PENERAPAN KONTROL LOGIKA FUZZY (KLF) SEBAGAI PENGENDALI SUHU DAN KADAR KEASAMAN (pH) PADA KOLAM PEMBENIHAN IKAN LELE

SKRIPSI

TEKNIK ELEKTRO KONSENTRASI TEKNIK KONTROL

Ditujukan untuk memenuhi persyaratan
memperoleh gelar Sarjana Teknik

DANANG INDRA PERMANA
NIM. 125060307111035

UNIVERSITAS BRAWIJAYA
FAKULTAS TEKNIK
MALANG
2016

LEMBAR PERSETUJUAN

PENERAPAN KONTROL LOGIKA FUZZY (KLF) SEBAGAI PENGENDALI SUHU DAN KADAR KEASAMAN (pH) PADA KOLAM PEMBENIHAN IKAN LELE

SKRIPSI

TEKNIK ELEKTRO KONSENTRASI TEKNIK KONTROL

Diajukan untuk memenuhi persyaratan
memperoleh gelar Sarjana Teknik

DANANG INDRA PERMANA

NIM. 125060307111035

Telah diperiksa dan disetujui oleh :

Dosen Pembimbing I	Dosen Pembimbing II
M. Aziz Muslim, S.T., M.T.,Ph.D.	**Ir. Purwanto, M.T.**
NIP. 19741203 200012 1 001	**NIP. 19540424 198601 1 001**

PERNYATAAN ORISINALITAS SKRIPSI

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

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

Malang, 3 Juni 2016

Mahasiswa,

DANANG INDRA PERMANA
NIM. 125060307111035

RINGKASAN

Danang Indra Permana, Jurusan Teknik Elektro, Fakultas Teknik Universitas Brawijaya, Juni 2016, *Penerapan Kontrol Logika Fuzzy (KLF) sebagai Pengendali Suhu dan Kadar Keasaman (pH) pada Kolam Pembenihan Ikan Lele*, Dosen Pembimbing: Muhammad Aziz Muslim dan Purwonto.

Budidaya benih ikan lele perlu memperhatikan suhu dan kadar keasaman (pH) air kolam. Suhu *ideal* untuk benih lele agar dapat hidup secara optimal adalah 26 - 30 ^{o}C dan pH ideal untuk benih lele adalah 6,5-9. Pada skripsi ini akan dibuat sistem pengendali suhu dan pH air kolam benih ikan lele agar suhu dan pH air kolam tetap terjaga sesuai dengan *setpoint* suhu 28 ^{o}C dan pH 7.8.

Pengendalian suhu pada skripsi ini dengan cara menambahkan air hangat dan biasa untuk menyesuaikan *setpoint*, pada pengendalian pH dengan cara menambahkan air asam dan basa. Air yang digunakan untuk pengendalian suhu dan pH adalah air yang keluar dari *valve* yang dihubungkan dengan motor *stepper*. Motor *stepper* berfungsi sebagai penggerak *valve.*

Kontroler yang digunakan pada skripsi ini adalah kontrol logika *fuzzy* (KLF). Perancangan KLF pada skripsi ini menggunakan masukan *error* dan *delta error* dengan 5 fungsi keanggotaan masukan dan keluaran. Metode inferensi yang digunakan adalah metode *min-max* dan metode defuzzifikasi *center of area*. Dengan mengatur *setpoint* 7.8 untuk pH diperoleh respon sistem *settling time* (ts) sebesar 240 detik dengan *error steady state* (ess) sebesar 0,98% dan *setpoint* 28 ^{o}C untuk suhu diperoleh respon sistem *settling time* (ts) sebesar 210 detik dengan *error steady state* (ess) sebesar 0,29%.

Kata kunci : *Fuzzy,* Suhu, *pH,* Benih ikan lele

SUMMARY

Danang Indra Permana, *Department of Electrical Engineering, Faculty of Engineering University of Brawijaya, Juni 2016, Application of Fuzzy Logic Control (FLC) as controlling temperature and acidity levels (pH) in the pool Hatchery Catfish*, *Academic Supervisor*: Muhammad Aziz Muslim dan Purwonto.

Cultivation of seeds catfish need to pay attention to the temperature and acidity (pH) of pool water. The ideal temperature for seed catfish in order to live optimally is 26-30 ° C and pH are ideal for seed catfish is 6.5 to 9. This thesis will make temperature control system and a pH of pool water catfish seed so that the temperature and pH of pool water is maintained in accordance with the setpoint temperature of 28 ° C and pH 7.8.

Controlling the temperature in this thesis by adding warm water and usual to adjust the setpoint, the control of pH by adding acid and alkaline water. Water used for controlling the temperature and the pH is water coming out of the valve connected to a stepper motor. stepper motor serves as the driving valve.

The controller used in this thesis is the fuzzy logic control (FLC). Design FLC in this thesis uses two inputs and two outputs with 5 input and output membership functions. Inference method used is the min-max method and the method of defuzzification center of the area. With a setpoint set to pH 7.8 obtained by the system response settling time of 240 seconds with steady state error of 0.98% and the setpoint temperature of 28 ° C to obtained the system response settling time of 210 seconds with steady state error of 0.29%.

Keyword : Fuzzy, Temperature, pH, Seed catfish

PENGANTAR

PENERAPAN KONTROL LOGIKA FUZZY (KLF) SEBAGAI PENGENDALI SUHU DAN KADAR KEASAMAN (pH) PADA KOLAM PEMBENIHAN IKAN LELE

Bismillahirrohmanirrohim. Alhamdulillah, puji syukur penulis panjatkan kehadirat Allah SWT yang telah memberikan rahmat dan hidayah-Nya, sehingga penulis dapat menyelesaikan skripsi yang berjudul "Penerapan Kontrol Logika Fuzzy (KLF) sebagai Pengendali Suhu dan Kadar Keasaman (pH) pada Kolam Pembenihan Ikan Lele" dengan baik. Tak lepas shalawat serta salam tercurahkan kepada junjungan kita Nabi Muhammad SAW yang telah menjadi suri tauladan bagi yang mengharapkan rahmat dan hidayah-Nya.

Penulis menyadari bahwa penyusunan skripsi ini tidak lepas dari bantuan, bimbingan serta dorongan dari berbagai pihak. Pada kesempatan ini penulis menyampaikan rasa terima kasih yang sebesar – besarnya kepada:

- Allah SWT yang telah memberikan kelancaran, kemudahan dan hidayah-Nya.
- Nabi Muhammad SAW yang telah menjadi suri tauladan.
- Kedua orang tua Sugeng dan Fatmiatun yang telah banyak memberikan do'a, kasih sayang, dukungan, serta semangat dan banyak hal yang tidak bisa penulis tuliskan satu persatu.
- Bapak M. Aziz Muslim, S.T., M.T., Ph.D. selaku dosen pembimbing skripsi dan Ketua Jurusan Teknik Elektro Fakultas Teknik Universitas Brawijaya Malang.
- Bapak Hadi Suyono S.T., M.T., Ph.D selaku Sekretaris Jurusan Teknik Elektro Fakultas Teknik Universitas Brawijaya.
- Bapak Ir. Purwanto, M.T. selaku dosen pembimbing skripsi dan KKDK Sistem Kontrol.
- Ibu Dr.Ir. Erni Yudaningtyas, M.T selaku Ketua Laboratorium Sistem Kontrol.
- Ibu Dr. Rini Nur Hasanah, S.T., M.Sc. selaku dosen penasehat akademik.
- Bapak, Ibu dosen serta segenap staf dan karyawan Jurusan Teknik Elektro baik secara langsung maupun tidak langsung yang telah banyak membantu dalam menyelesaikan skripsi ini.
- Mbak Eka selaku Laboran Lab. Sistem Kontrol

- Sahabat-Sahabat, Mahesutora, Ipin Mergan, Galih Blitar, Mico, Ray Selvi, Hilmy, Yogi, Yufrizal, Yudha Frankenstein, Tyo Bokong, Dirga Galer, Panji Bintang, Graha Peler, Faris dan Fian Tuban yang telah memberikan semangat, doa, dan canda tawa.
- Team skripsi bareng-bareng, Hesa, Hanip, Tyo Pinter, dan Angga terimakasih telah membantu dalam segala hal.
- Faradhila Ratu Sejagat yang telah membantu membuat animasi dan desain plant.
- Sam Dimas terimakasih telah membantu kesulitan dalam hal programing dan konsep.
- Bagus, Bidin dan Rifqa termiakasih telah membantu kesulitan dalam hal elektrik dan selaku team PKM.
- Tak lupa keluarga besar VOLTAGE'12 dan CONTROL ENGINEERING'12 yang memberikan semangat, do'a, dan dukungan.
- Semua pihak yang telah memberikan bantuan serta dukungan baik secara langsung maupun tidak langsung atas penyusunan skripsi ini.

Dalam penyusunan skripsi ini, penulis menyadari bahwa skripsi ini belumlah sempurna, karena keterbatasan ilmu dan kendala – kendala lain yang terjadi selama pengerjaan skripsi ini. Oleh karena itu, penulis berharap kritik dan saran untuk penyempurnaan tulisan di masa yang akan datang, semoga tulisan ini dapat bermanfaat dan dapat digunakan untuk pengembangan lebih lanjut.

Malang, Juni 2016
Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1 Latar belakang

Pemenuhan Kebutuhan protein di Indonesia menurut Data SUSENAS (Survey Sosial Ekonomi Nasional) – BPS (Badan Pusat Statistik) tahun 2013 terdiri dari 68.3% protein nabati dan 31.7% dari protein hewani. Rincian 31.7% protein hewani yaitu: 57.2% ikan, 19.6% daging, 23.2% telur dan susu. Namun pemenuhan kebutuhan protein hewani tersebut masih terbilang sangat rendah. Kurangnya pemenuhan kebutuhan protein hewani mengakibatkan pembangunan manusia Indonesia tertinggal dibandingkan negara Asia lain. "Data SUSENAS (Survey Sosial Ekonomi Nasional) – BPS (Badan Pusat Statistik) tahun 2013 menunjukkan bahwa sumbangan protein ikan terhadap konsumsi protein hewani masyarakat Indonesia mencapai 57%, ini terjadi seiring dengan kecenderungan pergeseran konsumen dalam pemenuhan kebutuhan protein hewani dari *red meat* ke *white meat"*.

Budidaya ikan juga dapat memanfaatkan efisiensi lahan/tanah yang tidak cocok untuk pertanian atau perkebunan, jadi lahan tersebut dapat dimanfaatkan untuk budidaya ikan tawar. "Berdasarkan data DJPB-KKP (Direktorat Jenderal Perikanan Budidaya-Kementerian Kelautan dan Perikanan) tahun 2013 bahwa target dengan peningkatan tertinggi adalah komoditas patin dengan kenaikan rata-rata sebesar 70 persen setiap tahunnya. Lele berada diurutan kedua dengan peningkatan rata-rata sebesar 35 persen kemudian ikan nila sebesar 26 persen dan peningkatan rendah adalah ikan mas dan gurame dengan kenaikan rata-rata sebesar 7 persen dan 5 persen". Tapi dilihat dari aspek lahan, ekonomi, dan sosial, dari kelima jenis ikan, ikan lele yang paling mumpuni untuk dibudidayakan. Dengan lahan yang terbatas,sumber air yang kurang baik, modal sedikitpun sudah bisa untuk dibudidayakan. Komoditas hasil budidaya lele juga dapat diterima oleh berbagai lapisan masyarakat. Selain itu juga rasio pakan menjadi daging ikan lele bisa mencapai 1:1. Artinya setiap pemberian pakan sebanyak 1 kg akan dihasilkan 1 kg pertambahan berat lele (Kurniawan Putro S,2016).

Di Desa Mandirejo Kecamatan Merakurak, Tuban. Khususnya di dusun Kebondalem banyak warga yang memiliki usaha budidaya ikan lele dengan media kolam yang terbuat dari semen maupun dari terpal. Banyaknya pembudidaya disana sangat didukung oleh sumberdaya alam khususnya air, tapi pembudidayaan masih belum maksimal. Ada kendala yang dihadapi oleh pembudidaya di Desa Mandirejo, yaitu

sedikitnya benih ikan lele berusia kurang dari tiga minggu (burayak) yang mampu bertahan hidup". Ada dua cara pembenihan yang digunakan di desa Mandirejo: Pertama, pembelian benih dari pembudidaya lain. Kedua, melakukan pemijahan sendiri.

Cara kedua saat ini sedang ditekuni pembudidaya, karena jika dibandingkan dengan cara pertama manfaat yang akan diterima akan berdeda jauh. Kekurangan dari cara pertama adalah lokasi pembelian benih itu sendiri berada diluar kota, kondisi seperti ini sangat tidak menguntungkan karena benih lele akan mudah *stress* akibat goncangan dan kondisi lingkungan yang berbeda dari tempat pembelian. Ketika benih *stress* maka keberhasilan untuk hidup itu sangat kecil. Sedangkan cara kedua kekurangannya adalah pada kondisi lingkungan (suhu dan pH) yang berubah-ubah, tapi masalah ini juga dialami oleh cara pertama juga. Jadi cara kedua akan lebih bermanfaat bagi pembudidaya kedepannya.

Keberhasilan dalam memijahkan sendiri / menetaskan sendiri bibit lele ini sangat bergantung pada kondisi lingkungan. Kondisi lingkungan ini yang sangat dikeluhkan pembudidaya disana, sebab benih lele yang masih berupa burayak dengan usia kurang dari tiga minggu sangat rentan terhadap perubahan pH dan suhu air kolam yang dapat menyebabkan benih mudah *stress* akibatnya benih akan mati. Kondisi lingkungan seperti ini membuat burayak yang berhasil bertahan hidup hingga mencapai usia tiga minggu atau ukuran 1-2 cm yang sudah bisa disebut sebagai benih sangat kurang maksimal sebab kematian burayak tinggi.

Pada pertumbuhan benih lele diperlukan kisaran suhu anatara 26 sampai 30 ºC dan tingkat kadar keasaman (pH) 6,5 – 9,0 (Cut Nina Herlina, 2015). Pada suhu dibawah 25 ºC, biasanya akan terbentuk bintik putih pada benih yang menyebabkan kematian massal. Apabila terjadi perubahan suhu, usahakan tidak terjadi secara ekstrim. Perubahan suhu kolam sebaiknya tidak berfluktuasi lebih dari 1 ºC. Banyak larva yang tidak mentolerir suhu yang berubah-ubah. (Rahmat Hidayat, 2013). Dengan didukung sumber air yang mendukung ini merupakan potensi/peluang untuk dapat menghasilkan benih secara mandiri untuk memaksimalkan hasil panen dan tidak menutup kemungkinan untuk bisa jadi produsen dengan menghasilkan benih yang berkualitas. Berdasarkan latar belakang tersebut akan dibuat sistem pengendali suhu dan pH pada air kolam benih ikan lele dengan cara mengendalikan motor *stepper* yang dikopel dengan *valve* untuk mengontrol suhu dan pH pada kolam pembenihan.

Pada penelitian ini mengangkat judul "Penerapan Kontrol Logika Fuzzy sebagai Pengendali Suhu dan Kadar Keasaman (pH) pada Kolam Pembenihan Ikan Lele". Dengan

menggunakan kontrol logika fuzzy yang bersifat fleksibel akan mempermudah perancangan sistem pengendalian suhu dan pH. Kontrol logika fuzzy bekerja berdasarkan rule-rule yang dibuat sesuai dengan pengetahuan seorang pakar (Ross, 2010).

Pada penelitian ini diharapkan dapat merancang suatu sistem pengendalian suhu dan kadar keasaman (pH) pada kolam pembenihan ikan lele untuk meningkatkan hasil produksi ikan lele.

1.2 Rumusan masalah

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

1. Bagaimana merancang dan membuat sistem pengendalian suhu dan kadar keasaman (pH) pada pembenihan ikan lele menggunakan kontrol logika *fuzzy* ?
2. Bagaimana merancang *software* sistem pengendalian suhu dan kadar keasaman (pH) pada pembenihan ikan lele menggunakan kontrol logika *fuzzy* ?
3. Apakah sistem pengendalian suhu dan kadar keasaman (pH) menggunakan kontrol logika *fuzzy* dapat diterapkan pada pembenihan ikan lele ?

1.3 Batasan masalah

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

1. Kolam pembenihan yang digunakan adalah miniatur untuk penelitian.
2. Aktuator yang digunakan adalah motor *stepper* DC.
3. Kontroler yang digunakan adalah mikrokontroler Arduino Mega 2560.
4. Kinerja *driver* dan elektronik tidak dibahas mendalam.
5. Gangguan pada pengontrolan suhu berupa penambahan air dengan suhu 90˚C sebanyak 2 liter.
6. Gangguan pada pengontrolan pH berupa penambahan larutan asam dengan pH 6.78 sebanyak 2 liter.
7. Pembahasan ditekankan pada pengendalian suhu dan kadar keasaman (pH) menggunakan kontrol logika *fuzzy*.

1.4 Tujuan

Penelitian ini bertujuan untuk merancang sistem kolam pembenihan yang dilengkapi pengaturan suhu dan kadar keasaman (pH) dengan menerapkan kontrol logika *fuzzy* sebagai pengendalinya sehingga suhu dan pH pada kolam pembenihan ikan lele dapat dipertahankan sesuai dengan *setpoint* untuk meningkatkan kuantitas dan menguji kelayakan sistem pengendalian suhu dan kadar keasaman (pH) pada kolam pembenihan ikan lele menggunakan kontrol logika *fuzzy*.

1.5 Manfaat

Manfaat dari penelitian ini adalah memberikan teknik pembenihan ikan lele yang dapat dikontrol sesuai dengan parameter suhu dan kadar keasaman yang di inginkan.

1.6 Sistematika pembahasan

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.

BAB IV Hasil dan Pembahasan

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

BAB V Penutup

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 sistem pengendalian suhu dan kadar keasaman (pH) pada kolam pembenihan ikan lele menggunakan kontrol logika *fuzzy*.

2.1 Benih ikan lele

Ikan lele termasuk ikan yang sangat kuat dengan berbagai macam kondisi air bahkan ikan lele mampu bertahan hidup beberapa menit bahkan sampai beberapa jam tanpa air. Walaupun demikian ikan lele yang berumur 1-30 hari atau yang disebut dengan benih lele rentan terhadap perubahan kondisi air yang terjadi. Benih ikan lele berumur 1-30 hari ditunjukkan pada Gambar 2.1.

Gambar 2.1 Benih ikan Lele
Sumber: https://ternakikanlelewordpress.com

Pada pertumbuhan benih diperlukan kisaran suhu anatara 26 sampai 30 ºC dan tingkat kadar keasaman (pH) 6,5 – 9,0 (Cut Nina Herlina, 2015). Pada suhu dibawah 25 ºC, biasanya akan terbentuk bintik putih pada larva yang menyebabkan kematian massal (Rahmat Hidayat, 2013).

2.2 Sensor suhu

Sensor adalah sesuatu yang digunakan untuk mendeteksi adanya perubahan lingkungan fisik atau kimia. Sensor suhu merupakan jenis sensor yang digunakan untuk mendeteksi setiap perubahan suhu. Pada perancangan sistem ini saya menggunakan sensor suhu DS18B20 adalah digital thermometer, dimana keluaran sensor berupa data digital

berupa sinyal pulsa yang mengindikasikan suhu tertentu. Sensor DS18B20 *waterproof* ditunjukkan pada Gambar 2.2. Range DS18B20 dari -55°C hingga 125°C. Sensor DS18B20 menggunakan 1-*wire*. Pin-pin nya adalah pin 1 = *output*,pin 2 = 5V,pin 3 = GND.

Gambar 2.2 Sensor suhu DS18B20
Sumber: *https://www.sparkfun.com*

2.3 Sensor pH

pH adalah pengukuran dari kadar keasaman atau kadar alkali sebuah larutan. Asam adalah zat yang mengandung hidrogen dan menyerahkan ion-ion hidrogen dalam larutan atau disebut donor proton. Basa mengandung OH dan menyerahkan ion-ion hidroksida dalam larutan encer atau disebut penerima proton (Zhang, Ju dan Wang, 2008). pH dapat didefinisikan sebagai berikut :

$$pH = -\log[H^+]$$

Dimana $[H^+]$ adalah konsentrasi ion hydrogen dalam mol/L.

Gambar 2.3 sensor pH SKU: SEN0161
Sumber*: www.sensorex.com/pH_Sensor*

Pada perancangan sistem ini saya menggunakan sensor pH kit SKU:SEN0161, yang dirancang untuk arduino, Sensor pH SKU SEN0161 ditunjukkan pada Gambar 2.3. Sensor pH kit SKU:SEN0161 memiliki LED (*Light Emitting Diode*) yang bekerja sebagai Indikator *Power*, BNC konektor dan PH2.0 antarmuka sensor.

2.4 Motor *stepper*

Motor *Stepper* adalah motor DC dengan prinsip stepping yang gerakannya bertahap (*step* per- *step*) atau motor *stepper* bergerak berdasarkan urutan pulsa yang diberikan

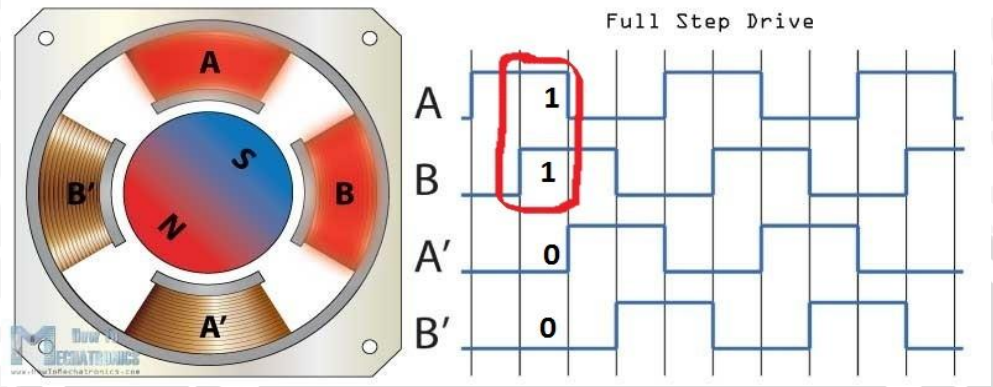
kepada motor. Karena itu, untuk menggerakkannya diperlukan *driver* motor *stepper* yang membangkitkan pulsa-pulsa periodik. *Driver* bekerja dengan mengkonversikan bit-bit masukan menjadi posisi rotor. Bit-bit tersebut berasal dari terminal-terminal *input* yang ada pada motor *stepper* yang menjadi kutub-kutub magnet dalam motor. Bila salah satu terminal diberi sumber tegangan, terminal tersebut akan mengaktifkan kutub di dalam magnet sebagai kutub utara dan kutub yang tidak diberi tegangan sebagai kutub selatan. Dengan terdapatnya dua kutub di dalam motor ini, rotor di dalam motor yang memiliki kutub magnet permanen akan mengarah sesuai dengan kutub-kutub *input*. Kutub utara rotor akan mengarah ke kutub selatan stator sedangkan kutub selatan rotor akan mengarah ke kutub utara stator.

Motor *stepper* tidak dapat bergerak sendiri secara kontinyu, tetapi bergerak secara diskrit per-*step* sesuai dengan spesifikasinya. Motor *stepper* mampu berputar untuk setiap *step*-nya dalam satuan sudut (0.75, 0.9, 1.8), makin kecil sudut per *step*-nya maka gerakan perstepnya motor *stepper* makin presisi. Kecepatan motor *stepper* ditentukan oleh kecepatan pemberian data pada komutatornya. Semakin cepat data yang diberikan, maka motor *stepper* akan semakin cepat berputarnya. . Dalam mengoperasikannya terdapat 2 prinsip yaitu *full-step* atau *half- step*, dengan *full- step* berarti motor stepper berputar sesuai dengan spesifikasi derajat setiap *step*-nya, sedangkan *half- step* berarti motor *stepper* berputar setengah derajat per-*step* dari spesifikasi motor *stepper* tersebut.

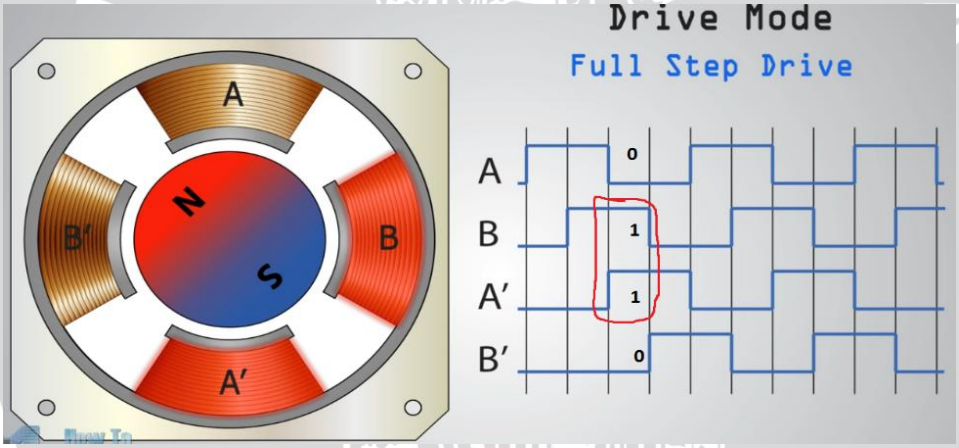
Gambar 2.4 Motor Stepper Nema 17
Sumber*: reprap.org/wiki/NEMA_**17**_Stepper_motor*

Pada perancangan sistem ini saya menggunakan Nema17 yang ditunjukkan pada Gambar 2.4. Motor *Stepper* Nema 17 membutuhkan *input* tegangan 8-42V, arus maksimum 3A, dan berjalan 1.8° setiap *step*-nya. *Mode full - step* yang saya gunakan untuk mengoperasikan motor *stepper*, maka motor *stepper* berputar sesuai dengan spesifikasinya yaitu 1.8° setiap *step*-nya yang artinya motor *stepper* membutuhkan 200 *step* persatu putaran penuh. Kelebihan menggunakan motor *stepper* ini, diantaranya sudut putar motor dapat diatur, menghasilkan perputaran yang lambat sehingga beban dapat

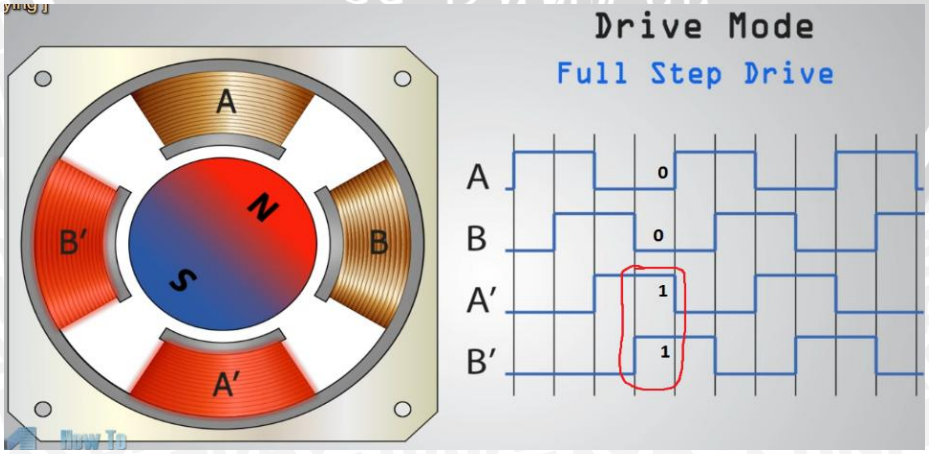
dikopel langsung ke porosnya dan menyediakan *output* torsi jauh lebih tinggi karena selalu memiliki 2 kumparan aktif pada waktu tertentu. Namun ini tidak meningkatkan resolusi *stepper* dan rotor berjalan pada siklus penuh dalam 4 langkah, untuk memahami prinsip kerja mode *full-step* pada motor stepper dapat dilihat pada Gambar 2.5, 2.6, 2.7, dan 2.8 yang akan menjelaskan bagaimana motor *stepper* bergerak setiap *step*-nya.



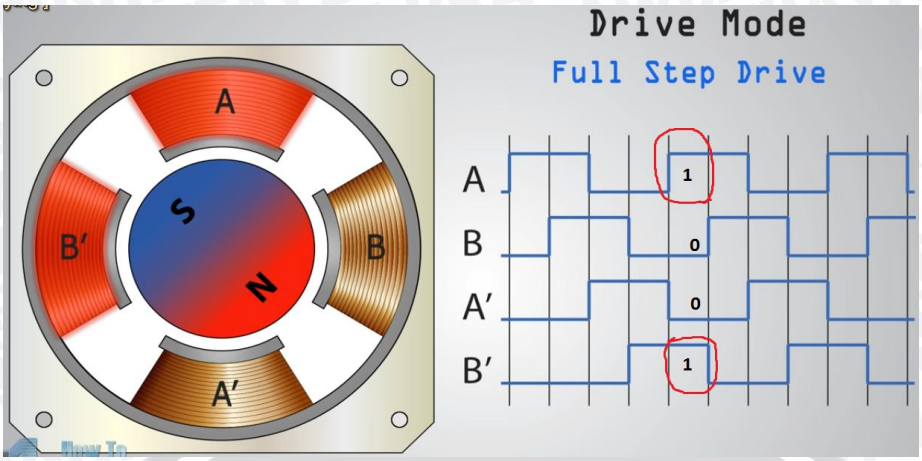
Gambar 2.5 *Full-Step Mode* koil AB aktif
Sumber: http://howtomechatronics.com/stepper-motor/



Gambar 2.6 *Full-Step Mode* koil BA' aktif
Sumber: http://howtomechatronics.com/stepper-motor/



Gambar 2.7 *Full-Step Mode* koil A'B' aktif
Sumber: *http://howtomechatronics.com/stepper-motor/*



Gambar 2.8 *Full-Step Mode* koil B'A aktif
Sumber*: http://howtomechatronics.com/stepper-motor/*

Pada Gambar 2.5, 2.6, 2.7, dan 2.8 terlihat untuk bergerak per-*step* pada mode *full-step* dibutuhkan 2 kutub berlogika satu dan 2 kutub berlogika nol berdasarkan *timing diagram*. Logika 1 dengan kata lain kutub tersebut tercatu oleh sumber tegangan dari *driver* dan logika 0 kutub tersebut tidak mendapat catu dari *driver*. Dua kutub yang tercatu tersebut aktif sebagai kutub utara dan kutub yang tidak diberi tegangan sebagai kutub selatan. Dengan terdapatnya dua kutub di dalam motor ini, rotor di dalam motor yang memiliki kutub magnet permanen akan mengarah sesuai dengan kutub-kutub *input*. Kutub utara rotor akan mengarah ke kutub selatan stator sedangkan kutub selatan rotor akan mengarah ke kutub utara stator.

2.5 *Driver* motor *stepper* DRV8825

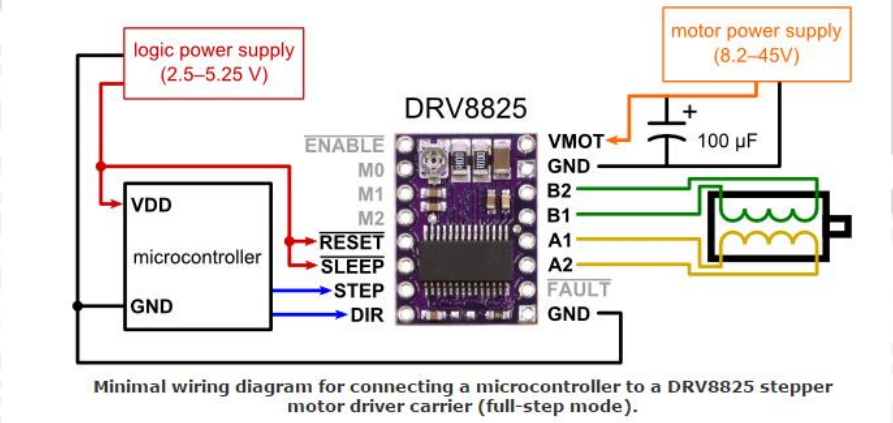
Pengendalian motor *stepper* tidak dapat langsung dilakukan oleh arduino. Motor *stepper* membutuhkan minimal tegangan 8.2 V sedangkan tegangan yang mampu diberi-kan arduino hanya 5, sehingga untuk dapat mengendalikan motor *stepper* diperlukan sumber tegangan eksternal yang dapat diberikan oleh *driver*. Pada perancangan sistem ini saya menggunakan *driver* DRV8825. *Driver* DRV8825 adalah *driver* motor *stepper* yang memiliki kemampuan melewati arus 2A serta mendukung 1/32 *micro stepping* sehingga pergerakan motor menjadi lebih halus. *Suplay driver* yang bisa diberikan tegangan 8.2-45 V akan terhubung di VMOT dan GND. *Driver* DRV8825 ditunjukkan pada Gambar 2.9.

Gambar 2.9 Driver motor stepper DRV825
Sumber*: www.ti.com/lit/ds/symlink/drv8825.pdf*

Pada *driver* DRV8825 dilengkapi pin yang memudahkan memilih mode yang akan digunakan sebagai dasar per-*step* motor *stepper* berjalan. Pin yang dimaksud adalah MS1, MS2 dan MS3 yang digunakan untuk memilih salah satu dari lima *mode step* sesuai dengan tabel kebenaran pada Gambar 2.10 . pada *driver* ini resistor *pull-down internal* sehingga jika pin MS1, MS2, MS3 dibiarkan terputus, maka secara otomatis *driver* akan beroperasi dalam *mode full-step*. Pada penelitian ini digunakan *mode full-step* sehingga pin MS1, MS2, MS3 dibiarkan terputus dan Pada Gambar 2.11 ditunjukkan *wiring* sederhana untuk *mode full-step*.

MS1	MS2	MS3	Resolution
LOW	LOW	LOW	Full Step
HIGH	LOW	LOW	Halft Step
LOW	HIGH	LOW	Quarter Step
HIGH	HIGH	LOW	Eighth step
HIGH	HIGH	HIGH	Sixteenth Step

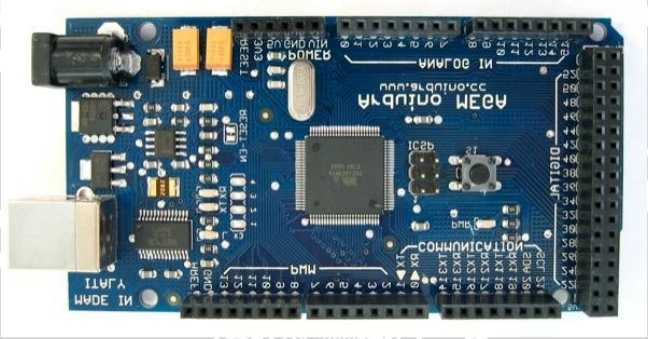
Gambar 2.10 Lima mode step driver DRV8825
Sumber*: www.ti.com/lit/ds/symlink/drv8825.pdf*



Gambar 2.11 *Wiring driver* DRV8825 dengan *mode full-step*
Sumber*: www.ti.com/lit/ds/symlink/drv8825.pdf*

2.6 Arduino Mega2560

Arduino Mega2560 adalah papan mikrokontroler berbasiskan ATmega2560. Arduino Mega2560 memiliki 54 pin digital *input/output*, 15 pin sebagai *output* PWM, 16 pin sebagai *input* analog, dan 4 pin sebagai UART (port serial hardware), 16 MHz kristal osilator, koneksi USB, *jack power*, *header* ICSP, dan tombol *reset*. Untuk menggunakan hanya dengan menghubungkannya ke komputer melalui kabel USB atau *power* dihubungkan dengan adaptor AC-DC atau baterai untuk mulai mengaktifkannya. Arduino Mega2560 ditunjukkan pada Gambar 2.12.



Gambar 2.12 Arduino Mega 2560
Sumber: http://arduino.cc/en/Main/ArduinoBoardMega

2.6.1 Daya

Arduino Mega dapat diaktifkan melalui koneksi USB atau dengan catu daya *eksternal*. Sumber daya dipilih secara otomatis. Sumber daya eksternal (non-USB) dapat berasal baik dari adaptor AC-DC atau baterai. Adaptor dapat dihubungkan dengan mencolokkan steker 2,1 mm yang bagian tengahnya terminal *positif* ke ke jack sumber tegangan pada papan. Jika tegangan berasal dari baterai dapat langsung dihubungkan melalui *header* pin Gnd dan pin Vin dari konektor *power*.

Arduino mega2560 dapat beroperasi dengan pasokan daya *eksternal* 6 Volt sampai 20 volt. Jika diberi tegangan kurang dari 7 Volt, maka, pin 5 Volt mungkin akan menghasilkan tegangan kurang dari 5 Volt dan ini akan membuat papan menjadi tidak stabil. Jika sumber tegangan menggunakan lebih dari 12 Volt, regulator tegangan akan mengalami panas berlebihan dan bisa merusak papan. Rentang sumber tegangan yang dianjurkan adalah 7 Volt sampai 12 Volt. Pin tegangan yang tersedia pada Arduino mega2560 adalah sebagai berikut :

- VIN : Adalah *input* tegangan untuk Arduino ketika menggunakan sumber daya *eksternal* (sebagai pengganti tegangan 5 Volt dari koneksi USB atau sumber daya dari regulator lainnya). Dapat diberikan tegangan melalui pin ini, atau jika

memasok tegangan untuk arduino melalui *jack power*, bisa mengakses/mengambil tegangan melalui pin ini.

- 5V : Sebuah pin yang mengeluarkan tegangan 5 Volt, dari pin ini tegangan sudah diatur dari regulator yang tersedia (*built-in*) pada papan. Arduino dapat diaktifkan dengan sumber daya baik berasal dari *jack power* DC (7-12 Volt), konektor USB (5 Volt), atau pin VIN pada *board* (7-12 Volt). Memberikan tegangan melalui pin 5V atau 3.3V secara langsung tanpa melewati regulator dapat merusak Arduino.
- 3V3 : Sebuah pin yang menghasilkan tegangan 3,3 Volt. Tegangan ini dihasilkan oleh regulator yang terdapat pada arduino (*on-board*). Arus maksimum yang dihasilkan adalah 50 mA.
- GND : Pin *Ground* atau *Massa*.
- IOREF : Pin ini pada papan Arduino berfungsi untuk memberikan referensi tegangan yang beroperasi pada mikrokontroler. Sebuah perisai (*shield*) dikonfigurasi dengan benar untuk dapat membaca pin tegangan IOREF dan memilih sumber daya yang tepat atau mengaktifkan penerjemah tegangan (voltage translator) pada *output* untuk bekerja pada tegangan 5 Volt atau 3,3 Volt.

2.6.2 Memori

Arduino mega2560 memiliki 256 KB *flash memory* untuk menyimpan kode (yang 8 KB digunakan untuk *bootloader*), 8 KB SRAM dan 4 KB EEPROM (yang dapat dibaca dan ditulis dengan perpustakaan EEPROM).

2.6.3 *Input* dan *Output*

Masing-masing dari 54 digital pin pada Arduino Mega dapat digunakan sebagai *input* atau *output*, menggunakan fungsi pinMode() , digitalWrite() , dan *digitalRead()*. Arduino Mega beroperasi pada tegangan 5 volt. Setiap pin dapat memberikan atau menerima arus maksimum 40 mA dan memiliki resistor *pull-up internal* (yang terputus secara *default*) sebesar 20-50 kOhms. Selain itu, beberapa pin memiliki fungsi khusus, antara lain :

- Serial : 0 (RX) dan 1 (TX); Serial 1 : 19 (RX) dan 18 (TX); Serial 2 : 17 (RX) dan 16 (TX); Serial 3 : 15 (RX) dan 14 (TX). Digunakan untuk menerima (RX) dan mengirimkan (TX) data serial TTL. Pins 0 dan 1 juga terhubung ke pin chip ATmega16U2 Serial USB-to-TTL.
- *Eksternal* Interupsi : Pin 2 (*interrupt* 0), pin 3 (*interrupt* 1), pin 18 (*interrupt* 5), pin 19 (*interrupt* 4), pin 20 (*interrupt* 3), dan pin 21 (*interrupt* 2). Pin ini dapat

dikonfigurasi untuk memicu sebuah interupsi pada nilai yang rendah, meningkat atau menurun, atau perubah nilai.

- SPI : Pin 50 (MISO), pin 51 (MOSI), pin 52 (SCK), pin 53 (SS). Pin ini mendukung komunikasi SPI menggunakan perpustakaan SPI. Pin SPI juga terhubung dengan header ICSP, yang secara fisik kompatibel dengan Arduino Mega2560, Arduino Duemilanove dan Arduino Diecimila.
- LED : Pin 13. Tersedia secara *built-in* pada papan Arduino ATmega2560. LED terhubung ke pin digital 13. Ketika pin diset bernilai HIGH, maka LED menyala (ON), dan ketika pin diset bernilai LOW, maka LED padam (OFF).
- TWI : Pin 20 (SDA) dan pin 21 (SCL). Yang mendukung komunikasi TWI menggunakan perpustakaan *Wire*. Perhatikan bahwa pin ini tidak di lokasi yang sama dengan pin TWI pada Arduino Duemilanove atau Arduino Diecimila.

Arduino Mega2560 memiliki 16 pin sebagai analog *input*, yang masing-masing menyediakan resolusi 10 bit (yaitu 1024 nilai yang berbeda). Secara *default* pin ini dapat diukur/diatur dari mulai *Ground* sampai dengan 5 Volt, juga memungkinkan untuk mengubah titik jangkauan tertinggi atau terendah mereka menggunakan pin AREF dan fungsi *analogReference()*.

Ada beberapa pin lainnya yang tersedia, antara lain:

- AREF : Referensi tegangan untuk input analog. Digunakan dengan fungsi *analogReference().*
- RESET : Jalur LOW ini digunakan untuk me-*reset* (menghidupkan ulang) mikrokontroler. Jalur ini biasanya digunakan untuk menambahkan tombol *reset* pada *shield* yang menghalangi Arduino.

2.6.4 Komunikasi

Arduino Mega2560 memiliki kemampuan untuk berkomunikasi dengan komputer, Arduino lain, atau dengan mikrokontroler lainnya. Sebuah perpustakaan SoftwareSerial memungkinkan untuk komunikasi serial pada salah satu pin digital Mega2560. Arduino mega2560 juga mendukung komunikasi TWI dan SPI. Perangkat lunak Arduino termasuk perpustakaan Wire digunakan untuk menyederhanakan penggunaan bus TWI. Untuk komunikasi SPI, menggunakan perpustakaan SPI.

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 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, 1993). 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*).
3. Himpunan *fuzzy* diatur agar saling menumpuk.
4. Fungsi keanggotaan yang digunakan bentuk segitiga atau trapesium.

2.7.2 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).

2.7.3 Metode Inferensi Min-Max

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 Min–Max* aturan operasi minimum Mamdani digunakan untukimplikasi *fuzzy*. Persamaan aturan minimum adalah

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

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 Min-Max fungsi keanggotaan konsekuen dinyatakan dengan

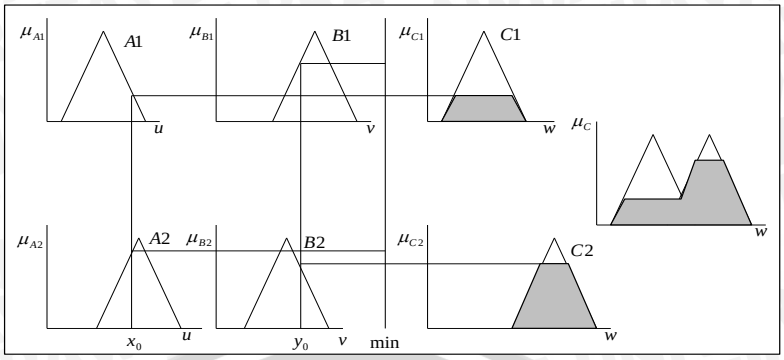
$$\mu_{C'}(w) = \mu_{c_1'} \vee \mu_{c_2'} = \left[\alpha_1 \wedge \mu_{C_1}(w)\right] \vee \left[\alpha_2 \wedge \mu_{C_2}(w)\right]$$

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 2.13.



Gambar 2.13 *Inferensi fuzzy* dengan metode Min-Max (Yan, Ryan dan Power, 1993)

2.7.4 Metode Defuzzifikasi *Center of Area*

Defuzzifikasi adalah proses untuk mendapatkan nilai numerik dari data *fuzzy* yang dihasilkan dari proses *inferensi* (Yan, 1994).

Proses defuzzifikasi dinyatakan sebagai berikut :

$$y_0 = defuzzifier(y)$$

Dengan

y = aksi kontrol *fuzzy*

y_0 = aksi kontrol *crisp*

$defuzzifier$ = operator *defuzzifikasi*

Metode *Center of Area* didefinisikan sebagai berikut :

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

Dengan

U = Keluaran

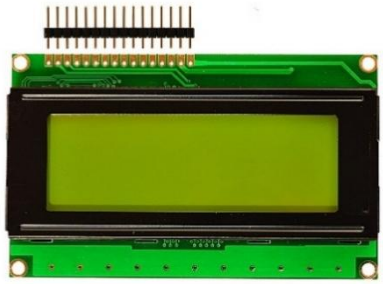
w_i = Bobot nilai benar w_i

u_i = Nilai linguistik pada fungsi keanggotaan keluaran

n = Banyak derajat keanggotaan

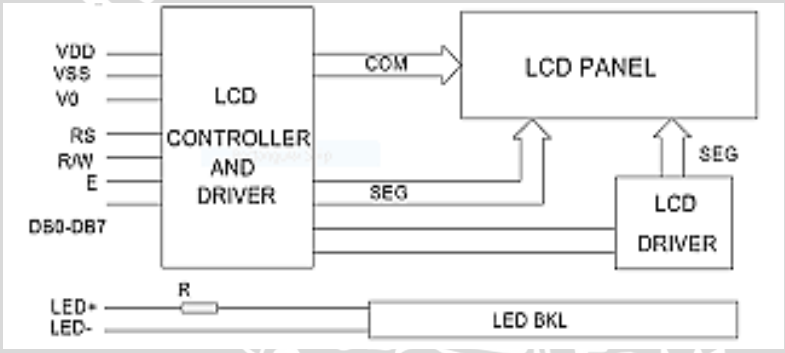
2.8 *Liquid Crystal Display* (LCD)

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 *Liquid crystal display* (LCD) ditunjukkan dalam Gambar 2.14.



Gambar 2.14 *Liquid crystal display* (LCD)
Sumber : *www.systronix.com*

Masukan modul *Liquid crystal display* (LCD) ini berupa bus data dan 3 sinyal kontrol yaitu RS, R/W dan E. Pengendali *dot* matriks *liquid crystal display* (LCD) dilakukan secara *internal* oleh kontroler yang sudah terpasang di dalam modul *liquid crystal display* (LCD). Diagram blok *liquid crystal display* (LCD) tipe QC1602A ditunjukkan dalam Gambar 2.15.



Gambar 2.15 Blok Diagram *liquid crystal display* (LCD) (*datasheet* QC1602A

Liquid crystal display (LCD) tipe QC1602A memiliki 16 pin koneksi antarmuka dimana setiap pin memiliki fungsi tertentu. Fungsi masing-masing pin ditunjukkan dalam Tabel 2.1.

Tabel 2.1 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

2.9 Heater

Alat pemanas (*heater*) ini digunakan sebagai penghangat air pada aktuator tangki air hangt. Pada perancangan sistem ini saya menggunakan *heater* eheim jager RH9000 dengan daya 75Watt yang mampu menghangatkan air dengan kapasitas maks 75 liter air. *Heater* RH9000 ini sudah dilengkapi kontrol *close loop* yang artinya *heater* mampu mengontrol suhu dalam tangki dengan pilihan suhu yang dapat diatur yaitu: 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, dan 32˚C. *Heater* yang digunakan ditunjukkan pada Gambar 2.16.

Gambar 2.16 Heater Ehem Jager
Sumber: *www.eheim.com*

2.10 *Valve (Stop Kran)*

Valve ini digunakan sebagai buka tutup aliran air yang digunakan sebagai aktuator. *Valve* di *couple* dengan motor *stepper* untuk dapat dibuka sesuai dengan *input step* yang diberikan. *Valve* yang digunakan dalam penelitian ini merupakan *stop kran* yang berukuran ½ dim atau berdiameter 1.25 cm yang ditunjukkan pada Gambar 2.17.

Gambar 2.17 *Stop Kran* ½ Dim
Sumber: *www.humijayatech.com*

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 Studi literatur
3.2 Perancangan sistem
 3.2.1 Blok diagram sistem
 3.2.2 Spesifikasi alat
 3.2.3 Prinsip kerja sistem
3.3 Perancangan perangkat keras
 3.3.1 Desain miniatur kolam
 3.3.2 Rangkaian catu daya
 3.3.3 *Couple* motor dengan *valve*
 3.3.4 Konfigurasi pin mikrokontroller
 3.3.5 Konfigurasi pin *driver* motor
 3.3.6 Konfigurasi pin *liquid crystal display* (LCD).
3.4 Perancangan perangkat lunak
 3.4.1 *Flowchart* program

3.1 Studi literatur

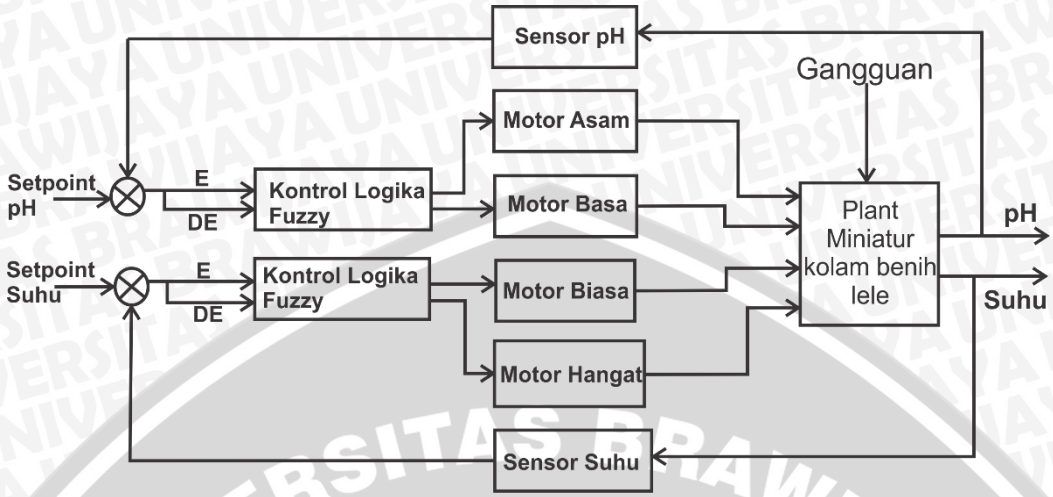
Studi literatur disini berfungsi agar dapat memahami komponen penyusun sistem, yang meliputi sensor suhu, sensor pH, motor *stepper*, Kontrol Logika *Fuzzy* (KLF), Arduino Mega2560.

3.2 Perancangan sistem

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

3.2.1 Blok diagram sistem

Diagram blok sistem yang dirancang ditunjukkan dalam Gambar 3.1



Gambar 3.1 Blok diagram sistem

Keterangan :

1. *Setpoint* sistem untuk pH 7.8 dan suhu 28˚C.
2. Kontroler yang digunakan adalah kontroler logika *fuzzy* yang diprogram ke dalam Arduino Mega2560.
3. *Input* kontroler logika *fuzzy* untuk pH dan suhu sama yaitu sinyal *error* dan *delta error*. *Output* kontroler logika *fuzzy* untuk pH dan suhu sama yaitu berupa sinyal kontrol untuk pH ditujukan pada motor asam dan basa, sedangkan untuk suhu ditujukan pada motor hangat dan biasa.
4. Aktuator yang digunakan adalah motor *stepper* yang di *couple* dengan *valve* yang berfungsi untuk membuka dan menutup aliran larutan asam dan basa.
5. Aktuator yang digunakan untuk suhu adalah motor *stepper* yang dipasangkan dengan *valve* yang berfungsi untuk membuka dan menutup aliran air hangat dan biasa.
6. Sebagai umpan balik digunakan sensor pH untuk mengukur kadar keasaman (pH) dan sensor suhu untuk mengukur suhu air pada *plant*.

3.2.2 Sepsifikasi alat.

Spesifikasi komponen-komponen sistem pengendalian kadar keasaman dan suhu menggunakan Arduino Mega2560 adalah sebagai berikut :

1. *Box* kolam yang digunakan terbuat dari bahan *plastic* dengan ukuran 48 cm x 33 cm x 30 cm.
2. *Volume* box kolam yang digunakan sebesar 43.56 liter.

3. Tangki aktuator yang digunakan sebesar 16 L untuk air asam, basa, dan biasa sedangkan untuk air hangat sebesar 10 L.
4. Perangkat kontrol yang digunakan adalah Arduino Mega 2560.
5. Sensor pH *SKUSEN0161* digunakan untuk mengukur kadar keasaman (pH)
6. Sensor suhu DS18B20 digunakan untuk mengukur suhu.
7. Aktuator yang digunakan untuk mengalirkan larutan asam atau basa adalah motor *stepper* 12 V 3A yang dipasangkan dengan *valve*.
8. Aktuator yang digunakan untuk mengalirkan air hangat dan air biasa adalah motor *stepper* 12 V 3A yang dipasangkan dengan *valve*.

3.2.3 Prinsip kerja sistem

Cara kerja sistem pengendalian suhu dan kadar keasaman (pH) pada kolam benih ikan lele menggunakan kontrol logika *fuzzy* sebagai berikut :

1. Sistem diberi catu daya. Catu daya sebesar 5 V digunakan untuk mencatu sensor dan rangkaian *driver* motor. Catu daya 12 V digunakan untuk mencatu motor *stepper* melalui *driver*.
2. Catu daya Arduino Mega2560 diambil dari komputer dengan menghubungkan *port* komunikasi antara Arduino dan komputer menggunakan perantara kabel USB dengan kecepatan transfer data sebesar 9600 *bit per second* (bps). Untuk pengendalian secara *realtime* catu daya yang digunakan adalah *regulator* dengan tegangan keluaran sebesar 9 V.
3. Kadar keasaman (pH) pada sistem diukur dengan menggunakan sensor pH SKU SEN0161 dan suhu diukur dengan sensor suhu DS18B20.
4. Sensor pH dihubungkan dengan rangkaian penguat agar keluaran sensor pH dapat dibaca oleh Arduino dengan rentang 0-5 V. Keluaran sensor suhu berupa sinyal digital sehingga dapat langsung dibaca oleh Arduino.
5. Sinyal keluaran rangkaian penguat sebagai masukan Arduino yang kemudian diproses menggunakan kontroler logika *fuzzy*.
6. Keluaran Arduino berupa sinyal *step* yang diberikan ke *driver* motor. *Driver* berfungsi penerus instruksi dari arduino sekaligus pengaman dari arus balik motor.
7. Motor *stepper* dipasangkan dengan *valve* yang berfungsi sebagai aktuator untuk pH dengan membuka aliran larutan asam dan basa, sedangkan untuk suhu dengan memebuka aliran air biasa dan air hangat.

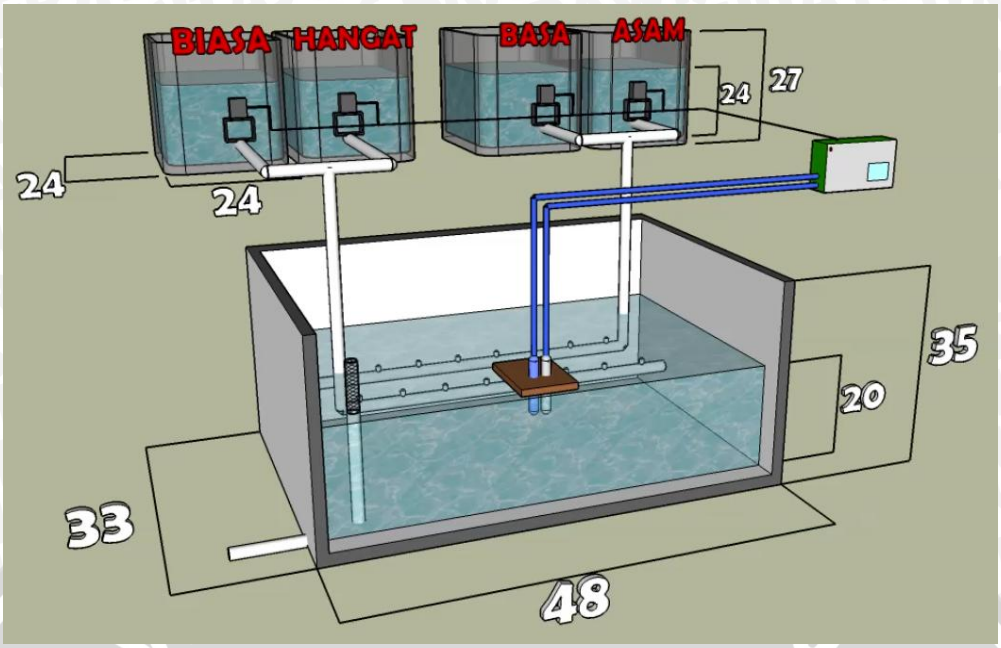
8. *Valve* asam akan mengalirkan larutan asam jika kadar keasaman (pH) pada *plant* terlalu basa atau pH > 7.8. Larutan asam yang digunakan adalah 12 ml pH *UP* dari *Fish-All* yang dicampur dengan 16 L air biasa dengan pH 6.9.
9. *Valve* basa akan mengalirkan larutan basa jika kadar keasaman (pH) pada *plant* terlalu asam atau pH < 7.8. Larutan basa yang digunakan adalah 12 ml pH *Down* dari *Fish-All* dicampur dengan 16 L air dengan pH 9.
10. *Valve* biasa akan mengalirkan air biasa (sumur) jika suhu pada plant terlalu panas atau suhu > 28. Air biasa yang digunakan adalah air sumur dengan suhuh 27.78 ˚C.
11. *Valve* hangat akan mengalirkan air hangat jika suhu pada *plant* terlalu dingin atau suhu < 28. Air hangat yang digunakan adalah air hasil pemanasan dari *heater* dengan suhu maksimum 32˚C.

3.3 Perancangan perangkat keras

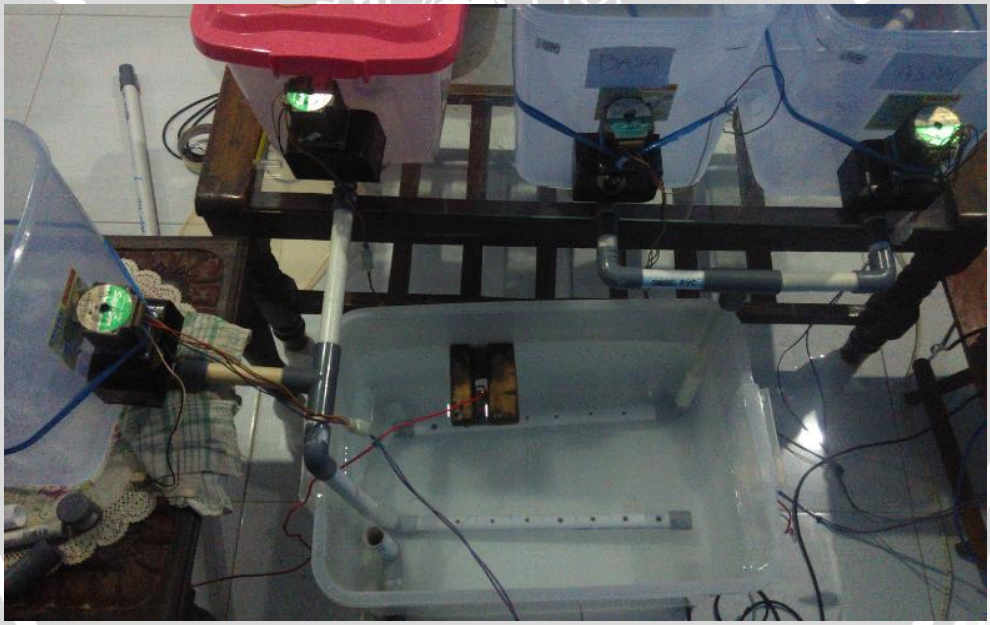
Pembuatan perangkat keras dilakukan sebagai langkah awal sebelum terbentuknya suatu sistem, hal ini bertujuan agar *output* pH dan suhu sesuai dengan masukan yang diinginkan dan sistem dapat bekerja dengan baik sesuai yang direncanakan.

3.3.1 Desain miniatur kolam

Desain miniatur kolam menggunakan 1 *box plastic* sebagai media kolam yang berukuran 48 cm x 33 cm x 30 cm dan 4 *Box plastic* sebagai tangki *aktuator*. Tangki untuk larutan asam, basa dan air biasa berukuran 24 cm x 24 cm x 24 cm dan tangki untuk air hangat berukuran 23 cm x 21.5 cm x 19 cm. Media kolam diisi air dengan volume 43.56 L dan tangki *aktuator* diisi air dengan masing-masing volume 13.8 L dan 9.4 L. Desain miniatur kolam ditunjukkan dalam Gambar 3.2 dan miniatur kolam ditunjukkan dalam Gambar 3.3.



Gambar 3.2 Desain Miniatur Kolam (dimensi dalam cm)



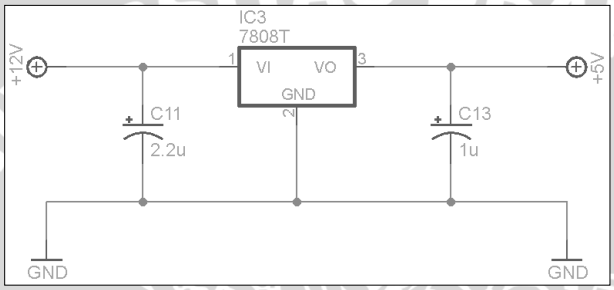
Gambar 3.3 Miniatur Kolam

Berdasarkan pada Gambar 3.2 dan 3.3, terlihat bahwa untuk menyebar / menyatukan air kolam dengan air dari tangki menggunakan pipa dengan lubang-lubang, lubang tersebut didesain dengan ukuran diameter semakin menjauhi dari sumber maka ukuran diameter lubang semakin besar. Pada miniatur juga dilengkapi *control level* manual dengan cara memasang pipa dengan tinggi 30 cm yang berfungsi sebagai pembatas tinggi air pada kolam apabila tinggi air melebihi 30 cm maka otomatis langsung terbuang melalui pipa ini.

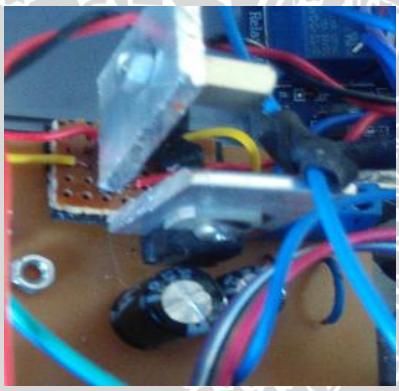
3.3.2 Rangkaian catu daya

Rangkaian catu daya yang digunakan adalah *positif regulator* dengan menggunakan IC 7805 yang memiliki tegangan keluaran sebesar 5 V dan *positif regulator* dengan menggunkan IC 7809 ang memiliki tegangan keluaran 9 V.

Rangkaian *positif regulator* dengan IC 7805 dan 7809. Rangkaian dengan IC 7805 digunakan untuk menurunkan tegangan dari 12 V menjadi 5 V. rangkaian dengan IC 7809 digunakan untuk menurunkan tegangan dari 12 V menjadi 9 V.Tegangan keluaran rangkaian dengan IC 7805 digunakan untuk mencatu driver motor, sensor, dan LCD. Tegangan keluaran rangkaian dengan IC 7809 digunakan untuk mencatu Arduino Mega2560. Rangkaian *positif regulator* ditunjukkan dalam Gambar 3.4.

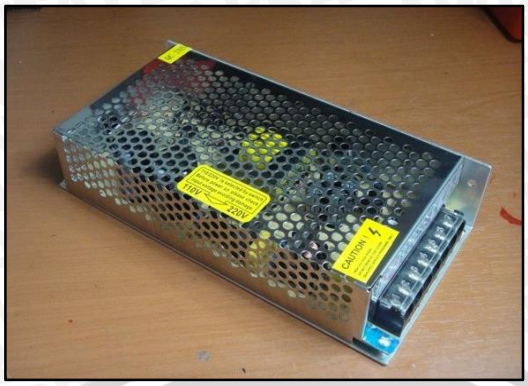


Gambar 3.4 Positif regulator (Perancangan)



Gambar 3.5 Penerapan IC 7805 dan IC 7809

Catu daya 12 volt yang digunakan sebagai sumber tegangan alat diambil dari *switching power supply* 12 V dengan arus maksimal sebesar 10 ampere. *Switching power supply* ditunjukkan dalam Gambar 3.6.



Gambar 3.6 *Switching Power Supply*
Sumber : *http://sfe-electronics.com/power-supply/2135-switching-ps-12v-10a*

Hasil pengukuran tegangan keluaran IC7805, IC 7809 dan *switching power supply* 12V 10A dengan menggunakan *multimeter* ditunjukkan pada Tabel 3.1 dibawah ini.

Tabel 3.1 Hasil pengukuran sumber catu daya dengan multimeter

No.	Sumber catu daya	Tegangan keluaran
1	IC 7805	4.94 V
2	IC 7809	9.12 V
3	*Switching power supply* 12V 10A	12.27 V

3.3.3 *Couple* motor dengan *valve*

Couple motor dengan *valve* berfungsi sebagai *aktuator* untuk membuka aliran air pada tangki. Diketahui bahwa *valve* membutuhkan 4.75 putaran untuk terbuka penuh dan motor *stepper* dalam satu *step* dengan *mode full-step* membentuk sudut 1.8˚ yang artinya untuk membentuk sudut 360˚ / 1 putaran motor *stepper* membutuhkan 200 *step*, jadi motor *stepper* membutuhkan 950 *step* untuk membuka *valve* secara penuh. *Couple* motor stepper dengan *valve* ditunjukkan pada Gambar 3.7.

Gambar 3.7 *Couple* Motor Stepper dengan *Valve*

3.3.4 Konfigurasi pin mikrokontroler

Sistem pengendalian kadar keasaman (pH) ini menggunakan Arduino Mega sebagai pusat pengolah utama dalam proses pengendalian. Konfigurasi *input*/*output* dari Arduino Mega ditunjukkan dalam Tabel 3.2.

Tabel 3.2 Konfigurasi pin Arduino Mega2560

No.	Pin	Fungsi
1	A8	Sensor pH
2	D14	Sensor Suhu
3	D13 dan D12	*Driver* motor asam
4	D11 dan D10	*Driver* motor basa
5	D6 dan D7	*Driver* motor biasa
6	D8 dan D9	*Driver* motor hangat
7	D42	RS.LCD
8	D44	E.LCD
9	D46	DB4.LCD
10	D48	DB5.LCD
11	D50	DB6.LCD
12	D52	DB7.LCD
13	5V	Catu untuk lcd, sensor dan *driver*
14	GND	*Ground* untuk lcd, sensor dan *driver*

3.1.1 Konfigurasi pin driver motor

Rangkaian *driver motor* yang digunakan adalah DRV8825. Satu DRV8825 digunakan untuk mengendalikan satu motor *stepper*. Motor *stepper* ini di *couple* dengan *valve*, motor *stepper* berfungsi untuk membuka dan menutup *valve* tangki air biasa, hangat, asam dan basa. *Mode* pengontrolan motor yang digunakan adalah jumlah *stepping* yang diberikan. Catu daya motor yang digunakan adalah catu daya *eksternal* 12 V. Konfigurasi pin *driver* motor ini ditunjukkan dalam Tabel 3.3.

Tabel 3.3 Konfigurasi pin *driver* motor *stepper*

No.	Pin Arduino	Pin *Driver*
1	D13	*Driver* dir asam
2	D12	*Driver step* asam
3	D11	*Driver* dir basa
4	D10	*Driver step* basa
5	D9	*Driver* dir biasa
6	D8	*Driver step* biasa
7	D7	*Driver* dir hangat
8	D6	*Driver step* hangat

3.1.2 Konfigurasi pin *liquid crystal display*

Liquid Crystal Display (LCD) digunakan untuk menampilkan suhu dan kadar keasaman (pH). Konfigurasi pin *Liquid Crystal Display* (LCD) ditunjukkan dalam Tabel 3.4.

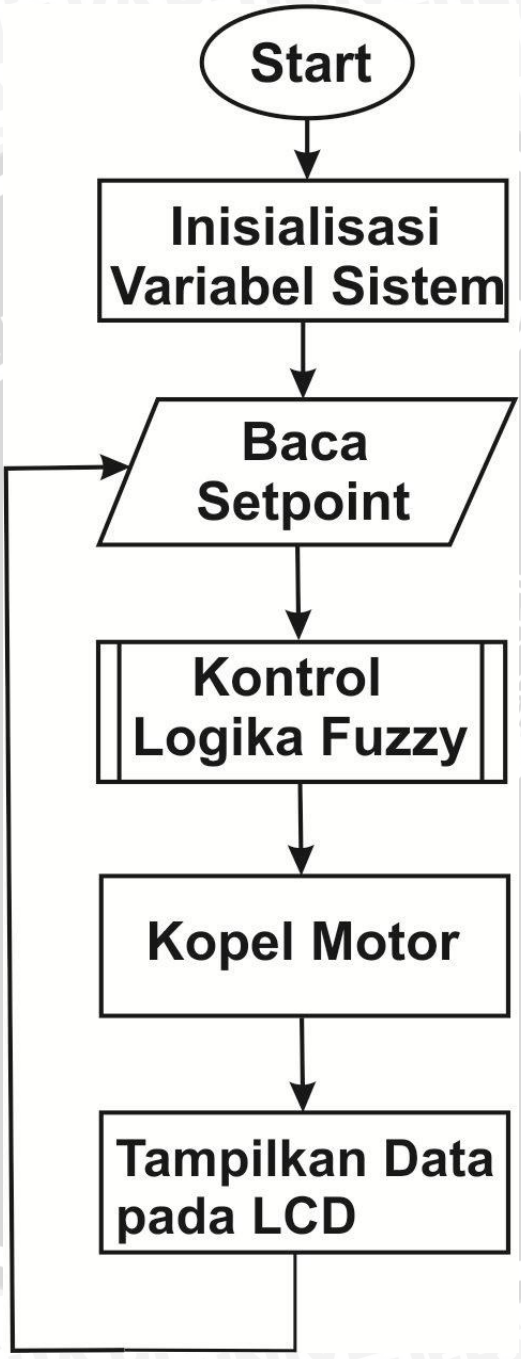
Tabel 3.4 Konfigurasi pin *Liquid Crystal Dislay* (LCD)

No.	Pin Arduino	Pin lcd
1	D42	RS.LCD
2	D44	E.LCD
3	D46	DB4.LCD
4	D48	DB5.LCD
5	D50	DB6.LCD
6	D52	DB7.LCD
7	5V	Vdd
8	gnd	Vss
9	Resistor	V0
10	gnd	R/W

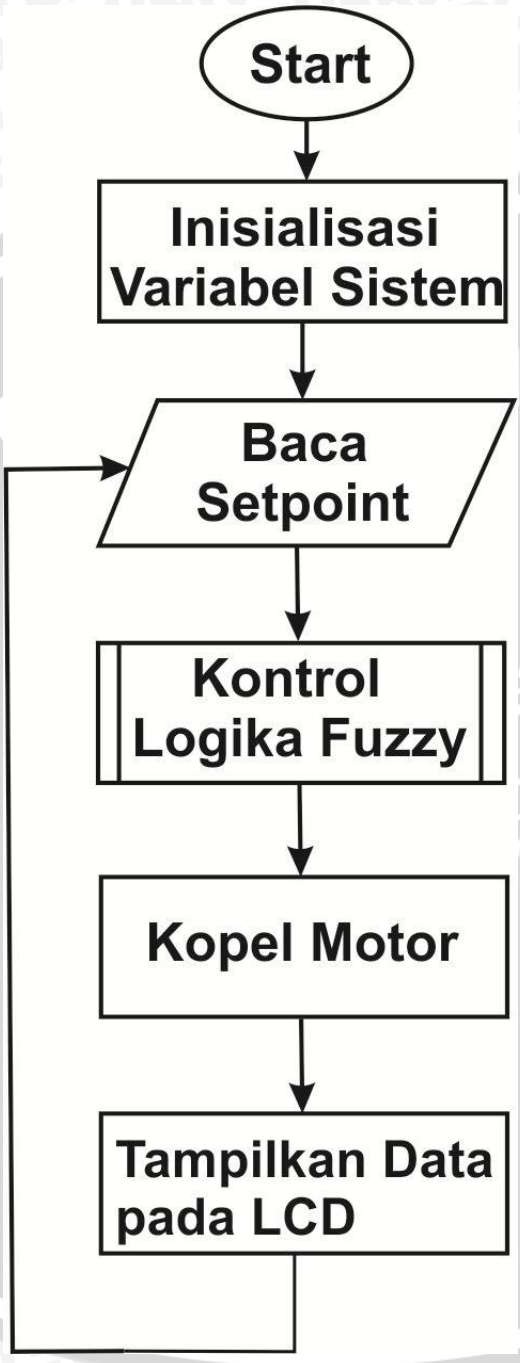
3.1 Perancangan Perangkat Lunak

3.1.1 *Flowchart* program

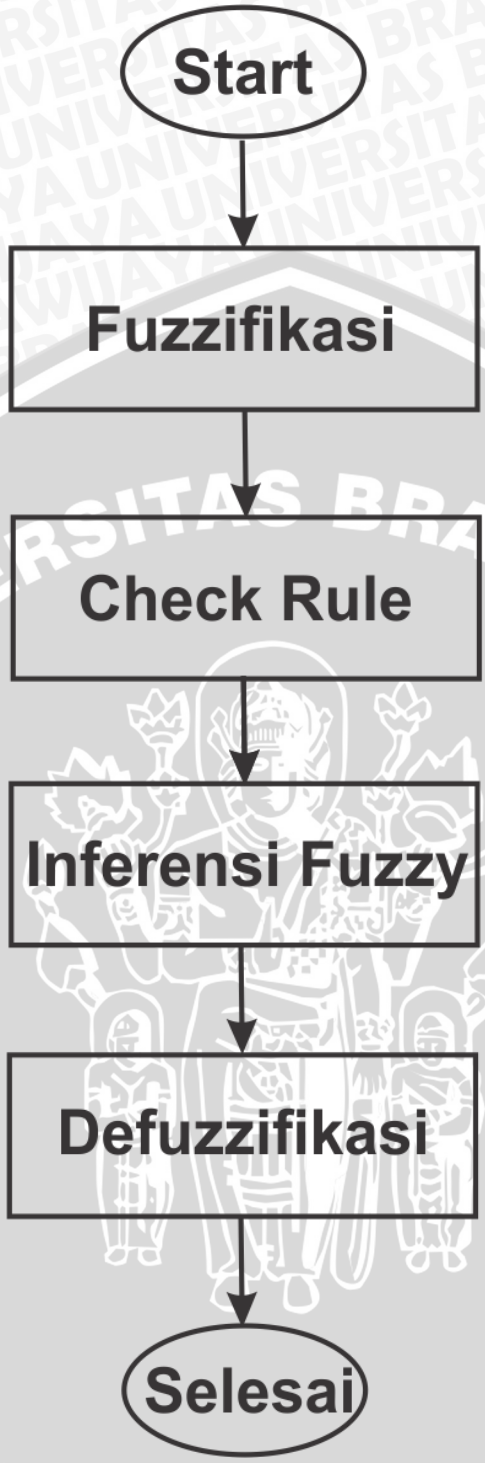
Flowchart program merupakan gambaran alur proses program yang dilakukan oleh kontroler pada saat implementasi. *Flowchart* program dalam skripsi ini dibagi menjadi dua yang ditunjukkan pada Gambar 3.8 dan Gambar 3.9. *Flowchart* kontrol logika *fuzzy* merupakan subrutin dari program utama yang ditunjukkan pada Gambar 3.10.



Gambar 3.8 *Flowchart* Keseluruhan Sistem pH



Gambar 3.9 *Flowchart* Keseluruhan Sistem Suhu



Gambar 3.10 *Flowchart* Keseluruhan Sistem Suhu

BAB IV
HASIL DAN PEMBAHASAN

Hasil dan pembahasan dilakukan untuk mengetahui apakah sistem telah bekerja sesuai dengan perancangan. Pengujian sistem dibagi menjadi dua bagian, yaitu :

4.1 Pengujian karakteristik setiap blok
- 4.1.1 Pengujian karakteristik sensor pH SKU SEN0161
- 4.1.2 Pengujian karakteristik sensor suhu DS18B20
- 4.1.3 Pengujian karakteristik *couple* motor dengan *valve*
- 4.1.4 Pengujian karakteristik *plant*

4.2 Desain kontroler

4.3 Pengujian keseluruhan sistem
- 4.3.1 Pengujian pH tanpa gangguan
- 4.3.2 Pengujian suhu tanpa gangguan
- 4.3.3 Pengujian pH dengan gangguan
- 4.3.4 Pengujian suhu dengan gangguan

4.4 Pengujian kelayakan sistem

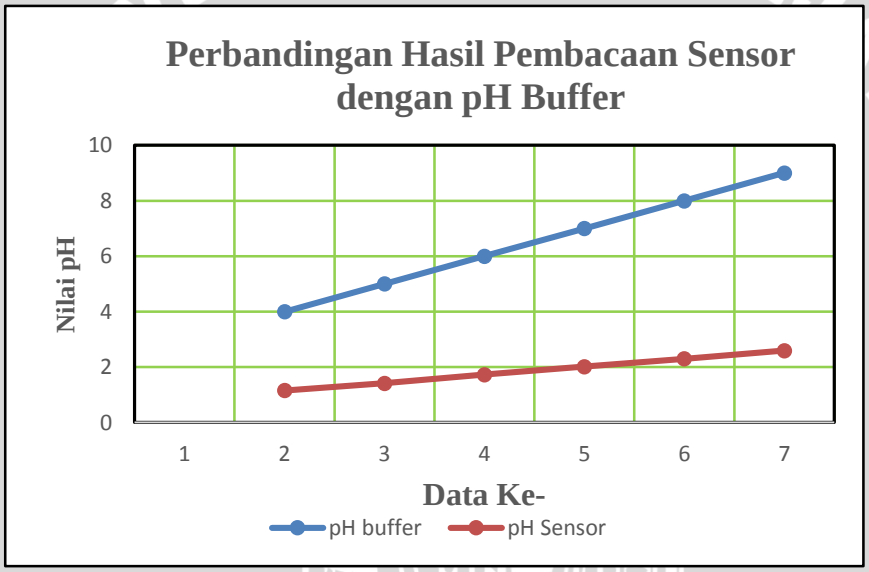
4.1 Pengujian Karateristik setiap blok

4.1.1 Pengujian karakteristik sensor pH SKU SEN0161

Tujuan dari karakteristik sensor pH SKU SEN0161 adalah mengetahui akurasi atau kemampuan pembacaan sensor pH terhadap pH *buffer*. pH *buffer* adalah larutan penyangga yang sudah diketahui nilai pH-nya. Hasil pengujian sensor pH SKU SEN0161 ditunjukkan dalam Tabel 4.1 dan Gambar 4.1.

Tabel 4.1 Hasil Pengujian Sensor pH SKU SEN0161 dengan pH *Buffer*

Larutan Uji	Pembacaan Sensor pH			Rata-rata Pengukuran Sensor pH	*Error* %
pH *buffer*	Pengukuran 1	Pengukuran 2	Pengukuran 3		
4	1.15	1.16	1.16	1.16	71.1
5	1.41	1.42	1.42	1.42	71.7
6	1.73	1.73	1.73	1.73	71.16
7	2.02	2.02	2.01	2.02	71.17
8	2.30	2.31	2.30	2.30	71.23
9	2.60	2.59	2.60	2.60	71.14
Error Rata-rata					71.25



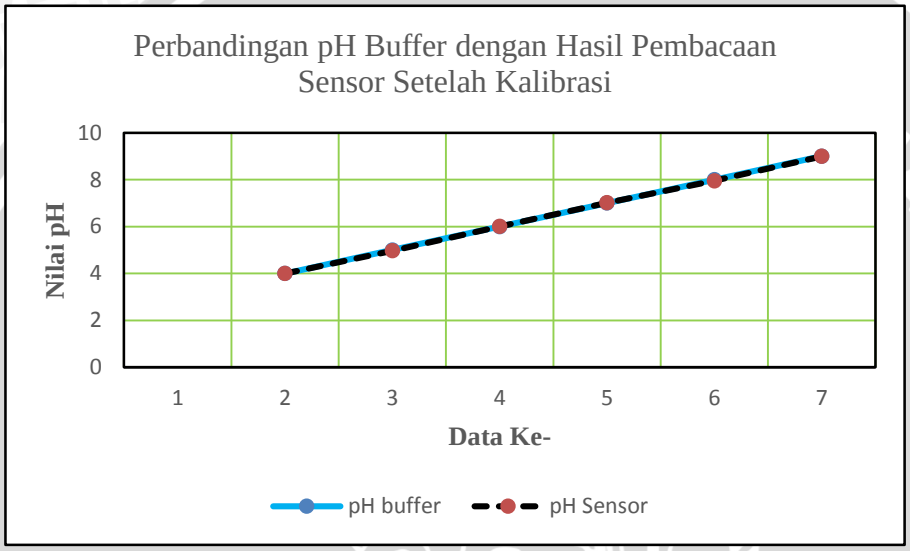
Gambar 4.1 Grafik Perbandingan Hasil Pembacaan Sensor pH dengan pH *Buffer*

Pada Tabel 4.1 pH *buffer* berguna sebagai acuan yang dibandingkan dengan pembacaan dari sensor pH. Hasil perbandingan tersebut berupa *error* yang dinyatakan dalam persen. Berdasarkan pengujian tersebut *error* rata-rata dari pembacaan sensor pH masih sangat besar yaitu 71.25 %.

Berdasarkan grafik pada Gambar 4.1 terlihat selisih antara pembacaan sensor pH dengan pH *buffer* membentuk garis *linear*, sehingga untuk mengurangi *error* dapat menggunkan cara kalibrasi dengan menggunakan *regresi linier*. Hasil *regresi linier* dilakukan pada program arduino untuk mengurangi *error* pembacaan sensor pH. Hasil kalibrasi sensor pH dapat dilihat pada Tabel 4.2 dan Gambar 4.2.

Tabel 4.2 Hasil Pengujian Sensor pH dengan pH *Buffer* setelah Kalibrasi

Larutan Uji	Pembacaan Sensor pH			Rata-rata	*Error %*
pH *buffer*	Pengukuran 1	Pengukuran 2	Pengukuran 3	Pengukuran Sensor pH	
4	4.00	3.96	4.06	4.01	0.2
5	5.00	4.94	5.00	4.98	0.4
6	5.97	6.04	6.04	6.02	0.32
7	7.01	7.05	6.98	7.01	0.2
8	7.95	7.85	8.02	7.94	0.73
9	8.96	8.99	9.03	8.99	0.09
Error Rata-rata					0.32



Gambar 4.2 Perbandingan pembacaan sensor pH setelah kalibrai

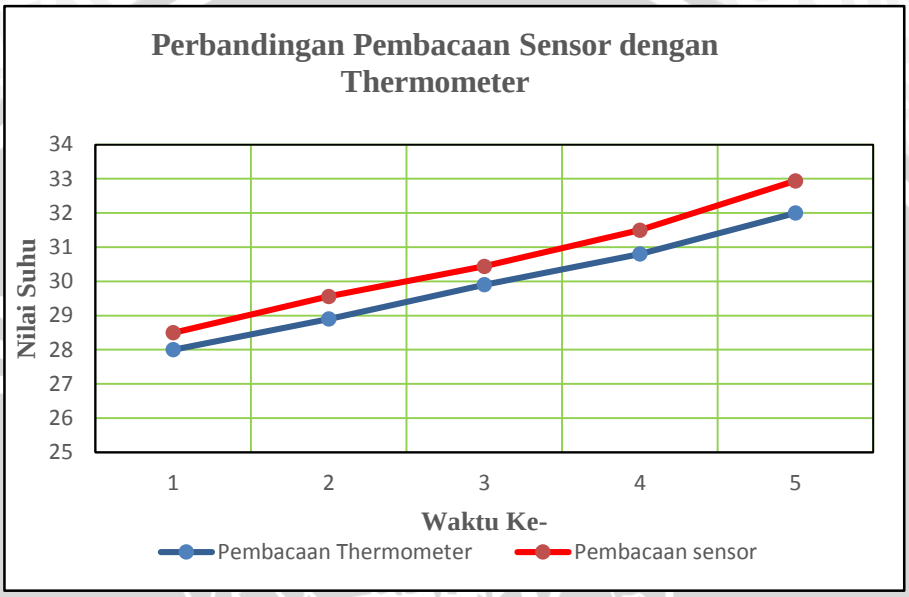
Berdasarkan hasil kalibrasi sensor pH SKU SEN0161 menghasilkan pH pembacaan yang mendekati pH acuan (pH *buffer*) dengan *error* rata-rata sebesar 0,32% yang dapat dilihat pada Tabel 4.2 dan Gambar 4.2, sehingga kalibrasi yang dilakukan telah sesuai dan sensor pH SKU SEN0161 dapat bekerja dengan baik.

4.1.2 Karakteristik sensor suhu DS18B20

Tujuan dari karakteristik sensor suhu DS18B20 adalah mengetahui akurasi atau kemampuan pembacaan sensor suhu terhadap *thermometer*. Hasil pengujian sensor suhu DS18B20 ditunjukkan dalam Tabel 4.3 dan Gambar 4.3.

Tabel 4..3 Hasil Pengujian Sensor Suhu DS18B20 dengan *Thermometer*

No.	Pembacaan *Thermometer*	Pembacaan sensor	*Error %*
1	28	28.5	1.79
2	28.9	29.56	2.28
3	29.9	30.44	1.81
4	30.8	31.5	2.27
5	32	32.94	2.94
Error Rata-rata			2.22



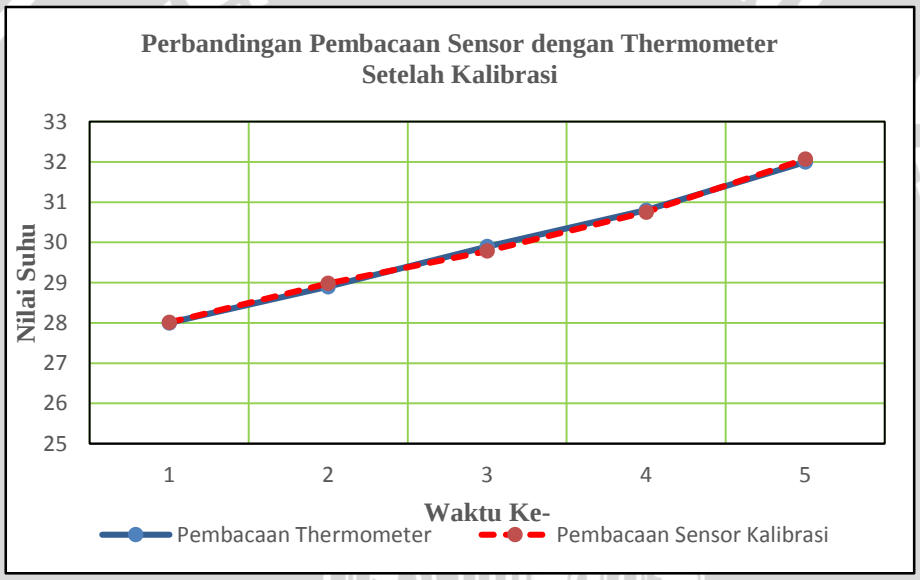
Gambar 4.3 Perbandingan Hasil Pembacaan Sensor Suhu dengan *Thermometer*

Pada Tabel 4.3 *Thermometer* sebagai acuan yang dibandingkan dengan pembacaan dari sensor suhu. Hasil perbandingan tersebut berupa *error* yang dinyatakan dalam *persen*. Berdasarkan pengujian tersebut *error* rata-rata dari pembacaan sensor pH masih besar yaitu 2.22 %.

Berdasarkan grafik pada Gambar 4.3 terlihat selisih antara pembacaan sensor pH dengan pH *buffer* membentuk garis *linear*, sehingga untuk mengurangi *error* dapat menggunkan cara kalibrasi dengan menggunakan *regresi linier*. Hasil *regresi linier* dilakukan pada program arduino untuk mengurangi error pembacaan sensor suhu. Hasil kalibrasi sensor suhu dapat dilihat pada Tabel 4.4 dan Gambar 4.4

Tabel 4.4 Hasil pengujian sensor suhu dengan *thermometer* setelah kalibras

No.	Pembacaan *Thermometer*	Pembacaan sensor	*Error %*
1	28	28.01	0.04
2	28.9	28.98	0.28
3	29.9	29.79	0.37
4	30.8	30.75	0.16
5	32	32.07	0.22
Error Rata-rata			0.21



Gambar 4.4 Perbandingan pembacaan sensor suhu setelah kalibrasi.

Berdasarkan hasil kalibrasi sensor suhu DS18B20 menghasilkan suhu pembacaan yang mendekati suhu acuan (*Thermometer*) dengan *error* rata-rata sebesar 0,21% yang dapat dilihat pada Tabel 3.4 dan Gambar 3.5, sehingga kalibrasi yang dilakukan telah sesuai dan sensor suhu DS18B20 dapat bekerja dengan baik.

4.1.3 Karakteristik *couple* motor dengan *valve*

Tujuan dari karakteristik *couple* motor dengan *valve* adalah mengetahui hubungan *step* motor terhadap terbukanya *valve* dan mengetahui hubungan *step* terhadap perubahan suhu dan pH. Hasil pengujian *step* motor terhadap *valve* ditunjukkan dalam Tabel 4.6 dan hasil pengujian *step* terhadap perubahan suhu dan pH ditunjukkan dalam Tabel 4.7, Tabel 4.8, Tabel 4.9, dan Tabel 4.10. Pada Tabel 4.5 adalah hasil pengukuran *output* tegangan

dari *driver* yang diukur menggunakan multimeter dengan tujuan mengetahui pengaruh perubahan *step* terhadap tegangan keluaran *driver*.

Karakteristik *valve* yang perlu diketahui bahwa *valve* membutuhkan 4.75 putaran untuk terbuka penuh dan motor *stepper* dalam satu *step* membentuk sudut 1.8˚ yang artinya untuk membentuk sudut 360˚ / 1 putaran motor *stepper* membutuhkan 200 *step*, jadi motor *stepper* membutuhkan 950 *step* untuk membuka *valve* secara penuh

Tabel 4.5 Hasil Pengukuran *Output* Driver dengan Multimeter

Jumlah Step	Output Driver Perkoil
0	0.00
100	4.85
200	4.87
300	4.83
400	4.86
500	4.86
600	4.87
700	4.86
800	4.87
900	4.87
950	4.86

Berdasarkan Tabel 4.5 Hasil pengukuran *output driver* dengan menggunakan *multimeter* menunjukkan dengan *step* berapa *output* tegangan dari *driver realtif* sama dapat diartikan bahwa jumlah *step* tidak mempengaruhi *output* tegangan *driver*. Perlu diketahui bahwa *driver* telah diset untuk *current limit* sebesar 1 A/koil, maka arus maksimal yang dikeluarkan *driver* sebesar 1 A/koil.

Tabel 4.6 Hasil Pengujian Step Motor terhadap *Valve*

Jumlah Step	Persentase kran terbuka %	Volume Air mL/20s
0	0.00	0
50	5.26	0
100	10.53	0
150	15.79	100
200	21.05	400
250	26.32	1000
300	31.58	1400
350	36.84	1800
400	42.11	2000
450	47.37	2100
500	52.63	2200
550	57.89	3400
600	63.16	4200
650	68.42	5000
700	73.68	5800
750	78.95	6600
800	84.21	7200
850	89.47	7800
900	94.74	8600
950	100.00	9200

Berdasarkan Tabel 4.6 Hasil pengujian *step* motor terhadap *valve* menunjukkan dengan perubahan *step* yang semakin besar maka *presentase valve* terbuka juga semakin besar dan ketika semakin besar *valve* terbuka maka *volume* air yang dapat dialirkan juga semakin besar. Berdasarkan hasil pengujian tersebut dapat disimpulkan bahwa *step* dengan presentase *valve* terhubung secara *linier*.

Tabel 4.7 Hasil pengujian *step* terhadap suhu aktuator biasa

Uji *Step* Terhadap Perubahan Suhu Aktuator Biasa				
Step	**Waktu**	**Suhu Awal**	**Suhu Akhir**	**Delta Suhu**
100	20s	33.86	33.86	0
200	20s	33.86	33.8	0.06
300	20s	33.8	33.54	0.26
400	20s	33.5	33.2	0.3
500	20s	33.18	32.87	0.31
600	20s	32.87	32.54	0.33
700	20s	32.5	32.15	0.35
800	20s	32.15	31.76	0.39
900	20s	31.76	31.33	0.43

Tabel 4.8 Hasil pengujian *step* terhadap suhu aktuator hangat

Uji *Step* Terhadap Perubahan Suhu Aktuator Hangat				
Step	**waktu**	**Suhu Awal**	**Suhu Akhir**	**Delta Suhu**
100	20s	28.04	28.04	0
200	20s	28.04	28.16	0.12
300	20s	28.16	28.35	0.19
400	20s	28.35	28.59	0.24
500	20s	28.59	28.84	0.25
600	20s	28.59	28.9	0.31
700	20s	28.66	29.09	0.43
800	20s	28.78	29.4	0.62
900	20s	28.97	29.77	0.8

Berdasarkan Tabel 4.7 dan Tabel 4.8 Hasil pengujian *step* motor terhadap perubahan suhu dengan aktuator air biasa dan air hangat menunjukkan bahwa perubahan *step* dengan waktu 20s waktu untuk *valve* terbuka terjadi perubahan suhu yang dapat diamati perubahannya dari suhu awal ke suhu akhir dengan *delta* suhu sebagai hasil perubahan tersebut.

Tabel 4.9 Hasil pengujian *step* terhadap pH aktuator asam

Uji *Step* Terhadap Perubahan pH Aktuator Asam				
Step	**Waktu**	**pH Awal**	**pH Akhir**	**Delta pH**
100	20s	7.63	7.63	0
200	20s	7.63	7.49	0.14
300	20s	7.36	7.16	0.2
400	20s	7.16	6.8	0.36
500	20s	6.72	6.32	0.4
600	20s	6.32	5.81	0.51
700	20s	5.81	5.39	0.42
800	20s	5.39	5.04	0.35
900	20s	5.04	4.71	0.33

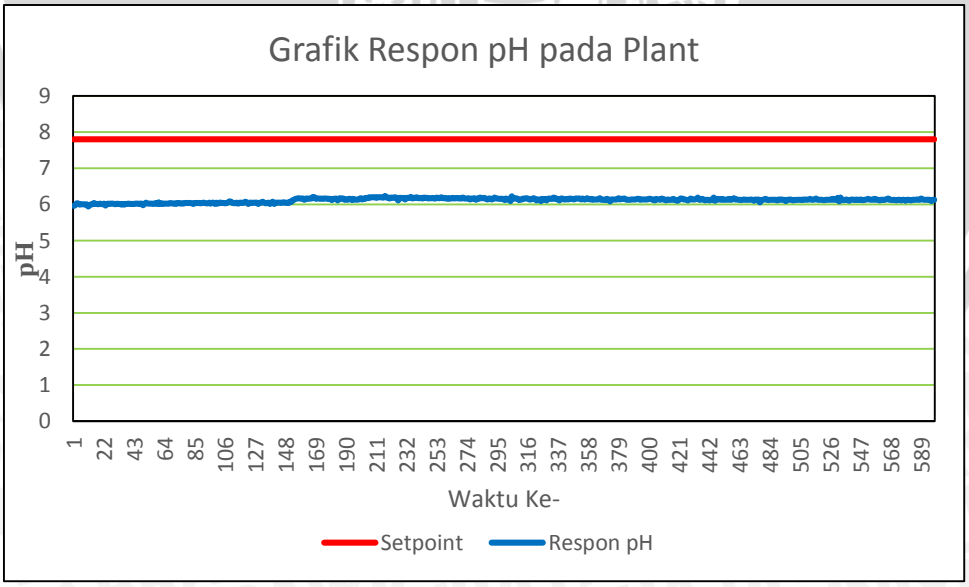
Tabel 4.10 Hasil pengujian *step* terhadap pH aktuator basa

Uji *Step* Terhadap Perubahan pH Aktuator Basa				
Step	**Waktu**	**pH Awal**	**pH Akhir**	**Delta pH**
100	20s	4.94	4.94	0
200	20s	4.94	4.94	0
300	20s	4.94	4.97	0.03
400	20s	4.97	5.02	0.05
500	20s	5.02	5.09	0.07
600	20s	5.09	5.22	0.13
700	20s	5.22	5.37	0.15
800	20s	5.37	5.64	0.27
900	20s	5.47	5.84	0.37

Berdasarkan Tabel 4.9 dan Tabel 4.10 Hasil pengujian *step* motor terhadap perubahan pH dengan aktuator air asam dan air basa menunjukkan bahwa perubahan *step* dengan waktu 20s waktu untuk *valve* terbuka terjadi perubahan pH yang dapat diamati perubahannya dari pH awal ke pH akhir dengan delta pH sebagai hasil perubahan tersebut.

4.1.4 Karakteristik *plant*

Pengujian *plant* ini dilakukan untuk mengetahui respon sistem secara keseluruhan tanpa melakukan pengendalian pH dan suhu. Hasil pengujian *plant* ditunjukkan dalam Gambar 4.5 dan Gambar 4.6.



Gambar 4.5 Grafik respon pH pada pengujian plant

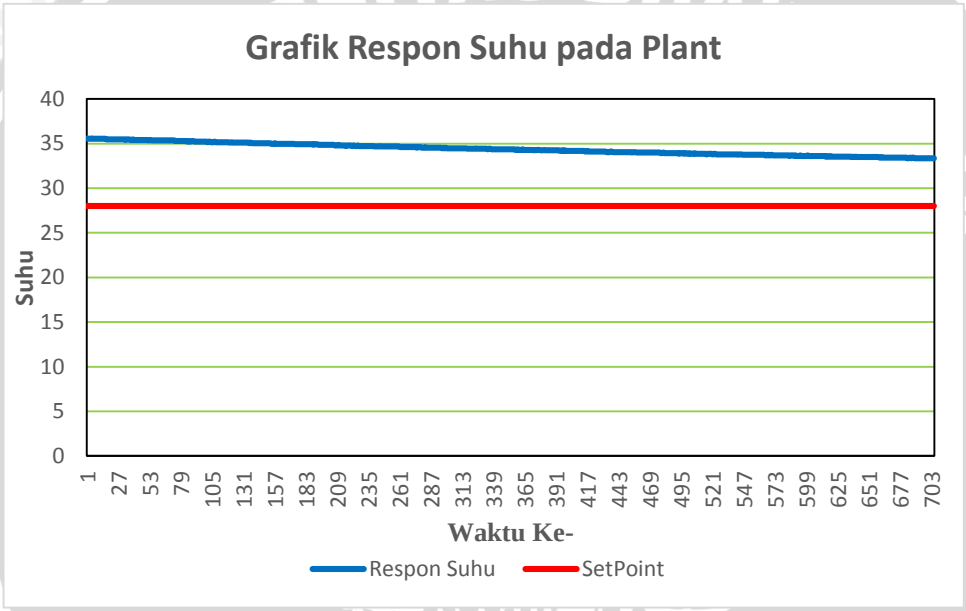
Berdasarkan grafik yang ditunjukkan dalam Gambar 4.5, *respon* pH masih belum mencapai nilai *setpoint* yang ditentukan. Nilai *error* yang terjadi adalah sebagai berikut.

$$\%Error\ max = \frac{|maximum\ pH\ reading - setpoint|}{setpoint} x100\%$$

$$\%Error\ \max = \frac{|5.7 - 7.8|}{7.8}\ x\ 100\%$$

$$\%Error\ max = \ 26.92\%$$

Error maksimum yang dimiliki *plant* sebesar 26.92 %. *Error* maksimum ini digunakan sebagai acuan untuk menentukan batas nilai *member function input error* dan *delta error* pH.



Gambar 4.6 Grafik *respon* suhu pada pengujian *plant*

Berdasarkan grafik yang ditunjukkan dalam Gambar 4.6, *respon* suhu masih belum mencapai nilai *setpoint* yang ditentukan. Nilai *error* yang terjadi adalah sebagai berikut.

$$\%Error\ max = \frac{|maximum\ suhu\ reading - setpoint|}{setpoint} x100\%$$

$$\%Error\ \max = \frac{|35.6 - 28|}{28}\ x\ 100\%$$

$$\%Error\ max = \ 27.14\%$$

Error maksimum yang dimiliki *plant* sebesar 27.14 %. *Error* maksimum ini digunakan sebagai acuan untuk menentukan batas nilai *member function input error* dan *delta error* suhu.

4.2 Desain kontrol logika *fuzzy*

Kontrol logika *fuzzy* yang dikembangkan dalam penelitian ini mempunyai masing-masing dua *crisp input* untuk pH dan suhu yaitu *error* dan *delta error*. Serta masing-masing dua *crisp output* untuk pH dan suhu. *Output* berupa sinyal *step* untuk aktuator pH yaitu motor untuk membuka *valve* air asam dan basa ,sedangkan untuk aktuator suhu yaitu motor air hangat dan air biasa. *Error* dan *delta error* didefinisikan dalam persamaan :

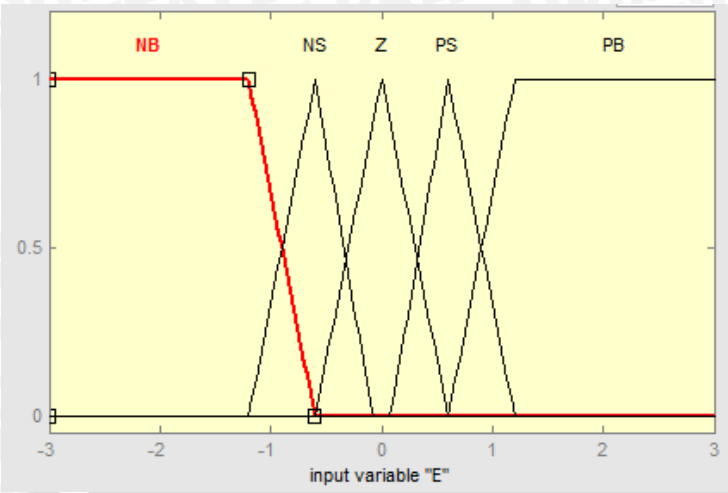
Error = *SP-PV*(*t*)
SP = *Setpoint*
PV(*t*) = *Present Value* pada waktu *t*
Delta error = *Error(t)* – *Error*(*t-1*)
Error(*t*) = *Error* pada waktu *t*
Error(*t-1*) = *Error* pada waktu *t-1*

Kontrol logika *fuzzy* pH dan suhu ini menggunakan 5 fungsi keanggotaan masukan dan 5 fungsi keanggotaan keluaran. 5 fungsi keanggotaan masukan dan 5 fungsi keanggotaan keluaran. Metode KLF yang digunakan pada penelitian ini adalak metode mamdani dengan inferensi Min-Max. Inferensi Min-Max ini menggunakan implikasi fungsi min dan agregasi fungsi max.

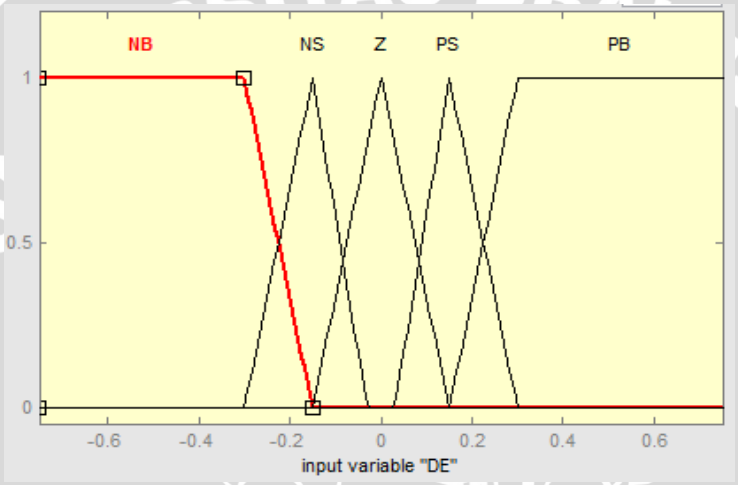
Fungsi keanggotaan masukan pH yaitu *error* dan *delta error* dengan 5 fungsi keanggotaan yang didefinisikan :

NB = *Negatif Big* (PV > SV)
NS = *Negatif small* (PV > SV)
Z = Zero (PV = SV)
PS = *Positif Small* (PV < SV)
PB = *Positif Big* (PV < SV)

Fungsi keanggotaan masukan *error* dan *delta error* pH ditunjukkan dalam Gambar 4.7 dan Gambar 4.8.

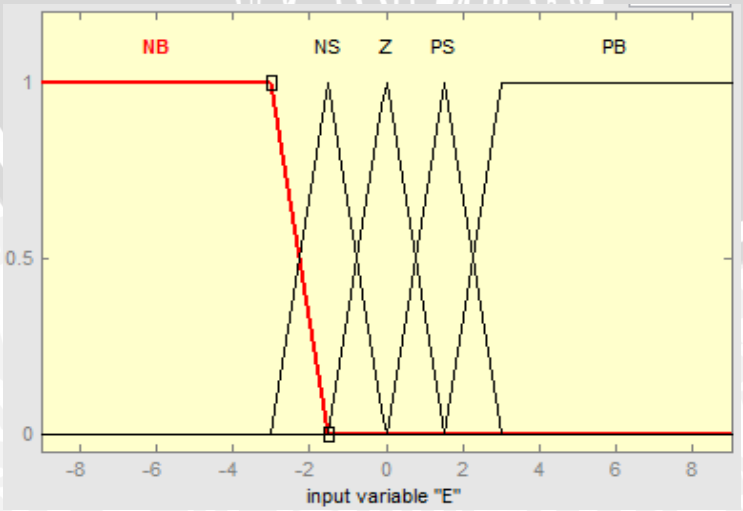


Gambar 4.7 Fungsi keanggotaan masukkan *error* pH

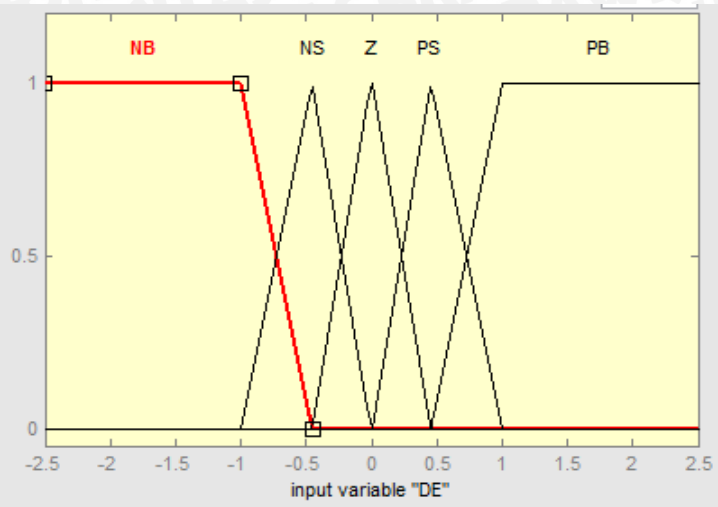


Gambar 4.8 Fungsi keanggotaan masukkan *delta error* pH

Fungsi keanggotaan masukan suhu yaitu *error* dan *delta error* dengan 5 fungsi keanggotaan yaitu NB, NS, Z, PS, PB yang definisinya sama dengan fungsi keanggotaan pH dapat dilihat pada Gambar 4.9 dan Gambar 4.10.



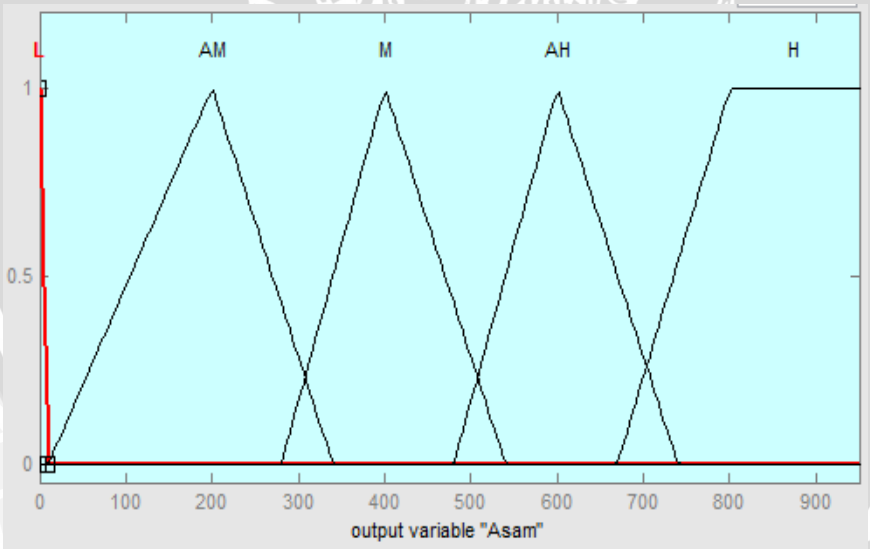
Gambar 4.9 Fungsi keanggotaan masukkan *error* Suhu



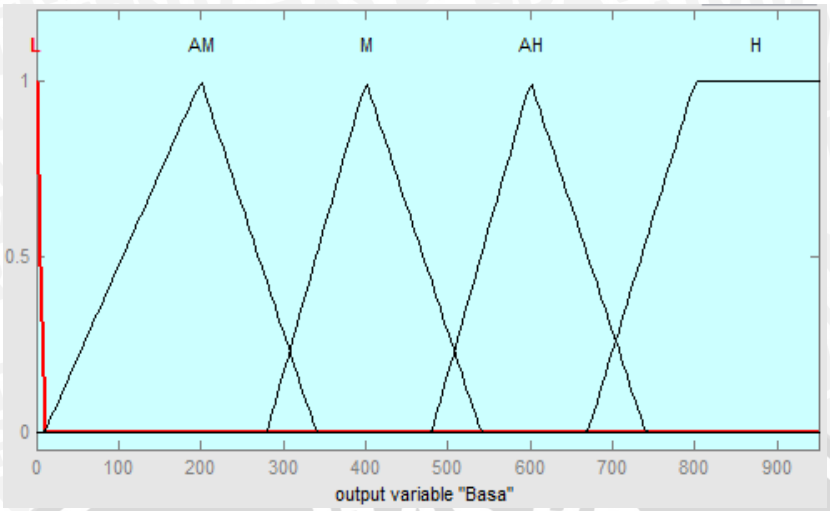
Gambar 4.10. Fungsi keanggotaan masukkan *delta error* Suhu

Fungsi keanggotaan keluaran merupakan representasi *output step* yang dikeluarkan oleh Arduino Mega berupa hasil perhitungan metode defuzzifikasi *Center of Area*. 5 Fungsi keanggotaan keluaran pH dan suhu memiliki definisi yang sama, yaitu : L, AM, M, AH, H.

Fungsi keanggotaan keluaran pH ditujukan untuk menjalankan *aktuator* yaitu motor untuk membuka *valve* air asam dan basa yang ditunjukkan gambar 4.11 dan 4.12.

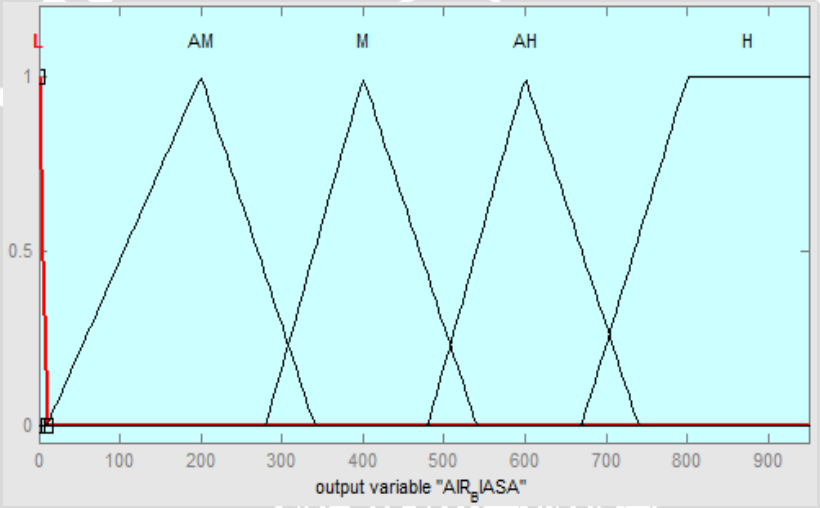


Gambar 4.11. Fungsi keanggotaan keluaran motor asam

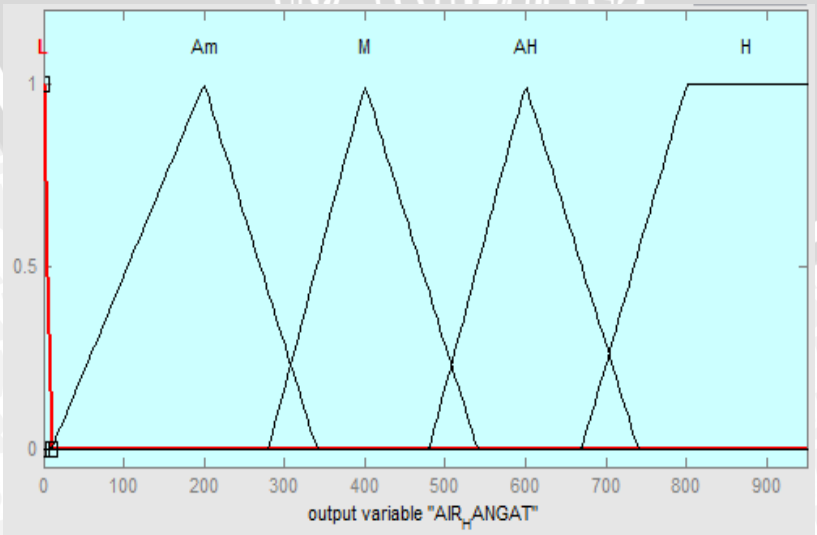


Gambar 4.12. Fungsi keanggotaan keluaran motor basa

Fungsi keanggotaan keluaran suhu ditujukan untuk menjalankan aktuator yaitu motor untuk membuka *valve* air biasa dan hangat yang ditunjukkan Gambar 4.13 dan 4.14.



Gambar 4.13. Fungsi keanggotaan keluaran motor biasa



Gambar 4.14. Fungsi keanggotaan keluaran motor hangat

Berikut ini merupakan aturan *fuzzy* yang digunakan ditunjukkan dalam Tabel 4.11 dan Tabel 4.12.

Tabel 4.11 *Aturan Fuzzy* untuk pH

DELTA ERROR \ ERROR	NB	NS	Z	PS	PB
NB	H / L	AH / L	L / AM	L / AM	L / AH
NS	H / L	M / L	L / L	L / AM	L / AH
Z	H / L	M / L	L / L	L / M	L / H
PS	AH / L	AM / L	L / L	L / M	L / H
PB	AH / L	AM / L	AM / L	L / AH	L / H

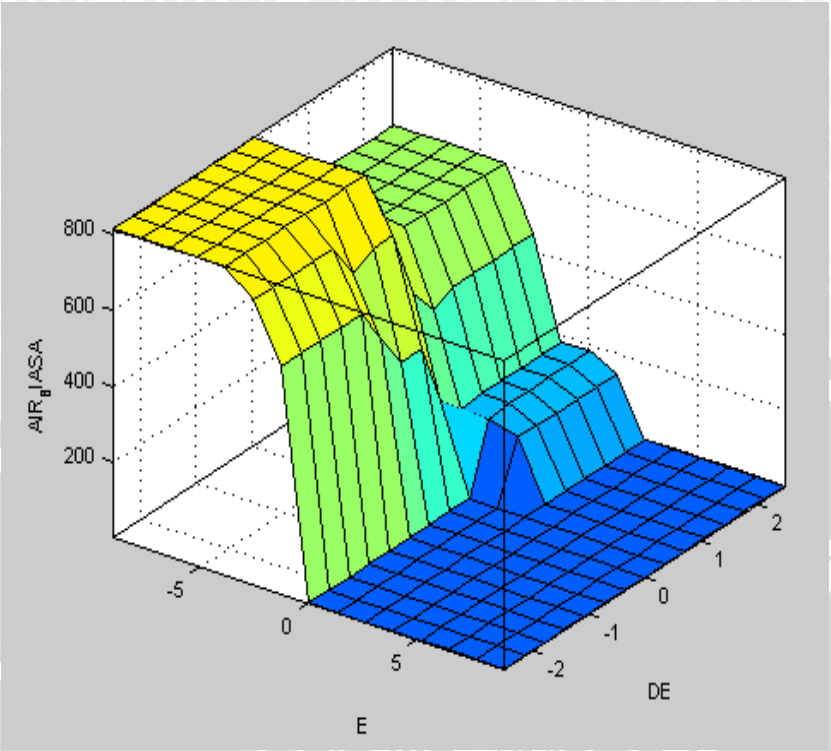
Motor Asam
Motor Basa

Tabel 4.12 Aturan *Fuzzy* untuk Suhu

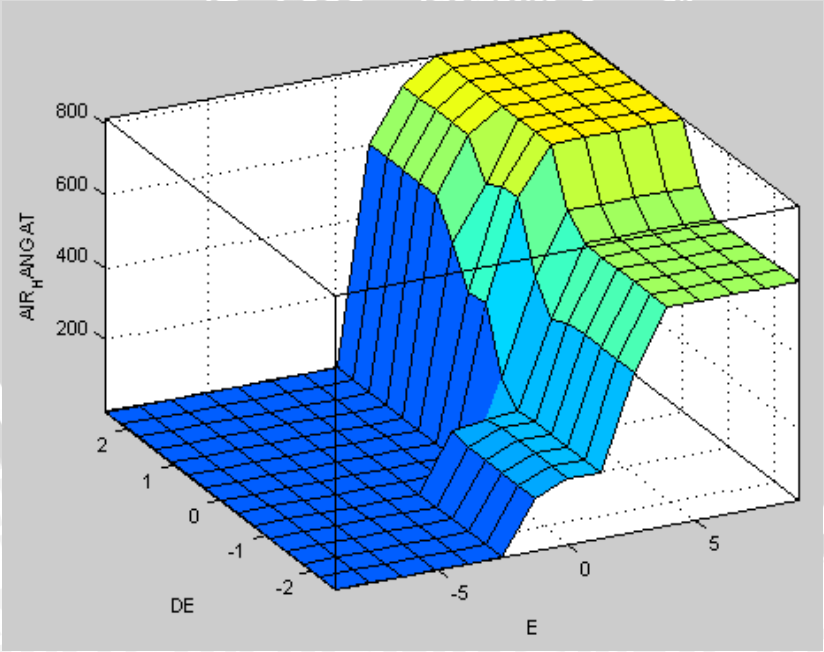
DELTA ERROR \ ERROR	NB	NS	Z	PS	PB
NB	H / L	AH / L	L / AM	L / AM	L / AH
NS	H / L	M / L	L / L	L / AM	L / AH
Z	H / L	M / L	L / L	L / M	L / H
PS	AH / L	AM / L	L / L	L / M	L / H
PB	AH / L	AM / L	AM / L	L / AH	L / H

Motor Biasa
Motor Hangat

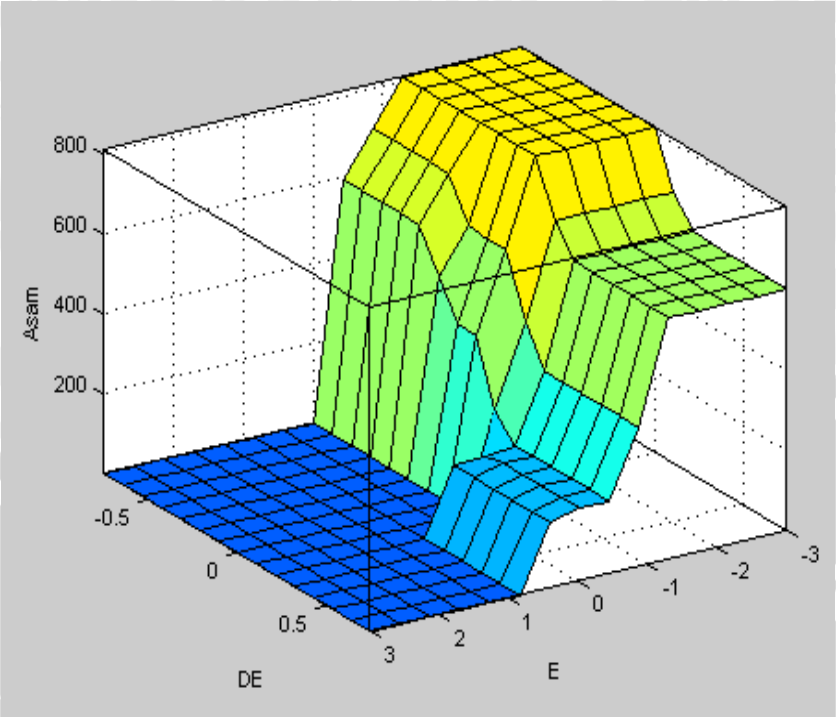
Ruang solusi *fuzzy* ditunjukkan dalam Gambar 4.15, 4.16, 4.17 dan 4.18



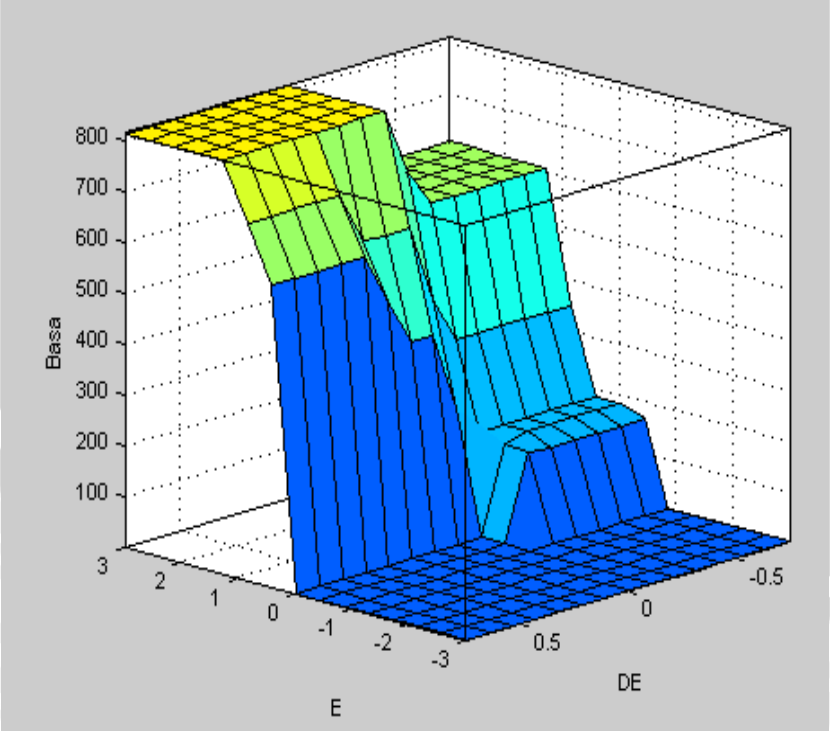
Gambar 4.15. Ruang solusi untuk motor biasa



Gambar 4.16. Ruang solusi untuk motor hangat



Gambar 4.17. Ruang solusi untuk motor asam



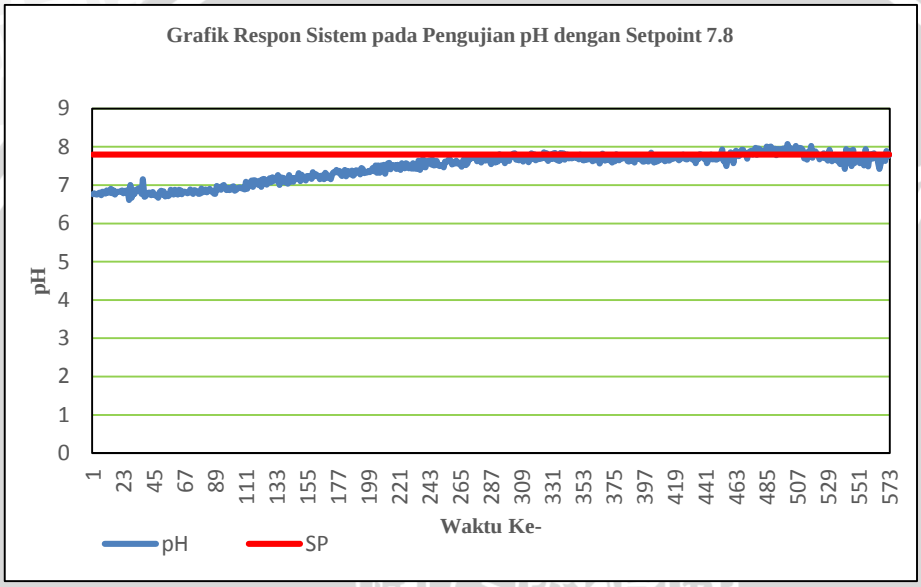
Gambar 4.18. Ruang solusi untuk motor basa

4.3 Pengujian keseluruhan sistem

Pengujian keseluruhan bertujuan untuk mengetahui kerja sistem apakah telah sesuai perancangan dengan melihat respon sistemnya. Pengujian keseluruhan sistem yang dilakukan sebagai berikut :

4.3.1 Pengujian pH tanpa gangguan

Pengujian pengontrolan pH dengan setpoint 7.8 menggunakan kontroler fuzzy dengan 5 fungsi keanggotaan masukan dan 5 fungsi keanggotaan keluaran dengan metode *inferensi Min-Max* dan metode *defuzzifikasi Center of Area* didapatkan hasil *output* yang ditunjukkan dalam Gambar 4.19.



Gambar 4.19 Grafik respon sistem pH tanpa gangguan

Dari hasil pengujian pengontrolan pH dengan *setpoint* 7.8 diperoleh *settling time* sebesar 240 detik dan dapat dihitung nilai %Ess sebagai berikut.

$$\%Ess = \frac{|average\ pH\ steady - setpoint|}{setpoint} x100\%$$

$$\%Ess = \frac{|7.723 - 7.8|}{7.8} x100\%$$

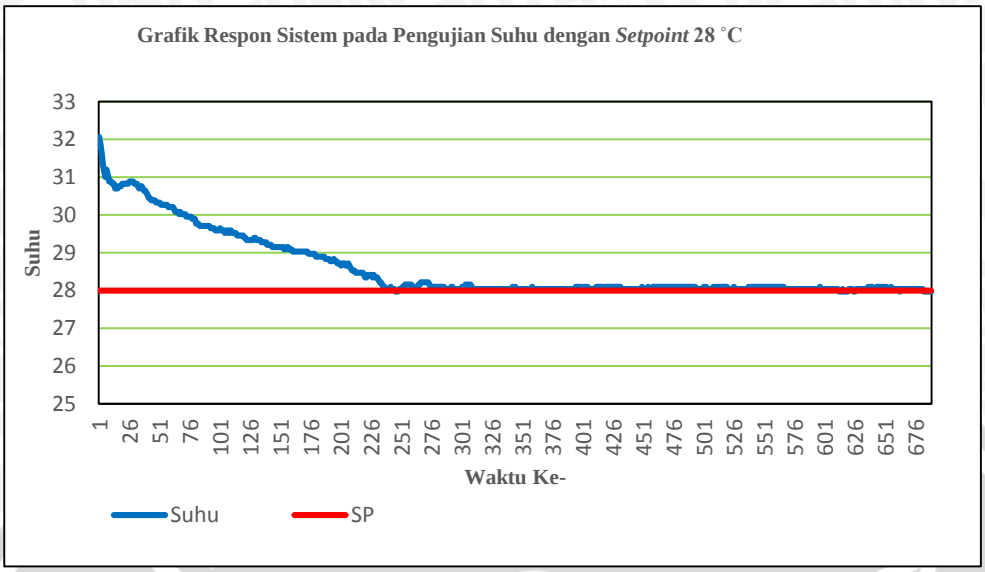
$$\%Ess = 0{,}987\%$$

Dari perhitungan %Ess dapat diketahui nilai *error steady state* yaitu sebesar 0.987%.

4.3.2 Pengujian suhu tanpa gangguan

Pengujian pengontrolan pH dengan setpoint 28 menggunakan kontroler fuzzy dengan 5 fungsi keanggotaan masukan dan 5 fungsi keanggotaan keluaran dengan metode

inferensi Min-Max dan metode defuzzifikasi *Center of Area* didapatkan hasil *output* yang ditunjukkan dalam Gambar 4.20.



Gambar 4.20 Grafik respon sistem suhu tanpa gangguan

Dari hasil pengujian pengontrolan suhu dengan *setpoint* 28 diperoleh *settling time* sebesar 210 detik dan dapat dihitung nilai %Ess sebagai berikut.

$$\%Ess = \frac{|average\ suhu\ steady - setpoint|}{setpoint} x100\%$$

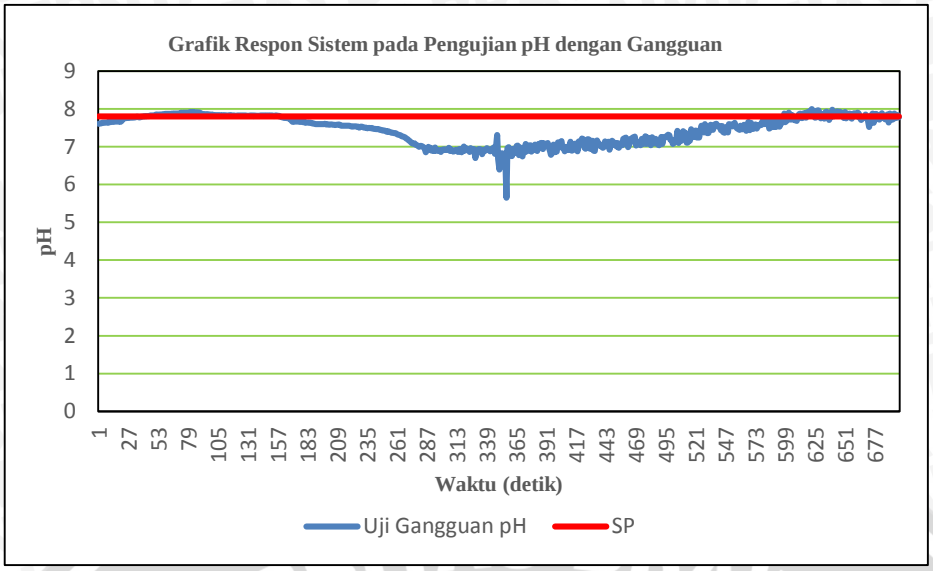
$$\%Ess = \frac{|28.08 - 28|}{28} x100\%$$

$$\%Ess = 0.286\%$$

Dari perhitungan %Ess dapat diketahui nilai *error steady state* yaitu sebesar 0.286%.

4.3.3 Pengujian pH dengan gangguan

Pada pengujian sistem ini diberikan gangguan berupa larutan asam CH3COOH sebanyak 2L dengan pH 6.78 untuk mengetahui ketika sistem mendapat gangguan perubahan pH apakah sistem dapat mempertahankan pH sesuai dengan *setpoint*. Hasil pengujian ditunjukkan Gambar 4.21.

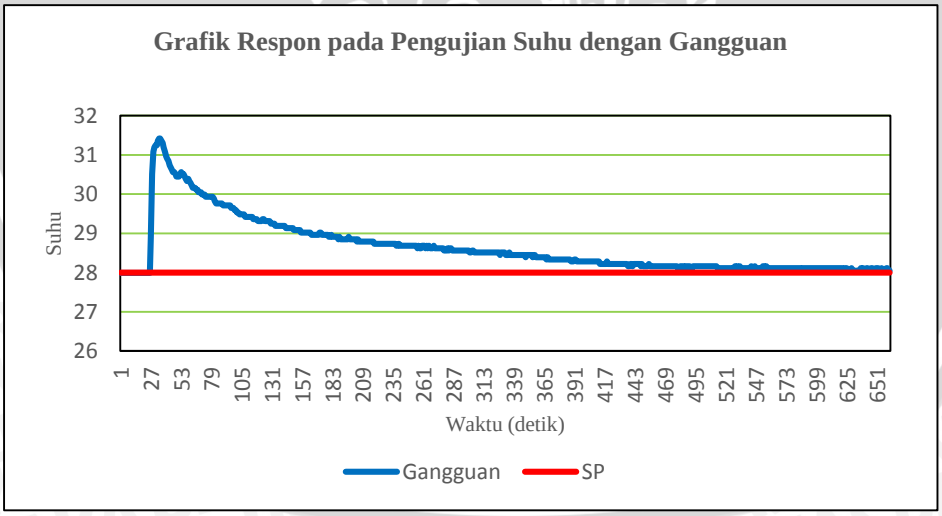


Gambar 4.21 Grafik respon sistem pH dengan gangguan

Dari hasil pengujian pengontrolan pH dengan gangguan, terlihat pada grafik respon sistem dapat menuju kembali ke kondisi *steady state* dengan *recovery time* sebesar 349 detik dan *error steady state* sebesar 0.83%.

4.3.4 Pengujian suhu dengan gangguan

Pada pengujian sistem ini diberikan gangguan berupa air panas sebanyak 2L dengan suhu 90 ºC. untuk mengetahui ketika sistem mendapat gangguan perubahan suhu apakah sistem dapat mempertahankan suhu sesuai dengan *setpoint*. Hasil pengujian ditunjukkan Gambar 4.22.



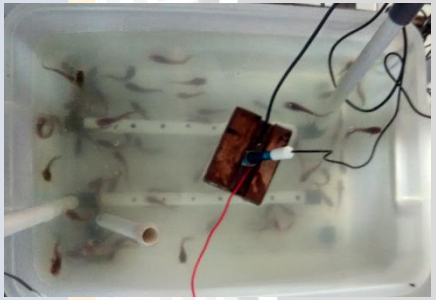
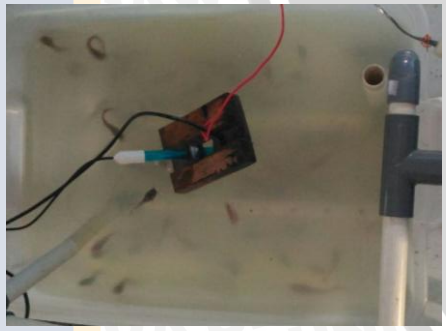
Gambar 4.22 Grafik respon sistem suhu dengan gangguan

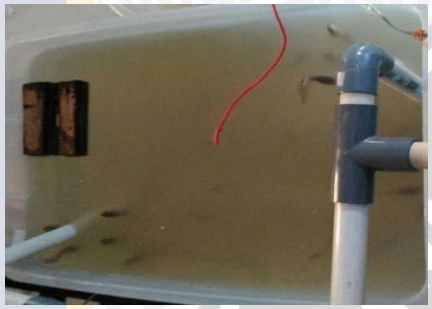
Dari hasil pengujian pengontrolan pH dengan gangguan, terlihat pada grafik respon sistem dapat menuju kembali ke kondisi *steady state* dengan *recovery time* sebesar 252 detik dan *error steady state* sebesar 0.82%.

4.4 Pengujian kelayakan sistem

Pengujian kelayakan sistem pengendalian kadar keasaman (pH) dan suhu yang diaplikasikan pada sistem pembenihan kolam ikan lele. Perlakuan yang diberikan selama enam hari yaitu pada masing-masing plant dimasukkan 100 ekor ikan dan diamati perbandingan jumlah ikan yang hidup selama 6 hari. Pada kolam pembenihan tanpa kontrol dilakukan penggantian air kurang lebih seperempat dari isi kolam yang dilakukan sehari sekali. Hasil pengujian selama enam hari ditunjukkan pada Tabel 4.13.

Tabel 4.13 Perbandingan Perlakuan Pembenihan

Hari Ke-	Pembenihan dengan Kontrol	Jumlah ikan hidup	Pembenihan tanpa Kontrol	Jumlah ikan hidup	Keterangan
1		100 ekor		100 ekor	Pada hari pertama terlihat bahwa kondisi air masih baik dan ikan dalam kondisi sehat
2		100 ekor		100 ekor	Pada hari kedua terlihat bahwa kondisi air pada pembenihan tanpa kontrol terlihat keruh berwarna kecoklatan dan ikan terlihat masih sama-sama sehat.
3		100 ekor		100 ekor	Pada hari ketiga terlihat bahwa kondisi air pada pembenihan tanpa kontrol terlihat keruh berwarna semakin kecoklatan dan ikan terlihat masih sama-sama sehat.

Hari Ke-	Pembenihan dengan Kontrol	Jumlah ikan hidup	Pembenihan tanpa Kontrol	Jumlah ikan hidup	Keterangan
4		100 ekor		100 ekor	Pada hari keempat terlihat bahwa kondisi air pada pembenihan tanpa kontrol terlihat keruh berwarna coklat kemerahan dan kondisi ikan pada kedua plant kurang sehat dengan melihat adanya ikan pada posisi menggantung.
5		100 ekor		99 ekor	Pada hari kelima terlihat bahwa kondisi air pada pembenihan tanpa kontrol terlihat keruh berwarna coklat kehitaman dan terdapat 1 benih yang mati sedangkan plant dengan kontrol ikan terlihat sehat dan tidak ada kematian benih.
6		100 ekor		96 ekor	Pada hari keenam terlihat bahwa kondisi air pada pembenihan tanpa kontrol terlihat keruh berwarna kemerahan dan terdapat 3 benih yang mati sedangkan plant dengan kontrol ikan terlihat sehat dan tidak ada kematian benih.

Berdasarkan Tabel 4.13 dapat dilihat bahwa *plant* dengan kontrol lebih sehat, ciri benih lele yang sehat adalah gerakan aktif, lincah, tidak menggantung serta tidak bergerombol di pojok kolam sedangkan pada *plant* tanpa kontrol terlihat dari banyak benih lele pada posisi menggantung dan bergerombol dipojok yang menunjukkan kondisi air kolam memburuk, kondisi ini biasanya disebabkan kadar amonia yang terlalu tinggi akibat penumpukan sisa makanan dan kotoran lele yang tidak terurai di dasar kolam (Rahadian Surya,2016). Persentase amoniak dalam perairan akan semakin meningkat seiring meningkatnya pH air. Pada saat keasamaan tinggi (pH rendah) ammonium yang terbentuk tidak terionisasi dan bersifat toksik pada ikan (Kordi, 2009).

Pada hari ke-5 pada *plant* tanpa kontrol ada satu benih ikan yang mati dan pada hari ke-6 ada 3 benih yang mati sedangkan pada *plant* kontrol tidak ada benih ikan yang mati. Hal ini menunjukkan bahwa perlakuan khusus yang diberikan pada pembenihan ikan lele yaitu dengan mengatur suhu dan kadar keasaman (pH) telah sesuai yang diharapkan pada tujuan.

BAB V
PENUTUP

5.1 Kesimpulan

Berdasarkan pengujian dan analisis yang telah dilakukan, dapat ditarik kesimpulan sebagai berikut:

1. Sistem pengendalian pH dan suhu pada sistem pembenihan kolam lele menggunakan kontroler logika *fuzzy*, setelah diimplementasikan sistem dapat mencapai *setpoint* pH 7.8 dan suhu 28 ºC. *Settling time* (ts) pH sebesar 240 detik dan *error steady state* sebesar 0.98%. *Settling time* (ts) suhu sebesar 210 detik dan *error steady state* sebesar 0.29%.
2. Hasil pengujian dengan gangguan sistem dapat kembali pada keadaan *steady* dengan *recovery time* untuk pH sebesar 349 detik dengan *error steady state* sebesar 0,83% dan suhu sebesar 252 detik dengan *error steady state* sebesar 0,82%.
3. Hasil pengujian kelayakan sistem diketahui bahwa perlakuan khusus yang diberikan yaitu dengan mengatur suhu dan kadar keasaman (pH) telah sesuai dengan tujuan dengan parameter jumlah ikan yang bertahan hidup.

5.2 Saran

Adapun saran – saran untuk menyempurnakan kerja sistem dan pengembangan lebih lanjut adalah sebagai berikut :

1. Tangki aktuator yang digunakan dalam penelitian ini kurang besar akibatnya sering kali isi ulang tangki maka tidak *efisien*. Sebagai saran untuk penelitian selanjutnya untuk menggunakan tangki berkapasitas lebih besar bila menggunakan ukuran plant yang sama.
2. Perlu pengujian dengan jumlah benih yang lebih banyak, apabila menggunakan ukuran *plant* yang sama dengan sekripsi ini setidaknya 200 ekor benih.
3. Perlu ditambahkan adanya perangkat tambahan seperti *keypad* sebagai *input setpoint* pH dan suhu untuk lebih memudahkan pengaturan, apabila jenis ikan yang dimasukkan dalam *plant* berbeda jenis dari ikan yang digunakan dalam penelitian ini.

DAFTAR PUSTAKA

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

Arduino.2014. *Arduino Mega2560*. *http://arduino.cc/en/Main/arduinoBoardMega2560*. (diakses 24 Oktober 2015).

Dwika, H. 2002. *Petunjuk Pelaksanaan. Monitoring Lingkungan Budidaya Air Tawar*. Departemen Kelautan dan Perikanan, Direktorat Jenderal Perikanan Budidaya, Direktorat Kesehatan Ikan dan Lingkungan.

Herlina, Cut Nina.S.Pi. 2015. *Budidaya Ikan Lele Dumbo dikolam Terpal*. Badan Penelitian dan Pengembangan Pertanian : http://nad.litbang.pertanian.go.id. (diakses 24 Oktober 2015).

Kordi. 2009. Budidaya Ikan Lele. *www.umi.ac.id.* (diakses 25 April 2016)

Kurniawan, Putro S. 2016. *Potensi Usaha Budidaya Ikan Lele air tawar*. Alamtani : *www.alamtani.com*. (diakses 12 Februari 2016).

Lee, C. 1990. *Fuzzy Logic in Control System : Fuzzy Logic Controller Part I.* IEEE.

Lusia Kus Anna. 2012. *Konsumsi protein Hewani Rendah*. Alamat : *http://health.kompas.com*. (diakses 21 September 2015).

M. Kretschmar and S. Welsby (2005). *Capacitive and Inductive Displacement Sensors, in Sensor Technology Handbook*. J. Wilson editor, Newnes: Burlington, MA, *id.wikipedia.org/wiki/Sensor*. Diakses 20 September 2015.

Meyer, Curtis A. 2006. *Basic Electronics An Introduction to Electronics for Science Students*. Carnegie Mellon University.

Nuh, M. 2013. *Elektronika Dasar*. Jakarta: Direktorat Jenderal Peningkatan Mutu Pendidik & Tenaga Kependidikan.

Rahadian Surya. 2013. *Benih Ikan Lele Menggantung*. *www.bibitikan.net*. (diakses 25 April 2016).

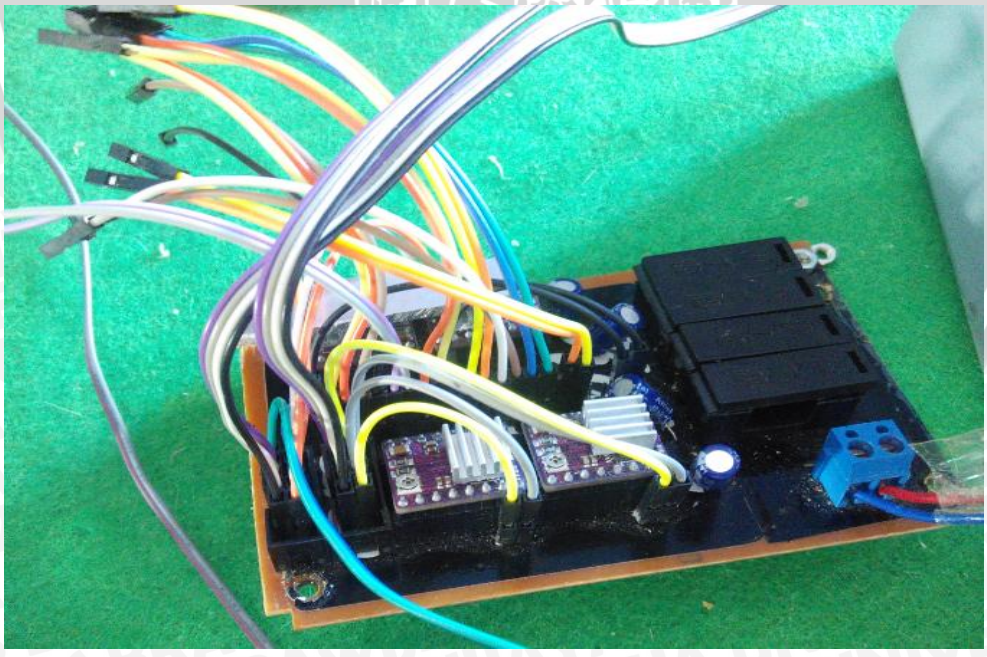
Rahmat Hidayat. 2013. *Cara Pembenihan Ikan Lele*. Alamtani : *http://alamtani.com*. (diakses 25 Oktober 2015).

Ross, J. Timothy. 2010. *Fuzzy logic with engineering applications*. USA: John Wiley & Sons, Ltd.

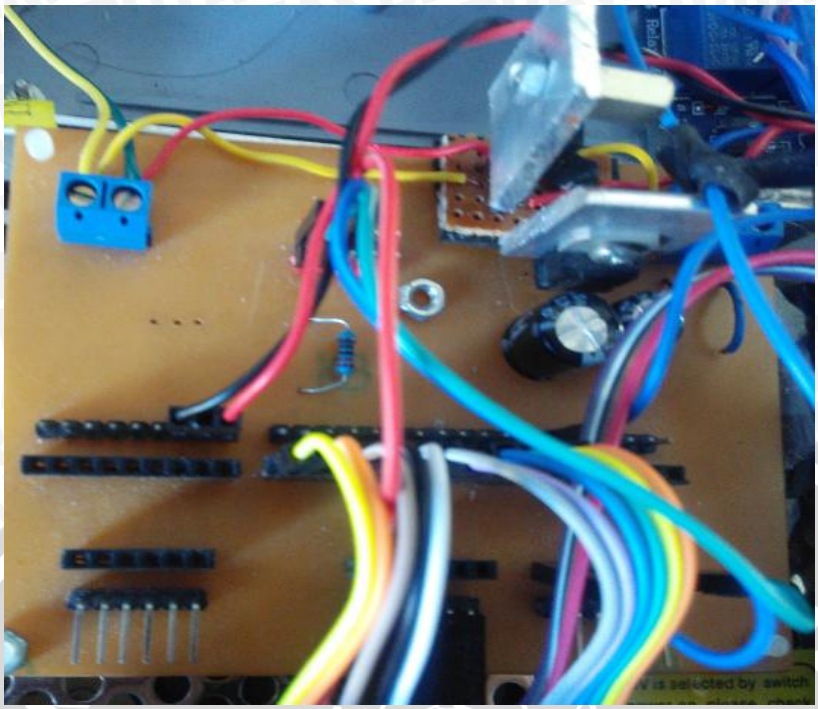
Simanjutak, R.H. 1996. *Pembudidayaan Ikan Lele Lokal dan Dumbo*. Bhratara. Jakarta.

LAMPIRAN I
FOTO ALAT

Box tempat komponen elektrik diletakkan

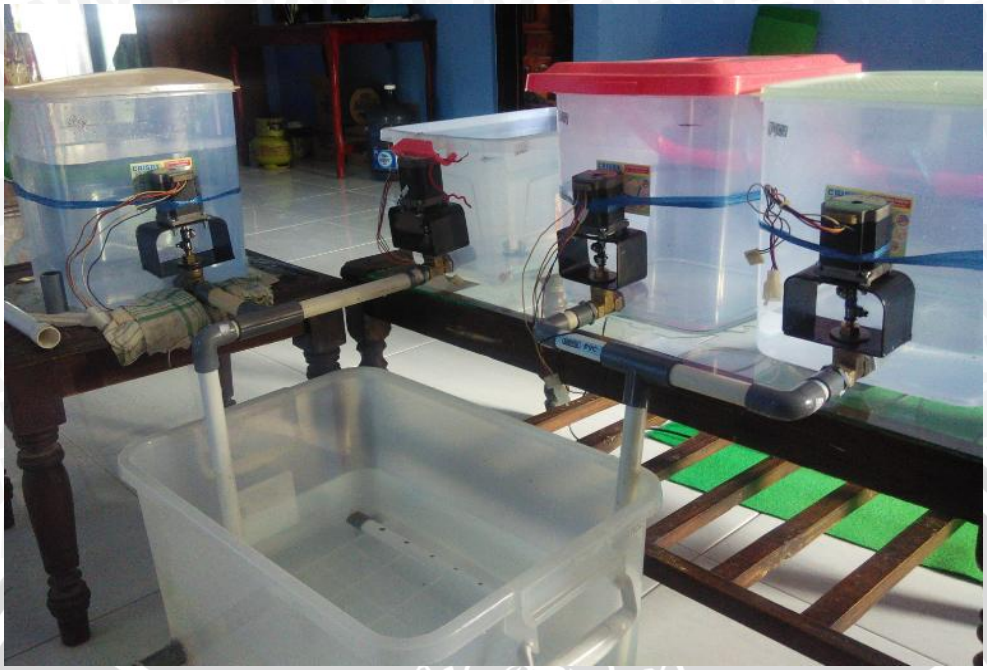


Rangkaian driver motor stepper

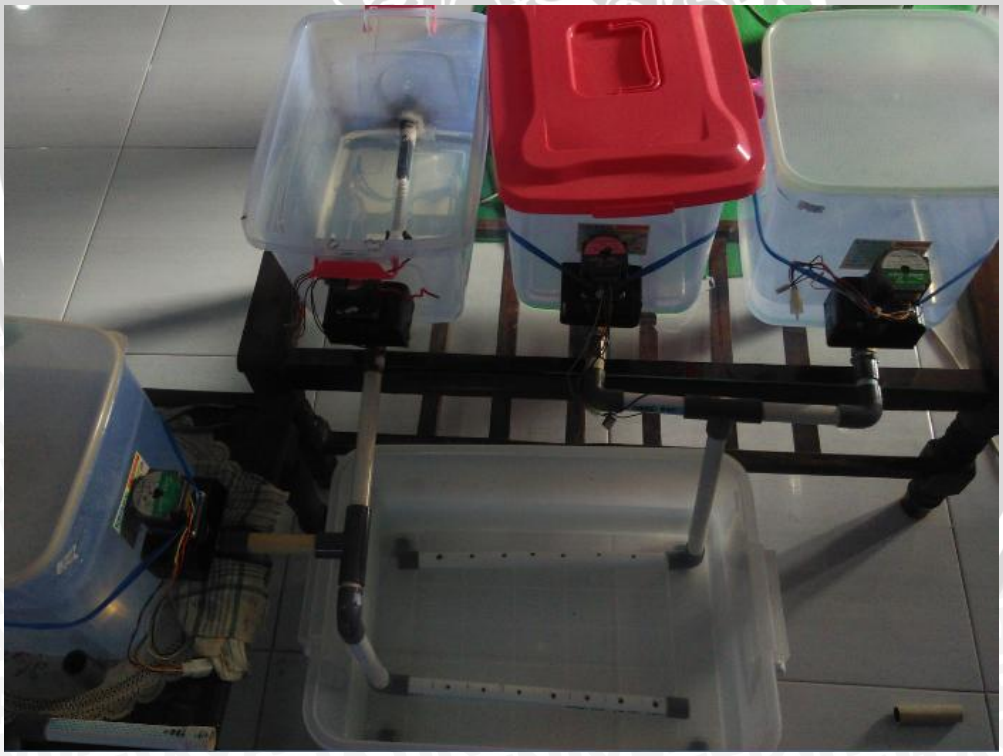


rangkaian regulator 9 v dan 5 v

Pemasangan Heater pada tangki aktuator



Desain perangkat keras sistem pengendali suhu dan pH pada miniatur *plant*



Desain perangkat keras sistem pengendali suhu dan pH pada miniatur *plant*

LAMPIRAN II
LISTING PROGRAM

```
//library Fuzzy//
#include <FuzzyRule.h>
#include <FuzzyComposition.h>
#include <Fuzzy.h>
#include <FuzzyRuleConsequent.h>
#include <FuzzyOutput.h>
#include <FuzzyInput.h>
#include <FuzzyIO.h>
#include <FuzzySet.h>
#include <FuzzyRuleAntecedent.h>  //lib.suhu//
#include <OneWire.h>
#include <DallasTemperature.h>
#include <LiquidCrystal.h>;
#define ONE_WIRE_BUS 14 //pin 14

//pin A8 sen.pH
#define SensorPin A8
unsigned long int avgValue;  //Store the average value of the sensor feedback
float i;
int data[10],temp;
// variable for reading the sensor voltage
float SensorValue = 0, readValue = 0;

//pin LCD       (RS, EN, D4, D5, D6,D7);
LiquidCrystal lcd(42, 44, 46, 48, 50, 52);

// LIBRARY KOMUNIKASI BUAT SENSOR SUHU > Setup a oneWire instance to
communicate with any OneWire devices (not just Maxim/Dallas temperature ICs)
OneWire oneWire(ONE_WIRE_BUS);

// NILAI REFERENSI > Pass our oneWire reference to Dallas Temperature.
DallasTemperature sensors(&oneWire);

//NILAI SET AWAL INPUT ERROR, DELTA_ERROR, SETPOINT  pH
float ERROR_pH = 0, lastErrorpH = 0, DE_pH = 0;
float setpointpH = 7.8;
float pHValue = 0;  // pH value

//NILAI SET AWAL INPUT ERROR, DELTA_ERROR, SETPOINT Suhu
float ERROR_SH = 0, lastErrorSH = 0, DE_SH = 0;
float setpointSH = 28.0;
float SuhuValue = 0;  //nilai awal
```

```
// inisial pin motor ASAM & BASA
const int MotorStep1 = 12, MotorDir1 = 13; // motor 1 Asam
const int MotorStep2 = 10, MotorDir2 = 11; // motor 3 Basa

// inisial pin motor BIASA & HANGAT
const int MotorStep3 = 8, MotorDir3 = 9; // motor 2 Biasa
const int MotorStep4 = 6, MotorDir4 = 7; // motor 4 Hangat

//NILAI AWAL MOTOR ASAM & BASA
int lastStep1 = 0, currentStep1 = 0, lastStep2 = 0, currentStep2 = 0;
int stepValue1 = 0, stepValue2 = 0;
int balikstep1, balikstep2;

//NILAI AWAL MOTOR BIASA & HANGAT
int lastStep3 = 0, currentStep3 = 0, lastStep4 = 0, currentStep4 = 0;
int stepValue3 = 0, stepValue4 = 0;
int balikstep3, balikstep4;

//NILAI AWAL OUTPUT SEBAGAI INPUT MOTOR
int output1 = 0, output2 = 0; // output motor pH
int output3 = 0, output4 = 0; // output motor suhu


// Fuzzy Input Inisialisasi
Fuzzy* fuzzy = new Fuzzy();

//INPUT ERROR pH
FuzzySet* pH_E_NB    = new FuzzySet(-3, -3, -1.2, -0.6);
FuzzySet* pH_E_NS    = new FuzzySet(-1.2, -0.6, -0.6, -0.2);
FuzzySet* pH_E_Z     = new FuzzySet(-0.6, 0, 0, 0.6);
FuzzySet* pH_E_PS    = new FuzzySet( 0.2, 0.6, 0.6, 1.2);
FuzzySet* pH_E_PB    = new FuzzySet( 0.6, 1.2, 3, 3);
//INPUT DELTA_ERROR 25%E
FuzzySet* pH_DE_NB    = new FuzzySet(-0.75, -0.75, -0.3, -0.8);
FuzzySet* pH_DE_NS    = new FuzzySet(-0.3, -0.15, -0.15, -0.03);
FuzzySet* pH_DE_Z     = new FuzzySet(-0.15, 0, 0, 0.15);
FuzzySet* pH_DE_PS    = new FuzzySet( 0.03,  0.15,  0.15,  0.3);
FuzzySet* pH_DE_PB    = new FuzzySet( 0.15,  0.3,  0.75,  0.75);

//INPUT ERROR SUHU
FuzzySet* SH_E_NB    = new FuzzySet(-9, -9, -3, -1.5);
FuzzySet* SH_E_NS    = new FuzzySet(-3, -1.5, -1.5, 0);
FuzzySet* SH_E_Z     = new FuzzySet(-1.5, 0, 0, 1.5);
FuzzySet* SH_E_PS    = new FuzzySet( 0, 1.5, 1.5, 3);
FuzzySet* SH_E_PB    = new FuzzySet( 1.5, 3, 9, 9);
//INPUT DELTA_ERROR 30%E
FuzzySet* SH_DE_NB    = new FuzzySet(-2.5, -2.5, -1, -0.45);
FuzzySet* SH_DE_NS    = new FuzzySet(-1, -0.45, -0.45, 0);
FuzzySet* SH_DE_Z     = new FuzzySet(-0.45, 0, 0, 0.45);
FuzzySet* SH_DE_PS    = new FuzzySet( 0,  0.45,  0.45,  1);
```

```
FuzzySet* SH_DE_PB    = new FuzzySet( 0.45,  1,  2.5,  2.5);

void setup(void)
{

  //set up the LCD's number of columns n rows
  lcd.begin(20, 4);

  // start serial port
  Serial.begin(9600);

    //serial suhu
  Serial.println("Dallas Temperature IC Control Library Demo");
  // Start up the library suhu
  sensors.begin();

///////////////INISIALISASI PIN MOTOR pH//////////////////////////////
//MOTOR ASAM//
  pinMode(MotorStep1,OUTPUT);
  pinMode(MotorDir1,OUTPUT);
//MOTOR BASA//
  pinMode(MotorStep2,OUTPUT);
  pinMode(MotorDir2,OUTPUT);

///////////////INISIALISASI PIN MOTOR SUHU//////////////////////////////
//MOTOR BIASA//
  pinMode(MotorStep3,OUTPUT);
  pinMode(MotorDir3,OUTPUT);
 //MOTOR HANGAT//
  pinMode(MotorStep4,OUTPUT);
  pinMode(MotorDir4,OUTPUT);

                  ////////////////////
  ////////////////    FUZZY INPUT ERROR pH ////////////////////
  FuzzyInput* ERROR_pH = new FuzzyInput(1);
  ERROR_pH->addFuzzySet(pH_E_NB  );
  ERROR_pH->addFuzzySet(pH_E_NS  );
  ERROR_pH->addFuzzySet(pH_E_Z   );
  ERROR_pH->addFuzzySet(pH_E_PS  );
  ERROR_pH->addFuzzySet(pH_E_PB  );

  fuzzy->addFuzzyInput(ERROR_pH);

////////////////    FUZZY INPUT DELTA_ERROR pH//////////////////
  FuzzyInput* DE_pH = new FuzzyInput(2);
  DE_pH->addFuzzySet(pH_DE_NB  );
  DE_pH->addFuzzySet(pH_DE_NS  );
  DE_pH->addFuzzySet(pH_DE_Z   );
  DE_pH->addFuzzySet(pH_DE_PS  );
  DE_pH->addFuzzySet(pH_DE_PB  );
```

```
fuzzy->addFuzzyInput(DE_pH);

                    //////////////////////
//////////////////    FUZZY INPUT ERROR SUHU       //////////////////////
  FuzzyInput* ERROR_SH = new FuzzyInput(3);
  ERROR_SH->addFuzzySet(SH_E_NB  );
  ERROR_SH->addFuzzySet(SH_E_NS  );
  ERROR_SH->addFuzzySet(SH_E_Z   );
  ERROR_SH->addFuzzySet(SH_E_PS  );
  ERROR_SH->addFuzzySet(SH_E_PB  );

  fuzzy->addFuzzyInput(ERROR_SH);

//////////////////    FUZZY INPUT DELTA_ERROR SUHU    ///////////////////
  FuzzyInput* DE_SH = new FuzzyInput(4);
  DE_SH->addFuzzySet(SH_DE_NB  );
  DE_SH->addFuzzySet(SH_DE_NS  );
  DE_SH->addFuzzySet(SH_DE_Z   );
  DE_SH->addFuzzySet(SH_DE_PS  );
  DE_SH->addFuzzySet(SH_DE_PB  );

  fuzzy->addFuzzyInput(DE_SH);

                    ////////////////////////////
//////////////////////// OUTPUT MF pH & SUHU//////////////////////////
                    ////////////////////////////

                    //////////////////////
  //////////////////    FUZZY OUTPUT MOTOR ASAM         //////////////////
  FuzzyOutput* M_ASAM    = new FuzzyOutput(1);

  FuzzySet* L  = new FuzzySet(   -10,   0, 0,  10 );
  M_ASAM->addFuzzySet(L  );

  FuzzySet* AM    = new FuzzySet(  10, 200, 200, 340 );
  M_ASAM->addFuzzySet(AM    );


  FuzzySet* M     = new FuzzySet( 280, 400, 400, 540 );
  M_ASAM->addFuzzySet(M     );

  FuzzySet* AH    = new FuzzySet(  480, 600, 600, 740 );
  M_ASAM->addFuzzySet(AH    );

  FuzzySet* H     = new FuzzySet( 680, 800, 800, 950);
  M_ASAM->addFuzzySet(H     );

  fuzzy->addFuzzyOutput(M_ASAM);
```

```
//////////////////    FUZZY OUTPUT MOTOR BASA          //////////////////
  FuzzyOutput* M_BASA    = new FuzzyOutput(2);

  FuzzySet* L1  = new FuzzySet(   -10,   0, 0,  10 );
  M_BASA->addFuzzySet(L1  );

  FuzzySet* AM1    = new FuzzySet(  10, 200, 200, 340 );
  M_BASA->addFuzzySet(AM1    );

  FuzzySet* M1     = new FuzzySet( 280, 400, 400, 540 );
  M_BASA->addFuzzySet(M1     );

  FuzzySet* AH1    = new FuzzySet(  480, 600, 600, 740 );
  M_BASA->addFuzzySet(AH1    );

  FuzzySet* H1     = new FuzzySet( 680, 800, 800, 950);
  M_BASA->addFuzzySet(H1     );

  fuzzy->addFuzzyOutput(M_BASA);

                ////////////////////////
//////////////////    FUZZY OUTPUT MOTOR BIASA         //////////////////
FuzzyOutput* M_BIASA    = new FuzzyOutput(3);

  FuzzySet* L2  = new FuzzySet(   -10,   0, 0,  10 );
  M_BIASA->addFuzzySet(L2  );


  FuzzySet* AM2    = new FuzzySet(  10, 200, 200, 340 );
  M_BIASA->addFuzzySet(AM2    );


  FuzzySet* M2     = new FuzzySet( 280, 400, 400, 540 );
  M_BIASA->addFuzzySet(M2     );


  FuzzySet* AH2    = new FuzzySet(  480, 600, 600, 740 );
  M_BIASA->addFuzzySet(AH2    );


  FuzzySet* H2     = new FuzzySet( 680, 800, 800, 950);
  M_BIASA->addFuzzySet(H2     );

  fuzzy->addFuzzyOutput(M_BIASA);
```

```
//////////////////    FUZZY OUTPUT MOTOR HANGAT          //////////////////
   FuzzyOutput* M_HANGAT    = new FuzzyOutput(4);

  FuzzySet* L3  = new FuzzySet(   -10,   0, 0,  10 );
  M_HANGAT->addFuzzySet(L3  );


  FuzzySet* AM3    = new FuzzySet(  10, 200, 200, 340 );
  M_HANGAT->addFuzzySet(AM3    );


  FuzzySet* M3     = new FuzzySet( 280, 400, 400, 540 );
  M_HANGAT->addFuzzySet(M3    );


  FuzzySet* AH3    = new FuzzySet(  480, 600, 600, 740 );
  M_HANGAT->addFuzzySet(AH3    );


  FuzzySet* H3     = new FuzzySet( 680, 800, 800, 950);
  M_HANGAT->addFuzzySet(H3     );

  fuzzy->addFuzzyOutput(M_HANGAT);


            ////////////////////////////////////
//////////////////  PEMBENTUK FUZZY RULE  pH     ////////////////////////
            ////////////////////////////////////

                ////////////////////////
//////////////////////  FUZZY ANTECEDENT pH   ////////////////////
//ERROR_pH = NB//
 FuzzyRuleAntecedent*ifERROR_pHpH_E_NBandDE_pHpH_DE_NB=new FuzzyRuleAntecedent();
 ifERROR_pHpH_E_NBandDE_pHpH_DE_NB->joinWithAND(pH_E_NB,pH_DE_NB);
 FuzzyRuleAntecedent* ifERROR_pHpH_E_NBandDE_pHpH_DE_NS = new FuzzyRuleAntecedent();
 ifERROR_pHpH_E_NBandDE_pHpH_DE_NS  ->joinWithAND(pH_E_NB, pH_DE_NS);
 FuzzyRuleAntecedent* ifERROR_pHpH_E_NBandDE_pHpH_DE_Z = new FuzzyRuleAntecedent();
 ifERROR_pHpH_E_NBandDE_pHpH_DE_Z  ->joinWithAND(pH_E_NB, pH_DE_Z);
 FuzzyRuleAntecedent* ifERROR_pHpH_E_NBandDE_pHpH_DE_PS = new FuzzyRuleAntecedent();
 ifERROR_pHpH_E_NBandDE_pHpH_DE_PS  ->joinWithAND(pH_E_NB, pH_DE_PS);
 FuzzyRuleAntecedent* ifERROR_pHpH_E_NBandDE_pHpH_DE_PB = new FuzzyRuleAntecedent();
 ifERROR_pHpH_E_NBandDE_pHpH_DE_PB  ->joinWithAND(pH_E_NB, pH_DE_PB);

//ERROR_pH = NS//
 FuzzyRuleAntecedent* ifERROR_pHpH_E_NSandDE_pHpH_DE_NB = new FuzzyRuleAntecedent();
 ifERROR_pHpH_E_NSandDE_pHpH_DE_NB  ->joinWithAND(pH_E_NS, pH_DE_NB);
 FuzzyRuleAntecedent* ifERROR_pHpH_E_NSandDE_pHpH_DE_NS = new FuzzyRuleAntecedent();
 ifERROR_pHpH_E_NSandDE_pHpH_DE_NS  ->joinWithAND(pH_E_NS, pH_DE_NS);
```

```
FuzzyRuleAntecedent* ifERROR_pHpH_E_NSandDE_pHpH_DE_Z = new FuzzyRuleAntecedent();
ifERROR_pHpH_E_NSandDE_pHpH_DE_Z  ->joinWithAND(pH_E_NS, pH_DE_Z);
FuzzyRuleAntecedent* ifERROR_pHpH_E_NSandDE_pHpH_DE_PS = new FuzzyRuleAntecedent();
ifERROR_pHpH_E_NSandDE_pHpH_DE_PS  ->joinWithAND(pH_E_NS, pH_DE_PS);
FuzzyRuleAntecedent* ifERROR_pHpH_E_NSandDE_pHpH_DE_PB = new FuzzyRuleAntecedent();
ifERROR_pHpH_E_NSandDE_pHpH_DE_PB  ->joinWithAND(pH_E_NS, pH_DE_PB);
```

//ERROR_pH = Z//

```
FuzzyRuleAntecedent* ifERROR_pHpH_E_ZandDE_pHpH_DE_NB = new FuzzyRuleAntecedent();
ifERROR_pHpH_E_ZandDE_pHpH_DE_NB  ->joinWithAND(pH_E_Z, pH_DE_NB);
FuzzyRuleAntecedent* ifERROR_pHpH_E_ZandDE_pHpH_DE_NS = new FuzzyRuleAntecedent();
ifERROR_pHpH_E_ZandDE_pHpH_DE_NS  ->joinWithAND(pH_E_Z, pH_DE_NS);
FuzzyRuleAntecedent* ifERROR_pHpH_E_ZandDE_pHpH_DE_Z = new FuzzyRuleAntecedent();
ifERROR_pHpH_E_ZandDE_pHpH_DE_Z  ->joinWithAND(pH_E_Z, pH_DE_Z);
FuzzyRuleAntecedent* ifERROR_pHpH_E_ZandDE_pHpH_DE_PS = new FuzzyRuleAntecedent();
ifERROR_pHpH_E_ZandDE_pHpH_DE_PS  ->joinWithAND(pH_E_Z, pH_DE_PS);
FuzzyRuleAntecedent* ifERROR_pHpH_E_ZandDE_pHpH_DE_PB = new FuzzyRuleAntecedent();
ifERROR_pHpH_E_ZandDE_pHpH_DE_PB  ->joinWithAND(pH_E_Z, pH_DE_PB);
```

//ERROR_pH = PS//

```
FuzzyRuleAntecedent* ifERROR_pHpH_E_PSandDE_pHpH_DE_NB = new FuzzyRuleAntecedent();
ifERROR_pHpH_E_PSandDE_pHpH_DE_NB  ->joinWithAND(pH_E_PS, pH_DE_NB);
FuzzyRuleAntecedent* ifERROR_pHpH_E_PSandDE_pHpH_DE_NS = new FuzzyRuleAntecedent();
ifERROR_pHpH_E_PSandDE_pHpH_DE_NS  ->joinWithAND(pH_E_PS, pH_DE_NS);
FuzzyRuleAntecedent* ifERROR_pHpH_E_PSandDE_pHpH_DE_Z = new FuzzyRuleAntecedent();
ifERROR_pHpH_E_PSandDE_pHpH_DE_Z  ->joinWithAND(pH_E_PS, pH_DE_Z);
FuzzyRuleAntecedent* ifERROR_pHpH_E_PSandDE_pHpH_DE_PS = new FuzzyRuleAntecedent();
ifERROR_pHpH_E_PSandDE_pHpH_DE_PS  ->joinWithAND(pH_E_PS, pH_DE_PS);
FuzzyRuleAntecedent* ifERROR_pHpH_E_PSandDE_pHpH_DE_PB = new FuzzyRuleAntecedent();
ifERROR_pHpH_E_PSandDE_pHpH_DE_PB  ->joinWithAND(pH_E_PS, pH_DE_PB);
```

//ERROR_pH = PB//

```
FuzzyRuleAntecedent* ifERROR_pHpH_E_PBandDE_pHpH_DE_NB = new FuzzyRuleAntecedent();
ifERROR_pHpH_E_PBandDE_pHpH_DE_NB  ->joinWithAND(pH_E_PB, pH_DE_NB);
FuzzyRuleAntecedent* ifERROR_pHpH_E_PBandDE_pHpH_DE_NS = new FuzzyRuleAntecedent();
ifERROR_pHpH_E_PBandDE_pHpH_DE_NS  ->joinWithAND(pH_E_PB, pH_DE_NS);
FuzzyRuleAntecedent* ifERROR_pHpH_E_PBandDE_pHpH_DE_Z = new FuzzyRuleAntecedent();
ifERROR_pHpH_E_PBandDE_pHpH_DE_Z  ->joinWithAND(pH_E_PB, pH_DE_Z);
FuzzyRuleAntecedent* ifERROR_pHpH_E_PBandDE_pHpH_DE_PS = new FuzzyRuleAntecedent();
ifERROR_pHpH_E_PBandDE_pHpH_DE_PS  ->joinWithAND(pH_E_PB, pH_DE_PS);
FuzzyRuleAntecedent* ifERROR_pHpH_E_PBandDE_pHpH_DE_PB = new FuzzyRuleAntecedent();
ifERROR_pHpH_E_PBandDE_pHpH_DE_PB  ->joinWithAND(pH_E_PB, pH_DE_PB);
```

```
//////////////////////   FUZZY CONSEQUENT pH  ////////////////////
//M_BASA = berhenti
 FuzzyRuleConsequent* thenM_ASAMLandM_BASAL1 = new FuzzyRuleConsequent();
thenM_ASAMLandM_BASAL1    ->addOutput(L    );
thenM_ASAMLandM_BASAL1   ->addOutput(L1    );

FuzzyRuleConsequent* thenM_ASAMHandM_BASAL1 = new FuzzyRuleConsequent();
thenM_ASAMHandM_BASAL1    ->addOutput(H    );
thenM_ASAMHandM_BASAL1    ->addOutput(L1    );

FuzzyRuleConsequent*          thenM_ASAMAHandM_BASAL1          =          new
FuzzyRuleConsequent();
thenM_ASAMAHandM_BASAL1    ->addOutput(AH    );
thenM_ASAMAHandM_BASAL1    ->addOutput(L1    );

FuzzyRuleConsequent* thenM_ASAMMandM_BASAL1 = new FuzzyRuleConsequent();
thenM_ASAMMandM_BASAL1    ->addOutput(M    );
thenM_ASAMMandM_BASAL1    ->addOutput(L1   );

FuzzyRuleConsequent*          thenM_ASAMALandM_BASAAM1          =          new
FuzzyRuleConsequent();
thenM_ASAMALandM_BASAAM1    ->addOutput(L    );
thenM_ASAMALandM_BASAAM1    ->addOutput(AM1   );

//M_ASAM = berhenti
FuzzyRuleConsequent*          thenM_ASAMAMandM_BASAL1          =          new
FuzzyRuleConsequent();
thenM_ASAMAMandM_BASAL1    ->addOutput(AM    );
thenM_ASAMAMandM_BASAL1    ->addOutput(L1    );

FuzzyRuleConsequent* thenM_ASAMLandM_BASAM1 = new FuzzyRuleConsequent();
thenM_ASAMLandM_BASAM1    ->addOutput(L    );
thenM_ASAMLandM_BASAM1    ->addOutput(M1    );

FuzzyRuleConsequent*          thenM_ASAMLandM_BASAAH1          =          new
FuzzyRuleConsequent();
thenM_ASAMLandM_BASAAH1    ->addOutput(L    );
thenM_ASAMLandM_BASAAH1    ->addOutput(AH1    );

FuzzyRuleConsequent* thenM_ASAMLandM_BASAH1 = new FuzzyRuleConsequent();
thenM_ASAMLandM_BASAH1    ->addOutput(L    );
thenM_ASAMLandM_BASAH1    ->addOutput(H1    );

//////////FUZZY   RULE   YG   DIBENTUK   DARI   ANTECENDENT   DENGAN
CONSEQUENT pH//////////////

//////////////////////////      FUZZY RULE pH      ////////////////////////////////
//ERROR_pH IS pH.E_NB
 FuzzyRule*    fuzzyRule1    =    new    FuzzyRule(1,    ifERROR_pHpH_E_NBandDE_pHpH_DE_NB,
 thenM_ASAMHandM_BASAL1);
 fuzzy->addFuzzyRule(fuzzyRule1);
```

```
FuzzyRule* fuzzyRule2 = new FuzzyRule(2, ifERROR_pHpH_E_NBandDE_pHpH_DE_NS, thenM_ASAMHandM_BASAL1);
 fuzzy->addFuzzyRule(fuzzyRule2);

FuzzyRule* fuzzyRule3 = new FuzzyRule(3, ifERROR_pHpH_E_NBandDE_pHpH_DE_Z, thenM_ASAMHandM_BASAL1);
 fuzzy->addFuzzyRule(fuzzyRule3);

FuzzyRule* fuzzyRule4 = new FuzzyRule(4, ifERROR_pHpH_E_NBandDE_pHpH_DE_PS, thenM_ASAMAHandM_BASAL1);
 fuzzy->addFuzzyRule(fuzzyRule4);

FuzzyRule* fuzzyRule5 = new FuzzyRule(5, ifERROR_pHpH_E_NBandDE_pHpH_DE_PB, thenM_ASAMAHandM_BASAL1);
 fuzzy->addFuzzyRule(fuzzyRule5);

//ERROR_pH IS pH.E_NS
FuzzyRule* fuzzyRule6 = new FuzzyRule(6, ifERROR_pHpH_E_NSandDE_pHpH_DE_NB, thenM_ASAMAHandM_BASAL1);
 fuzzy->addFuzzyRule(fuzzyRule6);

FuzzyRule* fuzzyRule7 = new FuzzyRule(7, ifERROR_pHpH_E_NSandDE_pHpH_DE_NS, thenM_ASAMMandM_BASAL1);
 fuzzy->addFuzzyRule(fuzzyRule7);

FuzzyRule* fuzzyRule8 = new FuzzyRule(8, ifERROR_pHpH_E_NSandDE_pHpH_DE_Z, thenM_ASAMMandM_BASAL1);
 fuzzy->addFuzzyRule(fuzzyRule8);

FuzzyRule* fuzzyRule9 = new FuzzyRule(9, ifERROR_pHpH_E_NSandDE_pHpH_DE_PS, thenM_ASAMAMandM_BASAL1);
 fuzzy->addFuzzyRule(fuzzyRule9);

FuzzyRule* fuzzyRule10 = new FuzzyRule(10, ifERROR_pHpH_E_NSandDE_pHpH_DE_PB, thenM_ASAMAMandM_BASAL1);
 fuzzy->addFuzzyRule(fuzzyRule10);

//ERROR_pH IS pH.E_Z
FuzzyRule* fuzzyRule11 = new FuzzyRule(11, ifERROR_pHpH_E_ZandDE_pHpH_DE_NB, thenM_ASAMALandM_BASAAM1);
 fuzzy->addFuzzyRule(fuzzyRule11);

FuzzyRule* fuzzyRule12 = new FuzzyRule(12, ifERROR_pHpH_E_ZandDE_pHpH_DE_NS, thenM_ASAMLandM_BASAL1);
 fuzzy->addFuzzyRule(fuzzyRule12);

FuzzyRule* fuzzyRule13 = new FuzzyRule(13, ifERROR_pHpH_E_ZandDE_pHpH_DE_Z, thenM_ASAMLandM_BASAL1);
 fuzzy->addFuzzyRule(fuzzyRule13);

FuzzyRule* fuzzyRule14 = new FuzzyRule(14, ifERROR_pHpH_E_ZandDE_pHpH_DE_PS, thenM_ASAMLandM_BASAL1);
 fuzzy->addFuzzyRule(fuzzyRule14);

FuzzyRule* fuzzyRule15 = new FuzzyRule(15, ifERROR_pHpH_E_ZandDE_pHpH_DE_PB, thenM_ASAMAMandM_BASAL1);
 fuzzy->addFuzzyRule(fuzzyRule15);
```

```
//ERROR_pH IS pH.E_PS
FuzzyRule* fuzzyRule16 = new FuzzyRule(16, ifERROR_pHpH_E_PSandDE_pHpH_DE_NB, thenM_ASAMALandM_BASAAM1);
 fuzzy->addFuzzyRule(fuzzyRule16);

FuzzyRule* fuzzyRule17 = new FuzzyRule(17, ifERROR_pHpH_E_PSandDE_pHpH_DE_NS, thenM_ASAMALandM_BASAAM1);
 fuzzy->addFuzzyRule(fuzzyRule17);

FuzzyRule* fuzzyRule18 = new FuzzyRule(18, ifERROR_pHpH_E_PSandDE_pHpH_DE_Z, thenM_ASAMLandM_BASAM1);
 fuzzy->addFuzzyRule(fuzzyRule18);

FuzzyRule* fuzzyRule19 = new FuzzyRule(19, ifERROR_pHpH_E_PSandDE_pHpH_DE_PS, thenM_ASAMLandM_BASAM1);
 fuzzy->addFuzzyRule(fuzzyRule19);

FuzzyRule* fuzzyRule20 = new FuzzyRule(20, ifERROR_pHpH_E_PSandDE_pHpH_DE_PB, thenM_ASAMLandM_BASAAH1);
 fuzzy->addFuzzyRule(fuzzyRule20);

//ERROR_pH IS pH.E_PB
FuzzyRule* fuzzyRule21 = new FuzzyRule(21, ifERROR_pHpH_E_PBandDE_pHpH_DE_NB, thenM_ASAMLandM_BASAAH1);
 fuzzy->addFuzzyRule(fuzzyRule21);

FuzzyRule* fuzzyRule22 = new FuzzyRule(22, ifERROR_pHpH_E_PBandDE_pHpH_DE_NS, thenM_ASAMLandM_BASAAH1);
 fuzzy->addFuzzyRule(fuzzyRule22);

FuzzyRule* fuzzyRule23 = new FuzzyRule(23, ifERROR_pHpH_E_PBandDE_pHpH_DE_Z, thenM_ASAMLandM_BASAH1);
 fuzzy->addFuzzyRule(fuzzyRule23);

FuzzyRule* fuzzyRule24 = new FuzzyRule(24, ifERROR_pHpH_E_PBandDE_pHpH_DE_PS, thenM_ASAMLandM_BASAH1);
 fuzzy->addFuzzyRule(fuzzyRule24);

FuzzyRule* fuzzyRule25 = new FuzzyRule(25, ifERROR_pHpH_E_PBandDE_pHpH_DE_PB, thenM_ASAMLandM_BASAH1);
 fuzzy->addFuzzyRule(fuzzyRule25);


        ////////////////////////////////////
////////////////// PEMBENTUK FUZZY RULE SUHU ////////////////////////
        ////////////////////////////////////

////////////////////// FUZZY ANTECEDENT SUHU ////////////////////
//ERROR_SH = NB//
 FuzzyRuleAntecedent* ifERROR_SHSH_E_NBandDE_SHSH_DE_NB = new FuzzyRuleAntecedent();
  ifERROR_SHSH_E_NBandDE_SHSH_DE_NB ->joinWithAND(SH_E_NB, SH_DE_NB);
 FuzzyRuleAntecedent* ifERROR_SHSH_E_NBandDE_SHSH_DE_NS = new FuzzyRuleAntecedent();
  ifERROR_SHSH_E_NBandDE_SHSH_DE_NS ->joinWithAND(SH_E_NB, SH_DE_NS);
 FuzzyRuleAntecedent* ifERROR_SHSH_E_NBandDE_SHSH_DE_Z = new FuzzyRuleAntecedent();
  ifERROR_SHSH_E_NBandDE_SHSH_DE_Z ->joinWithAND(SH_E_NB, SH_DE_Z);
```

```
FuzzyRuleAntecedent* ifERROR_SHSH_E_NBandDE_SHSH_DE_PS = new FuzzyRuleAntecedent();
 ifERROR_SHSH_E_NBandDE_SHSH_DE_PS  ->joinWithAND(SH_E_NB, SH_DE_PS);
FuzzyRuleAntecedent* ifERROR_SHSH_E_NBandDE_SHSH_DE_PB = new FuzzyRuleAntecedent();
 ifERROR_SHSH_E_NBandDE_SHSH_DE_PB  ->joinWithAND(SH_E_NB, SH_DE_PB);

//ERROR_SH = NS//
FuzzyRuleAntecedent* ifERROR_SHSH_E_NSandDE_SHSH_DE_NB = new FuzzyRuleAntecedent();
 ifERROR_SHSH_E_NSandDE_SHSH_DE_NB  ->joinWithAND(SH_E_NS, SH_DE_NB);
FuzzyRuleAntecedent* ifERROR_SHSH_E_NSandDE_SHSH_DE_NS = new FuzzyRuleAntecedent();
 ifERROR_SHSH_E_NSandDE_SHSH_DE_NS  ->joinWithAND(SH_E_NS, SH_DE_NS);
FuzzyRuleAntecedent* ifERROR_SHSH_E_NSandDE_SHSH_DE_Z = new FuzzyRuleAntecedent();
 ifERROR_SHSH_E_NSandDE_SHSH_DE_Z  ->joinWithAND(SH_E_NS, SH_DE_Z);
FuzzyRuleAntecedent* ifERROR_SHSH_E_NSandDE_SHSH_DE_PS = new FuzzyRuleAntecedent();
 ifERROR_SHSH_E_NSandDE_SHSH_DE_PS  ->joinWithAND(SH_E_NS, SH_DE_PS);
FuzzyRuleAntecedent* ifERROR_SHSH_E_NSandDE_SHSH_DE_PB = new FuzzyRuleAntecedent();
 ifERROR_SHSH_E_NSandDE_SHSH_DE_PB  ->joinWithAND(SH_E_NS, SH_DE_PB);

//ERROR_SH = Z//
FuzzyRuleAntecedent* ifERROR_SHSH_E_ZandDE_SHSH_DE_NB = new FuzzyRuleAntecedent();
 ifERROR_SHSH_E_ZandDE_SHSH_DE_NB  ->joinWithAND(SH_E_Z, SH_DE_NB);
FuzzyRuleAntecedent* ifERROR_SHSH_E_ZandDE_SHSH_DE_NS = new FuzzyRuleAntecedent();
 ifERROR_SHSH_E_ZandDE_SHSH_DE_NS  ->joinWithAND(SH_E_Z, SH_DE_NS);
FuzzyRuleAntecedent* ifERROR_SHSH_E_ZandDE_SHSH_DE_Z = new FuzzyRuleAntecedent();
 ifERROR_SHSH_E_ZandDE_SHSH_DE_Z  ->joinWithAND(SH_E_Z, SH_DE_Z);
FuzzyRuleAntecedent* ifERROR_SHSH_E_ZandDE_SHSH_DE_PS = new FuzzyRuleAntecedent();
 ifERROR_SHSH_E_ZandDE_SHSH_DE_PS  ->joinWithAND(SH_E_Z, SH_DE_PS);
FuzzyRuleAntecedent* ifERROR_SHSH_E_ZandDE_SHSH_DE_PB = new FuzzyRuleAntecedent();
 ifERROR_SHSH_E_ZandDE_SHSH_DE_PB  ->joinWithAND(SH_E_Z, SH_DE_PB);

//ERROR_ZH = PS//
FuzzyRuleAntecedent* ifERROR_SHSH_E_PSandDE_SHSH_DE_NB = new FuzzyRuleAntecedent();
 ifERROR_SHSH_E_PSandDE_SHSH_DE_NB  ->joinWithAND(SH_E_PS, SH_DE_NB);
FuzzyRuleAntecedent* ifERROR_SHSH_E_PSandDE_SHSH_DE_NS = new FuzzyRuleAntecedent();
 ifERROR_SHSH_E_PSandDE_SHSH_DE_NS  ->joinWithAND(SH_E_PS, SH_DE_NS);
FuzzyRuleAntecedent* ifERROR_SHSH_E_PSandDE_SHSH_DE_Z = new FuzzyRuleAntecedent();
 ifERROR_SHSH_E_PSandDE_SHSH_DE_Z  ->joinWithAND(SH_E_PS, SH_DE_Z);
FuzzyRuleAntecedent* ifERROR_SHSH_E_PSandDE_SHSH_DE_PS = new FuzzyRuleAntecedent();
 ifERROR_SHSH_E_PSandDE_SHSH_DE_PS  ->joinWithAND(SH_E_PS, SH_DE_PS);
FuzzyRuleAntecedent* ifERROR_SHSH_E_PSandDE_SHSH_DE_PB = new FuzzyRuleAntecedent();
 ifERROR_SHSH_E_PSandDE_SHSH_DE_PB  ->joinWithAND(SH_E_PS, SH_DE_PB);
```

```
//ERROR_SH = PB//
 FuzzyRuleAntecedent* ifERROR_SHSH_E_PBandDE_SHSH_DE_NB = new FuzzyRuleAntecedent();
  ifERROR_SHSH_E_PBandDE_SHSH_DE_NB  ->joinWithAND(SH_E_PB, SH_DE_NB);
 FuzzyRuleAntecedent* ifERROR_SHSH_E_PBandDE_SHSH_DE_NS = new FuzzyRuleAntecedent();
  ifERROR_SHSH_E_PBandDE_SHSH_DE_NS  ->joinWithAND(SH_E_PB, SH_DE_NS);
 FuzzyRuleAntecedent* ifERROR_SHSH_E_PBandDE_SHSH_DE_Z = new FuzzyRuleAntecedent();
  ifERROR_SHSH_E_PBandDE_SHSH_DE_Z  ->joinWithAND(SH_E_PB, SH_DE_Z);
 FuzzyRuleAntecedent* ifERROR_SHSH_E_PBandDE_SHSH_DE_PS = new FuzzyRuleAntecedent();
  ifERROR_SHSH_E_PBandDE_SHSH_DE_PS  ->joinWithAND(SH_E_PB, SH_DE_PS);
 FuzzyRuleAntecedent* ifERROR_SHSH_E_PBandDE_SHSH_DE_PB = new FuzzyRuleAntecedent();
  ifERROR_SHSH_E_PBandDE_SHSH_DE_PB  ->joinWithAND(SH_E_PB, SH_DE_PB);


////////////////////////   FUZZY CONSEQUENT SUHU  ////////////////////

//M_HANGAT = berhenti
FuzzyRuleConsequent*          thenM_BIASAL2andM_HANGATL3          =          new
FuzzyRuleConsequent();
thenM_BIASAL2andM_HANGATL3    ->addOutput(L2    );
thenM_BIASAL2andM_HANGATL3    ->addOutput(L3    );

FuzzyRuleConsequent*          thenM_BIASAH2andM_HANGATL3          =          new
FuzzyRuleConsequent();
thenM_BIASAH2andM_HANGATL3    ->addOutput(H2    );
thenM_BIASAH2andM_HANGATL3    ->addOutput(L3    );

FuzzyRuleConsequent*          thenM_BIASAAH2andM_HANGATL3          =          new
FuzzyRuleConsequent();
thenM_BIASAAH2andM_HANGATL3    ->addOutput(AH2    );
thenM_BIASAAH2andM_HANGATL3    ->addOutput(L3    );

FuzzyRuleConsequent*          thenM_BIASAM2andM_HANGATL3          =          new
FuzzyRuleConsequent();
thenM_BIASAM2andM_HANGATL3    ->addOutput(M2    );
thenM_BIASAM2andM_HANGATL3    ->addOutput(L3    );

FuzzyRuleConsequent*          thenM_BIASAAM2andM_HANGATL3          =          new
FuzzyRuleConsequent();
thenM_BIASAAM2andM_HANGATL3    ->addOutput(AM2    );
thenM_BIASAAM2andM_HANGATL3    ->addOutput(L3    );

//M_BIASA = berhenti
FuzzyRuleConsequent*          thenM_BIASAL2andM_HANGATAM3          =          new
FuzzyRuleConsequent();
thenM_BIASAL2andM_HANGATAM3    ->addOutput(L2    );
thenM_BIASAL2andM_HANGATAM3    ->addOutput(AM3    );

FuzzyRuleConsequent*          thenM_BIASAL2andM_HANGATM3          =          new
FuzzyRuleConsequent();
```

```
thenM_BIASAL2andM_HANGATM3    ->addOutput(L2    );
thenM_BIASAL2andM_HANGATM3    ->addOutput(M3    );

FuzzyRuleConsequent*          thenM_BIASAL2andM_HANGATAH3          =          new
FuzzyRuleConsequent();
thenM_BIASAL2andM_HANGATAH3   ->addOutput(L2    );
thenM_BIASAL2andM_HANGATAH3   ->addOutput(AH3   );

FuzzyRuleConsequent*          thenM_BIASAL2andM_HANGATH3           =          new
FuzzyRuleConsequent();
thenM_BIASAL2andM_HANGATH3    ->addOutput(L2    );
thenM_BIASAL2andM_HANGATH3    ->addOutput(H3    );


//////////// FUZZY RULE YG DIBENTUK DARI ANTECENDENT DENGAN
CONSEQUENT SUHU//////////////

////////////////////////// FUZZY RULE SUHU ////////////////////////////////
//ERROR_pH IS pH.E_NB
 FuzzyRule* fuzzyRule26 = new FuzzyRule(26, ifERROR_SHSH_E_NBandDE_SHSH_DE_NB,
thenM_BIASAH2andM_HANGATL3);
  fuzzy->addFuzzyRule(fuzzyRule26);

 FuzzyRule* fuzzyRule27 = new FuzzyRule(27, ifERROR_SHSH_E_NBandDE_SHSH_DE_NS,
thenM_BIASAH2andM_HANGATL3);
   fuzzy->addFuzzyRule(fuzzyRule27);

 FuzzyRule* fuzzyRule28 = new FuzzyRule(28, ifERROR_SHSH_E_NBandDE_SHSH_DE_Z,
thenM_BIASAH2andM_HANGATL3);
  fuzzy->addFuzzyRule(fuzzyRule28);

 FuzzyRule* fuzzyRule29 = new FuzzyRule(29, ifERROR_SHSH_E_NBandDE_SHSH_DE_PS,
thenM_BIASAAH2andM_HANGATL3);
  fuzzy->addFuzzyRule(fuzzyRule29);

 FuzzyRule* fuzzyRule30 = new FuzzyRule(30, ifERROR_SHSH_E_NBandDE_SHSH_DE_PB,
thenM_BIASAAH2andM_HANGATL3);
  fuzzy->addFuzzyRule(fuzzyRule30);

//ERROR_pH IS pH.E_NS
 FuzzyRule* fuzzyRule31 = new FuzzyRule(31, ifERROR_SHSH_E_NSandDE_SHSH_DE_NB,
thenM_BIASAH2andM_HANGATL3);
  fuzzy->addFuzzyRule(fuzzyRule31);

 FuzzyRule* fuzzyRule32 = new FuzzyRule(32, ifERROR_SHSH_E_NSandDE_SHSH_DE_NS,
thenM_BIASAM2andM_HANGATL3);
  fuzzy->addFuzzyRule(fuzzyRule32);

 FuzzyRule* fuzzyRule33 = new FuzzyRule(33, ifERROR_SHSH_E_NSandDE_SHSH_DE_Z,
thenM_BIASAM2andM_HANGATL3);
  fuzzy->addFuzzyRule(fuzzyRule33);

 FuzzyRule* fuzzyRule34 = new FuzzyRule(34, ifERROR_SHSH_E_NSandDE_SHSH_DE_PS,
thenM_BIASAAM2andM_HANGATL3);
  fuzzy->addFuzzyRule(fuzzyRule34);
```

```
FuzzyRule* fuzzyRule35 = new FuzzyRule(35, ifERROR_SHSH_E_NSandDE_SHSH_DE_PB, thenM_BIASAAM2andM_HANGATL3);
  fuzzy->addFuzzyRule(fuzzyRule35);

//ERROR_pH IS pH.E_Z
 FuzzyRule* fuzzyRule36 = new FuzzyRule(36, ifERROR_SHSH_E_ZandDE_SHSH_DE_NB, thenM_BIASAL2andM_HANGATAM3);
  fuzzy->addFuzzyRule(fuzzyRule36);

 FuzzyRule* fuzzyRule37 = new FuzzyRule(37, ifERROR_SHSH_E_ZandDE_SHSH_DE_NS, thenM_BIASAL2andM_HANGATL3);
  fuzzy->addFuzzyRule(fuzzyRule37);

 FuzzyRule* fuzzyRule38 = new FuzzyRule(38, ifERROR_SHSH_E_ZandDE_SHSH_DE_Z, thenM_BIASAL2andM_HANGATL3);
  fuzzy->addFuzzyRule(fuzzyRule38);

 FuzzyRule* fuzzyRule39 = new FuzzyRule(39, ifERROR_SHSH_E_ZandDE_SHSH_DE_PS, thenM_BIASAL2andM_HANGATL3);
  fuzzy->addFuzzyRule(fuzzyRule39);

 FuzzyRule* fuzzyRule40 = new FuzzyRule(40, ifERROR_SHSH_E_ZandDE_SHSH_DE_PB, thenM_BIASAAM2andM_HANGATL3);
  fuzzy->addFuzzyRule(fuzzyRule40);

//ERROR_pH IS pH.E_PS
 FuzzyRule* fuzzyRule41 = new FuzzyRule(41, ifERROR_SHSH_E_PSandDE_SHSH_DE_NB, thenM_BIASAL2andM_HANGATAM3);
  fuzzy->addFuzzyRule(fuzzyRule41);

 FuzzyRule* fuzzyRule42 = new FuzzyRule(42, ifERROR_SHSH_E_PSandDE_SHSH_DE_NS, thenM_BIASAL2andM_HANGATAM3);
  fuzzy->addFuzzyRule(fuzzyRule42);

 FuzzyRule* fuzzyRule43 = new FuzzyRule(43, ifERROR_SHSH_E_PSandDE_SHSH_DE_Z, thenM_BIASAL2andM_HANGATM3);
  fuzzy->addFuzzyRule(fuzzyRule43);

 FuzzyRule* fuzzyRule44 = new FuzzyRule(44, ifERROR_SHSH_E_PSandDE_SHSH_DE_PS, thenM_BIASAL2andM_HANGATM3);
  fuzzy->addFuzzyRule(fuzzyRule44);

 FuzzyRule* fuzzyRule45 = new FuzzyRule(45, ifERROR_SHSH_E_PSandDE_SHSH_DE_PB, thenM_BIASAL2andM_HANGATAH3);
  fuzzy->addFuzzyRule(fuzzyRule45);

//ERROR_pH IS pH.E_PB
 FuzzyRule* fuzzyRule46 = new FuzzyRule(46, ifERROR_SHSH_E_PBandDE_SHSH_DE_NB, thenM_BIASAL2andM_HANGATAH3);
  fuzzy->addFuzzyRule(fuzzyRule46);

 FuzzyRule* fuzzyRule47 = new FuzzyRule(47, ifERROR_SHSH_E_PBandDE_SHSH_DE_NS, thenM_BIASAL2andM_HANGATAH3);
  fuzzy->addFuzzyRule(fuzzyRule47);

 FuzzyRule* fuzzyRule48 = new FuzzyRule(48, ifERROR_SHSH_E_PBandDE_SHSH_DE_Z, thenM_BIASAL2andM_HANGATH3);
  fuzzy->addFuzzyRule(fuzzyRule48);
```

```
FuzzyRule* fuzzyRule49 = new FuzzyRule(49, ifERROR_SHSH_E_PBandDE_SHSH_DE_PS, thenM_BIASAL2andM_HANGATH3);
  fuzzy->addFuzzyRule(fuzzyRule49);

 FuzzyRule* fuzzyRule50 = new FuzzyRule(50, ifERROR_SHSH_E_PBandDE_SHSH_DE_PB, thenM_BIASAL2andM_HANGATH3);
  fuzzy->addFuzzyRule(fuzzyRule50);

}

void loop(void)
{

 //////////MAIN SENSOR PH//////////////////

  for(int i=0;i<10;i++)       //Get 10 sample value from the sensor for smooth the value
 {
  data[i]=analogRead(SensorPin);
  delay(10);
 }
 for(int i=0;i<9;i++)        //sort the analog from small to large
 {
  for(int j=i+1;j<10;j++)
  {
   if(data[i] > data[j])
   {
    temp= data[i];
    data[i] = data[j];
    data[j]=temp;
   }
  }
 }
 avgValue=0;
 for(int i=2;i<8;i++)                    //take the average value of 6 center sample
  avgValue+= data[i];
 float SensorValue=(float)avgValue*5.0/(1024*6); //convert the analog into volt
 float pHValue = 0;
 pHValue= (SensorValue+0.0922)/0.2983;

//////////UNTUK BACA SEN.SUHU DS18B20//// //////
 sensors.requestTemperatures(); // Send the command to get temperatures
 float tempU = sensors.getTempCByIndex(0); // suhu yang terbaca
 SuhuValue = (tempU + 2.1547)/1.0943;

/////////input ERROR & DELTA_ERROR pH////////
 ERROR_pH = setpointpH-pHValue;
 DE_pH = ERROR_pH-lastErrorpH;

////////input ERROR & DELTA_ERROR SUHU//////////
 ERROR_SH = setpointSH-SuhuValue;
 DE_SH = ERROR_SH-lastErrorSH;
```

```
/////FUZZY LOGIC CONTROL/////////////
  fuzzy->setInput(1, ERROR_pH);
  fuzzy->setInput(2, DE_pH);
  fuzzy->setInput(3, ERROR_SH);
  fuzzy->setInput(4, DE_SH);

  fuzzy->fuzzify();

  output1 = fuzzy->defuzzify(1); //M_ASAM
  output2 = fuzzy->defuzzify(2); //M_BASA
  output3 = fuzzy->defuzzify(3); //M_BIASA
  output4 = fuzzy->defuzzify(4); //M_HANGAT

//AKTUATOR
  Motor_1();
  Motor_2();
  Motor_3();
  Motor_4();

// TAMPILAN LCD
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("pH");
  lcd.setCursor(4,0);
  lcd.print(pHValue);
  lcd.setCursor(0,1);
  lcd.print("pwm1");
  lcd.setCursor(5,1);
  lcd.print(output1);
  lcd.setCursor(9,1);
  lcd.print("pwm2");
  lcd.setCursor(14,1);
  lcd.print(output2);
  lcd.setCursor(0,2);
  lcd.print("SV1");
  lcd.setCursor(4,2);
  lcd.print(stepValue1);

  lcd.setCursor(8,2);
  lcd.print("SV2");
  lcd.setCursor(12,2);
  lcd.print(stepValue2);

  lcd.setCursor(0,3);
  lcd.print("PB1");
  lcd.setCursor(4,3);
  lcd.print(balikstep1);

  lcd.setCursor(8,3);
  lcd.print("PB2");
```

```
  lcd.setCursor(12,3);
  lcd.print(balikstep2);

//INISIALISASI LAST ERROR pH & SUHU////
  lastErrorpH = ERROR_pH;
  lastErrorSH = ERROR_SH;

//INISIALISASI LAST STEP
  lastStep1 = currentStep1;
  lastStep2 = currentStep2;
  lastStep3 = currentStep3;
  lastStep4 = currentStep4;

  delay(1000);
}
void Motor_1(){
  currentStep1 = output1;
  stepValue1 = currentStep1 - lastStep1;
  balikstep1 =(int) stepValue1*(-1);
  if(stepValue1 >= 0){
    digitalWrite(MotorDir1,LOW); // Enables the motor to move in a particular direction
    // Makes 200 pulses for making one full cycle rotation
    for(int i = 0; i < stepValue1; i++) {
      digitalWrite(MotorStep1,HIGH);
      delayMicroseconds(2000); //delay dalam microdetik
      digitalWrite(MotorStep1,LOW);
      delayMicroseconds(2000);
    }
  }
  else if(stepValue1 < 0){
    digitalWrite(MotorDir1,HIGH); // Enables the motor to move in a particular direction
    // Makes 200 pulses for making one full cycle rotation
    for(int i = 0; i < balikstep1; i++) {
      digitalWrite(MotorStep1,HIGH);
      delayMicroseconds(2000); //delay dalam microdetik
      digitalWrite(MotorStep1,LOW);
      delayMicroseconds(2000);
    }
  }
}

void Motor_2(){
  currentStep2 = output2;
  stepValue2 = currentStep2-lastStep2;
  balikstep2 = (int) stepValue2*(-1);
  if(stepValue2 >= 0){
    digitalWrite(MotorDir2,LOW); // Enables the motor to move in a particular direction
    // Makes 200 pulses for making one full cycle rotation
    for(int i = 0; i < stepValue2; i++) {
      digitalWrite(MotorStep2,HIGH;
```

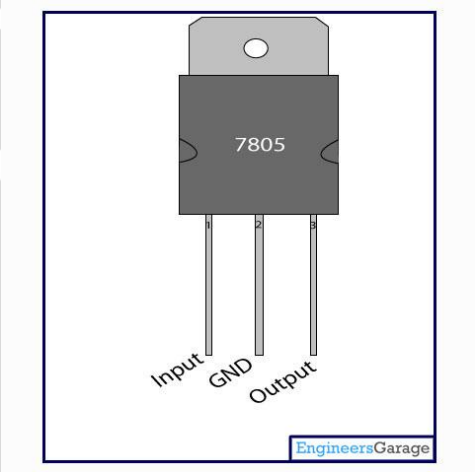
```
delayMicroseconds(2000); //delay dalam microdetik
      digitalWrite(MotorStep2,LOW);
      delayMicroseconds(2000);
    }
  }
  else if(stepValue2 < 0){
    digitalWrite(MotorDir2,HIGH); // Enables the motor to move in a particular direction
    // Makes 200 pulses for making one full cycle rotation
    for(int i = 0; i < balikstep2; i++) {
      digitalWrite(MotorStep2,HIGH);
      delayMicroseconds(2000); //delay dalam microdetik
      digitalWrite(MotorStep2,LOW);
      delayMicroseconds(2000);
    }
  }
}
void Motor_3(){
  currentStep3 = output3;
  stepValue3 = currentStep3 - lastStep3;
  balikstep3 =(int) stepValue3*(-1);
  if(stepValue3 >= 0){
    digitalWrite(MotorDir3,LOW); // Enables the motor to move in a particular direction
    // Makes 200 pulses for making one full cycle rotation
    for(int i = 0; i < stepValue3; i++) {
      digitalWrite(MotorStep3,HIGH);
      delayMicroseconds(2000); //delay dalam microdetik
      digitalWrite(MotorStep3,LOW);
      delayMicroseconds(2000);
    }
  }
  else if(stepValue3 < 0){
    digitalWrite(MotorDir3,HIGH); // Enables the motor to move in a particular direction
    // Makes 200 pulses for making one full cycle rotation
    for(int i = 0; i < balikstep3; i++) {
      digitalWrite(MotorStep3,HIGH);
      delayMicroseconds(2000); //delay dalam microdetik
      digitalWrite(MotorStep3,LOW);
      delayMicroseconds(2000);
    }
  }
}
void Motor_4(){
  currentStep4 = output4;
  stepValue4 = currentStep4 - lastStep4;
  balikstep4 = (int) stepValue4*(-1);
  if(stepValue4 >= 0){
    digitalWrite(MotorDir4,LOW); // Enables the motor to move in a particular direction
    // Makes 200 pulses for making one full cycle rotation
    for(int i = 0; i < stepValue4; i++) {
      digitalWrite(MotorStep4,HIGH);
```

```
delayMicroseconds(2000); //delay dalam microdetik
    digitalWrite(MotorStep4,LOW);
    delayMicroseconds(2000);
   }
  }
  else if(stepValue4 < 0){
   digitalWrite(MotorDir4,HIGH); // Enables the motor to move in a particular direction
   // Makes 200 pulses for making one full cycle rotation
   for(int i = 0; i < balikstep4; i++) {
    digitalWrite(MotorStep4,HIGH);
    delayMicroseconds(2000); //delay dalam microdetik
    digitalWrite(MotorStep4,LOW);
    delayMicroseconds(2000);
   }
  }
}
```

LAMPIRAN III
DATASHEET

Datasheet IC7805

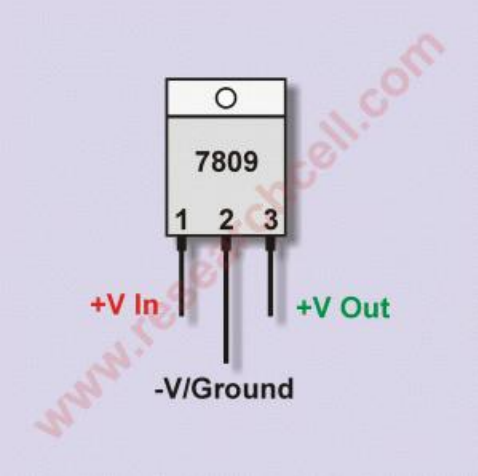
Pin diagram IC7805



Pin Description IC 7805:

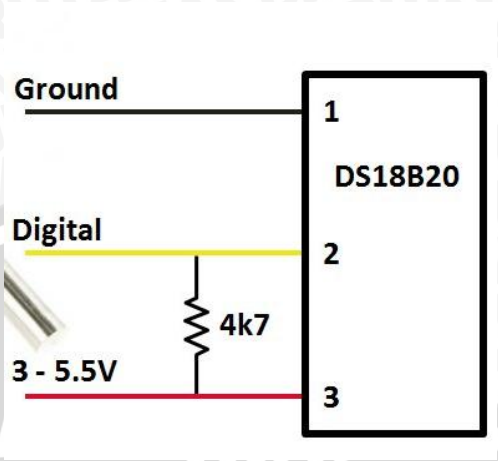
Pin No	Function	Name
1	Input voltage (5V-18V)	Input
2	Ground (0V)	Ground
3	Regulated output; 5V (4.8V-5.2V)	Output

Datasheet IC7805



Pin No	Function	Name
1	Input voltage (5V-23V)	Input
2	Ground (0V)	Ground
3	Regulated output; 5V (8.8V-9.2V)	Output

Datasheet DS18B20



FEATURES

- Unique 1-Wire interface requires only one port pin for communication
- Multidrop capability simplifies distributed temperature sensing applications
- Requires no external components
- Can be powered from data line. Power supply range is 3.0V to 5.5V
- Zero standby power required
- Measures temperatures from -55°C to +125°C. Fahrenheit equivalent is -67°F to +257°F
- 0.5°C accuracy from -10°C to +85°C
- Thermometer resolution is programmable from 9 to 12 bits
- Converts 12-bit temperature to digital word in 750 ms (max.)
- User-definable, nonvolatile temperature alarm settings
- Alarm search command identifies and addresses devices whose temperature is outside of programmed limits (temperature alarm condition)
- Applications include thermostatic controls, industrial systems, consumer products, thermometers, or any thermally sensitive system

PIN DESCRIPTION

GND - Ground
DQ - Data In/Out
VDD - Power Supply Voltage

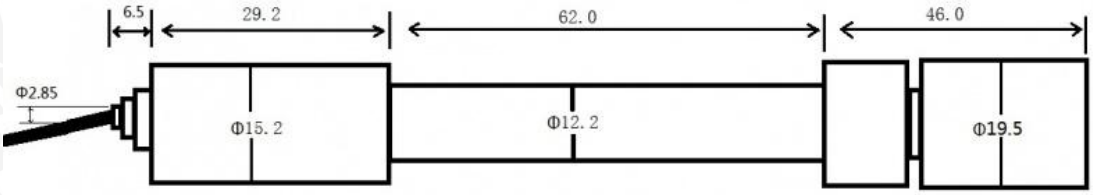
DESCRIPTION

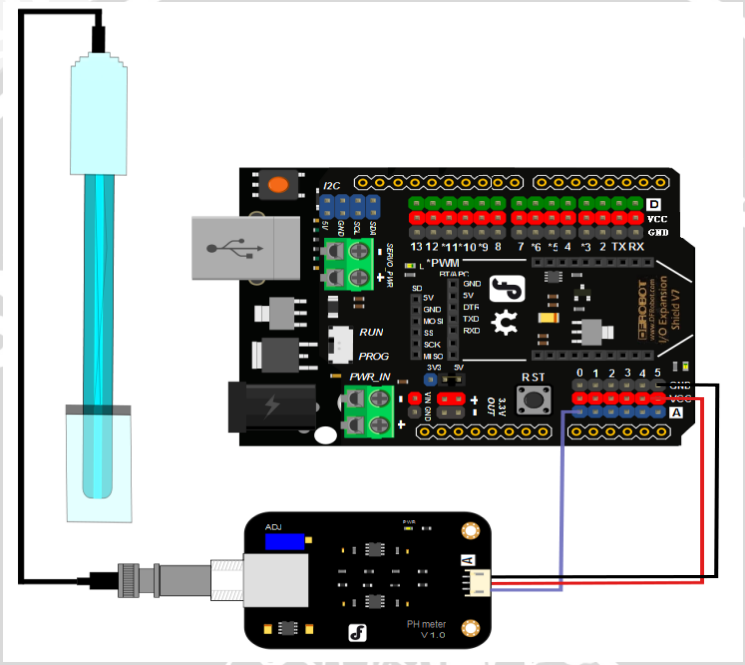
The DS18B20 Digital Thermometer provides 9 to 12-bit (configurable) temperature readings which indicate the temperature of the device.

Information is sent to/from the DS18B20 over a 1-Wire interface, so that only one wire (and ground) needs to be connected from a central microprocessor to a DS18B20. Power for reading, writing, and performing temperature conversions can be derived from the data line itself with no need for an external power source.

Because each DS18B20 contains a unique silicon serial number, multiple DS18B20s can exist on the same 1-Wire bus. This allows for placing temperature sensors in many different places. Applications where this feature is useful include HVAC environmental controls, sensing temperatures inside buildings, equipment or machinery, and process monitoring and control.

Datasheet SKU SEN0161





Specification Module Power : 5.00V Module Size : 43mm×32mm Measuring Range:0-14PH Measuring Temperature :060 ℃ 17/12/2014 PH meter(SKU: SEN0161) Robot Wiki http://dfrobot.com/wiki/index.php/PH_meter(SKU:_SEN0161) 2/6 Accuracy : ± 0.1pH (25 ℃) Response Time : ≤ 1min pH Sensor with BNC Connector PH2.0 Interface (3 foot patch) Gain Adjustment Potentiometer Power Indicator LED Cable Length from sensor to BNC connector:660mm pH Electrode Size pH Electrode Characteristics The output of pH electrode is Millivolts,and the pH value of the relationship is shown as follows (25℃).