- $I_{O\ max}$ = 3A
- $I_{O\ min}$ = 2A
- Vs = 2,5 – 46 V (*Power Supply)*
- Vss = 4,5 – 7 V (*Logic Supply Voltage*)
- Ven = -0,3 – 7 V (*Input and Enable Voltage*

Dalam perancangannya, motor DC yang digunakan memiliki tegangan catu sebesar 12 V dengan kecepatan 500 rpm. Apabila robot menggunakan

kecepatan maksimal, Arus yang dibutuhkan motor saat tak berbeban dengan menggunakan *driver* L298N ini sebesar 0,4 mA. Sedangakan ketika berbeban atau kondisi robot sedang berjalan sebesar 1,1 mA. Sehingga memenuhi dalam perancangan dengan menggunkan *driver* L298N. Skema rangkaian *driver* motor L298N ditunjukkan dalam Gambar 4.5



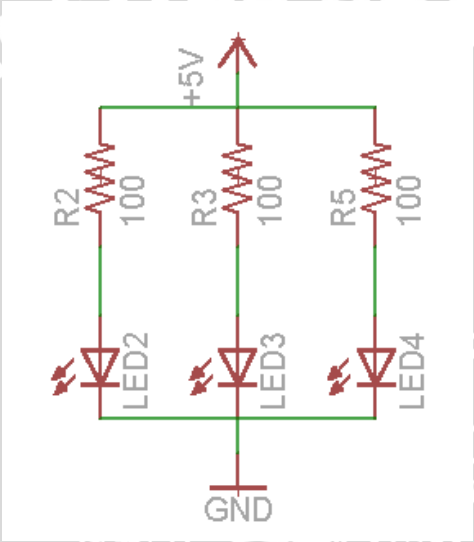
Gambar 4.5.Rangkaian *driver* motor DC

Dalam perancangan konfigurasi *driver* L298N ini untuk menjalankan motor, pin *enable* A dan *enable* B dihubungkan pada TIM2_CH3 dan TIM2_CH4 pada mikrokontroler. *Current sensing* A dan *current sensing* B dihubungkan ke *ground*. Pada *driver* L298N terdapat 4 input untuk menentukan arah putaran motor kiri dan kanan. Karena pada perancangan robot hanya perlu bergerak maju, maka input 1 dan 3 dihubungkan dengan VCC dan input 2 dan 4 dihubungkan dengan GND. Pada VDD menggunakan tegangan sebesar 11,1 V dari baterai *lithium polymer* dan VCC sebesar 5 V dari rangkaian catu daya 5V.

Output dari IC L298N ini tidak memiliki dioda pengaman oleh karena itu pada perancangan *driver* motor ini ditambahkan 2 buah dioda pada setiap titik output, sehingga total terdapat 8 buah dioda untuk 2 motor masing - masing motor 4 buah dioda. Menggunakan 4 buah LED sebagai indikator pergerakan motor.

4.1.2.3 Perancangan Rangkaian Lampu LED

Rangkaian lampu LED digunakan untuk menerangi lintasan supaya gambar yang ditangkap oleh kamera memiliki pencahayaan yang tetap sehingga nilai hitam dan putih jalur sesuai dengan setpoint yang telah ditentukan. Lampu LED yang digunakan adalah lampu LED Superbright 5mm. Menurut *datasheet* lampu LED ini membutuhkan tegangan 3.2 volt dan arus 20 mA. Rangkaian lampu LED terdapat pada Gambar 4.6.



Gambar 4.6.Rangkaian lampu LED

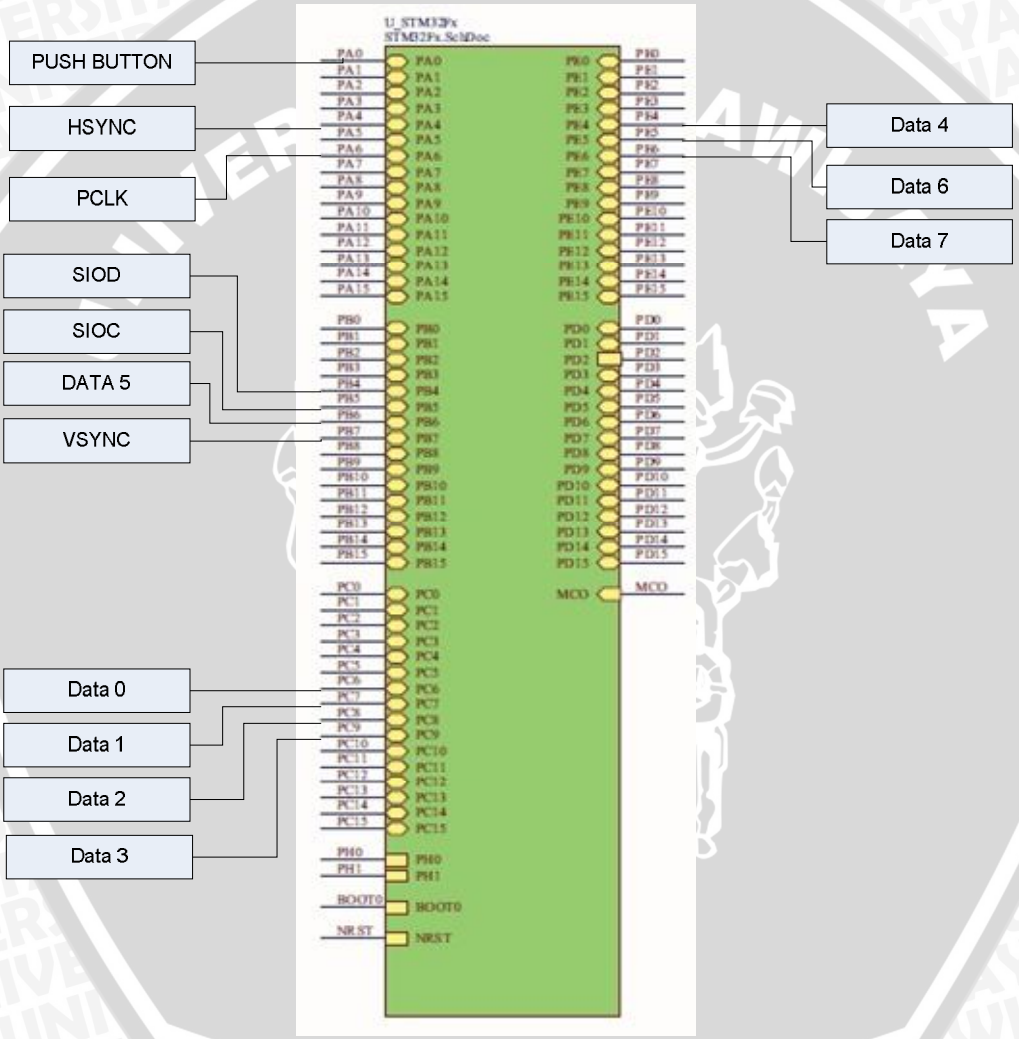
Karena catu yang digunakan menggunkan keluaran regulator dengan tegangan 5V, lampu LED perlu diseri dengan resistor. Nilai resistor yang digunakan adalah

$$R_{led} = \frac{VCC - V_{led}}{I_{led}}$$

Karena nilai tegangan lampu LED (V_{led}) yaitu 3.2 V dan arus lampu LED (I_{led}) yaitu 20mA maka resistor yang digunakan yaitu 90 ohm. Pada perancangan digunakan resistor 100 ohm karena mudah ditemukan di pasaran.

4.1.2.4. Antarmuka Mikrokontroller

Pada perancangan digunakan *development board* STM32F4Discovery sebagai pengolah utama dalam melakukan proses pengolahan gambar dan juga pengendalian logika *fuzzy*. Konfigurasi kaki I/O dari *development board* STM32F4Discovery ditunjukkan dalam Gambar 4.7.



Gambar 4.7.Konfigurasi I/O STM32F4Discovery

STM32F4Discovery memiliki 5 buah PORT 75 jalur yang bisa dijadikan masukkan, keluaran atau fungsi alternatif. Pada perancangan pin – pin yang digunakan adalah :

PINA.0 = diatur sebagai input dan dihungkan dengan tombol push botton untuk pemicu start

PINA.4 = diatur sebagai fungsi alternatif dan dihubungkan dengan pin HSYNC dari kamera OV7670.

PINA.6 = diatur sebagai fungsi alternatif dan dihubungkan dengan pin PCLK dari kamera OV7670.

PINB.7 = diatur sebagai fungsi alternatif dan dihubungkan dengan pin VSYNC dari kamera OV7670

PINC.6 = diatur sebagai fungsi alternatif dan dihubungkan dengan pin D0 dari kamera OV7670

PINC.7 = diatur sebagai fungsi alternatif dan dihubungkan dengan pin D1 dari kamera OV7670

PINC.8 = diatur sebagai fungsi alternatif dan dihubungkan dengan pin D2 dari kamera OV7670

PINC.9 = diatur sebagai fungsi alternatif dan dihubungkan dengan pin D3 dari kamera OV7670

PINE.4 = diatur sebagai fungsi alternatif dan dihubungkan dengan pin D4 dari kamera OV7670

PINB.6 = diatur sebagai fungsi alternatif dan dihubungkan dengan pin D5 dari kamera OV7670

PINE.5 = diatur sebagai fungsi alternatif dan dihubungkan dengan pin D6 dari kamera OV7670

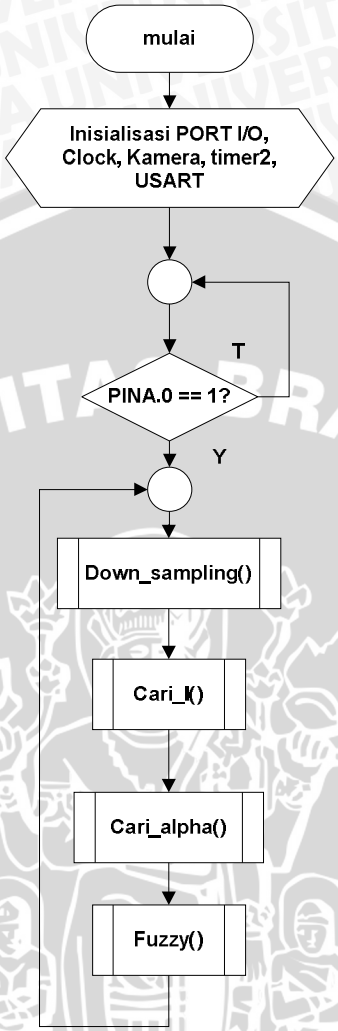
PINE.6 = diatur sebagai fungsi alternatif dan dihubungkan dengan pin D7 dari kamera OV7670

PINB.5 = diatur sebagai output dan dihubungkan dengan pin SIOC dari kamera OV7670

PINB.4 = diatur sebagai output dan dihubungkan dengan pin SIOD dari kamera OV7670

4.2. Perancangan Algoritma Pemrograman Mikrokontroller

Diagram alir program utama mikrokontroler meliputi proses inisialisasi, konfigurasi kamera, penurunan resolusi, penentuan nilai jarak robot terhadap garis (l) dan sudut simpangan robot dengan garis (α), dan perhitungan *fuzzy* sebagai proses berjalanya robot *line follower*. Diagram alir program utama ditunjukkan dalam Gambar 4.8.



Gambar 4.8. Diagram alir program utama mikrokontroler

Program utama mikrokontroler berisi inisialisasi *Clock*, PORT I/O, Kamera, timer2 untuk mengatur PWM motor kanan dan motor kiri dan USART. Selain itu juga terdapat inisialisasi variabel - variabel yang akan digunakan oleh program. Program utama akan menjalankan subrutin setting_awal(), subrutin down_sampling(), Cari_L(), Cari_alpha(), dan Fuzzy().

Pada mikrokontroller STM32F4 yang dipakai pada perancangan perlu dilakukan inisialisasi *clock*. Karena pada mikrokontroller ini tersedia 2 jenis *clock*, yaitu clock internal dan eksternal. *Clock* internal memiliki frekuensi 16 MHz

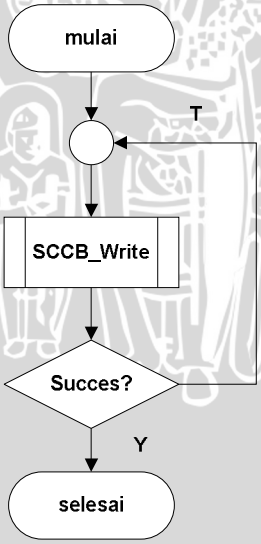
sedangkan *clock* eksternal memiliki frekuensi 8MHz. Untuk dapat memakai *clock* maksimal yaitu 168MHz. digunakan lah *clock* PLL (*Phase Locked Loop*) dengan rumus:

$$\text{PLLclock} = \frac{(Sumber\ clock\ /\ PLL\_M)\ x\ PLL\_N}{PLL\_P}$$

Pada perancangan sumber *clock* yang digunakan yaitu *clock* internal 16 MHz dan nilai PLL_M = 16, PLL_N = 336, dan PLL_P = 2. Jadi mikrokontroller bekerja pada frekuensi maksimal yaitu 168 MHz.

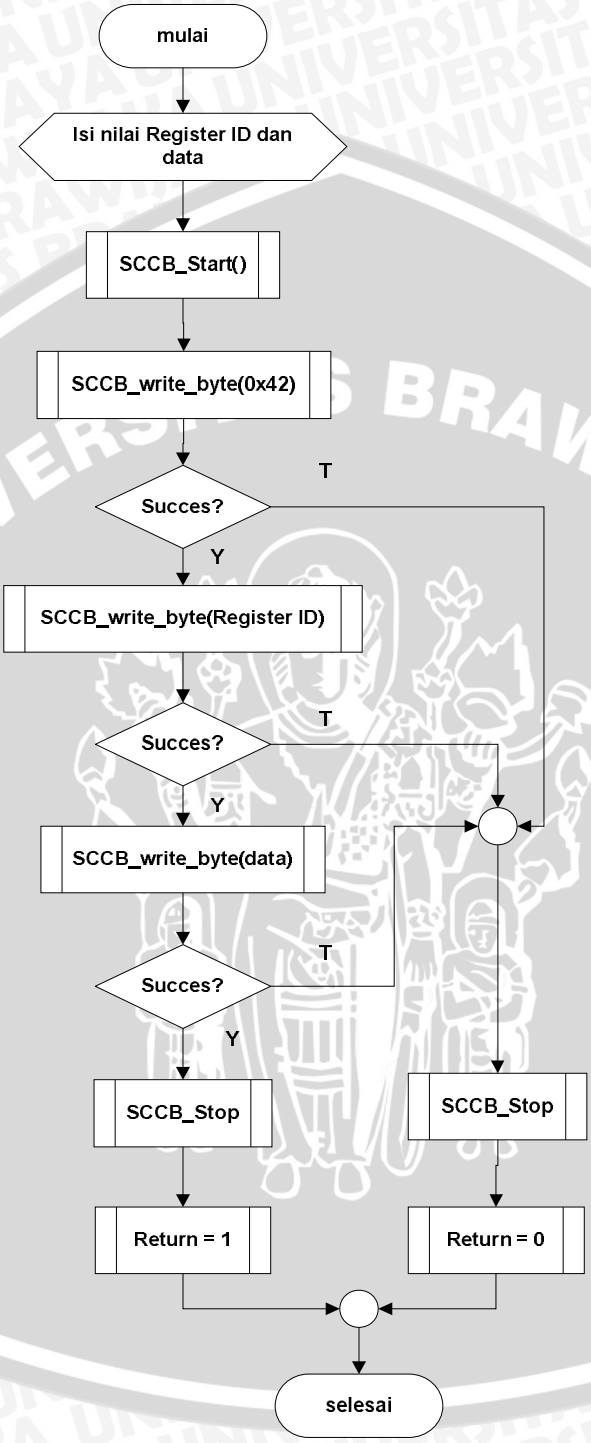
4.2.1. Proses Pengaturan Register Kamera

Pada proses pengaturan register kamera terjadi beberapa kali pemanggilan subfungsi sccb_write(register ID, data) yang berarti mengisi suatu alamat register dengan data baru supaya kamera dapat mengeluarkan gambar sesuai yang dibutuhkan.



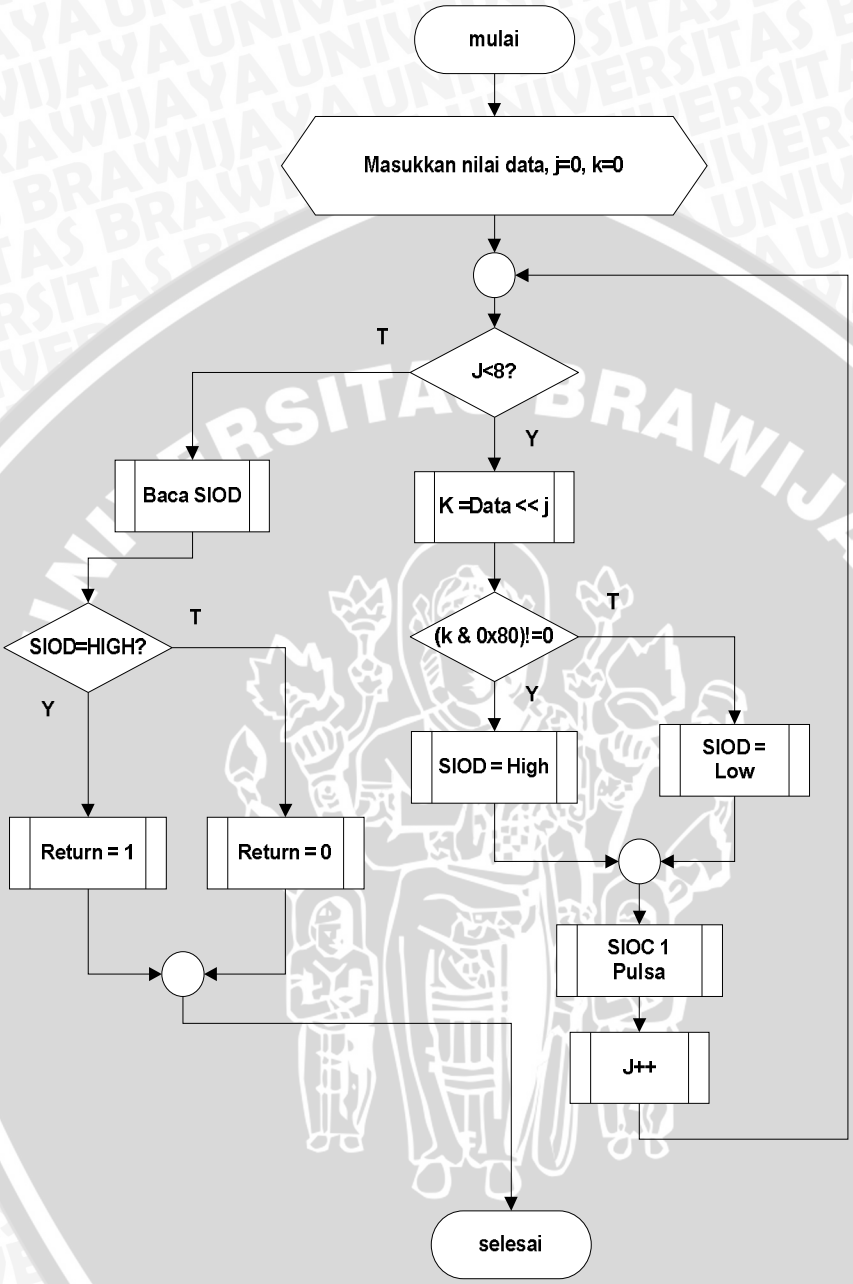
Gambar 4.9. Diagram Alir Proses Pengaturan Kamera

Diagram alir dalam Gambar 4.9 menunjukkan bila terjadi kegagalan penulisan register pada kamera maka akan diulang sampai berhasil. Sedangkan proses sub fungsi SCCB_Write ditunjukkan dalam Gambar 4.10.



Gambar 4.10. Diagram alir subfungsi SCCB_Write()

Pada proses subfungsi SCCB_Write(), diawali dengan penanda mulainya komunikasi SCCB. Dalam diagram tersebut ditunjukkan dengan subfungsi SCCB_Start(). Subfungsi SCCB_Start() merupakan proses pemberian sinyal start melalui pin SCCB yaitu SIOD sebagai jalur data dan SIOC sebagai jalur *clock*. Pada *datasheet* OV7670 telah ditunjukkan *timing* dari sinyal start untuk memulai komunikasi. Selanjutnya penulisan satu byte (8bit) melalui komunikasi SCCB yang berisi alamat dari OV7670. Pada *datasheet* OV7670 dapat diketahui bahwa alamat untuk menulis register yaitu 0x42 sedangkan untuk membaca nilai dari suatu register memakai alamat 0x43. Jika penulisan alamat berhasil maka akan ditandai dengan adanya sinyal balasan dari kamera OV7670 yang biasa disebut sinyal *acknowledge*. Sinyal ini terdapat pada pin SIOD dan berlogika *low*. Dilanjutkan dengan penulisan satu byte yang berisi alamat register yang akan dituju. Jika penulisan ini berhasil juga akan ditunjukkan oleh sinyal *acknowledge*. Yang terakhir yaitu penulisan satu byte yang berisi data yang akan ditulis ke register yang sudah dituju. Setiap pengiriman satu byte data pasti akan ada balasan berupa sinyal *acknowledge*. Jika ada salah satu pengiriman yang gagal maka komunikasi SCCB akan diakhiri dengan subfungsi SCCB_Stop. Dan memberi kan balasan (*return*) bernilai 0. Tapi jika semua pengiriman berhasil maka komunikasi SCCB juga diakhiri dengan subfungsi SCCB_Stop dan akan memberi nilai balasan (*return*) bernilai 1. Subfungsi SCCB_Stop() merupakan pemberian sinyal stop untuk mengakhiri komunikasi SCCB. Pada *datasheet* OV7670 telah ditunjukkan *timing* dari sinyal stop.



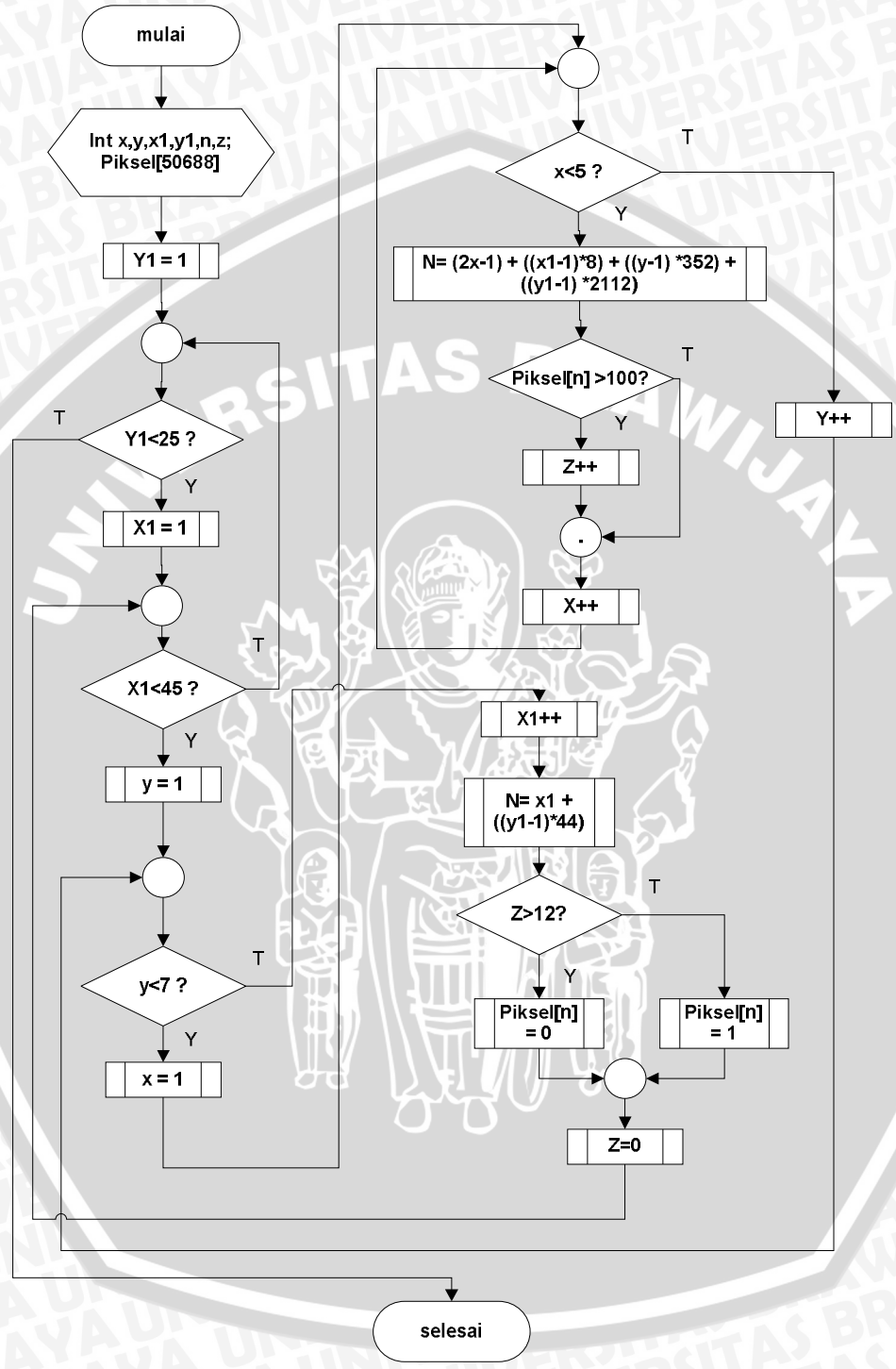
Gambar 4.11. Diagram alir subfungsi penulisan satu byte

Gambar 4.11. menunjukkan diagram alir subfungsi dari penulisan satu byte melalui SCCB. Data yang akan dikirim sebesar satu byte bisa berupa alamat untuk menulis atau membaca register, alamat register dan data yang akan di tuliskan ke register. Dalam komunikasi SCCB data dikirim secara serial. Data

akan keluar dari port SIOD. Dalam Gambar 4.11 ditunjukkan proses pengubahan nilai pin SIOD sesuai dengan data yang akan di kirim. Tiap bit data yang dikirim ditandai dengan adanya satu pulsa pada port SIOC. Setelah satu byte data selesai dikirim, selanjutnya yaitu menunggu sinyal balasan atau *acknowledge*. Jika sinyal balasan berlogika *high* maka keluaran dari subfungsi ini bernilai 0 dan jika balasan berlogika *low* maka keluaran dari subfungsi ini bernilai 1.

4.2.2. Proses Pengolahan Gambar

Gambar 4.12 menunjukkan diagram alir penurunan piksel

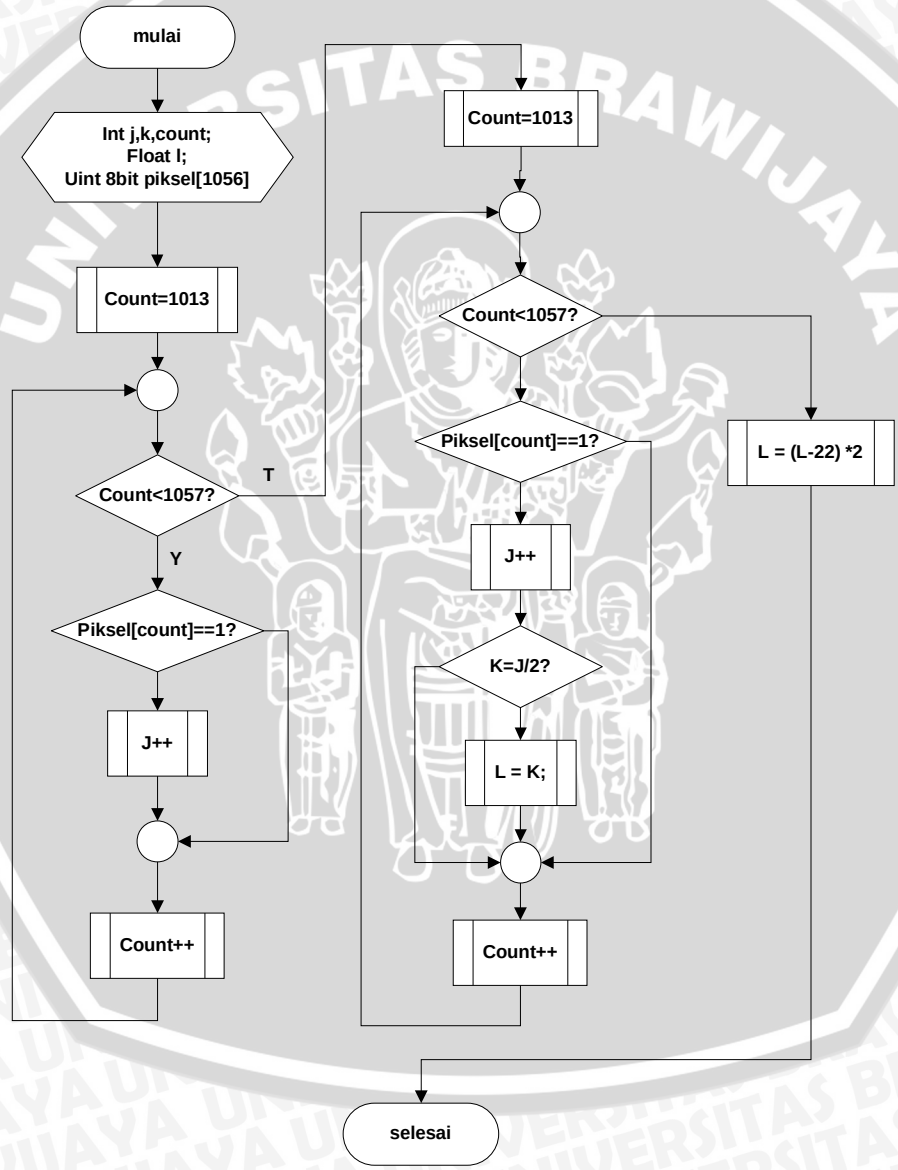


Gambar 4.12. Diagram alir subfungsi penurunan piksel

Pada diagram alir yang ditunjukkan Gambar 4.12. variabel Y1 mewakili baris dari gambar yang diturunkan resolusinya, X1 mewakili kolom dari gambar yang diturunkan resolusinya, sedangkan variabel X dan Y digunakan dalam pembentukan satu piksel baru. Pada proses penurunan piksel ini akan terbentuk X1 * Y1 piksel baru yang tiap pikselnya bernilai 1 atau 0.

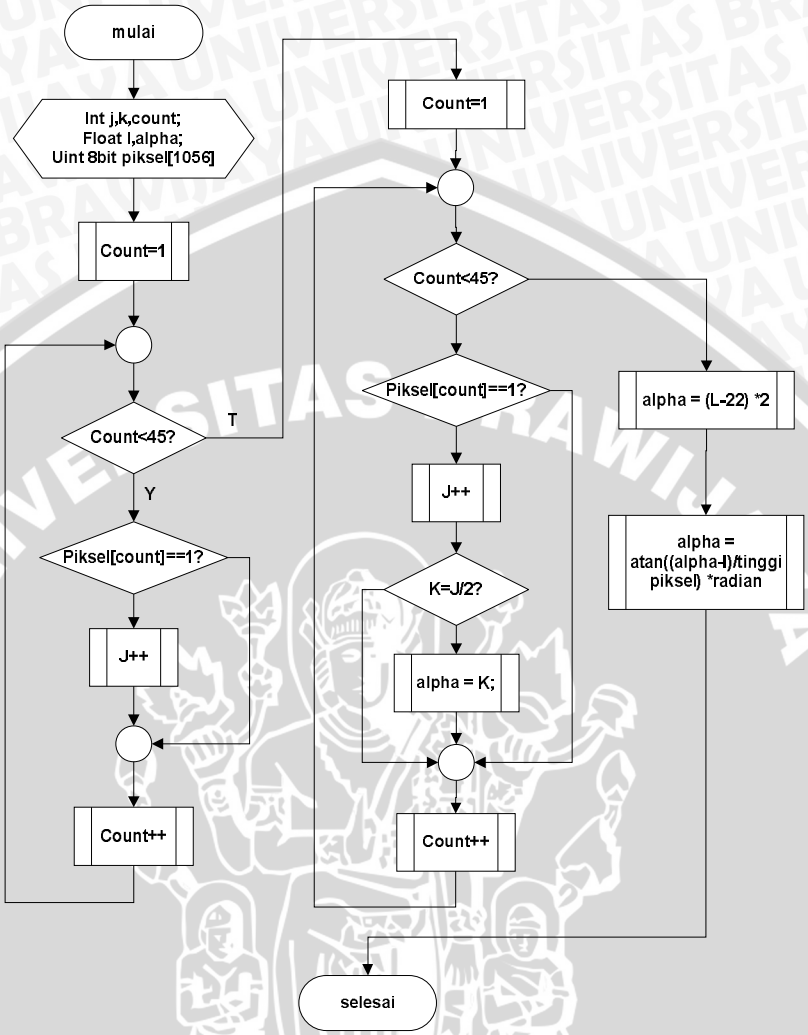
Setiap X * Y piksel akan dijadikan acuan dalam pembentukan satu piksel baru. Batas dari X yaitu 4 dan batas dari Y yaitu 6. Jadi setiap 4 kolom dan 6 baris piksel akan menjadi satu piksel. Karena letak data dalam *array* yang digunakan tidak berurutan maka digunakan variabel N. variabel N ini fungsinya untuk mengetahui letak data yang kemudian akan dibandingkan tingkat keabuannya. Nilai N ini dipengaruhi oleh X,Y,X1 dan Y1 dengan rumus N = (2X-1) + ((X1-1)*8) + ((Y-1)*352) + ((Y1-1*2112). Data dari piksel ke N dibandingkan dengan *setpoint* keabuan. Pada proses pembentukan satu piksel baru terjadi 24 kali pembandingan data. Jika terdapat lebih dari 12 data yang nilainya melebihi setpoint maka piksel baru akan bernilai 0 yang artinya berwarna putih. Dan bila terdapat kurang dari 12 data yang nilainnya melebihi *setpoint* maka piksel baru akan bernilai 1 yang menunjukkan warna hitam. Piksel baru ini nanti akan disimpan ke *array* yang digunakan dalam penyimpanan gambar menggantikan data yang lama. Piksel ini akan diatur letak penyimpanannya supaya berurutan. Data yang nilainya telah digantikan oleh data piksel baru memang sudah tidak digunakan lagi. Untuk proses penempatan data, digunakan bantuan variabel N. dengan nilai N dihasilkan oleh rumus N = X1 + ((Y1-1)*44). Dengan demikian nilai – nilai dari piksel baru tersimpan pada data piksel[1] sampai piksel[1056]. Piksel[1] sampai dengan piksel[44] merupakan baris pertama dari gambar yang telah diturunkan resolusinya.

Setelah didapat gambar dengan resolusi yang lebih kecil, maka gambar akan diolah untuk memperoleh sudut simpangan (α) dan jarak robot terhadap garis (*l*). Gambar 4.13 menunjukkan proses pencarian nilai sudut simpangan (α) dan jarak robot dengan garis (*l*).



Gambar 4.13. Diagram alir subfungsi pencarian posisi robot

Array piksel[1056] merupakan tempat penyimpanan gambar yang sudah diturunkan resolusinya. pada awalnya data pada piksel[1013] sampai piksel[1057] dibandingkan datanya dengan nilai satu. Piksel[1013] sampai piksel[1057] merupakan data baris terakhir pada gambar. Setiap piksel dibandingkan dengan nilai satu untuk mencari jumlah data yang bernilai satu. Data yang bernilai satu ini merupakan representasi dari garis yang terdeteksi. Jumlah data yang bernilai satu disimpan dalam variabel j. setelah itu piksel[1013] sampai piksel[1057] dibandingkan lagi dengan nilai satu untuk mencari posisi dari tengah garis. Tengah garis dianggap sebagai piksel berwarna hitam pada hitungan ke j/2. Setelah mengetahui posisi piksel tengah garis, dapat diketahui pula jarak dari sumbu pusat ke posisi tengah garis. Jarak ini masih dalam satuan piksel. Untuk mengetahui jarak dalam mm maka jarak ini dikalikan dengan suatu konstanta yang mewakili lebar satu piksel dalam mm.



Gambar 4.14. Diagram Alir Sub Fungsi Pencarian Nilai Sudut Simpangan Robot

Diagram alir pencarian nilai sudut simpangan (α) pada Gambar 4.26. hampir sama dengan diagram alir dari pencarian nilai jarak robot dengan garis (l). bedanya pencarian titik tengah dilakukan di baris pertama. Setelah diketahui jarak titik tengah dan sumbu pusat dalam satuan mm, maka dapat dihitung sudut simpangan robot dengan melibatkan jarak robot dengan garis(l). hasil pencarian sudut ini masih dalam bentuk radian. Untuk mengubahnya ke dalam bentuk derajat sudut simpangan di kali dengan konstanta yang bernilai 180/π.

4.2.3. Diagram Alir Pengendalian Logika *Fuzzy*

Secara umum proses pengendalian *fuzzy* melalui 3 tahapan yaitu fuzzifikasi, pengecekan aturan, lalu defuzzifikasi. Diagram alir dibawah ini menunjukkan tahap dari fuzzifikasi. Pada algoritma pengendali *fuzzy* digunakan beberapa definisi dan *array* untuk menyimpan aturan aturan. Definisi yang dilakukan yaitu :

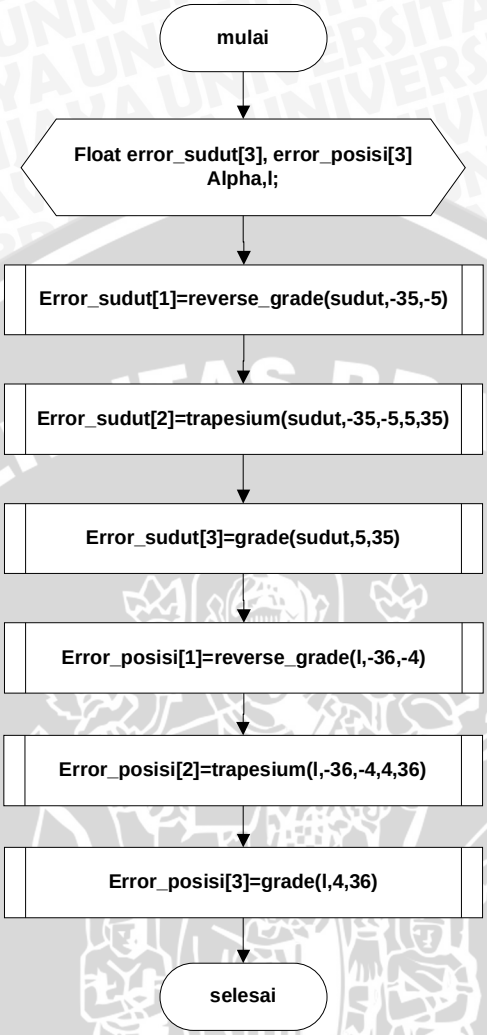
Vkac = 0 , Vkas = 1, Vt=2, Vkis=3, Vkic=4

Definisi tersebut sesuai dengan jumlah fungsi keanggotaan keluaran dari kendali logika *fuzzy* yang dirancang, yaitu belok kanan cepat (vkac), belok kanan sedang (vkas), lurus tengah (Vt), belok kiri sedang (Vkis), dan belok kiri cepat(Vkic).

Untuk pendeklarasian aturan, jumlahnya sesuai dengan aturan yang telah dibuat yaitu ada 9 aturan yang disimpan dalam matriks 3x3.

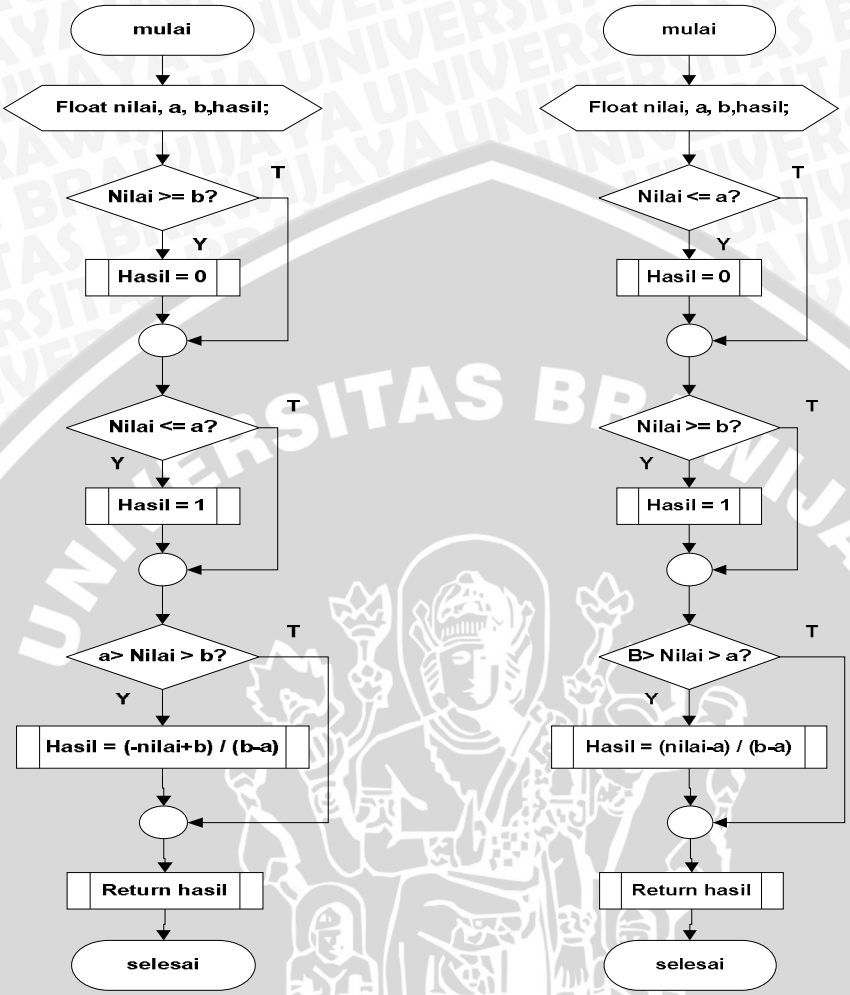
Rule[3][3] = {Vkic, Vkis, Vkas}, {Vkis, Vt, Vkas}, {Vkis, Vkas, Vkac}

Pada matriks Rule[3][3] terdapat 3 buah baris yang setiap baris nya memiliki 3 anggota. 3 buah baris menunjukkan posisi robot terhadap baris. Baris pertama robot berada di sebelah kanan garis, baris kedua robot berada ditengah garis dan baris ketiga robot berada di kiri garis. Pada setiap baris terdapat tiga anggota yang berupa aturan yang berlaku. Anggota pertama menunjukkan aturan yang berlaku saat sudut garis miring ke kiri, anggota kedua saat garis lurus, dan anggota ketiga saat garis miring ke kanan.



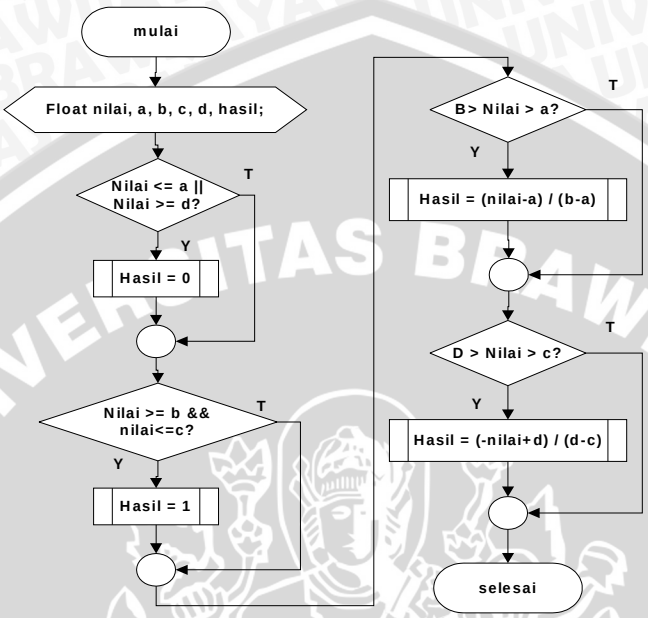
Gambar 4.15. Diagram Alir Sub Fungsi Fuzzifikasi

Pada diagram alir fuzzifikasi yang ditunjukkan Gambar 4.15 terjadi pemanggilan beberapa subfungsi yaitu reverse_grade(), trapesium(), dan grade().



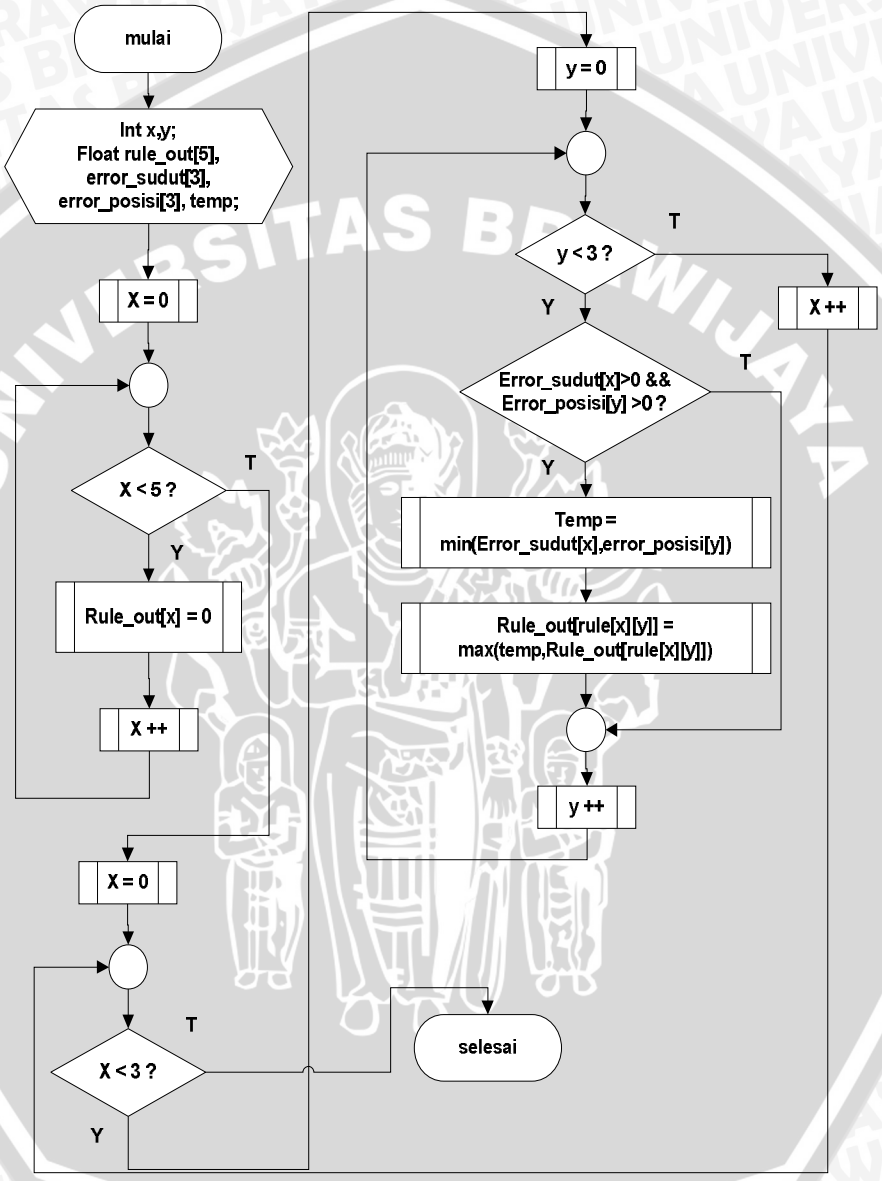
Gambar 4.16. Diagram alir subfungsi (A) Reverse Grade (B) Grade

Pada diagram alir untuk subfungsi reverse_grade() dan grade() memiliki alur yang sama. Hanya saling berlawanan, pada subfungsi reverse_grade() bila nilai masukkan lebih kecil atau sama dengan batas a maka akan menghasilkan nilai keluaran 1. Jika nilai masukkan lebih besar atau sama dengan batas B maka akan menghasilkan nilai keluaran 0. Dan jika nilai masukkan berada di antara batas A dan batas B maka keluarannya bernilai = (-nilai masukkan + batas B) / (batas B – batas A).



Gambar 4.17. Diagram Alir Sub Fungsi trapesium

Perbedaan reverse_grade() dan sub fungsi trapesium yang ditunjukkan dalam Gambar 4.17 yaitu pada trapesium terdapat 4 buah batas, jika nilai masukkan lebih besar dari batas d atau lebih kecil dari batas a maka nilai keluaran bernilai 0. Jika nilai masukkan berada di antara nilai batas B dan batas C maka nilai keluaran bernilai 0. Jika nilai masukkan berada diantara nilai batas A dan B maka nilai keluaran = (nilai masukkan – batas A) / (batas B – batas A). dan jika nilai masukkan berada diantara batas C dan D maka nilai keluaran = (-nilai masukkan + batas D) / (batas D – batas C).

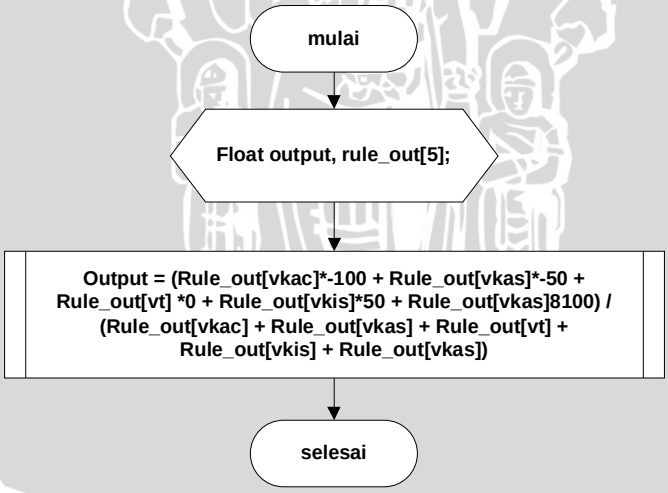


Gambar 4.18. Diagram Alir Sub Fungsi penentuan aturan *fuzzy*

Pada proses penentuan aturan yang ditunjukkan dalam Gambar 4.18, hasil dari perhitungan fuzzifikasi menjadi masukkan, dan keluaran nya adalah nilai dari masing – masing anggota dari fungsi keluaran logika *fuzzy*. Terdapat 6 masukkan

yang terdiri dari error sudut dan error posisi, masing - masing memiliki tiga nilai. Pada proses penentuan aturan ini lah digunakan *rule* yang sudah didefinisikan.

Proses pertama dalam penentuan aturan yaitu pengosongan dari nilai keluaran sebelumnya. Nilai keluaran yang tersimpan dalam variabel rule_out[6], semua nilainya dig anti dengan 0. Proses selanjutnya yaitu pembandingan nilai antara setiap anggota error_sudut dan setiap anggota error_posisi. terjadi Sembilan kali pembandingan karena terdapat tiga anggota pada masing masing variabel. Pada saat pembandingan, jika kedua nilai variabel lebih dari nol, maka akan ditampung nilai yang lebih kecil dari kedua nilai yang dibandingkan. Nilai yang ditampung ini juga akan dibandingkan dengan salah satu nilai keluaran rule_out. Karena terdapat lima anggota pada variable rule_out, pemilihan anggota rule_out yang akan dibandingkan berdasarkan kepada aturan yang sudah dibuat. Jika nilai yang ditampung lebih besar, maka nilai yang ditampung ini menggantikan nilai dari rule_out yang telah dibandingkan. Selanjutnya yaitu proses dufuzzifikasi.



Gambar 4.19. Diagram Alir Sub Fungsi Defuzzifikasi

Pada proses defuzzifikasi yang ditunjukkan dalam Gambar 4.19 terjadi perhitungan yang menggunakan rule_out yang nilainya didapat pada proses penentuan rule. keluaran dari defuzzifikasi ini merupakan presentase kecepatan

sudut yang nantinya akan digunakan untuk menentukan pwm motor kanan dan motor kiri sehingga robot dapat mengikuti garis.

BAB V

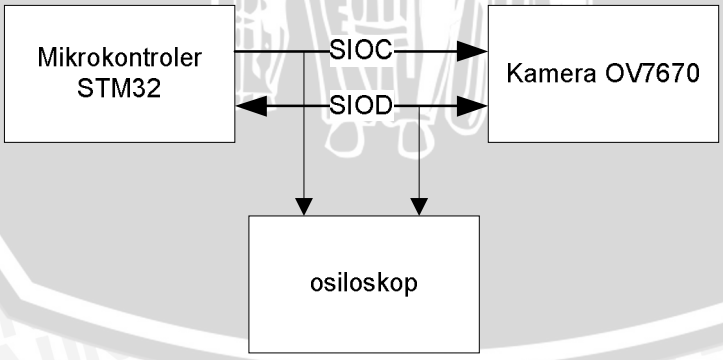
PENGUJIAN DAN ANALISIS

Pengujian dan analisis dilakukan untuk mengetahui apakah sistem telah bekerja sesuai perancangan yang telah dilakukan. Pengujian dilakukan perblok sistem kemudian secara keseluruhan. Adapun pengujian yang dilakukan sebagai berikut:

1) Pengujian komunikasi SCCB antara mikrokontroller dengan kamera OV7670
2) Pengujiaan hasil pengolahan gambar mikrokontroller
3) Pengujian respon sistem dengan kontroler logika *fuzzy*
4) Pengujian keseluruhan sistem
   a) Pengujian robot dengan garis lurus dan tikungan ke kanan
   b) Pengujian robot dengan garis lurus dan tikungan ke kiri

5.1 Pengujian Komunikasi SCCB

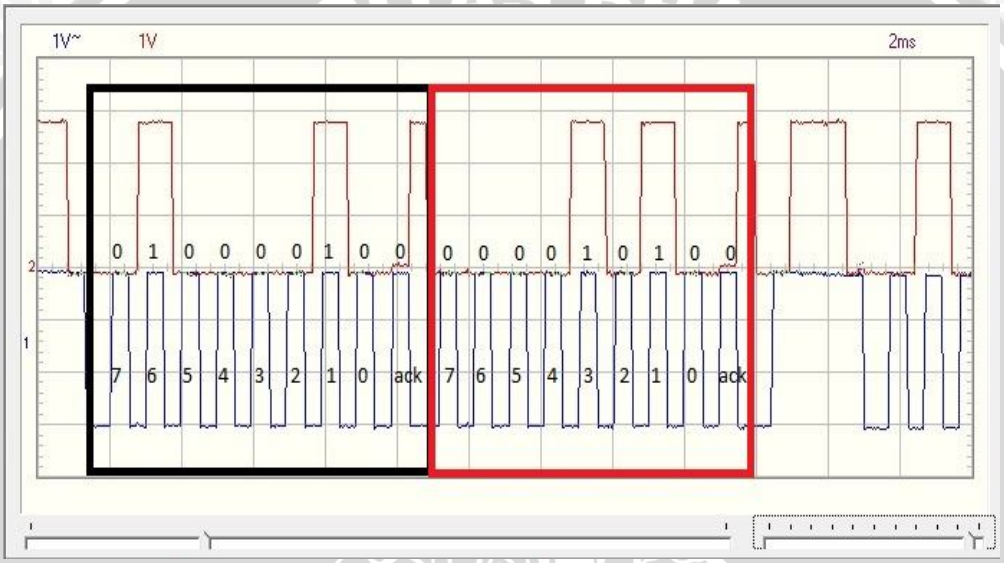
Pengujian ini dilakukan dengan tujuan untuk mengetahui apakah komunikasi SCCB untuk pengaturan kamera bisa dilakukan. Prosedur pengujian dilakukan dengan mikrokontroller membaca beberapa register dari kamera dan menampilkan sinyal pembacaan pada osiloskop.



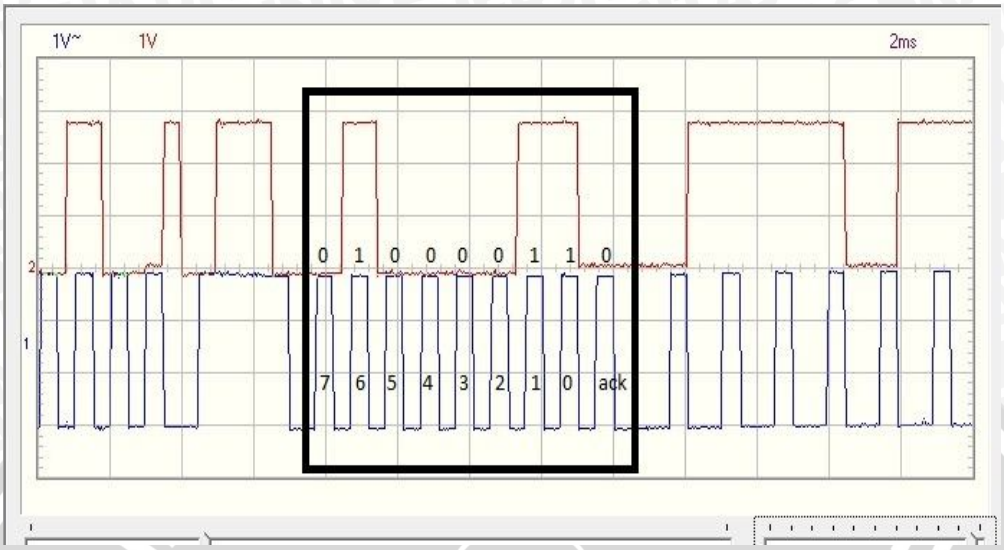
Gambar 5.1. Diagram Blok Pengujian Komunikasi SCCB

Pada pengujian ini, alamat register yang dibaca adalah 0x0A. register ini merupakan Product ID yang hanya bisa dibaca. Nilai dari register ini menurut *datasheet* OV7670 yaitu 0x76.

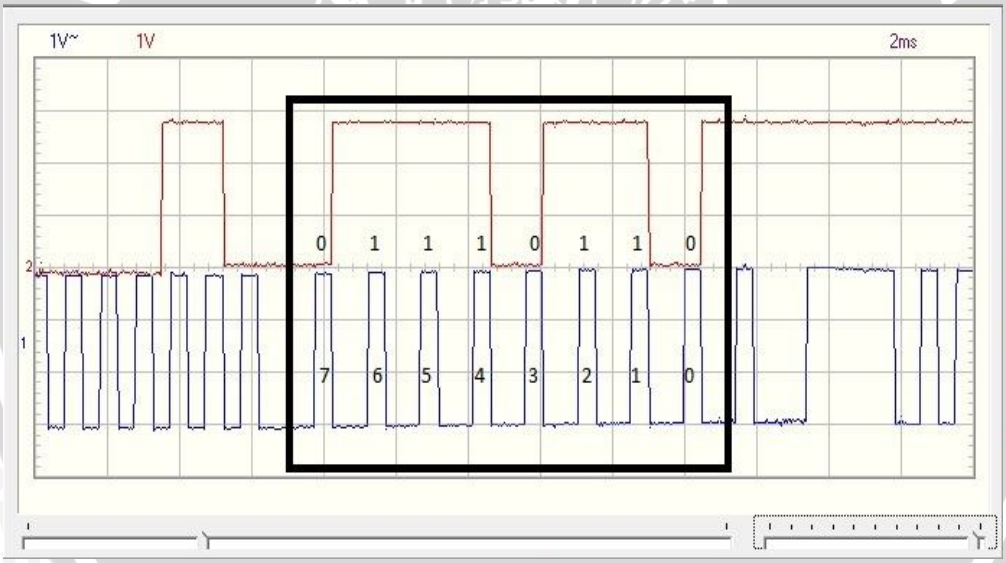
Untuk mendapat nilai dari register yang diinginkan terdapat tiga tahapan. Pertama pemilihan register yang terdiri atas instruksi tulis (0x42) yang dilanjutkan dengan alamat register yang dituju. Kemudian instruksi baca (0x43) dan yang ketiga adalah pembacaan isi register.



(a)



(b)



(c)

Gambar 5.2. (a) Instruksi pemilihan register, (b) Instruksi pembacaan register,

(c) Hasil pembacaan register

Gambar 5.2 menunjukkan sinyal SIOC dan SIOD. grafik atas menunjukkan SIOD dan grafik bawah menunjukkan SIOC. Pada Gambar 5.2.a dan Gambar 5.2.b setelah proses pengiriman satu byte data terdapat satu bit

tambahan yang berupa *acknowledge.* Pada percobaan *acknowledge* selalu berlogika *low* sehingga komunikasi berhasil. Hasil pengujian pembacaan data dari beberapa *register* OV7670 ditunjukkan dalam tabel 5.1

Tabel 5.1 Hasil Pengujian Komunikasi SCCB

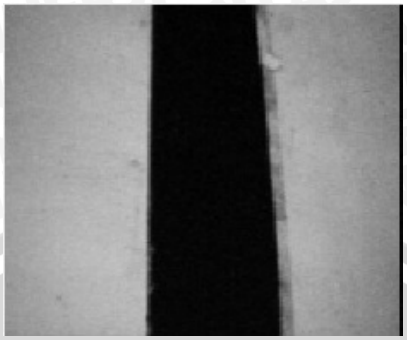
No.	Alamat Register (Hex)	Data pada datasheet (hex)	Data terbaca(hex)
1	0x0A	0x76	0x76
2	0x0B	0x73	0x73
3	0x1C	0x7F	0x7F
4	0x1D	0xA2	0xA2

Berdasarkan hasil pengujian pada Tabel 5.1 dapat disimpulkan bahwa komunikasi antara mikrokontroller dan kamera OV7670 berjalan baik, ditunjukkan oleh data yang terbaca oleh mikrokontroller sesuai data dalam *datasheet* OV7670.

5.2 Pengujiaan Pengolahan Gambar Oleh Mikrokontroller

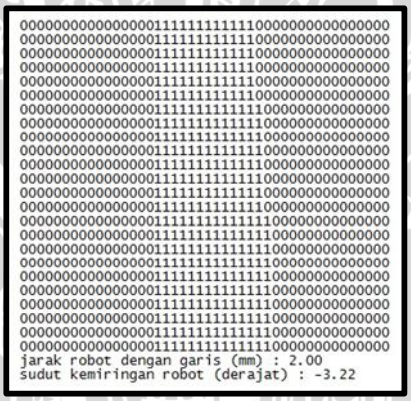
Pengujian ini dilakukan untuk mengetahui bagaimana hasil gambar yang sudah diolah oleh mikrokontroller dan juga untuk mengetahui ketelitian dari jarak robot terhadap garis (l) dan juga sudut robot(α). Ketelitian didapat dengan membandingkan data perhitungan dari gambar yang telah diolah dan data perhitungan dari gambar asli. Untuk pengujian ini, mikrokontroler mengirimkan data pada terminal komputer menggunakan konfigurasi *baudrate* 38400 bps, 8 bit data, tanpa paritas dan 1 stop bit. Data yang dikirimkan merupakan gambar yang telah diolah oleh mikrokontroller, dan juga hasil perhitungan nilai posisi robot terhadal garis (*l*) dan juga kemiringan robot terhadap garis (α) Prosedur pengujian dilakukan dengan menghitung nilai dari (*l*) dan (α) dari gambar asli, kemudian dibandingkan dengan hitungan dari gambar yang telah diolah oleh mikrokontroler.

Gambar 5.3 merupakan gambar asli sebelum gambar diolah oleh mikrokontroller. Gambar diambil saat robot berada di tengah garis.



Gambar 5.3 Gambar saat robot berada di tengah garis

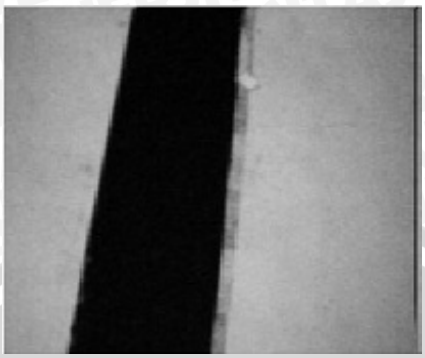
Pada perhitungan yang dilakukan secara manual pada Gambar 5.3 didapatkan jarak robot dengan garis (*l*) sejauh 2.36 mm dan juga sudut hadap robot (α) sebesar -2.13°. Gambar 5.4 merupakan hasil pengolahan gambar dari Gambar 5.3



Gambar 5.4 Gambar hasil pengolahan Gambar 5.3

Terdapat kesalahan pembacaan jarak robot dengan garis(*l*) sebesar 0.36 mm dan kesalahan pembacaan sudut sebesar 1.09°.

Gambar 5.5 menunjukkan gambar asli saat robot berada disebelah kanan garis.

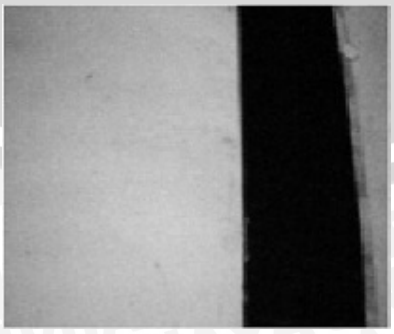


Gambar 5.5 Gambar saat robot berada di kanan garis

Pada perhitungan yang dilakukan secara manual pada Gambar 5.5 didapatkan jarak robot dengan garis (*l*) sejauh -16.08 mm dan juga sudut hadap robot (α) sebesar 7°. Gambar 5.6 merupakan hasil pengolahan gambar dari Gambar 5.5.

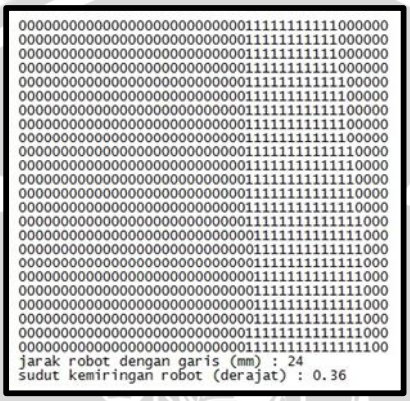
Gambar 5.6 Gambar 5.5 yang telah diolah oleh mikrokontroller

Terdapat kesalahan pembacaan jarak robot dengan garis(*l*) sebesar 0.08 mm dan kesalahan pembacaan sudut sebesar 0.76°. Gambar 5.7 menunjukkan gambar asli saat robot berada disebelah kiri garis.



Gambar 5.7 gambar hasil tangkapan kamera saat robot berada di kiri garis

Pada perhitungan yang dilakukan secara manual pada Gambar 5.7 didapatkan jarak robot dengan garis (*l*) sejauh 24.6 mm dan juga sudut hadap robot (α) sebesar 0°. Gambar 5.8 merupakan hasil pengolahan gambar dari Gambar 5.7.



Gambar 5.8 Gambar 5.7 yang telah diolah oleh mikrokontroller

Terdapat kesalahan pembacaan jarak robot dengan garis(*l*) sebesar 0.6 mm dan kesalahan pembacaan sudut sebesar 0.36°.

Tabel 5.2 Hasil pengujian ketelitian pembacaan (*l*) dan (α)

No	perhitungan gambar asli		perhitungan mikrokontoller		|Error *l*(%)|	|Error α(%)|
	l(mm)	α(°)	*l*(mm)	α(°)		
1	8.98	-2.72	8	-2.24	10.91	0.13
2	17.03	-6.85	18	-7.67	5.69	0.22
3	2.36	-2.13	2	-3.22	15.25	0.30
4	-16.08	7	-16	6.24	0.49	0.21
5	24.6	0	24	0.3	2.43	0.08
Error rata rata pembacaan			*l* (%)		6.954	
Error rata rata pembacaan			α(%)		0.18	

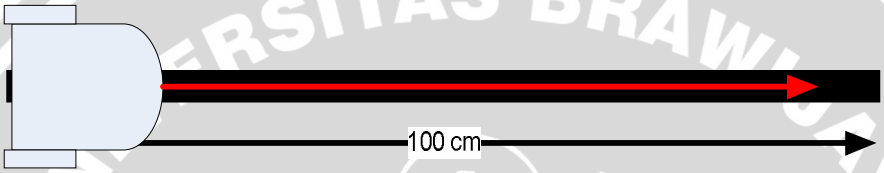
Berdasarkan hasil pengujian pada Tabel 5.2 yang telah dilakukan, dapat disimpulkan bahwa mikrokontroller dapat mengolah gambar dan menghitung nilai

(α) dan (*l*) dengan kesalahan rata rata pembacaan (*l*) 6.954 % dan kesalahan rata rata pembacaan (α) 0.18%.

5.3 Pengujiaan Kontroler Logika *Fuzzy*

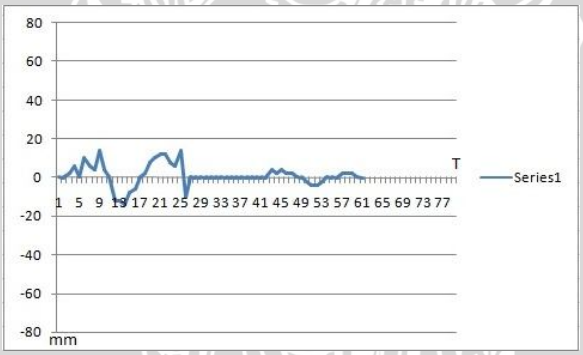
Tujuan dari pengujian ini adalah untuk mengamati respons sistem dengan menggunakan kontroler logika *fuzzy*. Prosedur pengujian memonitoring pergerakan dari robot pada lintasan lurus dan menikung dengan posisi awal yang bervariasi

5.3.1 Hasil Pengujian Lintasan Lurus



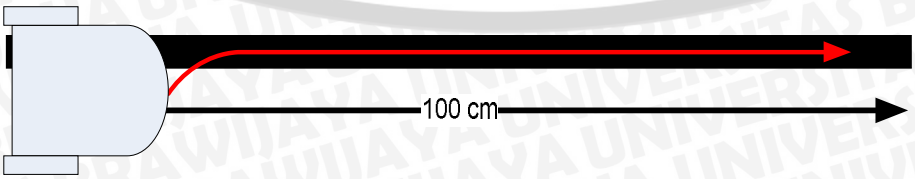
Gambar 5.9 Pengujian lintasan lurus saat kondisi awal robot di tengah garis

Hasil pengujian untuk respons posisi robot *line follower* saat robot berada di tengah garis ditunjukkan dalam Gambar 5.10.

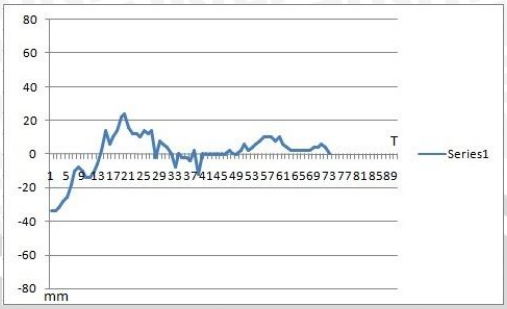


Gambar 5.10 Grafik respon posisi pada saat posisi awal robot di tengah garis

Terlihat bahwa saat robot mulai bergerak terjadi sedikit osilasi kemudian robot dapat mencapai *steady* dengan waktu 27T, T merupakan periode pengambilan data dari mikrokontroller dengan nilai $T = \frac{1}{15}\ sekon$. Maka dibutuhkan waktu 1.8 sekon untuk robot mencapai *steady*.

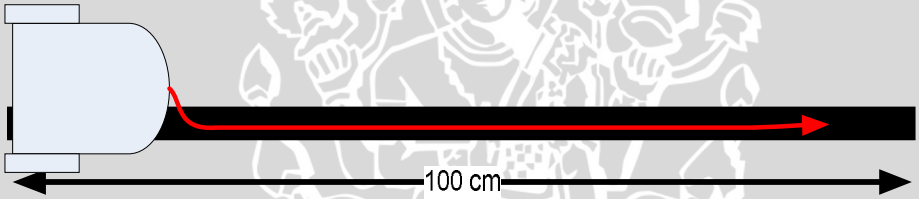


Gambar 5.11 Pengujian lintasan lurus saat kondisi awal robot di kanan garis

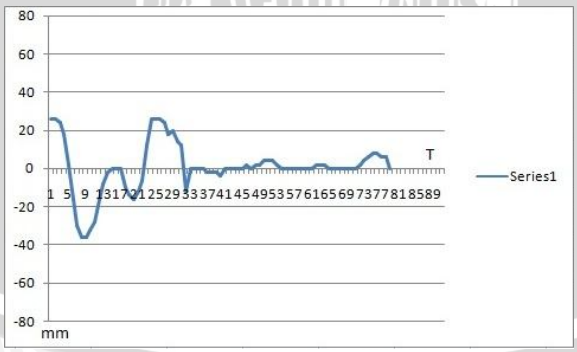


Gambar 5.12 Grafik respon posisi saat posisi awal robot di kanan garis

Gambar 5.12 menunjukkan grafik respons saat posisi awal robot berjarak -35 mm dari tengah lintasan. Robot berada di sebelah kanan dari lintasan. Dari grafik tersebut, pada T = 35 atau setelah 2.3 sekon robot sudah berada ditengah lintasan dan mengikuti lintasan dengan sedikit osilasi. Osilasi maksimum yang nampak dari grafik setelah robot mencapai *steady* yaitu ±10 mm.



Gambar 5. 13 Pengujian lintasan lurus saat kondisi awal robot di kiri garis

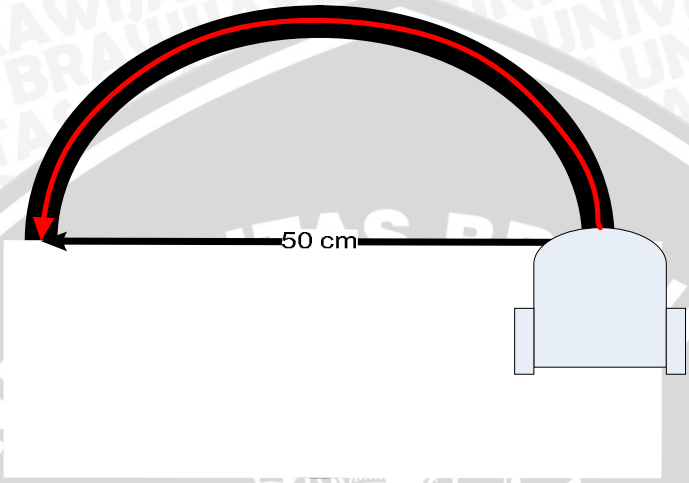


Gambar 5.14 Grafik respon posisi saat posisi awal robot di kiri garis

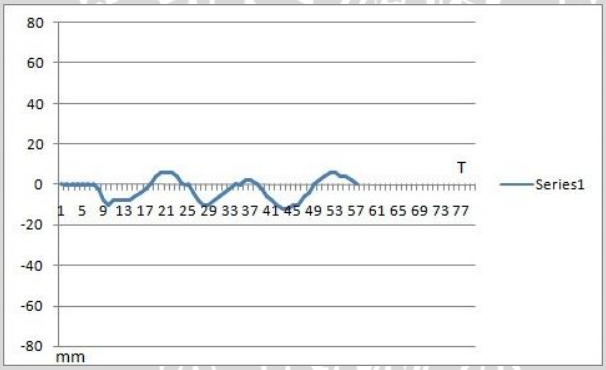
Gambar 5.14 merupakan grafik respon saat posisi awal robot berada pada jarak 28 mm dari tengah lintasan seperti yang ditunjukkan Gambar 5.13. Robot memerlukan waktu selama 34T atau 2.26 sekon untuk mencapai *steady*. Saat

robot telah mencapai *steady*, tampak terjadi sedikit osilasi dengan osilasi maksimum sebesar 10mm.

5.3.2 Hasil Pengujian Lintasan Menikung Kiri

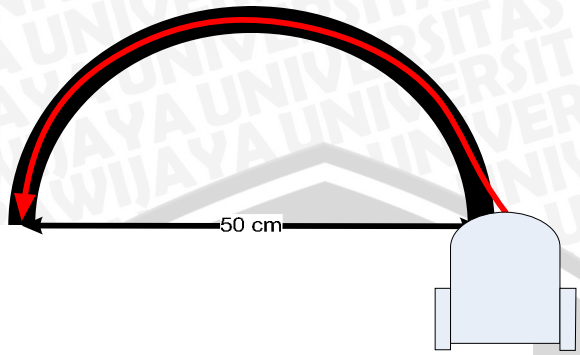


Gambar 5.15 Pengujian lintasan menikung kiri saat kondisi awal robot di tengah garis

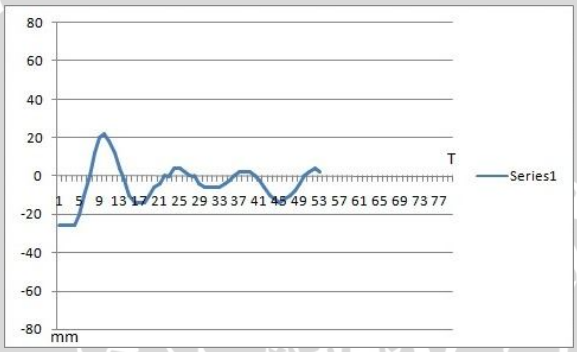


Gambar 5.16 Grafik respon posisi saat posisi awal robot di tengah garis

Pada pengujian dengan lintasan menikung yang ditunjukkan pada Gambar 5.15. Diperoleh respons posisi robot yang ditunjukkan dalam Gambar 5.16 yang selalu berosilasi. Hal ini disebabkan karena lintasan yang menikung sehingga robot tidak bisa bertahan di *set point*. Osilasi yang terjadi yaitu sebesar -10 mm.

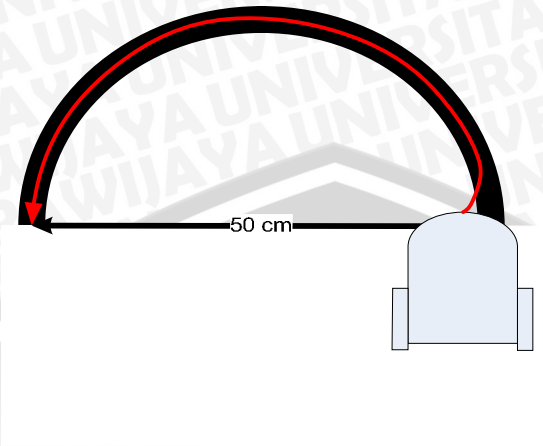


Gambar 5.17 Pengujian lintasan menikung kiri saat kondisi awal robot di kanan garis

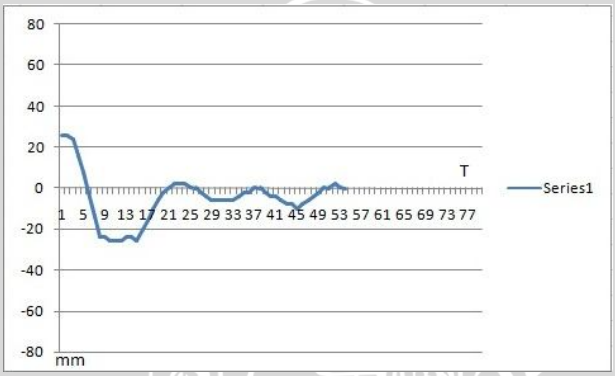


Gambar 5.18 Grafik respon posisi saat posisi awal robot di kanan lintasan

Dalam Gambar 5.17 , posisi robot berada -22 mm dari garis. Grafik respons posisi robot yang ditunjukkan Gambar 5.18 yang selalu berosilasi karena lintasan yang menikung, puncak dari osilasi pertama merupakan yang paling besar yaitu sebesar 20 mm. setelah itu gelombang osilasi bernilai negatif karena arah tikungan ke kiri.



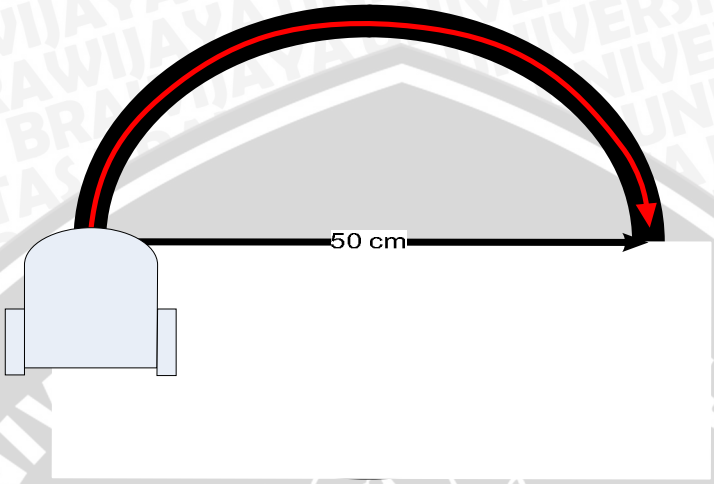
Gambar 5.19 Pengujian lintasan menikung kiri saat kondisi awal robot di kiri garis



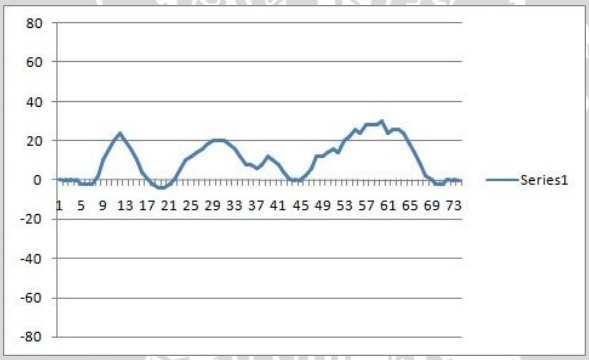
Gambar 5.20 Grafik respon posisi saat posisi awal robot di kiri lintasan

Dari hasil pengujian dengan kondisi awal robot seperti yang ditunjukkan dalam Gambar 5.19. Terlihat bahwa grafik respons posisi yang ditunjukkan Gambar 5.20 selalu berosilasi karena lintasan yang menikung pada osilasi pertama puncak osilasi mencapai -25mm atau 25mm di sebelah kanan lintasan.

5.3.3 Hasil Pengujian Lintasan Menikung Kanan

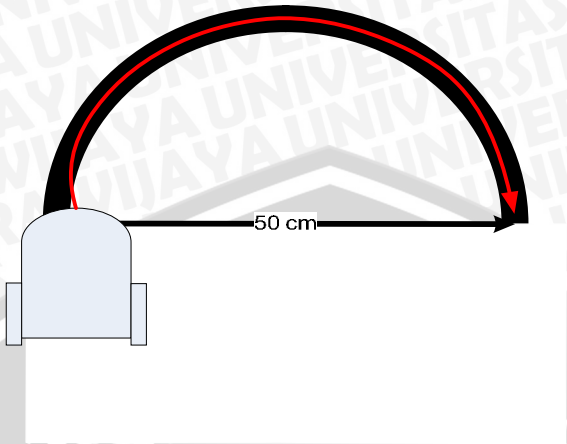


Gambar 5.21 Pengujian lintasan menikung kanan saat kondisi awal robot di tengah garis

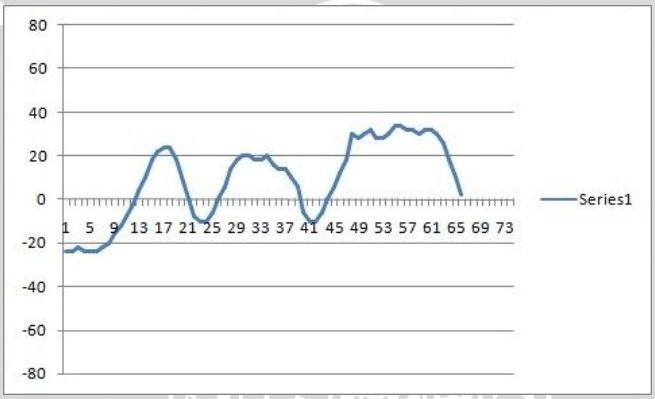


Gambar 5.22 Grafik respon posisi saat posisi awal robot di tengah garis

Pada pengujian dengan lintasan menikung ke kanan yang ditunjukkan pada Gambar 5.22. Posisi robot selalu berosilasi seperti yang ditunjukkan pada Gambar 5.22, hal ini disebabkan karena lintasan yang menikung sehingga robot tidak bisa bertahan di set point. Osilasi yang terjadi yaitu sebesar 30 mm. Karena menikung ke kanan maka gelombang osilasi bernilai positif.

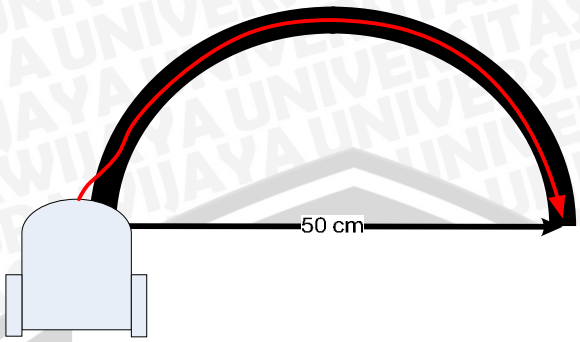


Gambar 5.23 Pengujian lintasan menikung kanan saat kondisi awal robot di kanan garis

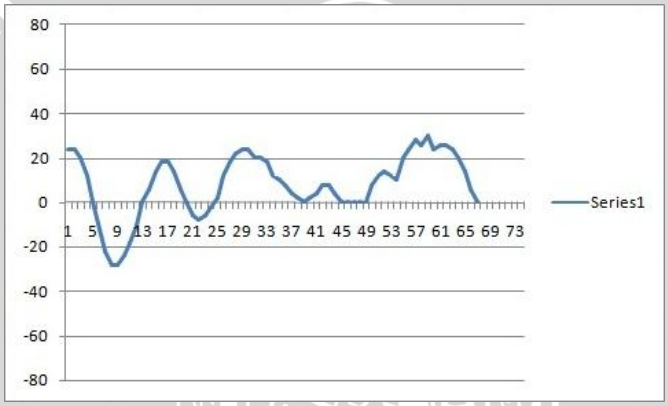


Gambar 5.24 Grafik respon posisi saat posisi awal robot di kanan lintasan

Pada pengujian saat kondisi awal robot berada -22 mm dari garis seperti yang ditunjukkan dalam Gambar 5.23. Grafik respons posisi robot dalam Gambar 5.24 juga selalu berosilasi karena lintasan yang menikung ke kanan dengan nilai osilasi positif.



Gambar 5.25 Pengujian lintasan menikung kanan saat kondisi awal robot di kiri garis



Gambar 5.26 Grafik respon posisi saat posisi awal robot di kiri lintasan

Hasil pengujian saat posisi awal robot berada di kiri garis seperti yang ditunjukkan pada gambar 5.25. Terlihat bahwa grafik respons posisi robot dalam Gambar 5.26 selalu berosilasi karena lintasan yang menikung ke kanan. Pada osilasi pertama puncak osilasi mencapai -25mm atau 25mm di sebelah kanan lintasan. Setelah itu gelombang osilasi bernilai positif.

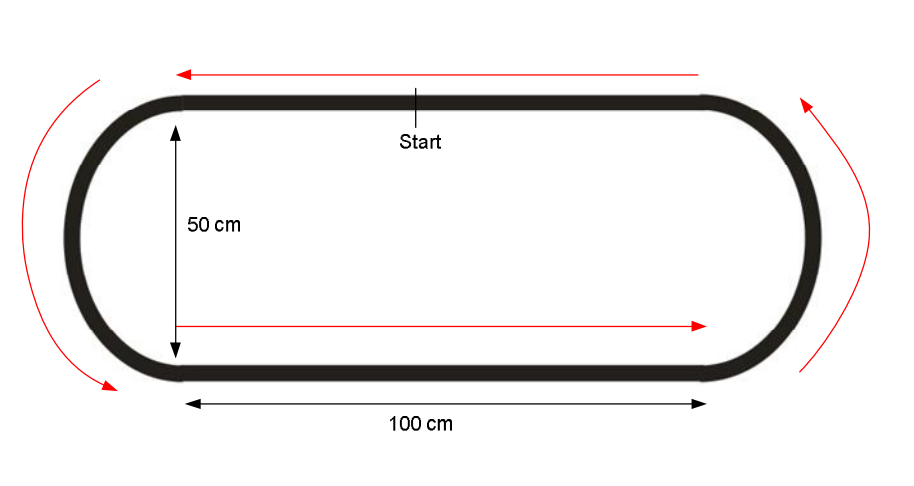
5.4 Pengujian Keseluruhan Sistem

Pengujian keseluruhan sistem terdiri atas dua bagian yaitu pengujian robot dengan tikungan ke kanan dan dengan tikungan ke kiri pada model lapangan pengujian. model lapangan yang memiliki dua lintasan lurus yang masing masing memiliki panjang 100cm dan dua buah tikungan berupa setengah lingkaran dengan diameter 50cm.

Prosedur pengujian dilakukan dengan meletakkan robot pada posisi awal dalam arena dan mengaktifkan robot agar bergerak megikuti lintasan dan mencatat lamanya waktu yang diperlukan robot untuk menyelesaikan satu puturan atau mencatat waktu dan posisi terakhir robot bila robot tidak dapat menyelesaikan satu putaran.

5.4.1 Pengujian Robot dengan Tikungan ke Kiri

Pengujian ini dilakukan untuk mengetahui performa sistem yang telah dirancang. Arena pengujian dan ilustrasi jalur pergerakan robot ditunjukan pada Gambar 5.27.



Gambar 5.27 Ilustrasi pergerakan robot pada arena pengujian dengan tikungan ke kiri

Robot diletakan pada posisi awal (*Start*) kemudian bergerak menelusuri arena berupa lintasan lurus dan juga lintasan menikung ke kiri. Arah panah merah dalam Gambar 5.27 menunjukan ilustrasi rute yang dilewati robot selama proses pengujian. Hasil pengujian ditunjukan pada Tabel 5.3.

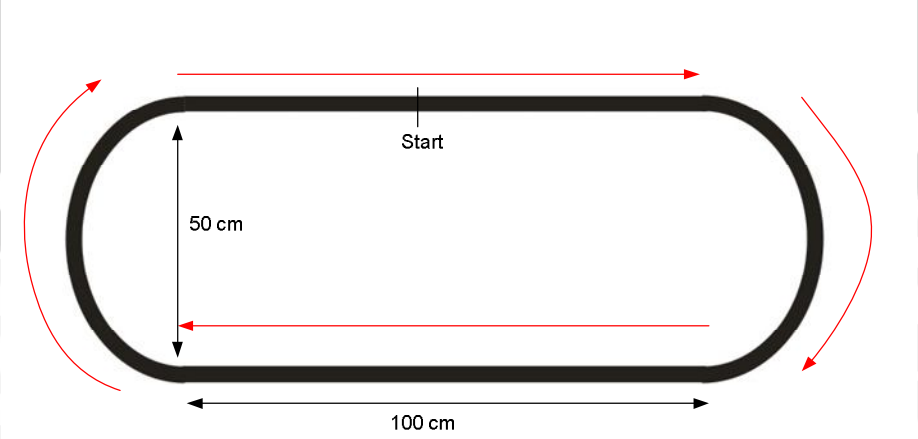
Tabel 5.3 Hasil pengujian robot dengan tikungan ke kiri

Pengujian ke-	Lama waktu (detik)	Hasil Pengujian
1	10,75	Berhasil menyelesaikan lintasan
2	11.7	Berhasil menyelesaikan lintasan
3	11.6	Berhasil menyelesaikan lintasan
4	10.2	Gagal menyelesaikan lintasan, keluar di tikungan ke tiga
5	11.9	Berhasil menyelesaikan lintasan

Berdasarkan hasil pengujian dapat diketahui bahwa robot *line follower* dengan mengimplementasikan kamera OV7670 pada mikrokontroller STM32F407VGT6 berhasil menjadi masukkan dari kontroler logika *fuzzy* yang membuat keputusan dalam menentukan pergerakan robot selama mengikuti lintasan.

5.4.2 Pengujian Robot dengan Tikungan ke Kanan

Pengujian ini dilakukan untuk mengetahui performa sistem yang telah dirancang. Arena pengujian dan ilustrasi jalur pergerakan robot ditunjukan pada Gambar 5.28.



Gambar 5.28 Ilustrasi pergerakan robot pada arena pengujian dengan tikungan ke kanan

Robot diletakan pada posisi awal (*Start*) bergerak menelusuri arena berupa lintasan lurus dan juga lintasan menikung ke kanan. Arah panah merah pada Gambar 5.28 menunjukan ilustrasi rute yang dilewati robot selama proses pengujian. Hasil pengujian ditunjukan pada Tabel 5.4.

Tabel 5.4 Hasil pengujian robot dengan tikungan ke kanan

Pengujian ke-	Lama waktu (detik)	Hasil Pengujian
1	11.0	Berhasil menyelesaikan lintasan
2	11.3	Berhasil menyelesaikan lintasan
3	11.07	Berhasil menyelesaikan lintasan
4	11.20	Berhasil menyelesaikan lintasan
5	11.84	Berhasil menyelesaikan lintasan

Berdasarkan hasil pengujian dapat diketahui bahwa robot *linefollower* dengan mengimplementasikan kamera OV7670 pada mikrokontroller STM32F407VGT6 berhasil menjadi masukkan dari kontroler logika *fuzzy* yang membuat keputusan dalam menentukan pergerakan robot selama mengikuti lintasan.

Secara garis besar, Hasil pengujian keseluruhan yang dilakukan baik pengujian dengan tikungan ke kanan dan pengujian lintasan dengan tikungan ke kiri mendapatkan kemiripan hasil karena pada perancangan yang telah dilakukan menggunakan aturan aturan dan fungsi keanggotaan yang sama.

BAB VI

KESIMPULAN DAN SARAN

6.1 Kesimpulan

Dari hasil perancangan dan pengujian yang telah dilakukan, maka dapat disimpulkan beberapa hal sebagai berikut.

1) Pengaturan kamera OV7670 oleh mikrokontroller melalui kamera SCCB berjalan dengan baik. ditunjukkan oleh data yang terbaca oleh mikrokontroller sesuai data pada *datasheet* OV7670.

2) Mikrokontroller dapat mengolah gambar dan menghitung nilai (α) dan (*l*) dengan kesalahan rata rata pembacaan (*l*) 6.954 % dan kesalahan rata rata pembacaan (α) 0.18%.

3) Dengan Penerapan kontroler logika *fuzzy* pada robot *line follower* membuat navigasi robot menjadi stabil dalam mengikuti lintasan yang berupa garis lurus dan tikungan baik ke kiri maupun ke kanan. Dan mampu untuk menyelesaikan lintasan dengan waktu rata – rata 11.37 detik.

4) Pada lintasan lurus robot line follower dapat mencapai keaadaan steady dengan waktu 2.3 sekon jika kondisi awal robot berada tidak di tengah garis. Saat kondisi awal robot di tengah garis robot mencapai keadaan steady setelah 1.8 sekon.

6.2 Saran

Beberapa hal yang direkomendasikan untuk pengembangan lebih lanjut adalah sebagai berikut:

1) Perlu adanya penambahan algoritma pembelajaran untuk menentukan parameter fungsi keanggotaan supaya kendali fuzzy bisa lebih optimal.

1) Perlu digunakan algoritma pendeteksi garis yang lebih baik supaya tidak bergantung pada penerangan lampu LED.

Daftar Pustaka

STMicroelectronics. 2012. *STM32F4DISCOVERY, STM32F4 high performance discovery board*. my.st.com. Diakses tanggal: 31 Mei 2013.

STMicroelectronics. 2012. *STM32F405xx/STM32F407xx, ARM Cortex-M4 MCU+FPU, 210 DMIPS, up to 1MB Flash/192+1KB RAM, USB OTG HS/FS, Ethernet, 17 TIMs, 3 ADCs, 15 comm, interface & camera*. my.st.com. Diakses tanggal: 31 Mei 2013.

OmniVision. 2006. *OV7670/OV7171CMOS VGA (640x480) CameraChip sensor with OmniPixel Technology*. www.atmel.com/literatur. Diakses tanggal: 31 Mei 2013.

OmniVision. 2003. *OmniVision Serial Camera Control Bus (SCCB) Function Specification.* www.atmel.com/literatur. Diakses tanggal: 31 Mei 2013.

Aparicio, Jorge. 2012. *Hacking the OV7670 camera module (SCCB cheat sheet inseide).*http://embeddedprogrammer.blogspot.com/2012/07/hacking-ov7670-camera-module-sccb-cheat.html diakses tanggal 30 Mei 2013

M. Ibrahim, Ahmad. 2004. *Fuzzy Logic for Embedded Systems Aplications*. USA: Newnes.

Cook, Gerald. 2011. *Mobile Robot : Navigation, Control and Remote Sensing.* New Jersey : John Willey & Son, Inc.

Kuswandi, Son. 2007. *Kendali Cerdas: Teori dan Aplikasi Praktisnya*. Yogyakarta: ANDI.

Siegwart, Roland dan Illah R. Nourbakhsh. 2004. *Introduction to Autonomous Mobile robots*. London: MIT Press.

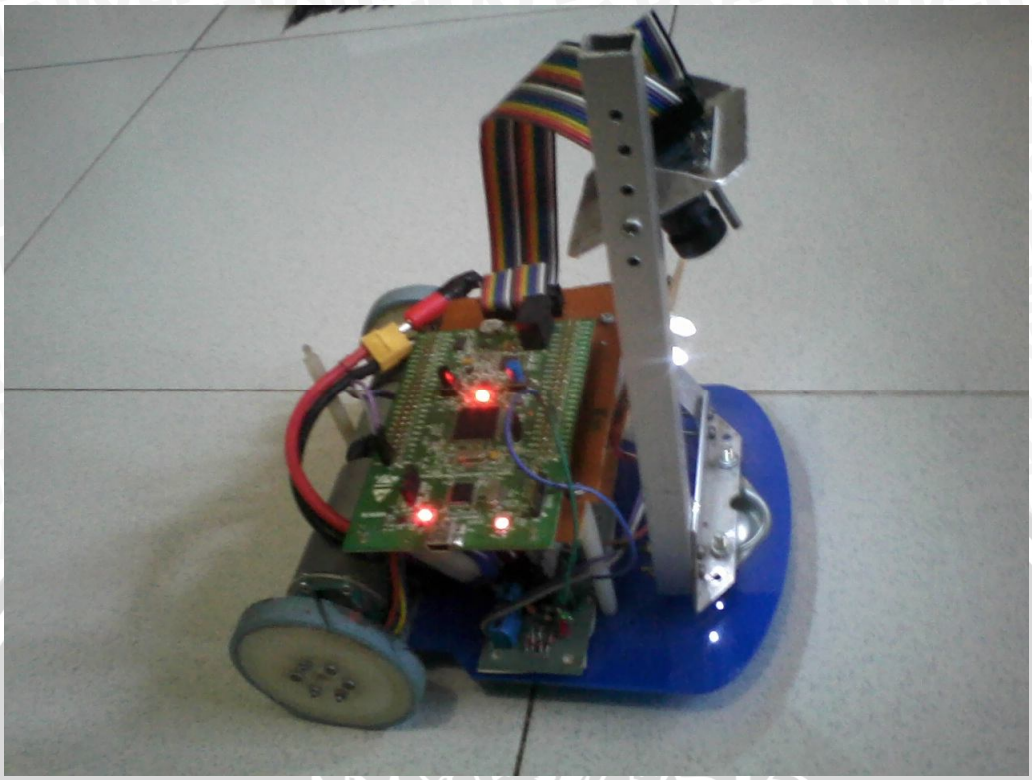
Tara, Rayi Yanu. 2010. *Desain dan Implementasi Logika Fuzzy sebagai Sistem Navigasi Wall Following pada Mobile Robot KRCI.* Malang:Skripsi Jurusan Teknik Elektro FT-UB.

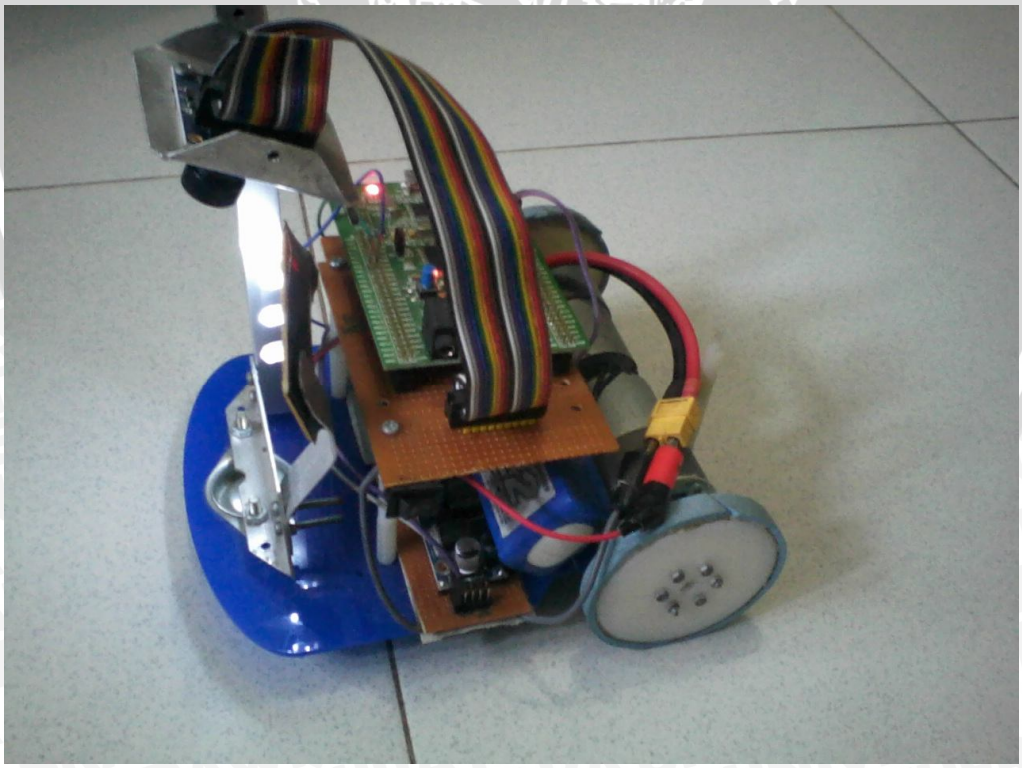
Thiang, Resmana Wahyudi. 2001. *Aplikasi Kendali Fuzzy Logic untuk Pengaturan Kecepatan Motor Universal.* Surabaya: Jurna; Jurusan Teknik Elektro FTI-Universitas Petra.

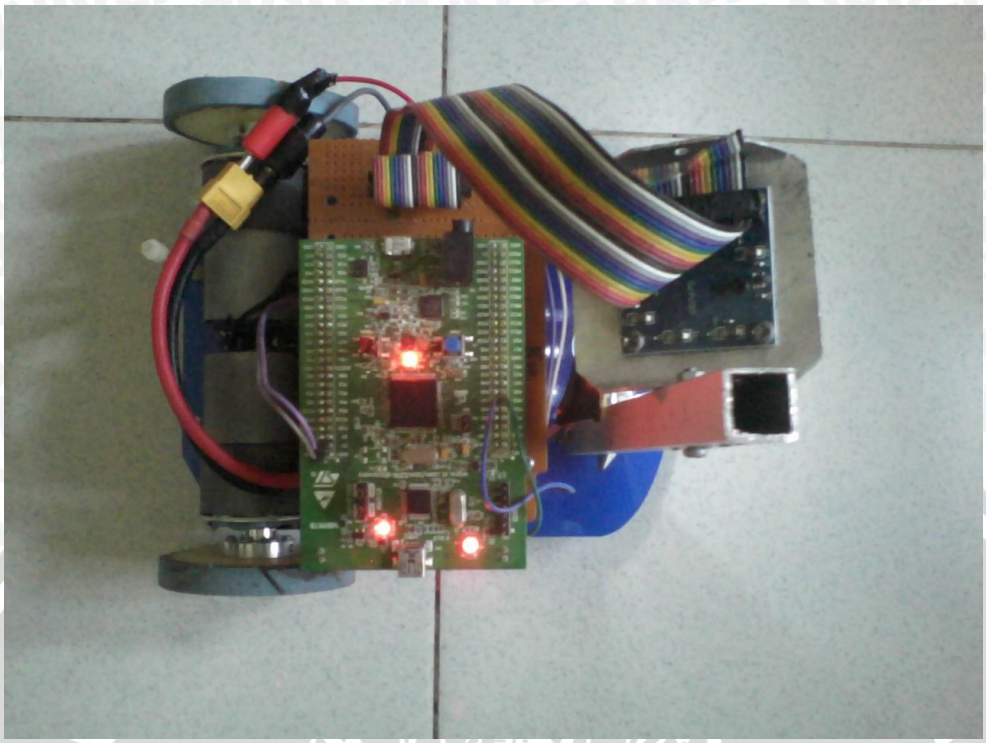
LAMPIRAN

LAMPIRAN

FOTO ALAT







LAMPIRAN II

LISTING PROGRAM

```
/************************************
=====================================
                LIBRARY
=====================================
************************************/
#include "stm32f4xx.h"
#include "stm32f4xx_rcc.h"
#include "stm32f4xx_flash.h"
#include "stm32f4xx_gpio.h"
#include "stm32f4xx_usart.h"
#include "stm32f4xx_dcmi.h"
#include "stm32f4xx_dma.h"
#include "stm32f4xx_tim.h"
#include "misc.h"
#include <stdio.h>
#include <math.h>

GPIO_InitTypeDef  GPIO_InitStructure;
DCMI_InitTypeDef DCMI_InitStructure;
DMA_InitTypeDef  DMA_InitStructure;
NVIC_InitTypeDef NVIC_InitStructure;
TIM_TimeBaseInitTypeDef TIM_TimeBaseInitStruct;
TIM_OCInitTypeDef TIM_OCInitStruct;

/************************************
=====================================
                Define
=====================================
************************************/
#define LED1_H GPIOD->BSRRL = GPIO_Pin_12  //Led 1 high(nyala)
#define LED2_H GPIOD->BSRRL = GPIO_Pin_13  //Led 2 high
#define LED3_H GPIOD->BSRRL = GPIO_Pin_14  //Led 3 high
#define LED4_H GPIOD->BSRRL = GPIO_Pin_15  //Led 4 high

#define LED1_L GPIOD->BSRRH = GPIO_Pin_12  //Led 1 low(mati)
#define LED2_L GPIOD->BSRRH = GPIO_Pin_13  //Led 2 low
#define LED3_L GPIOD->BSRRH = GPIO_Pin_14  //Led 3 low
#define LED4_L GPIOD->BSRRH = GPIO_Pin_15  //Led 4 low


#define SCL_H       GPIOB->BSRRL = GPIO_Pin_5              //PORTB.5 = 1
#define SCL_L       GPIOB->BSRRH  = GPIO_Pin_5         //PORTB.5 = 0

#define SDA_H       GPIOB->BSRRL = GPIO_Pin_4               //PORTB.4 = 1
#define SDA_L       GPIOB->BSRRH  = GPIO_Pin_4         //PORTB.4 = 0

#define SCL_read     GPIOB->IDR  & GPIO_Pin_5          //baca nilai input PINB.5
#define SDA_read     GPIO_ReadInputDataBit(GPIOB , GPIO_Pin_4)    //baca nilai input PINB.4
#define SDA_read_out  GPIOB->ODR & GPIO_Pin_4                //baca nilai output PORTB.4
#define ADDR_OV7725   0x42                              //alamat kamera

//definisi dimensi bidang tangkap kamera
#define lebar_piksel_L          2
#define lebar_piksel_al 2.14
#define tinggi_piksel           14
#define radian 57.32

//definisi fuzzy
#define ER_kiri                 0
#define ER_tengah   1
#define ER_kanan    2
#define Vkac                    0
#define Vkas                    1
#define Vt          2
#define Vkis                    3
#define Vkic                    4

uint8_t buff,gray=0;
uint32_t loop_y=0,loop_x=0,loop_y1=0,loop_x1=0,num;
uint16_t jap=0;
```

```
uint8_t frame_buffer[50688];
int piksel_l=0,piksel_alpha=0,pwm_ka,pwm_ki;
float alpha,l,output;
char buffer[32];
double rad;

//Aturan Fuzzy
unsigned char rule[3][3] =
        {
                        {Vkic,Vkis,Vkas},   //posisi kiri
                        {Vkis,Vt,Vkas},         //posisi tengah
                        {Vkis,Vkas,Vkac}    //posisi kanan
        };

float Error_sudut[3];
float Error_jarak[3];
float Rule_out[5];

int kp=7,kd=3,t;
float last_error,j;

/*************************************
=====================================
        Inisialisasi Sub Fungsi
=====================================
*************************************/
void STM32F4_Discovery_LEDInit(void);
void STM32F4_Discovery_MCO1init();
void SCCB_Configuration(void);
void SDA_OUT(void);
void SDA_IN(void);
void delay_us(unsigned int i);
void SCCB_Start(void);
void SCCB_Stop(void);
void SCCB_noAck(void);
unsigned char SCCB_writeByte(unsigned char m_data);
unsigned char SCCB_readByte(void);
unsigned char read_OV7670(unsigned char regID, unsigned char *regDat);
unsigned char write_OV7670(unsigned char regID, unsigned char regDat);
void usart_init(void);
void USART_puts(USART_TypeDef* USARTx, volatile char *s);
void Camera_Configuration();
void PWM_Configuration();
void Camera_Init();
void FIND_L();
void FIND_Alpha();
void ftoa(char *pStr,float op);
float triangle(float value, float x0, float x1, float x2);
float grade(float value, float x0, float x1);
float reverse_grade(float value, float x0, float x1);
float trapesium(float value,float x0,float x1,float x2,float x3);
void fuzzifikasi(float sudut, float jarak_l);
void check_rule();
float defuzzifikasi();


/*************************************
=====================================
                Program Utama
=====================================
*************************************/
int main(void)
{
        RCC_DeInit();                                   //reset peripheral
        FLASH_PrefetchBufferCmd(ENABLE);  //aktifkan flash
        RCC_HSEConfig(RCC_HSE_ON);                      //aktifkan clock eksternal
        while ( SUCCESS != RCC_WaitForHSEStartUp () ); //tunggu sampai sukses
        FLASH_SetLatency(FLASH_Latency_4); //set latency
        RCC_PLLConfig(RCC_PLLSource_HSI,16,336,2,4);            //set PLL
        RCC_SYSCLKConfig(RCC_SYSCLKSource_PLLCLK);      //pll menjadi sistem clock
```

```
        RCC_HCLKConfig(RCC_SYSCLK_Div1);        //set clock peripheral
        RCC_PCLK1Config(RCC_HCLK_Div1);         //set clock peripheral
        RCC_PCLK2Config(RCC_HCLK_Div1);         //set clock peripheral
        RCC_PLLCmd(ENABLE);                     //aktifkan PLL

        STM32F4_Discovery_MCO1init();           //set sumber clock utk kamera
        STM32F4_Discovery_LEDInit();            //set led
        SCCB_Configuration();                   //set komunikasi sccb
        PWM_Configuration();                    //set pwm
        usart_init();                           //set usart
        delay_us(500);
        Camera_Configuration();                 //isi register kamera
        delay_us(500);
        Camera_Init();                          //set antarmuka kamera


LED2_H;         //penanda led
LED1_L;
TIM2->CCR3 = 0;//pwm=0
TIM2->CCR4 = 0;
pwm_ka=0;
pwm_ki=0;
while(0==GPIO_ReadInputDataBit(GPIOA,GPIO_Pin_0))
{
                //tunggu tombol ditekan

}


        while(1)
{

        DMA_ClearITPendingBit(DMA2_Stream1,DMA_IT_TCIF1);
        while(DMA_GetITStatus(DMA2_Stream1,DMA_IT_TCIF1)== RESET); //tunggu flag DMA
        DCMI_Cmd(DISABLE);                      //matikan sementara fitur kamera
        DCMI_CaptureCmd(DISABLE);

        FIND_L();                       //hitung letak l
        FIND_Alpha();                   //hitung letak sudut
        DCMI_Cmd(ENABLE);               //aktifkan kamera
        DCMI_CaptureCmd(ENABLE);
        DMA_Cmd(DMA2_Stream1, ENABLE);


        if((piksel_l>80)||(piksel_alpha>80)) //jika keluar jalur
        {
                if((piksel_l>80) || (piksel_alpha>80))
                {

                        if(last_error > 0){pwm_ka=0;pwm_ki=360;}
                        else if(last_error < 0){pwm_ka=360;pwm_ki=0;}
                }

        }

        else if(piksel_l<80) //jika mendeteksi garis
        {
        l = piksel_l * lebar_piksel_L;    //konversi piksel ke mm
        alpha = ((piksel_alpha * lebar_piksel_al) - l) / tinggi_piksel;
        alpha= (atan (alpha)) * radian; //konversi piksel ke sudut

    fuzzifikasi(alpha,l);           //fuzy
        check_rule();
        output = defuzzifikasi();
        pwm_ki = ((40-((output * 40)/100))/80) * 760;
        pwm_ka = ((40+((output * 40)/100))/80) * 760;
        last_error = alpha;
        }

        //batasi keluaran
```

```
        if(pwm_ka>400)pwm_ka=400;
        if(pwm_ka<0)pwm_ka=0;
        if(pwm_ki>400)pwm_ki=400;
        if(pwm_ki<0)pwm_ki=0;
        TIM2->CCR3 = pwm_ka;
        TIM2->CCR4 = pwm_ki;
        GPIO_ToggleBits(GPIOD, GPIO_Pin_12); //penanda bergantian

    }
}


/************************************
=====================================
                Sub Fungsi
=====================================
************************************/
void FIND_L()
{
        uint8_t j=0,m=0;
        uint16_t k;
          //penurunan resolusi baris terakhir
           for(loop_x1=1;loop_x1<45;loop_x1++)
                        {
                                for(loop_y=1;loop_y<7;loop_y++)
                                {
                                        for(loop_x=1;loop_x<5;loop_x++)
                                        {
                                                //num  =((((loop_x1-1)*2)+loop_x)*2) - 1;
                                                num  = (loop_x*2) - 1;
                                                num += ((loop_x1-1)*8);
                                                num += ((loop_y-1)*352);
                                                num += 48576;
                                                t = (2*(frame_buffer[num]-80)) + 80;
                                                if(t>20) gray++;        //100
                                        }

                                }
                                num = loop_x1 + 1012;
                                if(gray>8)frame_buffer[num] = 0;
                                else frame_buffer[num] = 1;
                                gray=0;

                        }

        //jika piksel pertama bukan garis
        if(frame_buffer[1013]==0)
        {
                for (jap=1013; jap<1057; jap++)
                {
                        if(frame_buffer[jap]==1)j++;
                        if(j==8)
                        {
                                k=jap;

                        }
                }
                if(j>0)      //<15
                {
                        for (jap=1013; jap<1057; jap++)
                                        {
                                                if(frame_buffer[jap]==1)m++;
                                                if(m==(j/2))
                                                {
                                                        k=jap;

                                                }
                                        }

                }
```

```
}
//jika piksel pertama garis
if((frame_buffer[1013]==1) && (frame_buffer[1014]==1))
{
        for (jap=1057; jap>1013; jap--)
        {
                if(frame_buffer[jap]==1)j++;
                if(j==8)
                {
                        k=jap;

                }
        }
        if(j>0)
                        {
                                for (jap=1057; jap>1013; jap--)
                                                {
                                                        if(frame_buffer[jap]==1)m++;
                                                        if(m==(j/2))
                                                        {
                                                                k=jap;

                                                        }
                                                }

                        }
}

if(k<1035) piksel_l = k-1034;
if(k>1034) piksel_l = k-1035;
if(j==0)piksel_l=100;
}

void FIND_Alpha()
{
        uint8_t j=0,m=0;
        uint16_t k;


   //penurunan resolusi baris pertama
   for(loop_x1=1;loop_x1<45;loop_x1++)
                        {
                                for(loop_y=1;loop_y<7;loop_y++)
                                {
                                        for(loop_x=1;loop_x<5;loop_x++)
                                        {

                                                num  = (loop_x*2) - 1;
                                                num += ((loop_x1-1)*8);
                                                num += ((loop_y-1)*352);
                                                num += 40128;
                                                t = (2*(frame_buffer[num]-80)) + 80;
                                                if(t>20) gray++;        //100
                                        }

                                }
                                num = loop_x1;
                                if(gray>8)frame_buffer[num] = 0;
                                else frame_buffer[num] = 1;
                                gray=0;

                        }

        //jika piksel pertama bukan garis
        if(frame_buffer[1]==0)
        {
                for (jap=1; jap<45; jap++)
                {
                        if(frame_buffer[jap]==1)j++;
                        if(j==7)
```

```
                {
                        k=jap;
                        //break;
                }
        }
        if(j>0)     //<13
        {
                for (jap=1; jap<45; jap++)
                        {
                                if(frame_buffer[jap]==1)m++;
                                if(m==(j/2))
                                {
                                        k=jap;
                                        //break;
                                }
                        }
        }
}

        //jika piksel pertama garis
        if((frame_buffer[1]==1) && (frame_buffer[1]==1))
        {
                for (jap=45; jap>1; jap--)
                {
                        if(frame_buffer[jap]==1)j++;
                        if(j==7)
                        {
                                k=jap;
                                //break;
                        }
                }
                if(j>0)
                        {
                                for (jap=45; jap>1; jap--)
                                        {
                                                if(frame_buffer[jap]==1)m++;
                                                if(m==(j/2))
                                                {
                                                        k=jap;
                                                        //break;
                                                }
                                        }
                        }
        }
        if(k<23) piksel_alpha = k-22;
        if(k>22) piksel_alpha = k-23;
        if(j==0) piksel_alpha = 100;
}


void STM32F4_Discovery_LEDInit(void) //set I/O Led
{
        RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOD, ENABLE);
        GPIO_InitStructure.GPIO_Pin = GPIO_Pin_12 | GPIO_Pin_13| GPIO_Pin_14| GPIO_Pin_15;
        GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;
        GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
        GPIO_InitStructure.GPIO_Speed = GPIO_Speed_100MHz;
        GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
        GPIO_Init(GPIOD, &GPIO_InitStructure);
}


void STM32F4_Discovery_MCO1init()     //set sumber clock kamera 8 Mhz
{

         RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOA, ENABLE);
         GPIO_InitStructure.GPIO_Pin = GPIO_Pin_8;
         GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF;
         GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
         GPIO_InitStructure.GPIO_PuPd =GPIO_PuPd_NOPULL;
```

```
        GPIO_InitStructure.GPIO_Speed = GPIO_Speed_100MHz;
        GPIO_Init(GPIOA, &GPIO_InitStructure);

        GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0;
        GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN;
        GPIO_Init(GPIOA, &GPIO_InitStructure);

        RCC_MCO1Config(RCC_MCO1Source_HSE, RCC_MCO1Div_1);
        GPIO_PinAFConfig(GPIOA, GPIO_PinSource8, GPIO_AF_MCO);

}

void SCCB_Configuration(void)                   //set I/O SCCB
{
 RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOB, ENABLE);
 GPIO_InitStructure.GPIO_Pin =  GPIO_Pin_5 | GPIO_Pin_4;
 GPIO_InitStructure.GPIO_Speed = GPIO_Speed_100MHz;
 GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;
 GPIO_InitStructure.GPIO_OType = GPIO_OType_OD;
 GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
 GPIO_Init(GPIOB, &GPIO_InitStructure);
}


void SDA_OUT(void)          //set Sda(sccb) sebagai output
{

        GPIO_InitStructure.GPIO_Pin =  GPIO_Pin_4;
        GPIO_InitStructure.GPIO_Speed = GPIO_Speed_100MHz;
        GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;
        GPIO_InitStructure.GPIO_OType = GPIO_OType_OD;
        GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
        GPIO_Init(GPIOB, &GPIO_InitStructure);

}

void SDA_IN(void)   //set sda(sccb) sebagai input
{

      GPIO_InitStructure.GPIO_Pin =  GPIO_Pin_4;
      GPIO_InitStructure.GPIO_Speed = GPIO_Speed_100MHz;
      GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN;
      GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
      GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
      GPIO_Init(GPIOB, &GPIO_InitStructure);

}

void delay_us(unsigned int i)
 {
        uint8_t x;
  while( i--)
    {
                for(x=0;x<8;x++);
    }
 }

/**************************************
======================================
        Sub fungsi SCCB
======================================
**************************************/
void SCCB_Start(void)          //mulai sccb
{
   SDA_H;
   delay_us(500);
   SCL_H;
   delay_us(500);
   SDA_L;
   delay_us(500);
```

```
    SCL_L;
    delay_us(500);


}

void SCCB_Stop(void)            //stop sccb
{
    SDA_L;
    delay_us(500);
    SCL_H;
    delay_us(500);
    SDA_H;
    delay_us(500);
}

void SCCB_noAck(void)           //pemberian acknowledge
{
        SDA_H;
        delay_us(500);
        SCL_H;
        delay_us(500);
        SCL_L;
        delay_us(500);
        SDA_L;
        delay_us(500);
}

unsigned char SCCB_writeByte(unsigned char m_data)  //kirim data 1 byte
{
        unsigned char j,tem;

        for(j=0;j<8;j++)
        {
                if((m_data<<j)&0x80)
                {
                        SDA_H;
                }
                else
                {
                        SDA_L;
                }
                delay_us(250);
                SCL_H;
                delay_us(500);
                SCL_L;
                delay_us(250);

        }
        delay_us(250);
        SDA_IN();
        //delay_us(500);
        SCL_H;
        delay_us(500);
        if(SDA_read){tem=0;}
        else {tem=1;}
        SCL_L;
        delay_us(500);
    SDA_OUT();/*ÉèÖÃSDAÎª¹¦Äü*/

        return (tem);
}

unsigned char SCCB_readByte(void)       //baca data 1 byte
{
        unsigned char read,j;
        read=0x00;

        SDA_IN();
        delay_us(500);
```

```
	for(j=8;j>0;j--)
	{
		delay_us(500);
		SCL_H;
		delay_us(500);
		read=read<<1;
		if(SDA_read)
		{
			read=read+1;
		}
		SCL_L;
		delay_us(500);
	}
	return(read);
}

//register yang dibaca nilainya
unsigned char read_OV7670(unsigned char regID, unsigned char *regDat)
{

	SCCB_Start();
	if(0==SCCB_writeByte(0x42))
	{
		SCCB_Stop();
		return(0);
	}
	delay_us(100);
	if(0==SCCB_writeByte(regID))
	{
		SCCB_Stop();
		return(0);
	}
	SCCB_Stop();

	delay_us(100);

	SCCB_Start();
	if(0==SCCB_writeByte(0x43))
	{
		SCCB_Stop();
		return(0);
	}
	delay_us(100);
	*regDat=SCCB_readByte();
	SCCB_noAck();
	SCCB_Stop();
	return(1);
}

//tulis register
unsigned char write_OV7670(unsigned char regID, unsigned char regDat)
{
	SCCB_Start();
	if(0==SCCB_writeByte(0x42))
	{
		SCCB_Stop();
		return(0);
	}
	delay_us(100);
	if(0==SCCB_writeByte(regID))
	{
		SCCB_Stop();
		return(0);
	}
	delay_us(100);
	if(0==SCCB_writeByte(regDat))
	{
		SCCB_Stop();
		return(0);
	}
```

```
        SCCB_Stop();

        return(1);
}

//konfigurasi Uart

void usart_init(void)
{
  /* Konfigurasi:
      - BaudRate = 115200 baud
      - Word Length = 8 Bits
      - One Stop Bit
      - No parity
      - Hardware flow control disabled (RTS and CTS signals)
      - Receive and transmit enabled
  */

  USART_InitTypeDef USART_InitStructure;

  /* Enable GPIO clock */
  RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOC, ENABLE);

  /* Enable UART clock */
  RCC_APB1PeriphClockCmd(RCC_APB1Periph_USART3, ENABLE);

  /* Connect PXx to USARTx_Tx*/
  GPIO_PinAFConfig(GPIOC, GPIO_PinSource10, GPIO_AF_USART3);

  /* Connect PXx to USARTx_Rx*/
  GPIO_PinAFConfig(GPIOC, GPIO_PinSource11, GPIO_AF_USART3);

  /* Konfigurasi pin Tx sebagai fungsi alternatif */
  GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
  GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_UP;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF;

  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_10;
  GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
  GPIO_Init(GPIOC, &GPIO_InitStructure);

  /* Konfigurasi Rx sebagai fungsi alternatif  */
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF;
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_11;
  GPIO_Init(GPIOC, &GPIO_InitStructure);

  USART_InitStructure.USART_BaudRate = 38400;
  USART_InitStructure.USART_WordLength = USART_WordLength_8b;
  USART_InitStructure.USART_StopBits = USART_StopBits_1;
  USART_InitStructure.USART_Parity = USART_Parity_No;
  USART_InitStructure.USART_HardwareFlowControl = USART_HardwareFlowControl_None;
  USART_InitStructure.USART_Mode = USART_Mode_Tx | USART_Mode_Rx;

  /* USART configuration */
  USART_Init(USART3, &USART_InitStructure);

  /* Enable USART */
  USART_Cmd(USART3, ENABLE);
  //USART_ITConfig(USART3,USART_IT_RXNE,ENABLE);
}

//kirim data lewat usart
void USART_puts(USART_TypeDef* USARTx, volatile char *s)
{

        while(*s){

                while( !(USARTx->SR & 0x00000040) );
                USART_SendData(USARTx, *s);
                *s++;
```

```
        }
}

//isi register kamera
void Camera_Configuration()
{
        while(0==write_OV7670(0x12,0x80));      //reset register
        delay_us(4000);
        while(0==write_OV7670(0x11,0x01));      //clock prescaler
        while(0==write_OV7670(0x6B,0x8A));      //8A
        while(0==write_OV7670(0x12,0x00));      //use qcif
        while(0==write_OV7670(0x0C, 0x0C));     //COM3
        while(0==write_OV7670(0x3e, 0x11));     //COM14
        while(0==write_OV7670(0x70, 0x3a));     //Scaling_xsc
        while(0==write_OV7670(0x71, 0x35));     //Scaling_ysc
        while(0==write_OV7670(0x72, 0x11));     //Scaling_dcwctr
        while(0==write_OV7670(0x73, 0xf1));     //Scaling_pclk_div
        while(0==write_OV7670(0xa2, 0x52));     //Scaling_pclk_delay
        while(0==write_OV7670(0x17, 0x39));     //hstrt 39
        while(0==write_OV7670(0x18, 0x03));     //hstop
        while(0==write_OV7670(0x32, 0x80));     //href 80
        while(0==write_OV7670(0x19, 0x03));     //vstart
        while(0==write_OV7670(0x1a, 0x7b));     //vstop
        while(0==write_OV7670(0x03, 0x02));     //vref
        while(0==write_OV7670(0x13, 0xE0));     //com8
        while(0==write_OV7670(0x07, 0x3F));     //AECHh
        while(0==write_OV7670(0x10, 0x00));     //AECH
        while(0==write_OV7670(0x04, 0x00));     //com1
        while(0==write_OV7670(0x0D, 0x40));     //com4
        while(0==write_OV7670(0x14, 0x38));     //com9
        while(0==write_OV7670(0x9f, 0xCA));     //haec1
        while(0==write_OV7670(0xA0, 0x90));     //haec2
        while(0==write_OV7670(0x13, 0xE5));     //com8
        while(0==read_OV7670(0x01,&buff));
        LED1_H;
        }

//konfigurasi antarmuka kamera
void Camera_Init()
{
        RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOA | RCC_AHB1Periph_GPIOB
                                                    | RCC_AHB1Periph_GPIOC |
RCC_AHB1Periph_GPIOE ,ENABLE);
        GPIO_PinAFConfig(GPIOA, GPIO_PinSource4, GPIO_AF_DCMI);         // HSYNC
        GPIO_PinAFConfig(GPIOA, GPIO_PinSource6, GPIO_AF_DCMI);         // PCLK
        GPIO_PinAFConfig(GPIOB, GPIO_PinSource7, GPIO_AF_DCMI);         // VSYNC
        GPIO_PinAFConfig(GPIOC, GPIO_PinSource6, GPIO_AF_DCMI);         //D0
        GPIO_PinAFConfig(GPIOC, GPIO_PinSource7, GPIO_AF_DCMI);         //D1
        GPIO_PinAFConfig(GPIOC, GPIO_PinSource8, GPIO_AF_DCMI);         //D2
        GPIO_PinAFConfig(GPIOC, GPIO_PinSource9, GPIO_AF_DCMI);         //D3
        GPIO_PinAFConfig(GPIOE, GPIO_PinSource4, GPIO_AF_DCMI);         //D4
        GPIO_PinAFConfig(GPIOB, GPIO_PinSource6, GPIO_AF_DCMI);         //D5
        GPIO_PinAFConfig(GPIOE, GPIO_PinSource5, GPIO_AF_DCMI);         //D6
        GPIO_PinAFConfig(GPIOE, GPIO_PinSource6, GPIO_AF_DCMI);         //D7

        GPIO_InitStructure.GPIO_Pin = GPIO_Pin_4 | GPIO_Pin_6 ;
        GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF;
        GPIO_InitStructure.GPIO_Speed = GPIO_Speed_100MHz;
        GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
        GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_UP ;
        GPIO_Init(GPIOA, &GPIO_InitStructure);

        GPIO_InitStructure.GPIO_Pin = GPIO_Pin_7 | GPIO_Pin_6;
        GPIO_Init(GPIOB, &GPIO_InitStructure);

        GPIO_InitStructure.GPIO_Pin = GPIO_Pin_7 | GPIO_Pin_6 | GPIO_Pin_8 | GPIO_Pin_9;
        GPIO_Init(GPIOC, &GPIO_InitStructure);

        GPIO_InitStructure.GPIO_Pin =  GPIO_Pin_4 | GPIO_Pin_5 | GPIO_Pin_6;
        GPIO_Init(GPIOE, &GPIO_InitStructure);
```

```
	//atur antarmuka kamera
	RCC_AHB2PeriphClockCmd(RCC_AHB2Periph_DCMI, ENABLE);
	DCMI_DeInit();
	DCMI_InitStructure.DCMI_CaptureMode = DCMI_CaptureMode_Continuous;
	DCMI_InitStructure.DCMI_SynchroMode = DCMI_SynchroMode_Hardware;
	DCMI_InitStructure.DCMI_PCKPolarity = DCMI_PCKPolarity_Rising;
	DCMI_InitStructure.DCMI_VSPolarity = DCMI_VSPolarity_High;
	DCMI_InitStructure.DCMI_HSPolarity = DCMI_HSPolarity_Low;
	DCMI_InitStructure.DCMI_CaptureRate = DCMI_CaptureRate_All_Frame;
	DCMI_InitStructure.DCMI_ExtendedDataMode = DCMI_ExtendedDataMode_8b;
	DCMI_Init(&DCMI_InitStructure);
	DCMI_Cmd(ENABLE);
	DCMI_CaptureCmd(ENABLE);
	NVIC_InitStructure.NVIC_IRQChannel = DCMI_IRQn;
	NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
	NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;
	NVIC_InitStructure.NVIC_IRQChannelCmd = DISABLE;
	NVIC_Init(&NVIC_InitStructure);

	//atur DMA
	RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_DMA2, ENABLE);
	DMA_DeInit(DMA2_Stream1);
	DMA_StructInit(&DMA_InitStructure);
	DMA_InitStructure.DMA_Channel = DMA_Channel_1;
	DMA_InitStructure.DMA_PeripheralBaseAddr =  (uint32_t)(DCMI_BASE + 0x28);
	DMA_InitStructure.DMA_Memory0BaseAddr =  (uint32_t) frame_buffer;
	DMA_InitStructure.DMA_DIR = DMA_DIR_PeripheralToMemory;
	DMA_InitStructure.DMA_BufferSize =    12672;
	DMA_InitStructure.DMA_PeripheralInc = DMA_PeripheralInc_Disable;
	DMA_InitStructure.DMA_MemoryInc = DMA_MemoryInc_Enable;
	DMA_InitStructure.DMA_PeripheralDataSize = DMA_PeripheralDataSize_Word;
	DMA_InitStructure.DMA_MemoryDataSize = DMA_MemoryDataSize_HalfWord;
	DMA_InitStructure.DMA_Mode = DMA_Mode_Circular;
	DMA_InitStructure.DMA_Priority = DMA_Priority_High;
	DMA_InitStructure.DMA_FIFOMode = DMA_FIFOMode_Enable;
	DMA_InitStructure.DMA_FIFOThreshold = DMA_FIFOThreshold_Full;
	DMA_InitStructure.DMA_MemoryBurst = DMA_MemoryBurst_Single;
	DMA_InitStructure.DMA_PeripheralBurst = DMA_PeripheralBurst_Single;
	DMA_Init(DMA2_Stream1, &DMA_InitStructure);
	DMA_Cmd(DMA2_Stream1, ENABLE);
	DMA_ITConfig(DMA2_Stream1,DMA_IT_TC,ENABLE);
}


void PWM_Configuration() // konfigurasi PWM
{
	RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOA, ENABLE);
	GPIO_InitStructure.GPIO_Pin = GPIO_Pin_2 | GPIO_Pin_3 ;
	GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF;
	GPIO_InitStructure.GPIO_Speed = GPIO_Speed_100MHz;
	GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
	GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL ;
	GPIO_Init(GPIOA, &GPIO_InitStructure);
	GPIO_PinAFConfig(GPIOA, GPIO_PinSource2, GPIO_AF_TIM2);
	GPIO_PinAFConfig(GPIOA, GPIO_PinSource3, GPIO_AF_TIM2);

	RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM2, ENABLE);
	TIM_TimeBaseInitStruct.TIM_Period = 999;
	TIM_TimeBaseInitStruct.TIM_Prescaler = 167;
	TIM_TimeBaseInitStruct.TIM_ClockDivision = 0;
	TIM_TimeBaseInitStruct.TIM_CounterMode = TIM_CounterMode_Up;
	TIM_TimeBaseInit(TIM2, &TIM_TimeBaseInitStruct);

	TIM_OCInitStruct.TIM_OCMode = TIM_OCMode_PWM1;
	TIM_OCInitStruct.TIM_OutputState = TIM_OutputState_Enable;
	TIM_OCInitStruct.TIM_Pulse = 0;
	TIM_OCInitStruct.TIM_OCPolarity = TIM_OCPolarity_High;
	TIM_OC3Init(TIM2, &TIM_OCInitStruct);
```

```
	TIM_OC3PreloadConfig(TIM2, TIM_OCPreload_Enable);

	TIM_OCInitStruct.TIM_OCMode = TIM_OCMode_PWM1;
	TIM_OCInitStruct.TIM_OutputState = TIM_OutputState_Enable;
	TIM_OCInitStruct.TIM_Pulse = 0;
	TIM_OCInitStruct.TIM_OCPolarity = TIM_OCPolarity_High;
	TIM_OC4Init(TIM2, &TIM_OCInitStruct);
	TIM_OC4PreloadConfig(TIM2, TIM_OCPreload_Enable);

	TIM_ARRPreloadConfig(TIM2, ENABLE);
	TIM_BDTRInitTypeDef bdtr;
	bdtr.TIM_AutomaticOutput = TIM_AutomaticOutput_Enable;
	TIM_BDTRConfig(TIM2, &bdtr);
	TIM_Cmd(TIM2, ENABLE);
}

void ftoa(char *pStr,float op)    //konversi float -> array
{
char buff[8];
int po;
uint8_t num=3;

	po=op;
	sprintf(buff,"%d",po);
	num = PutString(pStr, buff);
	while(num--)*pStr++;
	*pStr++ = '.';

	po = (op-po) * 1000;
	if(po<0)po = po * -1;

	if(po<99)*pStr++ = '0';
	else
	{
		sprintf(buff,"%d#",po);
		PutString(pStr, buff);
	}
	if(po<9)
		{
		*pStr++ = '0';
		sprintf(buff,"%d#",po);
		PutString(pStr, buff);
		}
	else
		{
			sprintf(buff,"%d#",po);
			PutString(pStr, buff);
		}

}

/************************************
=====================================
	Sub Fungsi Fuzzy
=====================================
************************************/
//fungsi keanggotaan
float triangle(float value, float x0, float x1, float x2)
{
	float result=0;
	if((value <= x0) || (value >= x2))result=0;
	else if(value == x1)result=1;
	else if((value>x0) && (value<x1))result= ((value-x0)/(x1-x0));
	else result = (((-value)+x2)/(x2-x1));
	return result;
}

float grade(float value, float x0, float x1)
{
	float result=0;
```

```
	if(value <= x0) result = 0;
	else if (value >= x1) result =1;
	else result=((value-x0)/(x1-x0));
	return result;
}

float reverse_grade(float value, float x0, float x1)
{
	float result=0;
	if (value >= x1) result =0;
	else if(value<=x0) result = 1;
	else result = (((-value)+x1)/(x1-x0));
	return result;
}

float trapesium(float value,float x0,float x1,float x2,float x3)
{
	float result=0;
	if((value<=x0) || (value>=x3)) result=0;
	else if((value>=x1)&&(value<=x2))result=1;
	else if((value>x0)&&(value<x1))result=((value-x0)/(x1-x0));
	else result = ((-value+x3)/(x3-x2));
	return result;
}


void fuzzifikasi(float sudut, float jarak_l)
{
	Error_sudut[ER_kiri]	= reverse_grade(sudut,-35,-0);
	Error_sudut[ER_tengah]	= triangle(sudut,-35,0,35);
	Error_sudut[ER_kanan]	= grade(sudut,0,35);
	Error_jarak[ER_kiri]	= reverse_grade(jarak_l,-22,-0);
	Error_jarak[ER_tengah]	= triangle(jarak_l,-22,0,22);
	Error_jarak[ER_kanan]	= grade(jarak_l,0,22);
}

//pengecekan aturan fuzzy yang dipakai
void check_rule()
{
	float temp=0;
	uint8_t x,y;
	for(x=0;x<5;x++)Rule_out[x]=0;
	for(x=0;x<3;x++)
	{
		for(y=0;y<3;y++)
		{
			if((Error_sudut[x]>0) && (Error_jarak[y]>0))
			{

				if(Error_sudut[x]>Error_jarak[y]) temp = Error_jarak[y];
				else temp = Error_sudut[x];
				if (temp > Rule_out[rule[y][x]]) Rule_out[rule[y][x]] = temp;
			}

		}

	}
}

float defuzzifikasi() //defuzzifikasi Centre of Area
{
	float temp,temp1;
	temp  = Rule_out[Vkac] + Rule_out[Vkas] + Rule_out[Vt] + Rule_out[Vkis] + Rule_out[Vkic];
	temp1 = (Rule_out[Vkac]*-95) + (Rule_out[Vkas]*-42) + (Rule_out[Vt]*0)
	    + (Rule_out[Vkis]*42) + (Rule_out[Vkic]*95);
	temp = temp1/temp;
	return temp;
}
```

LAMPIRAN III

DATASHEET