IMPLEMENTASI *FUZZY LOGIC CONTROLLER* PADA SISTEM PENGATURAN SUHU INKUBATOR TELUR

SKRIPSI

Diajukan untuk Memenuhi Persyaratan
Memperoleh Gelar Sarjana Teknik

Disusun oleh :
I GUSTI MADE BAYU INDRA JAYA
NIM. 105060304111002 - 63

KEMENTERIAN RISET, TEKNOLOGI DAN PENDIDIKAN TINGGI
UNIVERSITAS BRAWIJAYA
FAKULTAS TEKNIK
JURUSAN TEKNIK ELEKTRO
MALANG
2015

LEMBAR PERSETUJUAN

IMPLEMENTASI *FUZZY LOGIC CONTROLLER* PADA SISTEM PENGATURAN SUHU INKUBATOR TELUR

SKRIPSI

JURUSAN TEKNIK ELEKTRO

Diajukan untuk Memenuhi Persyaratan
Memperoleh Gelar Sarjana Teknik

Disusun oleh:

I GUSTI MADE BAYU INDRA JAYA
NIM. 105060304111002 – 63

Telah diperiksa dan disetujui oleh :

Pembimbing 1	Pembimbing 2
M. Aziz Muslim, S.T., M.T., Ph.D	**Goegoes Dwi Nusantoro, S.T., M.T.**
NIP. 19741203 200012 1 001	**NIP.19711013 200604 1 001**

PENGANTAR

Puji dan syukur penulis panjatkan kepada Tuhan Yang Maha Esa yang telah memberikan rahmat dan hidayah-Nya, sehingga penulis dapat menyelesaikan skripsi ini.

Penyusunan skripsi ini tidak lepas dari bantuan, bimbingan serta dorongan dari berbagai pihak. Pada kesempatan yang baik ini penulis ingin menyampaikan rasa terima kasih kepada:

- Keluarga penulis, Bapak I Gusti Made Rai Supartama dan Ibu Ni Luh Putu Windari yang telah merawat, memberikan kasih sayang, membesarkan, membimbing, dan mendoakan penulis hingga sampai pada kesuksesan saat ini.
- Bapak M. Aziz Muslim, S.T., M.T., Ph.D selaku Ketua Jurusan Teknik Elektro Universitas Brawijaya.
- Bapak Hadi Suyono, S.T., M.T., Ph.D selaku Sekretaris Jurusan Teknik Elektro Universitas Brawijaya.
- Bapak Ir. Purwanto, M.T. selaku Dosen Pembimbing Akademik penulis atas segala nasihat dan bimbingan yang telah diberikan.
- Bapak Ir. Purwanto, M.T. selaku KKDK Teknik Kontrol Jurusan Teknik Elektro UB.
- Bapak M. Aziz Muslim, S.T., M.T., Ph.D selaku Dosen Pembimbing 1 atas segala ilmu, bimbingan, nasihat, gagasan, ide, saran, motivasi, dan bantuan yang telah diberikan.
- Bapak Goegoes Dwi Nusantoro, S.T., M.T. selaku Dosen Pembimbing 2 atas segala ilmu, bimbingan, nasihat, gagasan, ide, saran, motivasi, dan bantuan yang telah diberikan.
- Bapak Ibu Dosen, karyawan, staff recording dan RBTE atas segala bantuan dan kemudahan.
- Sahabat penulis Fajar, Gozali, Dwika, Dikma, Delief, Azwan dan Gigih yang selalu memberikan dukungan-dukungan yang membangun selama perkuliahan dan pengerjaan skripsi.

- Sahabat seperjuangan skripsi yang selalu membantu dan mendukung skripsi penulis: Fauzan, Rahman, Wawan, Radek, Dilo, Haris, dan teman-teman lain yang juga saling membantu untuk mencapai gelar Sarjana Teknik.
- Rekan Asisten Laboratorium Sistem Kontrol dan teman-teman Konsentrasi Teknik Kontrol: Ika, Dina, Sandi, Hamu, Khairul, Hakiki, Ayu, Rudito, Ade, Neta dkk yang membantu selama perkuliahan dan organisasi di Elektro.
- Seluruh Keluarga Besar Magnet'10, yang selalu memberi pengalaman-pengalaman, saran-saran, dan dukungan-dukungan yang membangun selama perkuliahan dan pengerjaan skripsi.
- Seluruh Keluarga Besar Jurusan Teknik Elektro Universitas Brawijaya, serta semua pihak yang tidak mungkin untuk dicamtumkan namanya satu per satu, terima kasih banyak atas bantuan dan dukungannya.

Penulis menyadari bahwa skripsi ini belum sempurna karena keterbatasan ilmu dan kendala yang terjadi selama pengerjaan skripsi. Oleh karena itu, penulis terbuka terhadap kritik dan saran untuk penyempurnaan tulisan di masa yang akan datang. Penulis juga berharap tulisan ini dapat bermanfaat bagi kita semua.

Malang, Agustus 2015

Penulis

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

ABSTRAK

I Gusti Made Bayu Indra Jaya, Jurusan Teknik Elektro Fakultas Teknik Universitas Brawijaya, Agustus 2015, Implementasi Fuzzy Logic Controller pada Sistem Pengaturan Suhu Ikubator Telur, Dosen Pembimbing: M. Aziz Muslim, S.T., M.T., Ph.D, GOEGOES DN. S.T., M.T.

Beternak adalah salah satu upaya manusia dalam mengembangkan produksi hewan ternak khususnya ayam. Sudah banyak para peternak memiliki alat untuk menetaskan telur ayam, akan tetapi alat tersebut masih ada yang bekerja secara manual. Manual dalam arti masih perlu adanya pembalikan pada telur agar panas yang dihasilkan inkubator penetasan telur merata diseluruh bagian telur. Untuk mempermudah para peternak menetaskan telur ayam tersebut maka telah dirancang dan dibuat suatu sistem pengendali suhu ruang inkubator telur ayam menggunakan mikrokontroler Atmega32 dan *Fuzzy Logic Controller*.

Kata kunci : Inkubator telur pemerataan panas, Mikrokontroler Atmega32, Fuzzy.

BAB I
PENDAHULUAN

1.1 Latar Belakang

Di Indonesia, usaha penetasan telur sudah ada sejak lama, dimulai dari cara tradisional hingga cara modern, salah satunya adalah penggunaan inkubator. Inkubator adalah sebuat alat untuk mengendalikan lingkungan organisme hidup agar tercapai tingkat pertumbuhan yang baik. Karena itu alat ini sering dipakai di peternakan ayam untuk mengeramkan telur-telur yang telah dibuahi. Dengan memanfaatkan fungsi sensor suhu, maka para peternak dapat menjalankan fungsi dari inkubator penetas telur otomatis. Sehingga dapat mempermudah pekerjaan para peternak dan untuk mendapatkan presisi suhu yang seoptimal mungkin.

Kontrol otomatis telah memegang peranan yang sangat penting dalam perkembangan ilmu. Kontrol otomatis telah menjadi bagian yang sangat penting dan terpadu dari proses-proses dalam pabrik dan industri modern.

Dalam Inkubator, sering terjadi pemusatan panas pada suatu titik tertentu yang dapat menyebabkan sebagian telur mengalami panas yang berlebihan. Oleh karena itu, dibutuhkan suatu sistem pengaturan pada Inkubator agar suhu dapat sesuai dengan *setpoint* dan merata.

1.2 Rumusan Masalah

Berdasarkan uraian latar belakang di atas, maka dapat disusun rumusan masalah sebagai berikut:

1. Bagaimana cara mengatur dan menjaga kemerataan suhu di dalam Inkubator
2. Bagaimana cara merancang hardware dan software pada sistem pengaturan suhu Inkubator dengan menggunakan Mikrokontroller ATMega32

1.3 Batasan Masalah

Dalam perancangan skripsi ini permasalahan dibatasi oleh hal-hal sebagai berikut:

1. Ukuran Inkubator disamakan dengan ukuran yang sudah ada di pasaran

2. Sensor suhu yang digunakan adalah LM35
3. Menggunakan Mikrokontroller ATMega32
4. Faktor kelembapan tidak dibahas

1.4 Tujuan

Tujuan dari penelitian ini adalah membuat sebuah alat yang mampu menjaga dan meratakan suhu saat kondisi *setpoint* sebesar 38°C dengan menggunakan *Fuzzy Logic Controller* sebagai pengendalinya.

1.5 Sistematika Penulisan

Sistematika penulisan yang digunakan dalam skripsi ini sebagai berikut:

BAB I Pendahuluan

Membahas tentang latar belakang, rumusan masalah, batasan masalah, tujuan dan sistematika pembahasan.

BAB II Tinjauan Pustaka

Membahas tentang teori-teori yang menunjang perancangan dan pembuatan alat.

BAB III Metode Penelitian

Membahas tentang metode penelitian yang terdiri dari perancangan alat, pembuatan alat, pengujian alat, analisis dan pengambilan kesimpulan.

BAB IV Perancangan dan Pembuatan Alat

Membahas tentang spesifikasi alat, perencanaan blok diagram, prinsip kerja dan pembuatan alat.

BAB V Pengujian dan Analisis

Membahas tentang hasil pengujian sistem dan analisis hasil yang diperoleh dari alat yang sudah dibuat.

BAB VI Kesimpulan dan Saran

Membahas tentang kesimpulan dari keseluruhan hasil yang diperoleh serta saran untuk pengembangan selanjutnya.

BAB II

TINJAUAN PUSTAKA

Dalam bab ini akan dijelaskan mengenai teori-teori yang akan menunjang perancangan pengendalian suhu pada sistem Inkubator menggunakan kontrol logika *fuzzy*.

2.1 Penetasan Telur

Semua unggas bertelur untuk melanjutkan kelangsungan hidup populasinya, karena di dalam telur itu terdapat "calon anak". Untuk membesarkan calon anak itu, induk memberikan cadangan makanan yang terbungkus rapi di sekitar embrio itu berkembang dan inilah yang dikenal dengan TELUR. (Rasyaf 1991)

Pada mulanya tidak ada cara khusus untuk menetaskan telur, orang pada zaman dahulu biasanya hanya memelihara ayam yang jinak dengan manusia dan dipelihara seadanya. Namun pada zaman sekarang orang-orang lebih memilih meggunakan Inkubator buatan dikarenakan efisiensi dan kuantitas yang di hasilkan.

Menurut (Rasyaf 1991), ada beberapa persiapan yang perlu dilakukan sebelum melakukan penetasan telur, diantaranya adalah sebagai berikut :

1. Telur yang akan dibuahkan harus diyakini sebagai telur yang telah dibuahi. Telur yang dapat ditetaskan hanyalah telur yang dihasilkan oleh ayam betina melalui perkawinan dengan ayam jantan.
2. Telur disimpan dengan baik pada kondisi temperatur yang ideal. Sel embrio pada ayam yang baru keluar dari tubuh ayam betina tidak akan segera tumbuh sebelum dierami oleh induknya. Pada temperatur penyimpanan antara 5-15°C, sel embrio yang ada di dalam telur tidak tumbuh dan tidak mati. Pada temperatur seperti inilah idealnya sebutir telur disimpan sebelum ditetaskan.
3. Usahakan telur selalu tetap kering, daerah yang lembab bukan tempat yang baik utuk penyimpana telur karena akan menyebabkan telur sulit menetas. Sebaliknya tempat penyimpanan yang terlalu kering akan menyebabkan telur menetas lebih cepat dari waktu normalnya. Terlalu cepat atau terlalu lambat menetas akan menyebabkan anak ayam mati saat menetas.

4. Uji coba mesin tetas sebelum benar-benar digunakan. Mesin tetas sederhana harus diujicobakan dengan tidak menggunakan telur selama 1 atau 2 hari penuh.
5. Jangan menyimpan telur lebih dari seminggu. Telur yang disimpan lebih dari seminggu sejak ditetaskan memiliki resiko kegagalan penetasan yang tinggi. Telur yang disimpan terlalu lama dapat saja sudah terpengaruh oleh temperatur kelembaban yang tidak ideal dan dapat menyebabkan kematian. Semakin cepat telur masuk ke mesin tetas, semakin baik hasilnya.
6. Waktu ideal yang diperlukan untuk menetaskan telur adalah 20 – 21 hari.
7. Telur yang ditetaskan berasal dari telur ayam betia yang diyakini merupakan ayam betina yang sehat dan terawat dengan baik.

2.2 Inkubator

Inkubator atau mesin penetas telur merupakan salah satu perkakas yang wajib dimilki oleh para peternak ayam. Ikubator telur suhunya musti disesuaikan dengan suhu alami ayam ketika mengeram yakni sekitar 37-38°C. Adapun untuk menjaga suhu dapat digunakan Thermostat yang terhubung dengan pemanas berupa lampu atau elemen pemanas. Setting suhu ditetapkan di Thermostat. Jika suhu Inkubator belum mencapai suhu target maka Thermostat akan menggerakkan relay sehingga pemanas akan menyala. Jika suhuya sudah tercapai maka pemanas akan mati. Demikian cara kerjanya, pemanas nyala – pemanas mati secara bergatian.(Riyanto 2001)

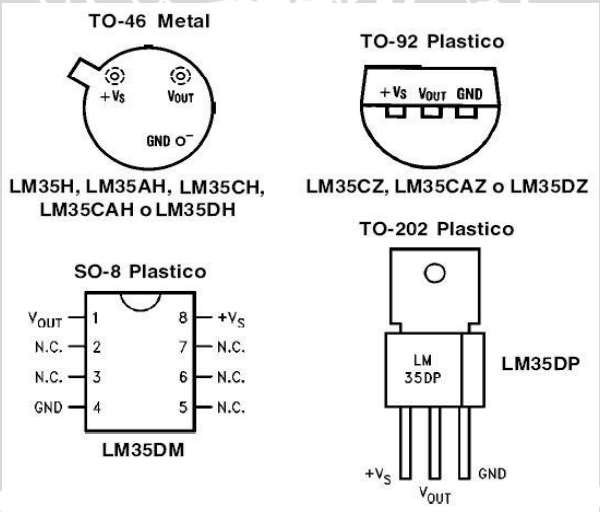
Akan tetapi, penggunaan Thermostat sebagai pengendali suhu Inkubator masih memiliki banyak kelemahan, diantaranya sebelum Inkubator dengan Thermostat bisa digunakan, kita harus mengatur Thermostat terlebih dahulu dengan cara mecoba melakukan pemanasan pada Inkubator yang bisa memakan waktu sampai 2 jam. Itupun seandainya *settingan* Thermostat masih jauh dari nilai *setpoint*, maka kita harus melakukan pengaturan ulang dengan cara memutar baut pada Thermostat dan melakukan pengujian seperti diawal hingga nanti didapatkan nilai *setpoint* yang sama dengan nilai pegendalian dari Thermostat.

Selain itu, sistem pengendali pada Thermostat yang membuat lampu nyala – lampu mati akan menyebabkan lampu yang digunakan tidak berumur panjang. Pengendalian suhu dengan menggunakan lampu nyala-lampu mati akan membuat suhu

di dalam Inkubator berada jauh diatas atau dibawah *setpoint*, yang menyebabkan respon sistem untuk menjaga suhu agar sesuai dengan setpoit menjadi kurang baik. Sehingga dibutuhkan waktu yang lebih lama bagi sistem untuk dapat menjaga kestabilan suhu pada Inkubator untuk dapat mendekati nilai *setpoint*.

2.3 Sensor Suhu LM35

Sensor suhu IC LM 35 merupakan chip IC produksi National *Semiconductor* yang berfungsi untuk mengetahui temperatur suatu objek atau ruangan dalam bentuk besaran elektrik, atau dapat juga di definisikan sebagai komponen elektronika yang berfungsi untuk mengubah perubahan temperatur yang diterima dalam perubahan besaran elektrik. Sensor suhu IC LM35 dapat mengubah perubahan temperatur menjadi perubahan tegangan pada bagian outputnya. Sensor suhu IC LM35 membutuhkan sumber tegangan DC +5 volt dan konsumsi arus DC sebesar 60 µA dalam beroperasi. Bentuk fisik sensor suhu LM 35 merupakan chip IC dengan kemasan yang berfariasi, pada umumnya kemasan sensor suhu LM35 adalah kemasan TO-92 sepeti yang terlihat pada gambar dbawah. (Texas Instrument. "LM35 datasheet")



Gambar 1 Sensor suhu LM35

Dari gambar diatas dapat diketahui bahwa sensor suhu IC LM35 pada dasarnya memiliki 3 pin yang berfungsi sebagai sumber supply tegangan, DC +5 volt sebagai pin output hasil penginderaan dalam bentuk perubahan tegangan DC pada Vout dan pin untuk Ground. Karakteristik Sensor suhu IC LM35 adalah : Memiliki sensitivitas suhu,

dengan faktor skala linier antara tegangan dan suhu 10 mVolt/ºC, sehingga dapat dikalibrasi langsung dalam celcius. Memiliki ketepatan atau akurasi kalibrasi yaitu 0,5ºC pada suhu 25 ºC. Memiliki jangkauan maksimal operasi suhu antara -55 ºC sampai +150 ºC. Bekerja pada tegangan 4 sampai 30 volt. Memiliki arus rendah yaitu kurang dari 60 µA. Memiliki pemanasan sendiri yang rendah (low-heating) yaitu kurang dari 0,1 ºC pada udara diam. Memiliki impedansi keluaran yang rendah yaitu 0,1 W untuk beban 1 mA. Memiliki ketidaklinieran hanya sekitar ± ¼ ºC.(Texas Instrument. "LM35 datasheet")

Sensor suhu IC LM35 memiliki keakuratan tinggi dan mudah dalam perancangan jika dibandingkan dengan sensor suhu yang lain, sensor suhu LM35 juga mempunyai keluaran impedansi yang rendah dan linieritas yang tinggi sehingga dapat dengan mudah dihubungkan dengan rangkaian kontrol khusus serta tidak memerlukan seting tambahan karena output dari sensor suhu LM35 memiliki karakter yang linier dengan perubahan 10mV/°C. Sensor suhu LM35 memiliki jangkauan pengukuran -55ºC hingga +150ºC dengan akurasi ±0.5ºC. Tegangan output sensor suhu IC LM35 dapat diformulasikan sebagai berikut : Vout LM35 = Temperature º x 10 mV Sensor suhu IC. (Texas Instrument. "LM35 datasheet")

2.4 Mikrokontroller ATMEGA32

a. Arsitektur CPU ATMEGA32

Fungsi utama CPU adalah memastikan pengeksekusian instruksi dilakukan dengan benar. Oleh karena itu CPU harus dapat mengakses memori, melakukan kalkulasi, mengontrol peripheral, dan menangani interupsi. Ada 32 buah General Purpose Register yang membantu ALU bekerja. Untuk operasi aritmatika dan logika, operand berasal dari dua buah general register dan hasil operasi ditulis kembali ke register. (Atmega32 datasheet)

Status and Control berfungsi untuk menyimpan instruksi aritmatika yang baru saja dieksekusi. Informasi ini berguna untuk mengubah alur program saat mengeksekusi operasi kondisional. Instruksi di jemput dari flash memory. Setiap byte flash memory memiliki alamat masing-masing. Alamat instruksi yang akan dieksekusi senantiasa disimpan Program Counter. Ketika terjadi interupsi atau pemanggilan rutin biasa, alamat di Program Counter disimpan terlebih dahulu di stack. Alamat interupsi atau rutin kemudian ditulis ke Program Counter, instruksi kemudian dijemput dan dieksekusi.

Ketika CPU telah selesai mengeksekusi rutin interupsi atau rutin biasa, alamat yang ada di stack dibaca dan ditulis kembali ke Program Counter. (Atmega32 datasheet)

b.Program Memori

ATMEGA 32 memiliki 32 KiloByte flash memori untuk menyimpan program. Karena lebar intruksi 16 bit atau 32 bit maka flash memori dibuat berukuran 16K x 16. Artinya ada 16K alamat di flash memori yang bisa dipakai dimulai dari alamat 0 heksa sampai alamat 3FFF heksa dan setiap alamatnya menyimpan 16 bit instruksi.

c. SRAM Data Memori

ATMEGA32 memiliki 2 KiloByte SRAM. Memori ini dipakai untuk menyimpan variabel. Tempat khusus di SRAM yang senantiasa ditunjuk register SP disebut stack. Stack berfungsi untuk menyimpan nilai yang dipush.

d. EEPROM Data Memori

ATMEGA32 memiliki 1024 byte data EEPROM. Data di EEPROM tidak akan hilang walaupun catuan daya ke sistem mati. Parameter sistem yang penting disimpan di EEPROM. Saat sistem pertama kali menyala paramater tersebut dibaca dan system diinisialisasi sesuai dengan nilai parameter tersebut.

e. Interupsi

Sumber interupsi ATMEGA32 ada 21 buah. Saat interupsi diaktifkan dan interupsi terjadi maka CPU menunda instruksi sekarang dan melompat ke alamat rutin interupsi yang terjadi. Setelah selesai mengeksekusi intruksi-instruksi yang ada di alamat rutin interupsi CPU kembali melanjutkan instruksi yang sempat tertunda.

f. I/O Port

ATMEGA32 memiliki 32 buah pin I/O. Melalui pin I/O inilah ATMEGA32 berinteraksi dengan sistem lain. Masing-masing pin I/O dapat dikonfigurasi tanpa mempengaruhi fungsi pin I/O yang lain. Setiap pin I/O memiliki tiga register yakni: DDxn, PORTxn, dan PINxn. Kombinasi nilai DDxn dan PORTxn menentukan arah pin I/O.

g. Clear Timer on Compare Match (CTC)

CTC adalah salah satu mode Timer/Counter1, selain itu ada Normal mode, FastPWM mode, Phase Correct PWM mode. Pada CTC mode maka nilai TCNT1 menjadi nol jika nilai TCNT1 telah sama dengan OCR1A atau ICR1. Jika nilai top ditentukan OCR1A dan interupsi diaktifkan untuk Compare Match A maka saat nilai TCNT1 sama dengan nilai OCR1A interupsi terjadi. CPU melayani interupsi ini dan nilai TCNT1 menjadi nol.

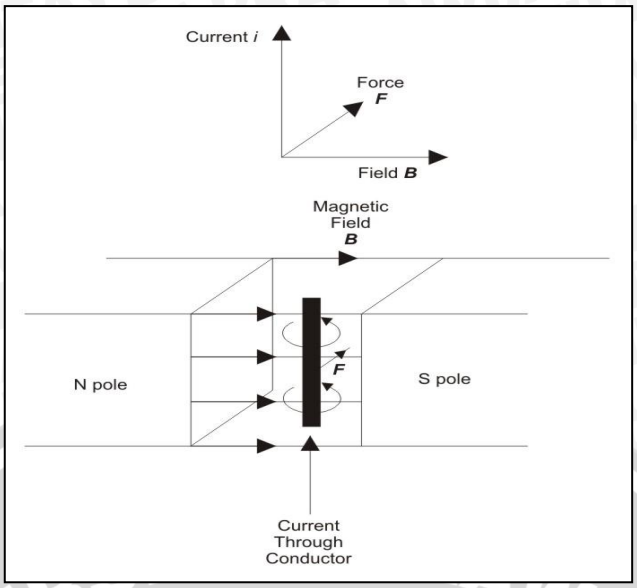
h. USART

Selain untuk general I/O, pin PD1 dan PD0 ATMEGA32 berfungsi untuk mengirim dan menerima bit secara serial. Pengubahan fungsi ini dibuat dengan mengubah nilai beberapa register serial. Untuk menekankan fungsi ini, pin PD1 disebut TxD dan pin PD0 disebut RxD. Nilai UBRR dan clock sistem menentukan laju bit pengirim dan penerima serial. (Atmega32 datasheet)

Gambar 2 Mikrokontroller ATMEGA32

2.5 Motor *Direct Current* (DC)

Motor DC mengubah energi lisrik arus searah (*direct current*) menjadi energi mekanik rotasi. Motor DC masih banyak digunakan dalam berbagai aplikasi kontrol seperti manipulator robot, *disk drive*, peralatan mesin dan aktuator katup **(**Silva dan Clarence, 1989).



Gambar 3 Prinsip Kerja Motor DC **(**Silva dan Clarence, 1989)

Prinsip kerja motor DC ditunjukkan dalam Gambar 3. Sebuah konduktor ditempatkan dalam medan magnet yang stabil. Jika konduktor dialiri arus searah, fluks magnetik yang disebabkan oleh arus akan berputar di sekitar konduktor. Perhatikan bidang melalui konduktor, sejajar dengan arah fluks magnet. Di satu sisi dari bidang ini, fluks arus dan fluks medan bersifat aditif, di sisi lain dua fluks magnetik yang berlawanan satu sama lain mengakibatkan ketidakseimbangan magnetik sehingga gaya F dihasilkan pada konduktor (Silva dan Clarence, 1989). Gaya ini diberikan oleh

$$F = B\, i\, l$$

Dimana:

B = kerapatan fluks medan

i = arus yang melalui konduktor

l = panjang konduktor

Vektor F dapat diartikan sebagai *cross product* dari vektor i dan B.

Jika konduktor bebas untuk bergerak, gaya akan menggerakannya pada suatu kecepatan v dalam arah gaya. Sebagai hasil dari gerakan ini pada medan magnet B, tegangan terinduksi pada konduktor. Ini disebut gaya gerak listrik, atau ggl, dan diberikan oleh

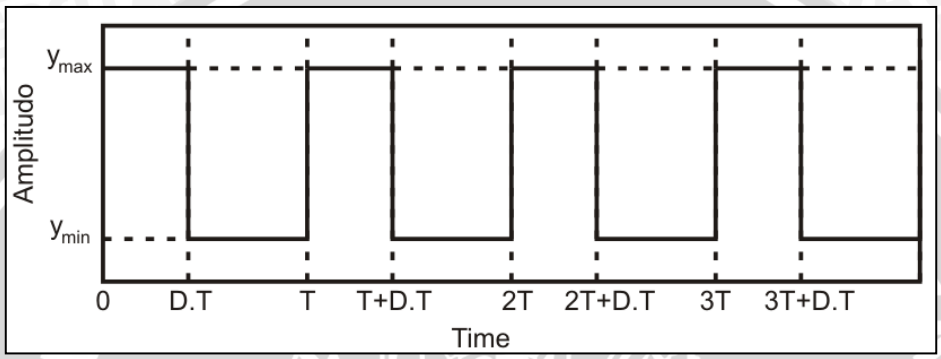
$$v_b = B\, l\, v$$

Berdasarkan hukum Lenz, fluks yang disebabkan oleh ggl balik v_b akan berlawanan dengan fluks arus melalui konduktor sehingga mencoba untuk menghentikan gerakan ini. Ini adalah penyebab redaman listrik pada motor (Silva dan Clarence, 1989).

2.6 *Pulse Width Modulation* (PWM)

Pulse Width Modulation (PWM) digunakan untuk mengatur kecepatan dari motor DC. Kecepatan motor DC tergantung pada besarnya *duty cycle* yang diberikan pada motor DC tersebut (Andrianto, 2013).

PWM menggunakan gelombang kotak dengan *duty cycle* tertentu menghasilkan berbagai nilai rata-rata dari suatu bentuk gelombang (Andrianto, 2013). Jika bentuk gelombang $f(t)$ dangan nilai batas bawah y_{min}, batas atas y_{max} dan *duty cycle* D, seperti dalam Gambar 4



Gambar 4 Gelombang kotak yang memiliki y_{max}, y_{min} dan D (Adrianto, 2013)

Nilai rata-rata dari bentuk gelombang dalam Gambar 2.6 adalah :

$$\overline{y} = \frac{1}{T}\int_0^T f(t)\,dt$$

Jika $f(t)$ adalah gelombang kotak, maka nilai y_{max} adalah dari $0 < t < D \cdot T$ dan nilai y_{min} dari $D \cdot T < t < T$. maka diperoleh :

$$\overline{y} = \frac{1}{T}\int_0^{DT} y_{max}\,dt + \int_{DT}^{T} y_{min}\,dt$$

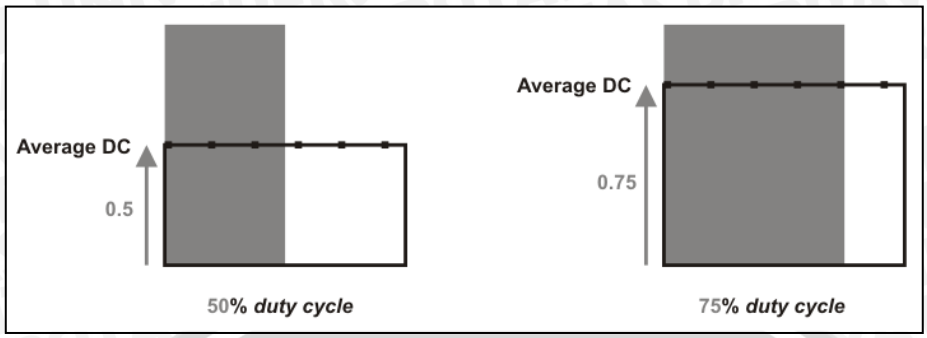
$$\overline{y} = \frac{D \cdot T \cdot y_{max} + T(1-D)y_{min}}{T}$$

$$\overline{y} = D \cdot y_{max} + (1-D)y_{min}$$

Persamaan tersebut dapat disederhanakan dimana $y_{min} = 0$ sehingga bentuk persamaan akhir $\overline{y} = D \cdot y_{max}$. Dari persamaan ini nilai rata-rata dari sinyal $(\overline{y})$ bergantung pada *duty cycle* D.

Duty Cycle menyatakan presentase keadaan logika *high* (pulse) dalam satu periode sinyal. Satu siklus diawali oleh transisi *low to high* dari sinyal dan berakhir pada transisi berikutnya. Selama satu siklus, jika waktu sinyal pada keadaan *high* sama dengan *low* maka dikatakan sinyal mempunyai *duty cycle* 50% (Andrianto, 2013).

Hubungan antara *Duty Cycle* dengan nilai rata-rata tegangan ditunjukkan dalam Gambar 5



Gambar 5 Hubungan antara *Duty Cycle* dengan nilai rata-rata tegangan (Adrianto, 2013)

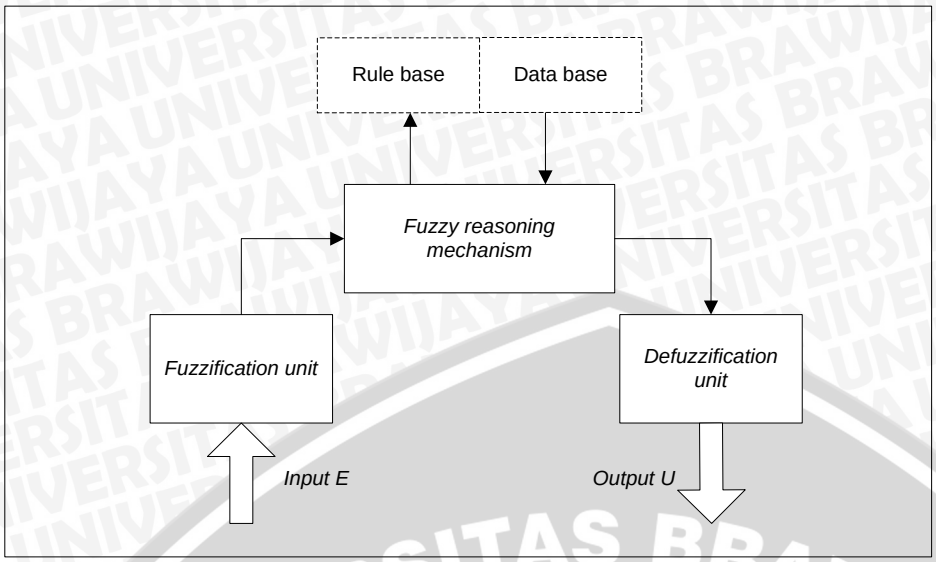
2.7 Kontrol Logika Fuzzy

Kontroler 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.

2.7.1 Struktur Dasar Kontrol Logika *Fuzzy*

Komponen utama penyusun kontrol logika *fuzzy* adalah unit fuzzifikasi, unit penalaran logika *fuzzy*, basis pengetahuan dan unit deffuzifikasi (Yan, Ryan dan Power, 1994). Strukur dasar kontrol logika *fuzzy* ditunjukkan dalam Gambar 6.



Gambar 6 Struktur dasar kontrol logika *fuzzy* (Yan, Ryan dan Power, 1994)

2.7.2 Fungsi Keanggotaan Kontrol Logika Fuzzy

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]. Secara matematika hal ini dinyatakan dengan

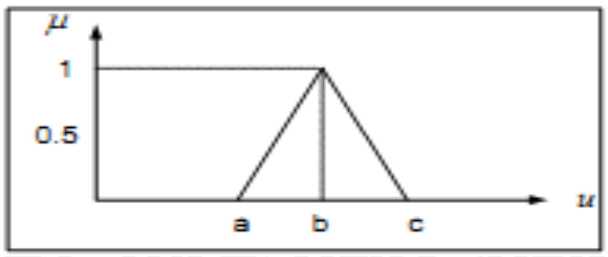
$$\mu_A : U \rightarrow [0,1]$$

a) Fungsi Keanggotaan Bentuk Triangular

Definisi fungsi triangular sebagai berikut

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

Fungsi keanggotaan bentuk triangular ditunjukkan dalam Gambar 7.



Gambar 7 Fungsi keanggotaan bentuk triangular (Yan, Ryan dan Power, 1994)

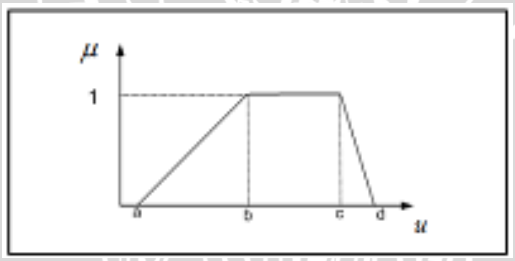
Fungsi keanggotaan bentuk triangular ini digunakan bila diinginkan himpunan *fuzzy* mempunyai nilai proporsional terhadap nol maupun satu.

b) Fungsi Keanggotaan Bentuk Trapesium

Definisi fungsi trapezium sebagai berikut

$$T(u,a,b,c) = \begin{cases} 0 & u < a \\ \dfrac{u-a}{b-a} & a \le u \le b \\ 1 & b \le u \le c \\ \dfrac{d-u}{d-c} & c \le u \le d \\ 0 & u \ge d \end{cases}$$

Fungsi keanggotaan bentuk trapesium ditunjukkan dalam Gambar 8



Gambar 8 Fungsi keanggotaan bentuk trapesium (Yan, Ryan dan Power, 1994)

2.7.3 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, Ryan dan Power, 1994). 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 Yan, J., Ryan, M., dan Power, J. menggunakan :

1. Himpunan *fuzzy* dengan distribusi simetris.
2. Himpunan *fuzzy* yang digunakan berjumlah ganjil, berkaitan erat dengan jumlah kaidah (*rules*). Umumnya digunakan himpunan *fuzzy* dengan jumlah 5 atau 7.
3. Himpunan *fuzzy* diatur agar saling menumpuk.
4. Fungsi keanggotaan yang digunakan bentuk segitiga atau trapesium.

2.7.4 Kaidah Aturan *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 *Multi Input Single Output* (MISO) basis aturan pengendalian *fuzzy* berbentuk seperti berikut

Rule 1 $if\ X\ is\ A_1\ and\ Y\ is\ B_1 then\ Z\ is\ C_1$

Rule 2 $if\ X\ is\ A_2\ and\ Y\ is\ B_2 then\ Z\ is\ C_2$

. . .

. . .

Rule n $if\ X\ is\ A_n\ and\ Y\ is\ B_n then\ Z\ is\ C_n$

Dengan X, Y, Z merupakan variabel linguistik, dimana X dan Y merupakan variabel masukan, dan Z merupakan variabel keluaran sistem. A_n , B_n , dan C_n merupakan nilai linguistik dari X, Y, dan Z (Lee, 1990).

2.7.5 Metode Inferensi *MAX-MIN*

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.

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

$$\mu_{C'} = \bigcup_{1}^{n} \alpha_i \wedge \mu_{C_i}$$

Dengan

$$\alpha_i = \mu_{A_i}(x_0) \wedge \mu_{B_i}(y_0)$$

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

Rule 1 :

$$jika\ x\ adalah\ A_1\ dan\ y\ adalah\ B_1 maka\ z\ adalah\ C_1$$

Rule 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

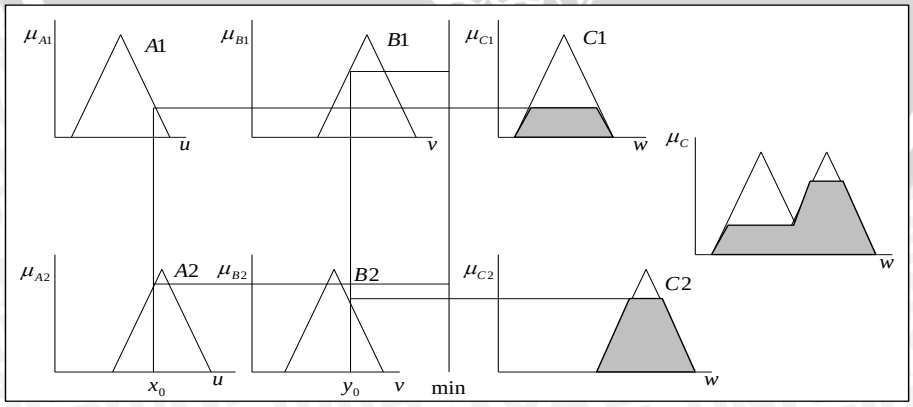
$$\mu_{C'}(w) = \mu_{c_1'} \vee \mu_{c_2'} = \left[\alpha_1 \wedge \mu_{C_1}(w)\right] \vee \left[\alpha_2 \wedge \mu_{C_2}(w)\right]$$

Dimana

$$\alpha_1 = \mu_{A_1}(x_0) \wedge \mu_{B_1}(y_0)$$

$$\alpha_2 = \mu_{A_2}(x_0) \wedge \mu_{B_2}(y_0)$$

Metode ini dideskripsikan dalam Gambar 9



Gambar 9 Inferensi *fuzzy* dengan metode *MAX-MIN* (Yan, Ryan dan Power, 1994)

2.7.6 Metode Defuzzifikasi *Weighted Average*

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

$$y_0 = defuzzifier(y)$$

Dengan

y = aksi kontrol *fuzzy*

y_0 = aksi kontrol *crisp*

$defuzzifier$ = operator defuzzifikasi

Metode *Weighted Average* menurut Ross didefinisikan sebagai berikut :

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

Dengan

U = Keluaran

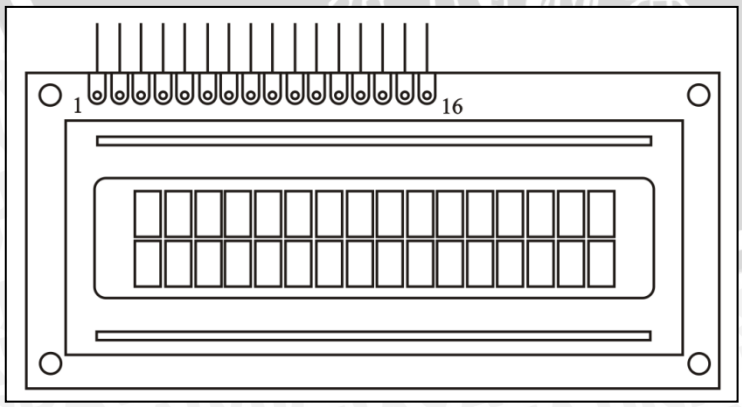
w_i = Bobot nilai benar w_i

u_i = Nilai linguistik pada fungsi keanggotaan keluaran

n = Banyak derajat keanggotaan

2.8 *Liquid Crystal Display* (LCD)

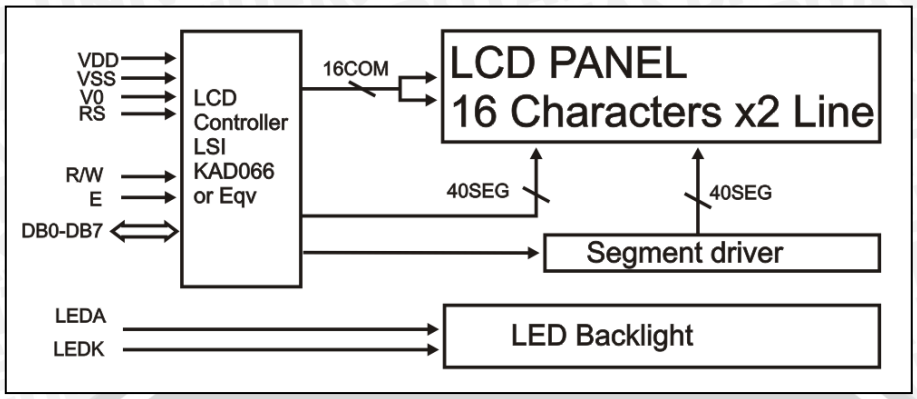
Liquid crystal display (LCD) adalah suatu *display* dari bahan cairan kristal yang pengoperasiannya menggunakan sistem *dot* matriks. LCD banyak digunakan sebagai *display* dari alat-alat elektronika seperti kalkulator, *multitester digital*, jam *digital* dan sebagainya (Andrianto, 2013). Ilustrasi LCD ditunjukkan dalam Gambar 10



Gambar 10 *Liquid crystal display* (LCD) (*datasheet* QC1602A)

Masukan modul LCD ini berupa bus data dan 3 sinyal kontrol yaitu RS, R/W dan E. Pengendali *dot* matriks LCD dilakukan secara internal oleh kontroler yang sudah

terpasang di dalam modul LCD. Diagram blok LCD tipe QC1602A ditunjukkan dalam Gambar 11.



Gambar 11 Blok Diagram *liquid crystal display* (LCD) (*datasheet* QC1602A)

LCD tipe QC1602A memiliki 16 pin koneksi antarmuka dimana setiap pin memiliki fungsi tertentu. Fungsi masing-masing pin ditunjukkan dalam Tabel 1

Tabel 1 Daftar fungsi pin *Liquid Crystal Display* (LCD) QC1602A

pin	**Nama pin**	**Level**	**Fungsi**
1	VSS	0V	*Power ground*
2	VDD	+5V	*Power supply for logic*
3	V0	-	*Contrast adjust*
4	RS	*High/Low*	*High: data* *Low: commmand*
5	R/W	*High/Low*	*High: read* *Low: write*
6	E	*High. High →Low*	*Enable signal*
7	DB0	*High/Low*	*Data bus bit 0*
8	DB1	*High/Low*	*Data bus bit 1*
9	DB2	*High/Low*	*Data bus bit 2*
10	DB3	*High/Low*	*Data bus bit 3*
11	DB4	*High/Low*	*Data bus bit 4*

12	DB5	*High/Low*	*Data bus bit 5*
13	DB6	*High/Low*	*Data bus bit 6*
14	DB7	*High/Low*	*Data bus bit 7*
15	LEDA	+5V	*Power supply for LED Backlight*
16	LEDK	0V	*Power supply for LED Backlight*

Sumber : *datasheet* QC1602A

BAB III
METODE PENELITIAN

Dalam penyelesaian rumusan masalah dan merealisasikan tujuan penelitian ini maka dibutuhkan metode penelitian dalam pelaksanaannya, berikut ini adalah langkah-langkah yang digunakan dalam penelitian:

3.1 Perancangan Sistem

Perancangan sistem dilakukan sebagai langkah awal sebelum terbentuknya suatu sistem, hal ini dimaksudkan agar sistem pengendalian suhu dapat berjalan sesuai deskripsi awal yang telah direncanakan. Perancangan sistem yang dilakukan meliputi :

1. Blok diagram sistem
2. Spesifikasi alat
3. Cara kerja sistem
4. Perancangan perangkat keras meliputi :

a. Perancangan Ikubator
b. Perancangan rangkaian elektrik meliputi :

i. Rangkaian catu daya
ii. Rangkaian penguat
iii. Konfigurasi pin mikrokontroler
iv. Konfigurasi pin *driver* motor
v. Rangkaian *liquid crystal display* (LCD)

5. Perancangan perangkat lunak meliputi:

a. Perancangan rutin sistem secara keseluruhan
b. Perancangan sub rutin pembacaan sensor LM35
c. Perancangan sub rutin kontroler logika *fuzzy*
d. Perancangan kontroler logika *fuzzy*

3.2 Pembuatan Sistem

Pembuatan sistem meliputi pembuatan perangkat keras (*hardware*) dan pembuatan perangkat lunak (*software*) sebagai berikut :

1. Pembuatan perangkat keras (*hardware*)

   Pembuatan perangkat keras (*hardware*) meliputi pembuatan Inkubator, pembuatan rangkaian catu daya, pembuatan rangkaian driver dan pembuatan rangkaian LCD.

2. Pembuatan perangkat lunak (*software*)

   Pembuatan perangkat lunak (*software*) yaitu pembuatan program dengan *software* Code Vision berdasarkan *flowchart* yang telah dirancang.

3.3 Pengujian dan Analisa Data

Pengujian sistem pengendalian suhu bertujuan untuk menguji apakah sistem ini bekerja sesuai dengan perancangan. Pengujian sistem ini meliputi pengujian setiap blok rangkaian dan pengujian keseluruhan sistem.

1. Pengujian setiap blok rangkaian, meliputi :

a. Pengujian catu daya
b. Pengujian sensor suhu
c. Penguian *driver motor*
d. Pengujian *liquid crystal display* (LCD)

2. Pengujian keseluruhan sistem

   Pengujian keseluruhan sistem dilakukan dengan menyambungkan seluruh blok perangkat keras maupun perangkat lunak yang telah dibuat. Sistem dikatakan berhasil jika sistem telah bekerja sesuai dengan spesifikasi rancangan.

3.4 Pengambilan Kesimpulan

Kesimpulan diperoleh berdasarkan data pengujian sistem secara keseluruhan. Jika hasil yang diperoleh sesuai dengan yang direncanakan, maka sistem kendali tersebut telah memenuhi harapan dan dapat dikembangkan untuk penelitian selanjutnya.

BAB IV
PERANCANGAN DAN PEMBUATAN ALAT

Pada bab ini akan dibahas perancangan dan pembuatan alat. Pembahasan akan dilakukan pada setiap blok rangkaian, cara kerja masing-masing blok rangkaian, perhitungan dan juga fungsi masing-masing blok rangkaian.

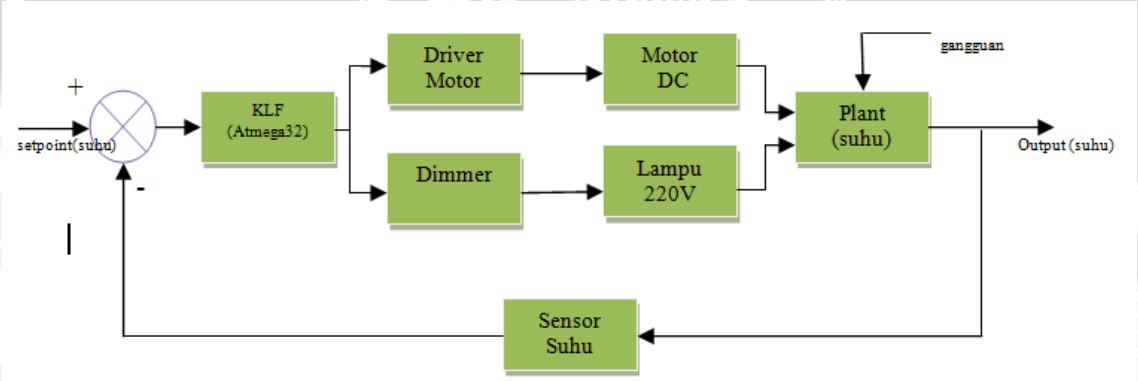
Secara garis besar terdapat dua bagian perangkat yang ada yaitu :

1. Perancangan perangkat keras (*hardware*)
2. Perancangan perangkat lunak (*software*)

Pada perancangan perangkat keras meliputi perangkat elektrik maupun mekanik yang digunakan pada sistem pengendalian suhu Inkubator Telur. Sedangkan pada perancangan perangkat lunak meliputi *flowchart* dan *software*.

4.1 Diagram Blok Sistem

Diagram blok sistem yang dirancang ditunjukkan dalam Gambar 12.



Gambar 12 Blok Diagram Sistem (Perancangan)

Keterangan :

1. *Setpoint* sistem berupa suhu sebesar 38°C.
2. Kontroler yang digunakan adalah kontroler logika *fuzzy* yang diprogram ke dalam Mikrokontroller Atmega32.
3. *Input* kontroler logika *fuzzy* berupa sinyal *error* dan *delta error*. *Output* kontroler logika *fuzzy* berupa sinyal kontrol untuk driver motor dan dimmer.
4. Aktuator yang digunakan adalah kipas motor DC sebagai perata panas di dalam Inkubator dan Lampu 220V sebagai pemanas di dalam Plant.

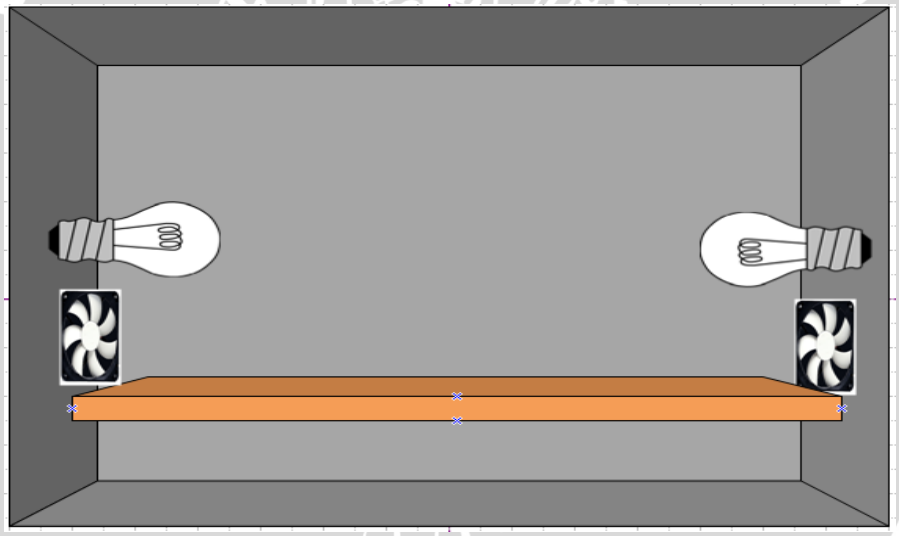
5. Sebagai umpan balik digunakan sensor LM35 untuk mengukur besar nilai suhu(°C) di dalam Inkubator.

4.2 Spesifikasi Alat

Spesifikasi komponen-komponen sistem pengendalian suhu Inkubator menggunakan Atmega32 adalah sebagai berikut :

1. Kotak yang digunakan terbuat dari kayu dengan ukuran 40 cm x 30 cm x 30 cm.
2. Adaptor yang digunakan memiliki sumber tegangan sebesar 12 V dan arus 1200 mA.
3. Perangkat kontrol yang digunakan adalah ATMEGA 32.
4. Driver Motor yang digunakan yaitu IC L298.
5. Sensor suhu yang digunakan yaitu sensor suhu LM35.
6. Aktuator yang digunakan adalah kipas motor DC dengan tegangan maksimal 12 V.

Skema perancangan sistem pengendalian suhu pada Inkubator menggunakan KLF ditunjukkan dalam Gambar 13



Gambar 13 Skema perancangan sistem(tampak depan)

4.3 Prinsip Kerja Sistem

Cara kerja sistem pengendalian suhu pada prosesor menggunakan Kontrol Logika *Fuzzy* sebagai berikut :

1. Sistem diberi catu daya. Catu daya sebesar 5 V digunakan untuk mencatu rangkaian LCD, Mikrokontroler ATMEGA 32 dan *driver* motor . Dan catu daya 12 V digunakan untuk mencatu kipas angin motor DC.

2. Pengukuran suhu pada Inkubator menggunakan sensor suhu.
3. Sensor suhu dihubungkan ATMEGA 32 dengan rentang 0-5 V.
4. Keluaran ATMEGA 32 berupa sinyal *pulse width modulation* (PWM) diberikan ke *driver* motor. *Driver* berfungsi untuk menguatkan sinyal *Pulse Width Modulation* (PWM) ATMEGA 32 dari tegangan sebesar 0-5 V menjadi 0-12 V.
5. Sistem dari alat ini bekerja sesuai dengan penilaian dari sensor suhu yang mengukur besaran suhu di dalam Inkubator. Setelah suhu didapatkan dengan LCD, maka kontroler akan memberikan perintah kepada kipas angin motor DC dan Lampu 220 VAC untuk bekerja.
6. Sensor suhu LM35 akan mengukur secara berkala, jika suhu output melebihi *setpoint* maka ATMEGA 32 akan memerintahkan pada Motor DC untuk berputar lebih cepat. Dan intensitas cahaya pada Lampu akan direndahkan.Dengan demikian suhu prosesor pada laptop diharapkan akan berada pada Suhu s*etpoint*.

4.4 Perancangan Perangkat Keras

4.4.1 INKUBATOR

Inkubator menggunakan *box* yang terbuat dari kayu dengan ukuran 40 cm x 30 cm x 30 cm. Kotak ini di modifikasi sedemikian rupa sehingga dapat memberikan tempat bagi kipas angin motor DC.

4.4.2 Rangkaian *Driver* Motor

Driver motor yang digunakan adalah IC L298N. *Driver* motor ini digunakan untuk mengendalikan motor DC yaitu kipas angin motor DC. Mode pengendalian kecepatan motor yang digunakan adalah mode sinyal *Pulse Width Modulation* (PWM). Catu daya motor yang digunakan adalah catu daya eksternal 12 V.

4.4.3 Konfigurasi pin *driver motor* ini ditunjukkan dalam Tabel 2

Tabel 2 Konfigurasi pin *driver* motor pada Atmega 32

Pin	Pin pada Atmega 32	Fungsi
5	9 (PWM)	Kontrol arah motor 1
7	10 (PWM)	Kontrol arah motor 1

6	8 (PWM)	Kontrol PWM motor 1
9	5 V	*Supply* tegangan 5 V

4.4.4 Rangkaian Mikrokontroler

Sistem pengendalian suhu ini menggunakan MikroKontroller ATMEGA 32 sebagai pusat pengolah utama dalam proses pengendalian. Konfigurasi pin dari ATMega 32 ditunjukkan dalam Tabel 3

Tabel 3 Konfigurasi pin Atmega 32

No	Pin	Fungsi
1	A0	Sensor suhu LM35
2	PWM 8	*Enable* (*driver* motor)
3	PWM 9	*Input* 1 (*driver* motor)
4	PWM 10	*Input* 2 (*driver* motor)
5	Vin	*Input* 5 V (*adaptor*)
6	GND	VSS, R/W, LEDK (*adaptor* & LCD)
7	D22	DB7 (LCD)
8	D24	DB6 (LCD)
9	D26	DB5 (LCD)
10	D28	DB4 (LCD)
11	D30	E (LCD)
12	D32	RS (LCD)

4.4.5 Konfigurasi pin Mikrokontroler

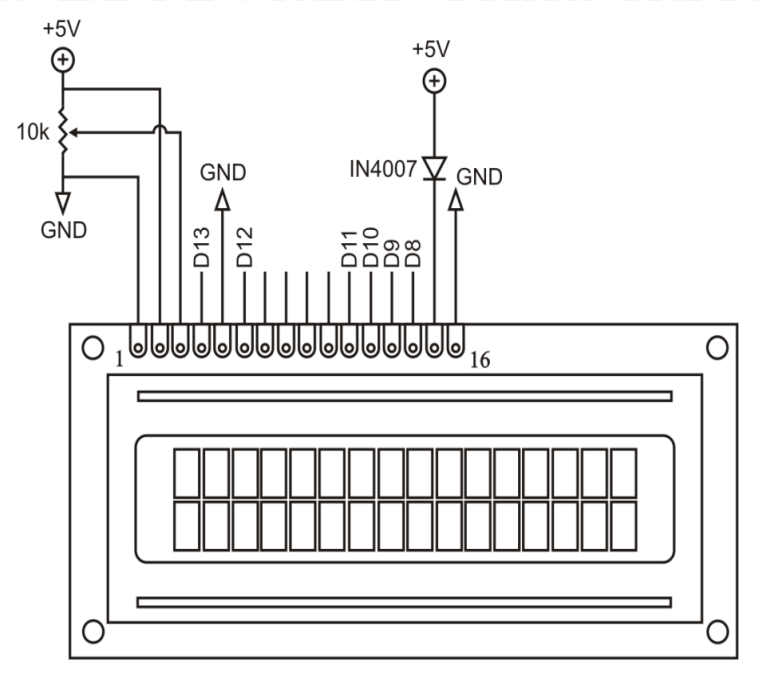
Sistem pengendalian suhu ini menggunakan Atmega32 sebagai pusat pengolah utama dalam proses pengendalian. Konfigurasi *input*/*output* dari Atmega32 ini ditunjukkan dalam Tabel 3.

Tabel 4 Konfigurasi input/output Atmega32

No	**Pin**	**Fungsi**
1	A0	Sensor 1
2	D4	M1
3	D5	E1
4	D6	E2
5	D7	M2
6	D8	DB7
7	D9	DB6
8	D10	DB5
9	D11	DB4
10	D12	E
11	D13	RS
12	5V	LEDA
13	GND	VSS, R/W, LEDK

4.4.6 Rangkaian *Liquid Crystal Display* (LCD)

Liquid Crystal Display (LCD) tipe QC1602A digunakan untuk menampilkan suhu Inkubator telur. Rangkaian *liquid crystal display* (LCD) ditunjukkan dalam Gambar 14.



Gambar 14 Rangkaian *liquid crystal display* (LCD) (Perancangan)

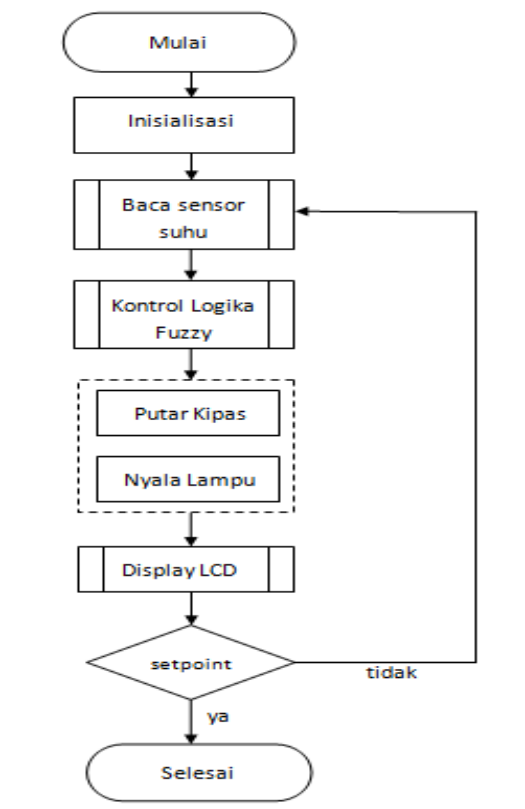
Keterangan :

1. Pin 1 atau pin VSS dihubungkan dengan *ground*.
2. Pin 2 atau pin VDD dihubungkan dengan +5V.
3. Pin 3 atau pin V0 dihubungkan dengan *variable resistor*. Besar nilai resistansi *variable resistor* diatur untuk mengatur *contrast*.
4. Pin 4 atau pin RS dihubungkan dengan pin *Digital* 13 Atmega32
5. Pin 5 atau pin R/W dihubungkan dengan *ground*.
6. Pin 6 atau pin E dihubungkan dengan pin *Digital* 12 Atmega32.
7. Pin 11 atau pin DB4 dihubungkan dengan pin *Digital* 11 Atmega32.
8. Pin 12 atau pin DB5 dihubungkan dengan pin *Digital* 12 Atmega32.
9. Pin 13 atau pin DB6 dihubungkan dengan pin *Digital* 13 Atmega32.
10. Pin 14 atau pin DB7 dihubungkan dengan pin *Digital* 14 Atmega32.
11. Pin 15 atau pin LEDA dihubungkan dengan diode 1N4007.
12. Pin 16 atau pin LEDK dihubungkan dengan *ground*.

4.5 Perancangan Perangkat Lunak

4.5.1 *Flowchart* Sistem Secara Keseluruhan

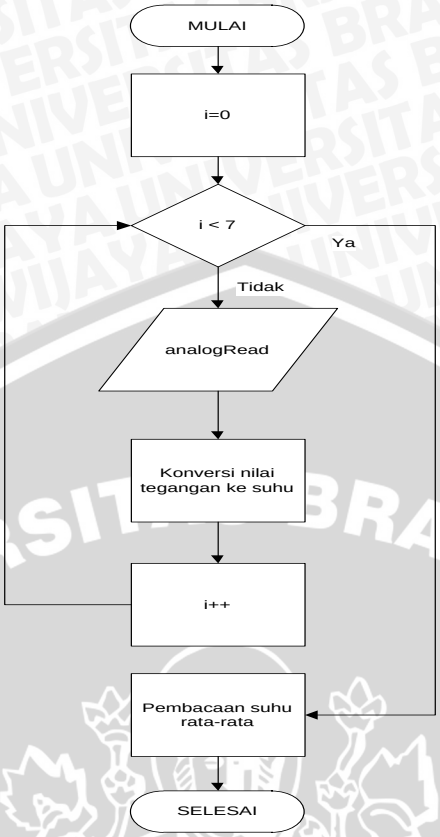
Flowchart keseluruhan sistem ditunjukkan dalam Gambar 15.



Gambar 15 *Flowchart* keseluruhan sistem (Perancangan)

4.5.2 *Flowchart* Pembacaan Sensor Suhu

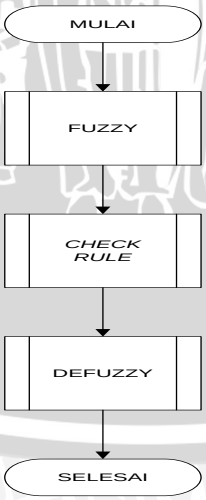
Flowchart pembacaan sensor suhu ditunjukan pada Gambar 16



Gambar 16 Flowchart pembacaan sensor suhu (perancangan)

4.5.3 *Flowchart* Kontroler Logika *Fuzzy*

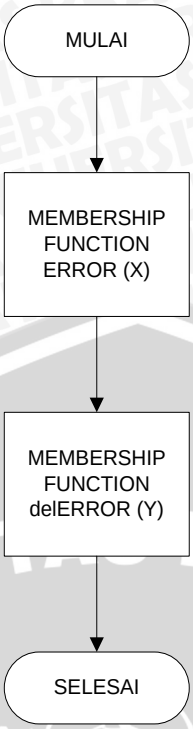
Flowchart kontroler logika *fuzzy* ditunjukkan dalam Gambar 17



Gambar 17 *Flowchart* kontroler logika *fuzzy* (Perancangan)

4.5.4 *Flowchart* sub rutin *fuzzy*

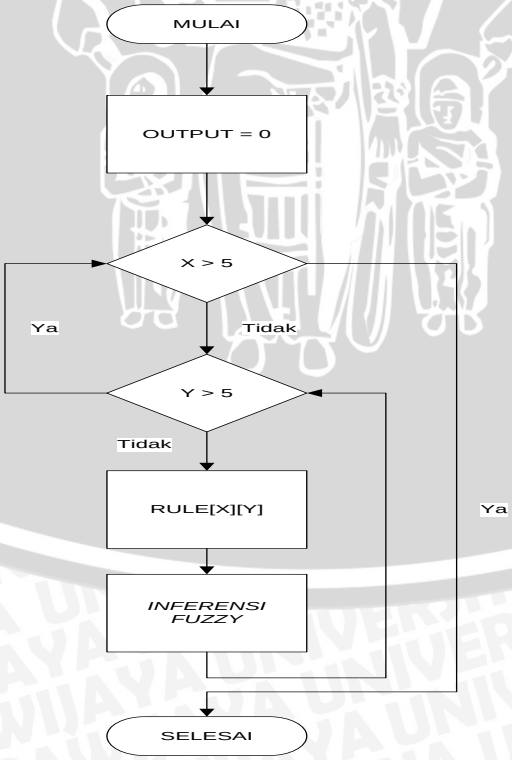
Flowchart sub rutin *fuzzy* ditunjukkan dalam Gambar 18



Gambar 18 *Flowchart* subrutin *fuzzy* (Perancangan)

4.5.5 Flowchart Sub Rutin *Check Rule*

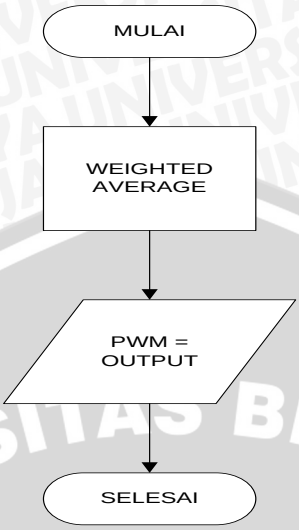
Flowchart sub rutin *check rule* ditunjukkan dalam Gambar 19



Gambar 19 *Flowchart* subrutin *check rule* (Perancangan)

4.5.6 *Flowchart* sub rutin *defuzzy*

Flowchart sub rutin *defuzzy* ditunjukkan dalam Gambar 20



Gambar 20 ***Flowchart*** subrutin *defuzzy* (Perancangan)

4.5.7 Perancangan Kontroler Logika *Fuzzy*

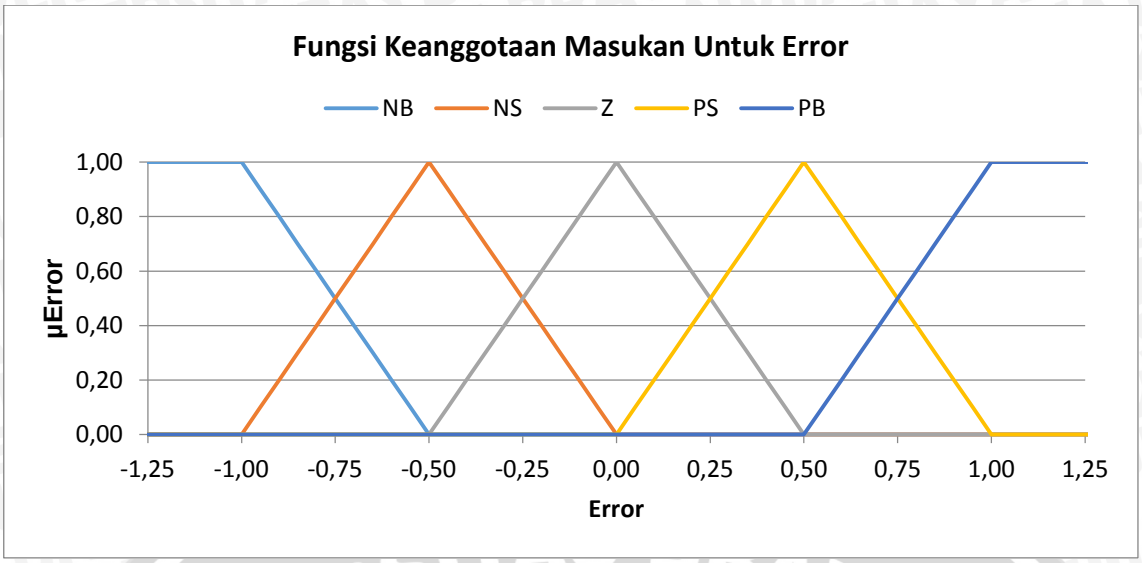
Pada perancangan kontroler logika *fuzzy* dilakukan langkah-langkah sebagai berikut :

1. Menentukan fungsi keanggotan masukan dan keluaran.

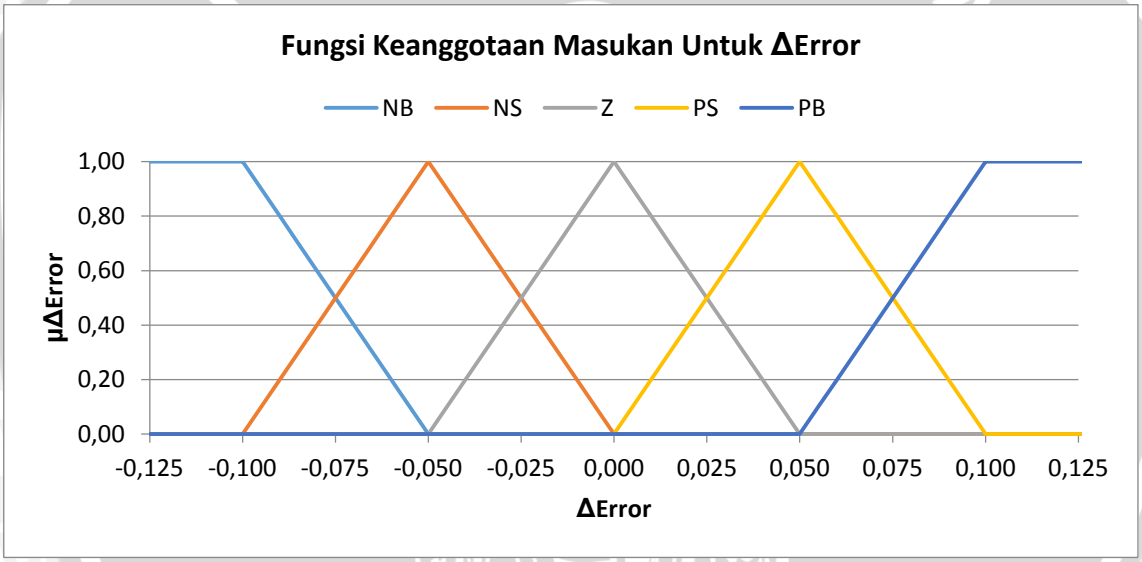
a. Fungsi keanggotaan masukan.

Variabel yang digunakan untuk masukan adalah *error* dan *delta error*. *Error* adalah nilai setpoint dikurangi nilai sebenarnya, sedangkan *delta error* adalah nilai *error* sekarang dikurangi *error* sebelumnya. *Fungsi* keanggotaan dari *error* dan *delta error* terdiri atas 5 label, yaitu *Negative Big* (NB), *Negative Small* (NS), *Zero* (Z), *Positive Small* (PS) dan *Positive Big* (PB). Data masukan dari fungsi keanggotaan diperoleh dari nilai pengukuran suhu oleh sensor LM35.

Fungsi keanggotaan berjumlah 5 dengan tujuan agar aturan *fuzzy* yang dihasilkan semakin banyak sehingga semakin banyak pula kondisi dan aksi kontrol yang akan dipilih. Fungsi keanggotaan masukan ditunjukkan dalam Gambar 21



(a)



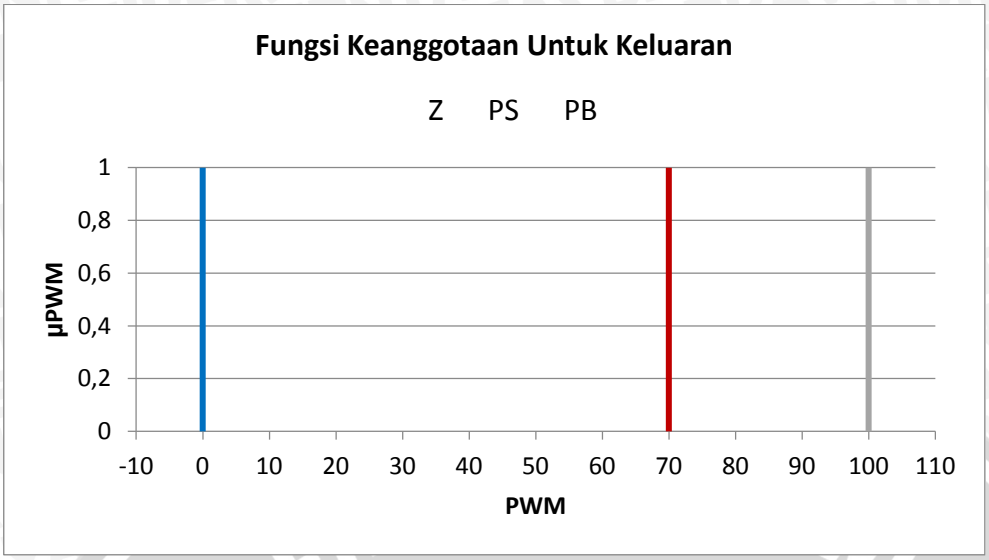
(b)

Gambar 21 (a) Fungsi keanggotaan *error dan* **(b)** Fungsi keanggotaan *delta error* (Perancangan)

b. Fungsi keanggotaan keluaran.

Fungsi keanggotaan keluaran motor DC merupakan representasi kecepatan putaran motor DC dan besarnya nilai PWM yang dikeluarkan oleh Atmega32 berupa hasil perhitungan dengan metode defuzzifikasi *weighted average* yang telah dibulatkan dalam program.

Fungsi keanggotaan motor DC berjumlah 3 yaitu *Zero* (Z), *Positive Small* (PS) dan *Positive Big* (PB) ditunjukkan dalam Gambar 22



Gambar 22 Fungsi keanggotaan motor DC (Perancangan)

2. Menentukan aturan *fuzzy*.

Aturan *fuzzy* (*rule*) digunakan sebagai penentu keluaran dari fuzzifikasi yang akan diolah dalam proses defuzzifikasi. Jumlah aturan *fuzzy* yang digunakan sebanyak 25. Aturan *fuzzy* diperoleh dari 5 fungsi keanggotaan masukan *error* dan *delta error*. Aturan *fuzzy* yang digunakan ditunjukan dalam Tabel 5.

Tabel 5 Aturan *Fuzzy* pada Sistem

		Error				
		NB	**NS**	**Z**	**PS**	**PB**
Delta Error	**NB**	S	S	S	M	M
	NS	S	S	S	M	MF
	Z	S	S	MS	MF	MF
	PS	S	MS	MS	MF	F
	PB	MS	MS	M	F	F

3. Menentukan metode inferensi *fuzzy*.

Metode inferensi *fuzzy* yang digunakan adalah metode inferensi MAX-MIN.

4. Menentukan metode defuzzifikasi.

Defuzzifikasi merupakan proses untuk mengubah keluaran *fuzzy* menjadi keluaran *crisp*. Hasil defuzzifikasi digunakan untuk mengatur kecepatan motor kipas. Metode defuzzifikasi yang digunakan adalah *weighted average*.

BAB V
PENGUJIAN DAN ANALISIS

Pengujian dan analisis sistem dilakukan untuk mengetahui apakah sistem telah bekerja sesuai dengan perancangan. Pengujian yang dilakukan meliputi pengujian sistem pengendalian suhu.

Pengujian sistem pengendalian suhu yang dilakukan meliputi pengujian setiap blok sistem kemudian dilakukan pengujian secara keseluruhan. Pengujian yang dilakukan sebagai berikut :

1. Pengujian sensor LM35
2. Pengujian motor kipas:
3. Pengujian *liquid crystal display* (LCD)
4. Pengujian keseluruhan sistem

5.1 Pengujian Sensor LM35

Pengujian sensor LM35 bertujuan untuk mengetahui apakah pembacaan sensor LM35 sudah sesuai dengan besar suhu yang sebenarnya. Pengujian dilakukan dengan meletakkan sensor di dekat lampu pijar 220V.Seperti yang terlihat pada Tabel 6

Tabel 6 Pengujian sensor LM35

Suhu	Tegangan Keluaran
35°C	0.35
40°C	0.40
45°C	0.45
50°C	0.51
55°C	0.55
60°C	0.65
65°C	0.71
70°C	0.76

Berdasarkan Tabel diatas dapat dilihat bahwa kenaikan suhu berbanding lurus dengan kenaikan tegangan keluaran. Sehigga sensor dapat dikatakan bekerja dengan baik. Akan tetapi sensor LM35 perlu melakukan kalibrasi agar data yang dibaca sama dengan besar data yang dibaca oleh thermometer ruangan. Berikut (tabel 7) kalibrasi dari sensor LM35.

Tabel 7 Kalibrasi Sensor LM35

t(s)	Suhu LM35 °C	Suhu thermometer°C
10	28,63	26,4
20	28,33	26,3
30	28,76	25,7
40	28,43	25,2
50	28,66	26,4
60	28,56	26,4
70	28,53	26,3
80	28,76	26,7
90	27,76	26,6
100	28,63	26,7
Rata-rata	28,5	26,2

Jadi dapat diketahui perbedaan pembacaan pada sensor lm35 dan thermometer ruangan adalah sebesar 2,3°C.

5.2 Pengujian *Driver Motor*

Pengujian *driver motor* bertujuan untuk mengetahui hubungan antara tegangan terhadap *pulse width modulation* (PWM). Pengujian dilakukan dengan menghubungkan Rangkaian Driver ke dalam Atmega32. Catu daya + 12 V dihubungkan ke Atmega32 yang berfungsi sebagai catu daya motor pompa DC. Pada Atmega32 dibuat program yang berisi perintah untuk memberikan sinyal PWM dari 0 sampai 255. Hasil pengujian *driver motor* ditunjukkan dalam Tabel 8.

Tabel 8 Pengujian Drive Motor

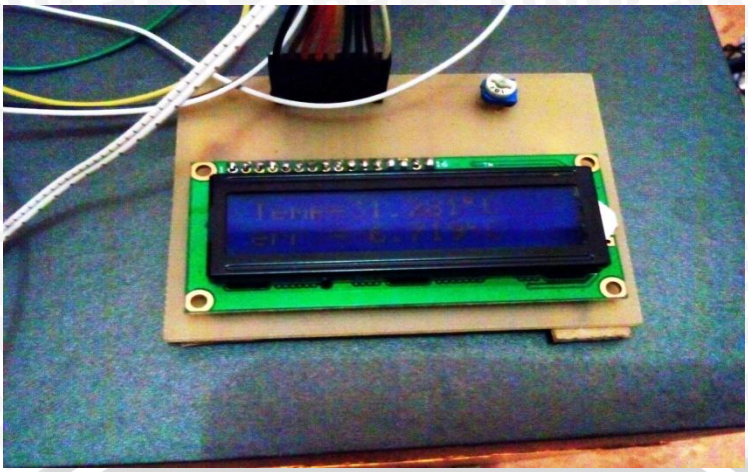
Sudut	*Duty Cycle* (%)	PWM
10	3.2	8.3
20	3.8	9.6
30	4.3	10.9
40	4.8	12.2
50	5.3	13.5
60	5.8	14.9
70	6.3	16.1
80	6.8	17.4
90	7.4	18.7
100	7.9	20.0
110	8.4	21.3
120	8.9	22.7
130	9.4	24.0
140	9.9	25.3
150	10.4	26.5
160	10.9	27.9
170	11.5	29.2
180	12	30.6

Pada pengujian *driver motor* yang telah dilakukan diperoleh hasil seperti dalam Tabel 8 diketahui bahwa *driver* sudah bekerja dengan baik, terlihat bahwa semakin naik sudut sesuai dengan kenaikan *pulse width modulation* (PWM)

.

5.3 Pengujian *Liquid Crystal Display* (LCD)

Pengujian *liquid crystal display* (LCD) dilakukan untuk mengetahui apakah LCD dapat menampilkan data yang diberikan oleh Atmega32 melalui program. Pengujian dilakukan dengan menghubungkan LCD dengan Atmega32. Pada mikrokontroler dibuat program yang berisi perintah untuk menampilkan karakter tertentu pada LCD. Program berisi perintah untuk menampilkan karakter “Tegangan” dan “suhu” pada baris pertama dan menampilkan karakter hasil pembacaan suhu oleh sensor lm35 pada baris kedua tampilan LCD. Pengujian LCD ditunjukkan dalam Gambar 23.



Gambar 23 Pengujian LCD

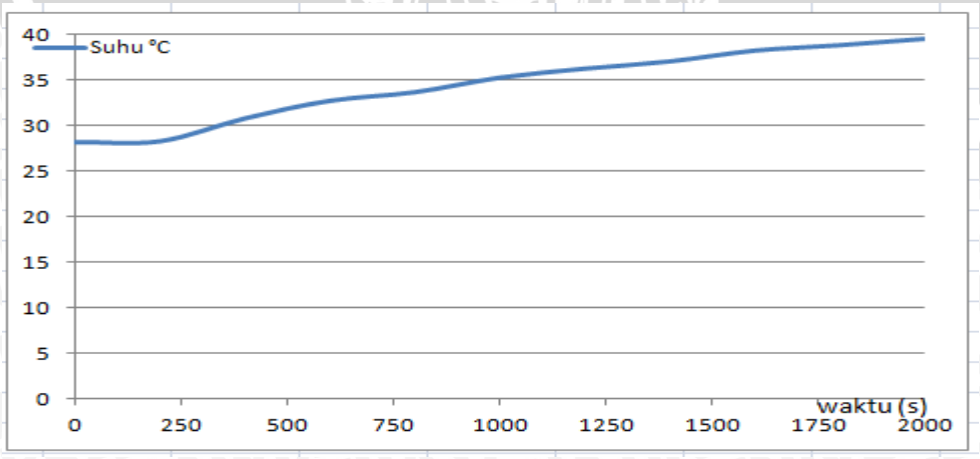
Pada pengujian *liquid crystal display* (LCD) yang telah dilakukan diperoleh hasil yaitu LCD mampu menampilkan karakter yang ada pada program.

5.4 Pengujian Keseluruhan Sistem

Pengujian sistem secara keseluruhan ini dilakukan untuk mengetahui kinerja perangkat keras dan perangkat lunak serta mengetahui respon sistem secara keseluruhan tanpa menggunakan kontroler maupun dengan menggunakan kontrol logika *fuzzy*.

5.4.1 Pengujian Tanpa Kontroller

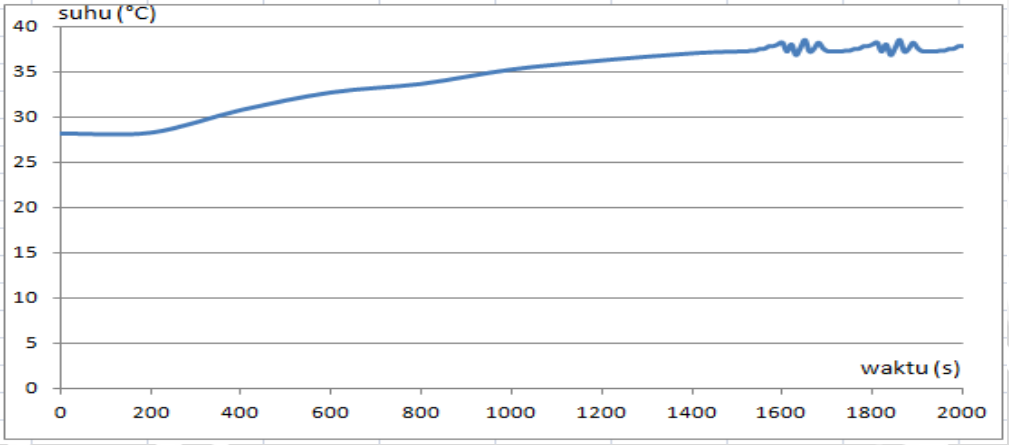
Pengujian tanpa kontroler ini dilakukan untuk mengetahui respon sistem secara keseluruhan tanpa melakukan pengendalian suhu. Hasil pengujian tanpa kontroler ditunjukkan dalam Gambar 24.



Gambar 24 Pengujian Tanpa Kontroller

5.4.2 Pengujian Tanpa Gangguan

Pengujian ini dilakukan untuk mengetahui kinerja sistem pengaturan suhu ketika tidak mengalami gangguan. *Setpoint* yang digunakan sebesar 38 °C. Seperti yang ditunjukan dalam gambar 25.



Gambar 25 Pengujian Tanpa Gangguan

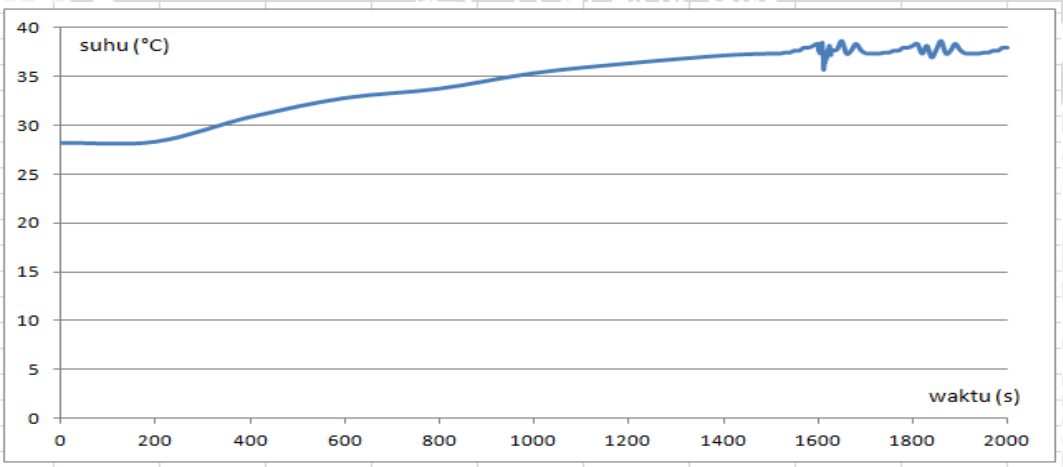
$$\% \, Ess = \frac{\sum_{i=1}^{n}\left(\frac{|data\ ke(n) - setpoint|}{setpoint}\right)}{banyak\ data} \text{ x } 100\%$$

$$= 0{,}023\text{x } 100\%$$

$$= 2{,}3\ \%$$

5.4.3 Pengujian Dengan Gangguan

Pengujian dengan gangguan bertujuan untuk mengetahui bagaimana kinerja sistem secara keseluruhan dan mengamati respon sistem kontrol logika *fuzzy* ketika mendapat gangguan perubahan Suhu. Seperti yang ditunjukkan pada gambar 26.



Gambar 26 Pengujian Dengan Gangguan

Dari grafik diatas dapat diketahui jika waktu yang dibutuhkan sistem untuk melakukan recovery adalah 12 s.

$$\%\ Ess = \frac{\sum_{i=1}^{n}\left(\frac{|data\ ke(n)-setpoint|}{setpoint}\right)}{banyak\ data} \text{ x } 100\%$$

$$= 0{,}0315 \text{x } 100\%$$

$$= 3{,}15\ \%$$

5.5 Perhitungan Manual dengan Logika Fuzzy

Perhitungan manual dibutuhkan sebagai bukti dari metode pada sistem. Data yang digunakan pada perhitungan ini berupa sampling dari data keseluruhan. Berikut hasil peritungan berdasarkan data Tabel 9.

Tabel 9 Data error, Setpoint dan Suhu terukur

Data ke -	Setpoint (Suhu °C)	Present Value (Suhu °C)	error	Delta error
28	38	37,33	0,67	-0,19
29	38	37,63	0,37	-0,3
30	38	37,67	0,33	-0,04

Perhitungan manual pada sistem

$$Error(t) = SP(t) - PV(t)$$

$$Error(28) = SP(28) - PV(28)$$

$$Error(28) = 38 - 37{,}33$$

$$Error(28) = 0{,}67$$

$$Error(t) = SP(t) - PV(t)$$

$$Error(29) = SP(29) - PV(29)$$

$$Error(29) = 38 - 37{,}63$$

$$Error(29) = 0{,}37$$

$$Error(t) = SP(t) - PV(t)$$

$$Error(30) = SP(30) - PV(30)$$

$Error(30) = 38 - 37{,}67$

$Error(30) = 0{,}33$

$delta\ Error(t) = Error(t) - Error(t-1)$

$delta\ Error(29) = Error(29) - Error(28)$

$delta\ Error(29) = 0{,}37 - (0{,}67)$

$delta\ Error(29) = -0{,}3$

Dari proses implentasi tersebut dihasilkan data sebagai berikut :

1. t_s (*settling time*) yaitu waktu yang diperlukan sistem untuk mencapai nilai akhir ketika *steady*. t_s berdasarkan pengujian adalah 27,30 menit. *Settling Time* didapat ketika suhu telah mencapai 38,34 °C.
2. Pada saat tanpa gangguan terdapat error terbesar yaitu:

$$\%\ Error\ maks = \frac{|36{,}9 - 38|}{38} \times 100\%$$
$$= 0{,}0289 \times 100\%$$
$$= 2{,}9\%$$

3. Pada saat dengan gangguan terdapat error terbesar yaitu:

$$\%\ Error\ maks = \frac{|35{,}67 - 38|}{38} \times 100\%$$
$$= 0{,}021 \times 100\%$$
$$= 6{,}1\%$$

4. Setelah diberi gangguan, sistem didapatkan *recovery time* sebesar 12 detik.

BAB VI
KESIMPULAN DAN SARAN

6.1 Kesimpulan

- Penggunaan KLF pada pengaturan suhu Inkubator mencapai nilai *setpoint* dengan nilai *error steady state* sebesar 2,3 % saat tidak terjadi gangguan.
- Penggunaan KLF pada pengaturan suhu Inkubator mencapai nilai *setpoint* dengan nilai *error steady state* sebesar 3,15%.
- t_s (*settling time*) yaitu waktu yang diperlukan sistem untuk mencapai nilai akhir ketika *steady*. t_s berdasarkan pengujian adalah 27,30 menit. *Settling Time* didapat ketika suhu telah mencapai 38,34 °C.

6.2 Saran

1. Menggunakan daya lampu yang lebih besar agar sistem dapat mecapai nilai steady state lebih cepat
2. Penggunaan daya lampu yang lebih besar juga akan membuat sistem memiliki waktu recovery time lebih cepat dari sebelumnya.

DAFTAR PUSTAKA

Andrianto, Heri. 2013. *Pemrograman Mikrokontroler AVR Atmega16 Menggunakan Bahasa C (CodeVisionAVR)*. Bandung: Informatika Bandung

Atmel. "Atmega32 datasheet". 11 Agustus 2015. http://www.datasheetcatalog.com/datasheets_pdf/A/T/M/E/ATMEGA32.shtml

Rasyaf. 1991. Pengelolaan Penetasan. Penerbit Kanisius. Yogyakarta.

Riyanto, A., 2001, *Sukses Menetaskan Telur Ayam*, Agro Media Pustaka, Jakarta.

Silva, De dan Clarence W. 1989. *Control Sensors and Actuators*. Englewood Cliffs: Prentice-Hall, Inc.

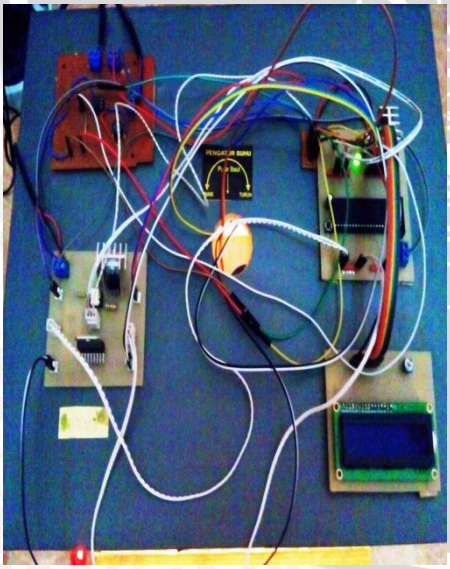
Texas Instrument. "LM35 datasheet". 11 Agustus 2015. http://www.datasheetcatalog.com/datasheets_pdf/L/M/3/5/LM35.shtml

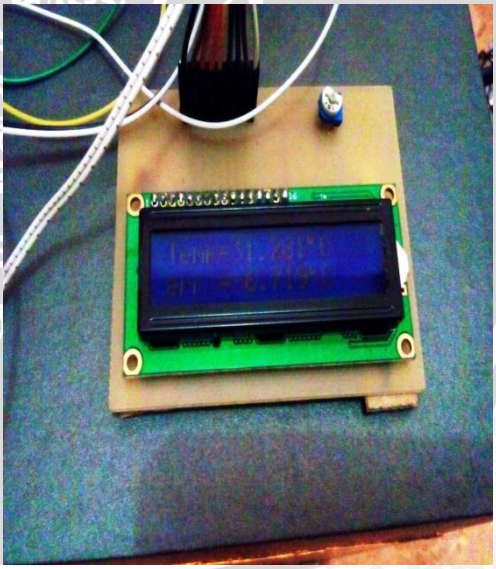
Unknow Manufacturer. *"datasheet* QC1602A". 11 Agustus 2015. http://www.datasheetcatalog.com/datasheets_pdf/A/C/1/6/AC1602A.shtml

Yan, Jun, Ryan, Michael, Power, James. 1994. Using Fuzzy Logic.UK: Prentice Hall International.

LAMPIRAN

Lampiran 1. Foto Alat dan Dokumentasi





Desain perangkat keras sistem pengaturan suhu Inkubator Telur

Lampiran 2. *Listing* Program Utama Kontroler Logika *Fuzzy*

```
This program was created by the
CodeWizardAVR V3.12 Advanced
Automatic Program Generator
ץ Copyright 1998-2014 Pavel Haiduc, HP InfoTech s.r.l.
http://www.hpinfotech.com

Project : Fuzzy Suhu
Version : Final
Date    : 5/4/2015
Author  : Bayu Indra
Company : 63 Elektro Teknik
Comments: This is a final version


Chip type               : ATmega32A
Program type            : Application
AVR Core Clock frequency: 16.000000 MHz
Memory model            : Small
External RAM size       : 0
Data Stack size         : 512
*****************************************************/

#include <mega32a.h>
```

```
#include <stdio.h>
#include <delay.h>
#include <alcd.h>   // Alphanumeric LCD functions

float adc0, adc1, suhu0, suhu1,temp, error, setp;
char lcd_buffer[33];
// Declare your global variables here

// Voltage Reference: AREF pin
#define ADC_VREF_TYPE ((0<<REFS1) | (0<<REFS0) | (0<<ADLAR))

// of the AD conversion result
unsigned char read_adc(unsigned char adc_input)
{
ADMUX=adc_input | ADC_VREF_TYPE;
// Delay needed for the stabilization of the ADC input voltage
delay_us(10);
// Start the AD conversion
ADCSRA|=(1<<ADSC);
// Wait for the AD conversion to complete
while ((ADCSRA & (1<<ADIF))==0);
ADCSRA|=(1<<ADIF);
return ADCW;
}
```

```
void pwm0()
{
    PORTC.0=0;
    PORTC.1=1;
    OCR1A =127;

    PORTC.6=0;
    PORTC.7=1;
    OCR1B =127;
}

void pwm1()
{
    PORTC.0=0;
    PORTC.1=1;
    OCR1A =102;

    PORTC.6=0;
    PORTC.7=1;
    OCR1B =102;
}

void pwm2()
```

```
{
    PORTC.0=0;
    PORTC.1=1;
    OCR1A =204;

    PORTC.6=0;
    PORTC.7=1;
    OCR1B =204;
}

void pwm3()
{
    PORTC.0=0;
    PORTC.1=0;
    OCR1A =0;

    PORTC.6=0;
    PORTC.7=0;
    OCR1B =0;
}

void delay0()
{
     delay_ms(2);
```

```
        PORTB.0=1;
        delay_us(100);
        PORTB.0=0;
}

void delay1()
{
        delay_ms(0);
        PORTB.0=1;
        delay_us(100);
        PORTB.0=0;
}

void delay2()
{
        delay_ms(4);
        PORTB.0=1;
        delay_us(100);
        PORTB.0=0;
}

void delay3()
{
        delay_ms(0);
```

```
    PORTB.0=0;
    delay_us(100);
    PORTB.0=0;
}

void bayu()
{
    lcd_clear();
    sprintf(lcd_buffer,"   Fuzzy_Suhu\n   BAYU INDRA");
    lcd_puts(lcd_buffer);
    delay_ms(3000);
}

void main(void)
{
// Declare your local variables here

// Input/Output Ports initialization
// Port A initialization
// Function: Bit7=In Bit6=In Bit5=In Bit4=In Bit3=In Bit2=In Bit1=In Bit0=In
DDRA=(0<<DDA7) | (0<<DDA6) | (0<<DDA5) | (0<<DDA4) | (0<<DDA3) |
(0<<DDA2) | (0<<DDA1) | (0<<DDA0);
// State: Bit7=T Bit6=T Bit5=T Bit4=T Bit3=T Bit2=T Bit1=T Bit0=T
PORTA=(0<<PORTA7) | (0<<PORTA6) | (0<<PORTA5) | (0<<PORTA4) |
(0<<PORTA3) | (0<<PORTA2) | (0<<PORTA1) | (0<<PORTA0);
```

// Port B initialization

// Function: Bit7=In Bit6=In Bit5=In Bit4=In Bit3=In Bit2=In Bit1=In Bit0=In

DDRB=(0<<DDB7) | (0<<DDB6) | (0<<DDB5) | (0<<DDB4) | (0<<DDB3) | (0<<DDB2) | (0<<DDB1) | (1<<DDB0);

// State: Bit7=T Bit6=T Bit5=T Bit4=T Bit3=T Bit2=T Bit1=T Bit0=T

PORTB=(0<<PORTB7) | (0<<PORTB6) | (0<<PORTB5) | (0<<PORTB4) | (0<<PORTB3) | (0<<PORTB2) | (0<<PORTB1) | (0<<PORTB0);

// Port C initialization

// Function: Bit7=Out Bit6=Out Bit5=In Bit4=In Bit3=In Bit2=In Bit1=Out Bit0=Out

DDRC=(1<<DDC7) | (1<<DDC6) | (0<<DDC5) | (0<<DDC4) | (0<<DDC3) | (0<<DDC2) | (1<<DDC1) | (1<<DDC0);

// State: Bit7=0 Bit6=0 Bit5=T Bit4=T Bit3=T Bit2=T Bit1=0 Bit0=0

PORTC=(0<<PORTC7) | (0<<PORTC6) | (0<<PORTC5) | (0<<PORTC4) | (0<<PORTC3) | (0<<PORTC2) | (0<<PORTC1) | (0<<PORTC0);

// Port D initialization

// Function: Bit7=In Bit6=In Bit5=Out Bit4=Out Bit3=In Bit2=In Bit1=In Bit0=In

DDRD=(0<<DDD7) | (0<<DDD6) | (1<<DDD5) | (1<<DDD4) | (0<<DDD3) | (0<<DDD2) | (0<<DDD1) | (0<<DDD0);

// State: Bit7=T Bit6=T Bit5=0 Bit4=0 Bit3=T Bit2=T Bit1=T Bit0=T

PORTD=(0<<PORTD7) | (0<<PORTD6) | (0<<PORTD5) | (0<<PORTD4) | (0<<PORTD3) | (0<<PORTD2) | (0<<PORTD1) | (0<<PORTD0);

// Timer/Counter 0 initialization

```
// Clock source: System Clock
// Clock value: Timer 0 Stopped
// Mode: Normal top=0xFF
// OC0 output: Disconnected
TCCR0=(0<<WGM00) | (0<<COM01) | (0<<COM00) | (0<<WGM01) | (0<<CS02) |
(0<<CS01) | (0<<CS00);
TCNT0=0x00;
OCR0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: 2000.000 kHz
// Mode: Fast PWM top=0x00FF
// OC1A output: Non-Inverted PWM
// OC1B output: Non-Inverted PWM
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer Period: 0.128 ms
// Output Pulse(s):
// OC1A Period: 0.128 ms Width: 0 us
// OC1B Period: 0.128 ms Width: 0 us
// Timer1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
```

```
TCCR1A=(1<<COM1A1) | (0<<COM1A0) | (1<<COM1B1) | (0<<COM1B0) |
(0<<WGM11) | (1<<WGM10);

TCCR1B=(0<<ICNC1) | (0<<ICES1) | (0<<WGM13) | (1<<WGM12) | (0<<CS12) |
(1<<CS11) | (0<<CS10);

TCNT1H=0x00;

TCNT1L=0x00;

ICR1H=0x00;

ICR1L=0x00;

OCR1AH=0x00;

OCR1AL=0x00;

OCR1BH=0x00;

OCR1BL=0x00;


// Timer/Counter 2 initialization

// Clock source: System Clock

// Clock value: Timer2 Stopped

// Mode: Normal top=0xFF

// OC2 output: Disconnected

ASSR=0<<AS2;

TCCR2=(0<<PWM2) | (0<<COM21) | (0<<COM20) | (0<<CTC2) | (0<<CS22) |
(0<<CS21) | (0<<CS20);

TCNT2=0x00;

OCR2=0x00;


// Timer(s)/Counter(s) Interrupt(s) initialization
```

```
TIMSK=(0<<OCIE2) | (0<<TOIE2) | (0<<TICIE1) | (0<<OCIE1A) | (0<<OCIE1B) |
(0<<TOIE1) | (0<<OCIE0) | (0<<TOIE0);

// External Interrupt(s) initialization
// INT0: On
// INT0 Mode: Falling Edge
// INT1: Off
// INT2: Off
GICR|=(0<<INT1) | (1<<INT0) | (0<<INT2);
MCUCR=(0<<ISC11) | (0<<ISC10) | (1<<ISC01) | (0<<ISC00);
MCUCSR=(0<<ISC2);
GIFR=(0<<INTF1) | (1<<INTF0) | (0<<INTF2);

// USART initialization
// USART disabled
UCSRB=(0<<RXCIE) | (0<<TXCIE) | (0<<UDRIE) | (0<<RXEN) | (0<<TXEN) |
(0<<UCSZ2) | (0<<RXB8) | (0<<TXB8);

// Analog Comparator initialization
// Analog Comparator: Off
// The Analog Comparator's positive input is
// connected to the AIN0 pin
// The Analog Comparator's negative input is
// connected to the AIN1 pin
```

```
ACSR=(1<<ACD) | (0<<ACBG) | (0<<ACO) | (0<<ACI) | (0<<ACIE) | (0<<ACIC) |
(0<<ACIS1) | (0<<ACIS0);

// ADC initialization
// ADC Clock frequency: 1000.000 kHz
// ADC Voltage Reference: AREF pin
// ADC Auto Trigger Source: Free Running
// Only the 8 most significant bits of
// the AD conversion result are used
ADMUX=ADC_VREF_TYPE;
ADCSRA=(1<<ADEN) | (0<<ADSC) | (1<<ADATE) | (0<<ADIF) | (0<<ADIE) |
(1<<ADPS2) | (0<<ADPS1) | (0<<ADPS0);
SFIOR=(0<<ADTS2) | (0<<ADTS1) | (0<<ADTS0);

// SPI initialization
// SPI disabled
SPCR=(0<<SPIE) | (0<<SPE) | (0<<DORD) | (0<<MSTR) | (0<<CPOL) | (0<<CPHA) |
(0<<SPR1) | (0<<SPR0);

// TWI initialization
// TWI disabled
TWCR=(0<<TWEA) | (0<<TWSTA) | (0<<TWSTO) | (0<<TWEN) | (0<<TWIE);

// Alphanumeric LCD initialization
// Connections are specified in the
```

```
// Project|Configure|C Compiler|Libraries|Alphanumeric LCD menu:
// RS - PORTB Bit 7
// RD - PORTB Bit 6
// EN - PORTB Bit 5
// D4 - PORTB Bit 4
// D5 - PORTB Bit 3
// D6 - PORTB Bit 2
// D7 - PORTB Bit 1
// Characters/line: 16
lcd_init(16);

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

bayu();

setp=38;

while (1)
    {
    // Place your code here
    adc0=read_adc(1);
    adc1=read_adc(3);
    suhu0=adc0*500/1023;
```

```
suhu1=adc1*500/1023;

temp=(suhu0+suhu1)/2;

if (temp==setp)
 {
    pwm0();
 }
else if (temp<setp)
 {
    pwm1();
 }
else if (temp>setp)
 {
    pwm2();
 }
else
 {
    pwm3(); // Semua mati
 };

error=temp-setp;

lcd_clear();
```

```
    sprintf(lcd_buffer,"Temp=%.3f\xdf" "C",temp);
    lcd_puts(lcd_buffer);
    lcd_gotoxy(0,1);
    sprintf(lcd_buffer,"err =%+.3f\xdf" "C",error);
    lcd_puts(lcd_buffer);
    delay_ms(1000);
    }
}


// External Interrupt 0 service routine
interrupt [EXT_INT0] void ext_int0_isr(void)
{
  if (temp==setp)
  {
    delay0();
  }
  else if (temp<setp)
  {
    delay1();
  }
  else if (temp>setp)
  {
    delay2();
```

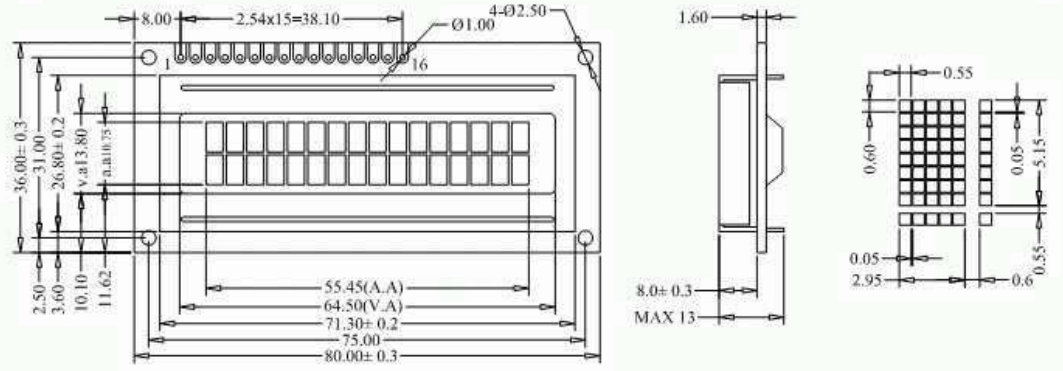
```
    }
    else
    {
       delay3(); // Semua Mati
    };

}
```

Lampiran 3. *DataSheet*

LCD

1.DIMENSION OUTLINE



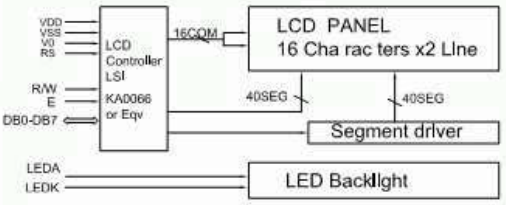
2.MECHANICAL SPECIFICATIONS

ITEM	SPECIFICATIONS	ITEM	REMARK
Modeule Size(L×W×H)	80.0×36.0×13.0	mm	Reference Dimensional Outline
View Area(W×H)	64.5×13.8	mm	
Effective V/Area	55.45×10.75	mm	
Number of Characters	16CH×2Lines	-	
Characters Size(W×H)	2.95×5.15	mm	
Dot Size(W×H)	0.55×0.60	mm	
Weight(Reflective/Led)	-	g	

3.ABSOLUTE MAXIMUM RATINGS

ITEM	SYMBOL	CONDITION	STANDARD	
			MIN	MAX
Logic Voltage	V_{DD}	Ta=25℃	-0.3V	7V
LCD Voltage	V_{LCD}		-0.3V	13V
Input Voltage	V_I		-0.3V	V_{DD}+0.3V
Operation Temperature	T_{OP}	—	-20℃	70℃
Storage Temperature	V_{OP}	—	-30℃	80℃

4.BLOCK DIAGRAMMECHANICAL



5.LED BACKLIGHT SPECIFICATIONS

ITEM	SYMBOL	TYPE	MAX	UNIT
Ta=25℃				
Forward Voltage	V_f	4.1	4.3	V
Forward Current	I_F	120	—	mA
Emission Vave Length	λ_P	568	—	nm

6.INTERFACE PIN CONNECTIONS

ITEM	SYMBOL	LEVEL	FUNCTIONS
1	VSS	0V	Power Ground
2	VDD	+5V	Power supply for logic
3	V0	—	Contrast adjust
4	RS	H/L	H:data L:command
5	R/W	H/L	H:read L:write
6	E	H.H→L	Enable signal
7-14	DB0-DB7	H/L	Data Bus
15	LEDA	+5V	Power supply for LED Backlight
16	LEDK	0V	

7.ELECTRICAL CHARACTERISTICS

ITEM	SYMBOL	MIN	TYPE	MAX	UNIT
Ta=25℃					
Logic Power	V_{DD}	4.5	5	5.5	V
Input High Voltage	V_{IH}	2.2	—	V_{DD}	V
Input Low Voltage	V_{IL}	-0.3	—	0.6	V
Output High Voltage	V_{OH}	2.4	—	V_{DD}	V
Output Low Voltage	V_{OL}	0	—	0.4	V
Logic Current	I_{DD}	—	1.5	3.0	mA
Operation Voltage For LCD	V_{0}-GND	—	5	—	V

Atmega32

Features
• High-performance, Low-power Atmel®AVR® 8-bit Microcontroller

• Advanced RISC Architecture
– 131 Powerful Instructions – Most Single-clock Cycle Execution
– 32 × 8 General Purpose Working Registers
– Fully Static Operation
– Up to 16 MIPS Throughput at 16MHz
– On-chip 2-cycle Multiplier

• High Endurance Non-volatile Memory segments
– 32Kbytes of In-System Self-programmable Flash program memory
– 1024Bytes EEPROM
– 2Kbytes Internal SRAM
– Write/Erase Cycles: 10,000 Flash/100,000 EEPROM
– Data retention: 20 years at 85°C/100 years at 25°C(1)
– Optional Boot Code Section with Independent Lock Bits In-System Programming by On-chip Boot Program True Read-While-Write Operation
– Programming Lock for Software Security

• JTAG (IEEE std. 1149.1 Compliant) Interface
– Boundary-scan Capabilities According to the JTAG Standard
– Extensive On-chip Debug Support
– Programming of Flash, EEPROM, Fuses, and Lock Bits through the JTAG Interface

• Peripheral Features
– Two 8-bit Timer/Counters with Separate Prescalers and Compare Modes
– One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
– Real Time Counter with Separate Oscillator
– Four PWM Channels
– 8-channel, 10-bit ADC, 8 Single-ended Channels, 7 Differential Channels in TQFP Package Only, 2 Differential Channels with Programmable Gain at 1x, 10x, or 200x
– Byte-oriented Two-wire Serial Interface
– Programmable Serial USART
– Master/Slave SPI Serial Interface
– Programmable Watchdog Timer with Separate On-chip Oscillator
– On-chip Analog Comparator

• Special Microcontroller Features
– Power-on Reset and Programmable Brown-out Detection
– Internal Calibrated RC Oscillator
– External and Internal Interrupt Sources
– Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby and Extended Standby

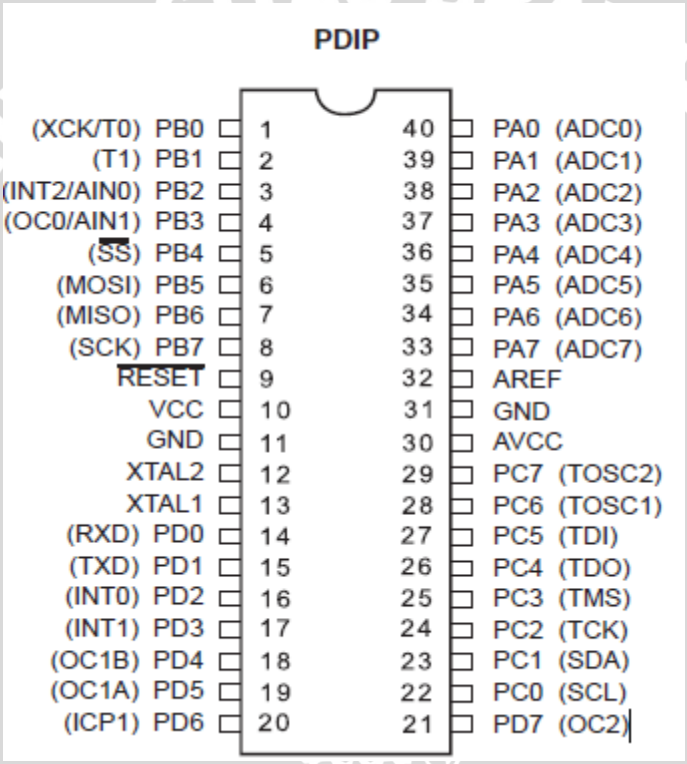
• I/O and Packages
– 32 Programmable I/O Lines

– 40-pin PDIP, 44-lead TQFP, and 44-pad QFN/MLF

• Operating Voltages
– 2.7V - 5.5V for ATmega32L
– 4.5V - 5.5V for ATmega32

• Speed Grades
– 0 - 8MHz for ATmega32L
– 0 - 16MHz for ATmega32

• Power Consumption at 1MHz, 3V, 25°C
– Active: 1.1mA
– Idle Mode: 0.35mA
– Power-down Mode: < 1µA



Atmega32 pin configuration

The Atmel®AVR®AVR core combines a rich instruction set with 32 general purpose working registers. All the 32 registers are directly connected to the Arithmetic Logic Unit (ALU), allowing two independent registers to be accessed in one single instruction executed in one clock cycle. The resulting architecture is more code efficient while achieving throughputs up to ten times faster than conventional CISC microcontrollers. The ATmega32 provides the following features: 32Kbytes of In-System Programmable Flash Program memory with Read-While-Write capabilities, 1024bytes EEPROM, 2Kbyte SRAM, 32 general purpose I/O lines, 32 general purpose working registers, a JTAG interface for Boundaryscan, On-chip Debugging support and programming, three flexible Timer/Counters with compare modes, Internal and External Interrupts, a serial programmable USART, a byte oriented Two-wire Serial Interface, an

8-channel, 10-bit ADC with optional differential input stage with programmable gain (TQFP package only), a programmable Watchdog Timer with Internal Oscillator, an SPI serial port, and six software selectable power saving modes. The Idle mode stops the CPU while allowing the USART, Two-wire interface, A/D Converter, SRAM, Timer/Counters, SPI port, and interrupt system to continue functioning. The Power-down mode saves the register contents but freezes the Oscillator, disabling all other chip functions until the next External Interrupt or Hardware Reset. In Power-save mode, the Asynchronous Timer continues to run, allowing the user to maintain a timer base while the rest of the device is sleeping.

The ADC Noise Reduction mode stops the CPU and all I/O modules except Asynchronous Timer and ADC, to minimize switching noise during ADC conversions. In Standby mode, the crystal/resonator Oscillator is running while the rest of the device is sleeping. This allows very fast start-up combined with low-power consumption. In Extended Standby mode, both the main Oscillator and the Asynchronous Timer continue to run. The device is manufactured using Atmel's high density nonvolatile memory technology. The Onchip ISP Flash allows the program memory to be reprogrammed in-system through an SPI serial interface, by a conventional nonvolatile memory programmer, or by an On-chip Boot program running on the AVR core. The boot program can use any interface to download the application program in the Application Flash memory. Software in the Boot Flash section will continue to run while the Application Flash section is updated, providing true Read-While-Write operation. By combining an 8-bit RISC CPU with In-System Self-Programmable Flash on a monolithic chip, the Atmel ATmega32 is a powerful microcontroller that provides a highly-flexible and cost-effective solution to many embedded control applications. The Atmel AVR ATmega32 is supported with a full suite of program and system development tools including: C compilers, macro assemblers, program debugger/simulators, in-circuit emulators and evaluation kits.

Sensor Lm35

FEATURES

• Calibrated Directly in ° Celsius (Centigrade)
• Linear + 10 mV/°C Scale Factor
• 0.5°C Ensured Accuracy (at +25°C)
• Rated for Full −55°C to +150°C Range
• Suitable for Remote
• Low Cost Due to Wafer-Level Trimming
• Operates from 4 to 30 V
• Less than 60-µA Current
• Low Self-Heating, 0.08°C in Still Air
• Nonlinearity Only ±¼°C Typical
• Low Impedance Output, 0.1 Ω for 1 mA Load

Description

The LM35 series are precision integrated- temperature sensors, with an output voltage linearly proportional to the Centigrade temperature. Thus the LM35 has an advantage over linear temperature sensors calibrated in ° Kelvin, as the user is not required to subtract a large constant voltage from the output to obtain convenient Centigrade scaling. The LM35 does not require any external calibration or trimming to provide typical accuracies of ±¼°C at room temperature and ±¾°C over a full −55°C to +150°C temperature range. Low cost is assured by trimming and calibration at the wafer level. The low output impedance, linear output, and precise inherent calibration of the LM35 make interfacing to readout or control circuitry especially easy. The device is used with single power supplies, or with plus and minus supplies. As the LM35 draws only 60 μA from the supply, it has very low self-heating of less than 0.1°C in still air. The LM35 is rated to operate over a −55°C to +150°C temperature range, while the LM35C is rated for a −40°C to +110°C range (−10° with improved accuracy). The LM35 series is available packaged in hermetic TO transistor packages, while the LM35C, LM35CA, and LM35D are also available in the plastic TO-92 transistor package. The LM35D is also available in an 8-lead surface-mount smalloutline package and a plastic TO-220 package.