TUGAS AKHIR – TF 145565

# RANCANG BANGUN PENGENDALIAN VARIABLE SPEED DRIVE PADA POMPA SUPPLY AIR DINGIN PADA ALAT PENUKAR PANAS

ENVER RAFID SADIDA
10 51 15 000 00 019

Dosen Pembimbing:
Dr. Ir. Totok Soehartanto, DEA.
Ahmad Fauzan 'Adziima, S.T., M.Sc.

PROGRAM STUDI DIII TEKNOLOGI INSTRUMENTASI
DEPARTEMEN TEKNIK INSTRUMENTASI
Fakultas Vokasi
Institut Teknologi Sepuluh Nopember
Surabaya 2018

**TUGAS AKHIR – TF 145565**

# RANCANG BANGUN PENGENDALIAN VARIABLE SPEED DRIVE PADA POMPA SUPPLY AIR DINGIN PADA ALAT PENUKAR PANAS

ENVER RAFID SADIDA
10 51 15 000 00 019

Dosen Pembimbing:
Dr. Ir. Totok Soehartanto, DEA.
Ahmad Fauzan 'Adziima, S.T., M.Sc.

PROGRAM STUDI DIII TEKNOLOGI INSTRUMENTASI
DEPARTEMEN TEKNIK INSTRUMENTASI
Fakultas Vokasi
Institut Teknologi Sepuluh Nopember
Surabaya 2018

*FINAL PROJECT – TF 145565*

# DESIGN AND CONTROL OF VARIABLE SPEED DRIVE CONTROL IN COLD WATER SUPPLY PUMP IN HEAT EXCHANGER

ENVER RAFID SADIDA
10 51 15 000 00 019

Supervisors:
Dr. Ir. Totok Soehartanto, DEA.
Ahmad Fauzan 'Adziima, S.T., M.Sc.

DIII PROGRAM OF INSTRUMENTATION TECHNOLOGY
INSTRUMENTATION ENGINEERING DEPARTMENT
Faculty of Vocation
Institut Teknologi Sepuluh Nopember
Surabaya 2018

**LEMBAR PENGESAHAN i**

**"RANCANG BANGUN PENGENDALIAN VARIABLE SPEED DRIVE PADA POMPA SUPPLY AIR DINGIN PADA ALAT PENUKAR PANAS"**

**TUGAS AKHIR**

**Oleh :**
**ENVER RAFID SADIDA**
**NRP.10511500000019**

**Surabaya, 16 Juli 2018**
**Mengetahui / Menyetujui**

Dosen Pembimbing I

Dr.Ir. Totok Soehartanto, DEA
NIP. 19650309 199002 1 001

Dosen Pembimbing II

Ahmad Fauzan 'Adziimaa, ST,MT
NIP. 199120171 1052

Kepala Departemen
Teknik Instrumentasi FV-ITS

Dr. Ir. Purwadi Agus Darwito, MSc
NIP. 19620822 198803 1 001

## LEMBAR PENGESAHAN II

# RANCANG BANGUN PENGENDALIAN VARIABLE SPEED DRIVE PADA POMPA SUPPLY AIR DINGIN PADA ALAT PENUKAR PANAS

**TUGAS AKHIR**

Disusun untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Ahli Madya
pada
Departemen Teknik Instrumentasi
Fakultas Vokasi
Institut Teknologi Sepuluh Nopember

Oleh:

**ENVER RAFID SADIDA**
**NRP 10 51 15 000 00 019**

Disetujui Tim Penguji: Tanggal Ujian: 16 Juli 2018
Periode Wisuda: September 2018

**Dr. Ir. Totok Soehartanto, DEA** ..... **(Pembimbing I)**

**Ahmad Fauzan 'Adziima, S.T.,M.Sc.** ..... **(Pembimbing II)**

**Ir. Tutug Danardono, M.T.** ..... **(Penguji)**

**RANCANG BANGUN PENGENDALIAN VARIABLE SPEED DRIVE PADA POMPA SUPPLY AIR DINGIN PADA ALAT PENUKAR PANAS**


**Nama** **: Enver Rafid Sadida**
**NRP** **: 10511500000019**
**Departemen** **: Teknik Instrumentasi FV-ITS**


## *ABSTRAK*


*Heat Exchanger* merupakan alat penukar kalor yang berfungsi untuk mengubah *temperature* dan fasa suatu jenis fluida. Proses tersebut terjadi dengan memanfaatkan proses perpindahan kalor dari *fluida temperature* tinggi menuju *fluida temperature* rendah. Pada penelitian ini, difokuskan untuk memaksimalkan pengendalian *flow* air dingin yang akan masuk ke alat *heat exchanger*. Secara umum cara kerjanya yaitu jika *flow* dingin yang masuk ke *inlet Heat exchanger* memiliki debit yang lebih rendah dari keluaran debit panas, maka campuran dari air tersebut akan memiliki suhu berbeda. Begitupun juga sebaliknya. Cara yang digunakan untuk memanipulasi *flow* adalah dengan mengatur kecepatan dari pompa dengan menggunakan *Variable Speed Drive*. Variable Speed Drive adalah suatu piranti elektronik yang dapat memanipulasi frekuensi alirn listrik. Untuk mencapai laju aliran 14 liter/meint dibutuhkan frekuensi 50hz sedangan kan untuk mencapai laju aliran 7 liter/menit dibutuhkan frekuensi 30hz.



**Kata Kunci :** *Heat Exchanger, Flow, Temperature, pompa, Variable Speed Drive*

***DESIGN AND CONTROL OF VARIABLE SPEED DRIVE CONTROL IN COLD WATER SUPPLY PUMP IN HEAT EXCHANGER***

**Name : Enver Rafid Sadida**
**NRP : 10511500000019**
**Department : Instrument Engineering FV-ITS**

***ABSTRACT***

*Heat Exchanger heat exchanger is a tool that serves to change the temperature of a fluid type and phase. The process occurs by utilizing a heat transfer processes of fluid temperature high temperature fluid low. In this study, focused to maximize control of the flow of cold water that would enter the appliance heat exchanger. In general how it works i.e. If flow inlet to the cold Heat exchanger has a lower discharge of heat discharge output, then a mix of the water will have different temperatures. As well as vice versa. The way used to manipulate flow is to set the speed of the pump by using a Variable Speed Drive. Variable Speed Drive is an electronic device can manipulate the frequency alirn electric. To achieve the flow rate 14 litres/50 Hz frequency is required while meint right to achieve the flow rate 7 liters/min required frequency 30 Hz .*

***Keywords*** **:** *Heat Exchanger, Flow, Temperature, Pump, Variable Speed Drive*

# KATA PENGANTAR

Puji syukur kehadirat Allah SWT atas berkat, rahmat dan kebesaran-Nya sehingga penulis dapat menyelesaikan laporan tugas akhir yang berjudul:

**"RANCANG BANGUN PENGENDALIAN VARIABLE SPEED DRIVE PADA POMPA SUPPLY AIR DINGIN KE ALAT PENUKAR PANAS"**

Tugas Akhir ini disusun guna memenuhi persyaratan untuk memperoleh gelar Diploma pada Departemen Teknik Instrumentasi, Fakultas Vokasi, Institut Sepuluh Nopember Surabaya. Selama menyelesaikan tugas akhir ini penulis telah banyak mendapatkan bantuan dari berbagai pihak. Oleh karena itu pada kesempatan ini penulis ingin mengucapkan terima kasih yang sebesar-besarnya kepada:

1. Keluarga tercinta, orangtua penulis terima kasih atas dorongan semangatnya, bantuan dan dukungannya selama ini sehingga laporan ini dapat selesai dengan baik.
2. Bapak Dr. Ir. Purwadi Agus Darwito, MSc selaku kepala departemen teknik instrumenatsi ITS, Surabaya.
3. Bapak Dr.Ir. Totok Soehartanto, DEA selaku dosen pembimbing pertama Tugas Akhir.
4. Bapak Ahmad Fauzan 'Adziimaa, ST,MT selaku dosen pembimbing kedua Tugas Akhir.
5. Muhammad Hidayat yang membantu dalam hal memberi saran dan membimbing dalam Tugas Akhir ini.
6. Rekan-rekan team tugas akhir Heat exchanger atas kekompakan dan kerjasamanya.
7. Teman-teman D3 teknik Instrumentasi angkatan 2015 yang membantu dan mensupport selama kegiatan Tugas Akhir berlangsung.
8. Seluruh karyawan dan staff Departemen Teknik Instrumentasi yang tidak dapat kami sebutkan satu persatu.

Penulis menyadari bahwa kesempurnaan hanya milik Allah SWT. Oleh karena itu, penulis sangat berterimakasih atas segala masukan, kritik dan saran yang membangun dari pembaca

agar laporan ini menjadi lebaih baik lagi untuk di kemudisn hari. Demikian laporan ini penulis buat, semoga laporan ini dapat memberikan manfaat selain bagi penulis sendiri, dan bagi pembaca sekalian.

Surabaya, 16 Juli 2018
Penulis

Enver Rafid Sadida
NRP. 10 5115 00000 019

# DAFTAR ISI

## DAFTAR GAMBAR

## DAFTAR TABEL

# BAB I
# PENDAHULUAN

## 1.1 Latar Belakang

Heat Exchanger adalah alat penukar kalor yang berfungsi untuk mengubah temperatur dan fasa suatu jenis fluida. Proses tersebut terjadi dengan memanfaatkan proses perpindahan kalor dari fluida bersuhu tinggi menuju fluida bersuhu rendah. Di dalam dunia industri peran dari heat exchanger sangat penting. Misal dalam industri pembangkit tenaga listrik, heat exchanger berperan dalam peningkatan efisiensi sistem. Contohnya adalah ekonomizer, yaitu alat penukar kalor yang berfungsi memanaskan feed water sebelum masuk ke boiler menggunakan panas dari exhaust gas (gas buang). Selain itu heat exchanger juga merupakan komponen utama dalam sistem mesin pendingin, yaitu berupa evaporator dan condenser.[1]

Pada simulator alat penukar panas bertipe *shell and tube* dipergunakan air panas yang diproduksi dari tangki pemanas dan juga air dingin yang diproduksi dari tangki pendingin, alat penukar panas ini dapat dioprasikan untuk proses pemanasan air dingin mempergunakan air panas dan proses pendinginan air panas menggunakan air dingin. Untuk itu diperlukan perancangan sistem pengendalian *mass flow rate* pada *supply* air panas dan *mass flow rate* pada air dingin.

Pada tugas akhir ini akan dilakukan perancangan *supply* air dingin yang dapat diatur *flow rate*nya melalui pengaturan RPM pompa *supply* air dingin dengan menggunakan *variable speed drive.* Untuk dapat mengatur *mass flow rate* melalui kecepatan putar pompa diperlukan sensor *flow rate* dan temperatur untuk diproses oleh mikrokontroler guna dipergunakan menjadi sinyal untuk mengatur kecepatan putar agar sesuai dengan fungsi proses yang dijalankan.

## 1.2 Rumusan Masalah

Pada pelaksanaan tugas akhir ini, permasalahan yang diangkat adalah :

a. Bagaimana merancang pengendalian *flow rate* untuk memanipulasi temperatur?
b. Bagaimana merancang *Variable Speed Drive* untuk dapat

mengatur pompa sentrifugal agar dapat mengahasilkan laju aliran yang sesuai?

## 1.3 Tujuan

Tujuan yang dicapai dalam tugas akhir ini adalah :

a. Mengetahui perancangan pengendalian *flow rate* untuk memanipulasi temperatur

**b.** Mengetahui perancangan *Variable Speed Drive* untuk dapat mengatur pompa sentrifugal agar dapat mengahasilkan laju aliran yang sesuai

## 1.4 Ruang Lingkup Tugas Akhir

Adapun ruang lingkup dari tugas akhir ini terdiri dari:

a. Pembuatan skeneraio proses pemansan dan pendinginan.
b. Pernacangan *temeprature transmitter* dan *flow transmitter* pada jalur *supply* air dingin ke alat alat penukar panas.
c. Perancangan alat penegndali kecepatan pompa sentrifugal agar dapat mengahsilkan laju aliran tertentu.

# BAB II
# DASAR TEORI

## 2.1 Heat Exchanger

Perpindahan kalor dari suatu zat ke zat lain seringkali terjadi dalam kehidupan sehari-hari baik penyerapan atau pelepasan kalor, untuk mencapai dan mempertahankan keadaan yang dibutuhkan sewaktu proses berlangsung. Kalor sendiri adalah salah satu bentuk energi. Hukum kekekalan energi menyatakan bahwa energi tidak musnah, contohnya hukum kekekalan massa dan momentum, ini artinya kalor tidak hilang. Energi hanya berubah bentuk dari bentuk yang pertama ke bentuk yang ke dua. Kalor dapat berpindah dengan tiga macam cara yaitu:

1. Pancaran, sering juga dinamakan radiasi.
2. Hantaran, sering juga disebut konduksi.
3. Aliran, sering juga disebut konveksi

Heat Exchanger adalah alat penukar kalor yang berfungsi untuk mengubah temperatur dan fasa suatu jenis fluida. Proses tersebut terjadi dengan memanfaatkan proses perpindahan kalor dari fluida bersuhu tinggi menuju fluida bersuhu rendah. Di dalam dunia industri peran dari heat exchanger sangat penting. Misal dalam industri pembangkit tenaga listrik, heat exchanger berperan dalam peningkatan efisiensi sistem. Contohnya adalah ekonomizer, yaitu alat penukar kalor yang berfungsi memanaskan feed water sebelum masuk ke boiler menggunakan panas dari exhaust gas (gas buang). Selain itu heat exchanger juga merupakan komponen utama dalam sistem mesin pendingin, yaitu berupa evaporator dan condenser.

Dalam perkembangannya heat exchanger mengalami transformasi bentuk yang bertujuan meningkatkan efisiensi sesuai dengan fungsi kerjanya. Bentuk heat exchanger yang sering digunakan ialah shell and tube. Dengan berbagai pertimbangan bentuk ini dinilai memiliki banyak keuntungan baik dari segi fabrikasi, biaya, hingga unjuk kerja.

Heat exchanger merupakan media vital didalam dunia industri. Untuk itu dalam tugas akhir ini direncanakan sebuah heat exchanger model shell and tube sederhana namun tetap mengacu

pada kaidah desain yang ada. Sehingga didapat keuntungan sebagai metode pembelajaran mengenai proses desain, mekanisme kerja, hingga unjuk kerja heat exchanger.[1]

## 2.2 Sensor *Termocouple*

Thermocouple merupakan sambungan dua jenis logam atau campuran yang salah satu sambungan logam tadi diberi perlakuan suhu yang berbeda dengan sambungan lainnya. Sambungan logam pada thermocouple terdiri dari dua sambungan, yaitu:

1. *Refrence Junction,* merupakan sambungan acuan yang suhunya konstan dan biasanya diberi suhu yang dingin.
2. *Measuring Junction,* merupakan sambungan yang dipakai untuk mengukur suhu.

Pada dunia elektronika, thermocouple adalah sensor suhu yang banyak digunakan untuk perbedaan suhu dalaam benda menjadi perubahan tegangan listrik (*voltage).* Thermocouple yang sederhana dapat dipasang, dan memiliki konektor standart yang sama, serta dapat mengukur suhu dalam jangkauan suhu yang cukup besar dengan batas kesalahan pengukuran kurang dari 1 ℃.pada banyak aplikas, salah satu sambungan (sambungan yang dingin) dijaga sebagai suhu refrensi, sedang yang lain dihubungkan pada objek pengukuran. Thermocouple dapat dihubungkan secara seri satu sama lain untuk membuat termopile, dimana tiap sambungan yang panas diarahkan ke suhu yang lebih tinggi dan semua sambungan dingin ke suhu lebih rendah. Dengan begitu, tegangan pada setiap thermocouple menjadi naik, yang memungkinkan untuk digunakan pada tegangan yang lebih tinggi. Dengan adanya suhu tetapan pada sambungan dingin, yang berguna untuk pengkuruan di laboratorium, secara sederhana thermocouple tidak mudah digunakan untuk kebanyakan indikasi sambunganang langsung dan instrumen kontrol.[2]

**Gambar 2.1** Sensor *Thermocouple*[2]

## 2.3 Flow Transmitter

Sensor flow water merupakan sensor yang digunakan untuk mengukur debit air yang mengalir pada pipa pelanggan. Sensor flow water terdiri dari bagian katup plastik (valve body), rotor air dan sebuah sensor half effect. Ketika air mengalir melalui rotor maka rotor akan berputar dan kecepatan dari rotor akan sesuai dengan aliran air yang masuk melewati rotor. Pulsa sinyal dari rotor akan diterima oleh sensor hall effect untuk selanjutnya diproses di mikrokontroller.

Sensor Hall effect merupakan salah satu tranduser yang sering digunakan untuk mendeteksi medan magnet. Hall Effect dapat digunakan untuk mendeteksi gerakan atau putaran apabila gerakan atau putaran tersebut dipengaruhi oleh medan magnet.[3]

**Gambar 2.3** Water Flow Sensor[3]

## 2.4 Variable Speed Driver

Variabel speed drive atau variabel frekuensi drive adalah suatu alat yang digunakan untuk mengendalikan kecepatan motor listrik

(AC) dengan mengontrol frekuensi daya listrik yang dipasok ke motor. Variabel frekuensi drive semakin popular karena kemampuannya untuk mengontrol kecepatan motor induksi. VSD mengontrol kecepatan motor induksid engan mengubah frekuensi dari grid untuk nilai disesuaikan pada sisi mesin sehingga memungkinkan motor listrik dengan cepat dan mudah menyesuaikan kecepatan dengan nilai yang diinginkan. Dua fungsi utama dari variabel frekuensi drive adalah untuk melakukan konversi listrik dari satu frekuensi ke yang lain, dan untuk mengontrol frekuensi keluaran. Aplikasi VSD digunakan dari mulai peralatan kecil sampai peralatan besar, yaitu pengaturan pabrik tambang, kompresor dan sistem ventilasi untuk bangunan besar. Selain itu VSD juga digunakan pada pompa, konveyor dan alat pengendali mesin. Penggunaan variabel frekuensi drive pada motor dapat menghemat energi sehingga mengurangi biaya listrik.[4]

**Gambar 2.4** Variable Speed Drive[4]

## 2.5 Spesifikasi Variable Speed Drive

Adapun Variable Speed Drive yang digunakan adalah LS is SV004iG5. Pada Speed Drive ini menggunakan masukan tegangan 1 fasa 200-230 V, dan keluaran 3 fasa dengan tegangan 200-230 V. Dengan frekuensi input 50-60 Hz dan frekuensi output 0.1-400 hz. Variable speed drive sendiri dapat dikontrol dengan berbagai mode, terdapat 3 mode yang dapat digunakan, yaitu mode kontrol PID, Up and Down.[5]

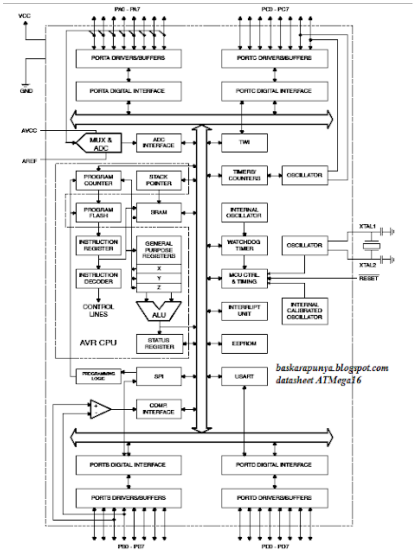
**230V Class (0.5~5.4HP)**

| Inverter Type (SVxxxiG5-x) | | 004-1 | 008-1 | 015-1 | 004-2 | 008-2 | 015-2 | 022-2 | 037-2 | 040-2 |
|---|---|---|---|---|---|---|---|---|---|---|
| Motor Rating[1] | HP | 0.5 | 1 | 2 | 0.5 | 1 | 2 | 3 | 5 | 5.4 |
| | kW | 0.37 | 0.75 | 1.5 | 0.37 | 0.75 | 1.5 | 2.2 | 3.7 | 4.0 |
| Output Ratings | Capacity[2] [kVA] | 1.1 | 1.9 | 3.0 | 1.1 | 1.9 | 3.0 | 4.5 | 6.1 | 6.5 |
| | FLA [A] | 3 | 5 | 8 | 3 | 5 | 8 | 12 | 16 | 17 |
| | Frequency | 0.1 ~ 400 Hz | | | | | | | | |
| | Voltage | 200 ~ 230 V [3] | | | | | | | | |
| Input Ratings | Voltage | 1 Phase 200 ~ 230 V (± 10 %) | | | 3 Phase 200 ~ 230 V (± 10 %) | | | | | |
| | Frequency | 50 ~ 60 Hz (±5 %) | | | | | | | | |
| Dynamic Braking | Braking Circuit | On Board | | | | | | | | |
| | Average Braking Torque | 20 % (Optional External DB Resistor: 100%, 150%) | | | | | | | | |
| | Max. Continuous Baking Time | 15 seconds | | | | | | | | |
| | Duty | 0 ~ 30 % ED | | | | | | | | |
| Weight [lbs] | | 2.65 | 3.97 | 4.63 | 2.65 | 2.65 | 3.97 | 4.63 | 4.85 | 4.85 |

**Gambar 2.5** Spesifikasi Variable Speed Drive[5]

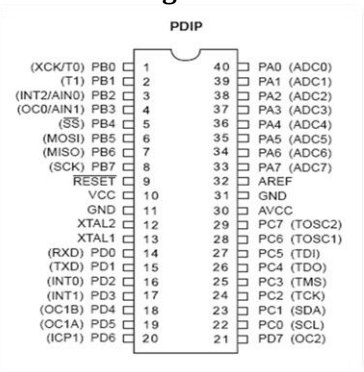
## 2.6 Mikrokontroler ATMega 16

AVR merupakan seri mikrokontroler CMOS 8-bit buatan Atmel,berbasis arsitektur RISC *(Reduced Instruction Set Computer).* Hampir semua instruksi dieksekusi dalam satu siklus clock. AVR mempunyai 32 register general-purpose, timer/counter fleksibel dengan mode compare, interrupt internal dan eksternal, serial UART, programmable Watchdog Timer, dan mode power saving, ADC dan PWM internal.AVR juga mempunyai *In-System Programmable Flash on-chip* yang mengijinkan memori program untuk diprogram ulang dalam sistem menggunakan hubungan serial SPI. ATMega16. ATMega16 mempunyai throughput mendekati 1 MIPS per MHz membuat disainer sistem untuk mengoptimasi konsumsi daya versus kecepatan proses.[6]




**Gambar 2.6** Blok Diagram ATMega 16[6]

## 2.7 Konfigurasi *Pin* ATMega16




**Gambar 2.7** Konfigurasi *Pin* ATMega16[6]

Gambar di atas merupakan susunan kaki standar 40 pin mikrokontroler AVR Atmega16. Berikut penjelasan umum susunan kaki Atmega16 tersebut:

a. VCC merupakan pin masukan positif catudaya. Setiap peralutan elektronika digital tentunya butuh sumber catu daya yang umumnya sebesar 5V.
b. GND sebagai ping ground.
c. Port A (PA0-PA7) merupakan pin I/O dua arah dan dapat diprogram sebagai masukan ADC.
d. Port B (PB0-PB7) merupakan pin I/O dua arah dan pin fungsi khusus, yaitu Timer/Counter, komparator analog, dan SPI
e. Port C (PC0-PC7) merupakan pin I/O dua arah dan pin fungsi khusus, yaitu TWI, komaparator analog, dan timer oscilator.
f. Port D (PD0-PD7) merupakan pin I/O dua arah dan pin fungsi khusus, yaitu komparator analog, Intrupsi Eksternal, dan komunikasi serial.
g. Reset merupakan pin yang digunakan untuk me-reset mikrokontroler ke kondisi semula.
h. XTAL1 dan XTAL2 sebagai pin masukan clock eksternal. Suatu mikrokontroler membutuhkan sumber detak (clock) agar daat mengeksekusi intruksi yang ada di memori. Semakin tinggi nilai kristalnya, maka semakin cepat pula mikrokontoler tersebut dalam mengeksekusi program.
i. AVCC sebagai pin masukan tegangan Auntuk ADC.
j. AREF sebagi pin masukan tegangan refrensi.[6]

## 2.8 Analog To Digital Converter (ADC)

AVR ATMega16 merupakan tipe AVR yang telah dilengkapi dengan 8 saluran ADC internal dengan resolusi 10 bit. Dalam mode operasinya, ADC dapat dikonfigurasi, baik single ended input maupun differential input. Selain itu, ADC ATMega16 memiliki konfigurasi pewaktuan, tegangan referensi, mode operasi, dan kemampuan filter derau (noise) yang amat fleksibel sehingga dapat dengan mudah disesuaikan dengan kebutuhan dari ADC itu sendiri. ADC pada ATMega16 memiliki fitur-fitur antara lain :

1. Resolusi mencapai 10-bit
2. Akurasi mencapai ± 2 LSB
3. Waktu konversi 13-260µs

4. 8 saluran ADC dapat digunakan secara bergantian
5. Jangkauan tegangan input ADC bernilai dari 0 hingga VCC
6. Disediakan 2,56V tegangan referensi internal ADC
7. Mode konversi kontinyu atau mode konversi tunggal
8. Interupsi ADC completeSleep Mode Noise canceler

## 2.9 *LCD (Liquid Crystal Display)*

Banyak sekali kegunaan LCD dalam perancangan suatu system yang menggunakan mikrokontroler. LCD berfungsi menampilkan suatu nilai hasil sensor,menampilkan teks,atau menampilkan menu pada aplikasi mikrokontroler. LCD yang digunakan adalah jenis LCD M1632. LCD M1632 merupakan modul LCD dengan tampilan 16 x 2 baris dengan konsumsi daya rendah. Modul tersebut dilengkapi dengan mikrokontroler yang didesain khusus untuk mengendalikan LCD.

Dengan alasan di atas maka dalam pembuatan alat tugas akhir ini penulis menggunakan LCD M1632 dengan tampilan 16 x 2 baris.

**Gambar 2.8** *Liquid Crystal Display*

## 2.10 Buck Converter

Buck-boost konverter adalah konverter DC (direct current) yang output tegangan dapat lebih besar atau lebih kecil dari tegangan input, dan juga tegangan outputnya selalu bernilai negatif. Sistem *buck-boost converter* merupakan salah satu regulator dc to dc tipe switching non –insolated yang dapat menjawab kebutuhan akan sebuah sumber tegangan searah dengan tegangan keluaran yang variabel. Dengan sistem buck-boost konverter nilai tegangan keluaran dapat diatur untuk lebih besar maupun lebih kecil dari nilai tegangan masukannya dengan

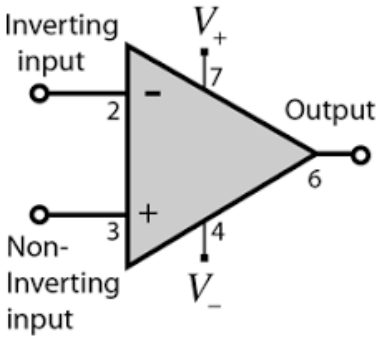
mengatur besar lebar pulsa *(duty cycle)* dari *PWM (Pulse Width Modulation*).karena itu dibandingkan dengan regulator tipe dc tipe pensakelaran yang lainnya, buck-boost konverter memiliki range tegangan keluaran yang lebih lebar

## 2.11 Op-Amp

Op-amp merupakan suatu komponen elektronika aktif yang dapat menguatkan sinyal dengan tingkat penguatan yang tinggi. Op-amp banyak digunakan untuk membangun blok rangkaian yang jauh lebih kompleks. Op-amp itu sendiri merupakan sebuah komponen yang terdiri dari banyak resistor, dioda, dan transistor yang dikemas dalam satu kemasan terintegrasi. Namun kita tidak perlu membahas atau mengetahui secara detail fungsi dan cara kerja dari komponen-komponen yang berada di dalam op-amp itu sendiri, yang terpenting adalah kita mengetahui bagaimana cara kerja mendasar dari op-amp itu sendiri.

Dalam skema rangkaian, op-amp biasa dilam- bangkan seperti tampak pada Gambar 1. Kaki- kaki utama dari op-amp terdiri dari kaki masukan inverting, kaki masukan non-inverting, dan kaki keluaran. Namun karena op-amp termasuk dalam kategori komponen aktif, op-amp membutuhkan koneksi catu daya yang terdiri dari koneksi dengan tegangan positif dan negatif.

Op-amp ideal memiliki karakteristik impedansi diantara kedua kaki masukannya adalah tak hingga sehingga tak membebani rangkaian yang memberi sinyal masukan, memiliki penguatan loop terbuka yang besarnya tak hingga, dan memiliki impedansi keluaran yang sama dengan nol. Namun pada prakteknya, tak ada op-amp yang ideal. Di dunia nyata, karakteristik dari op-amp ideal biasanya didekati oleh devais op-amp yang memiliki impedansi masukan yang tinggi, penguatan tegangan yang tinggi, pita respon frekuensi penggunaan yang lebar, serta impedansi keluaran yang rendah, sehingga bisa diasumsikan op-amp tersebut mendekati karakteristik dari op-amp ideal.




**Gambar 2.9** Simbol Rangkaian Op-Amp

# BAB III
# METODOLOGI

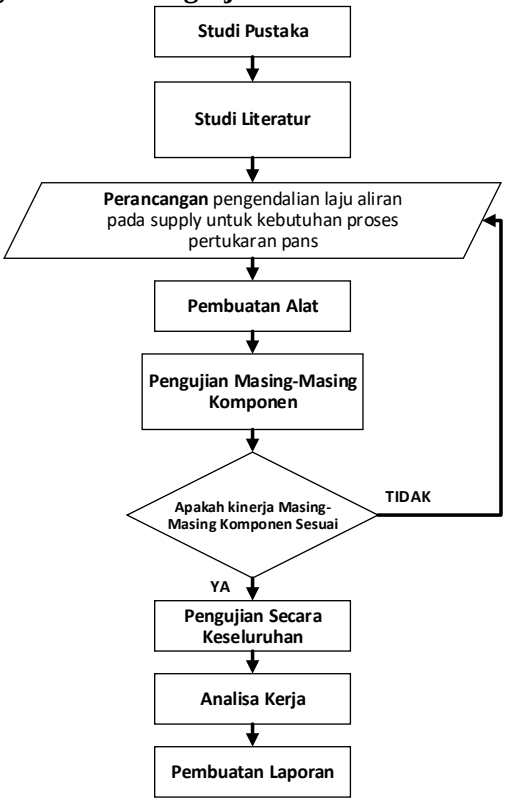
## 3.1 Alat dan Bahan

Adapun alat dan bahan yang digunakan adalah:

a. Pompa Panasonic
b. Water Flow Sensor (0.5-60 L/min)
c. Sensor Thermocouple
d. LS is SV004iC5
e. Mikronkontroler ATMega 16

## 3.2 Prosedur Penelitian

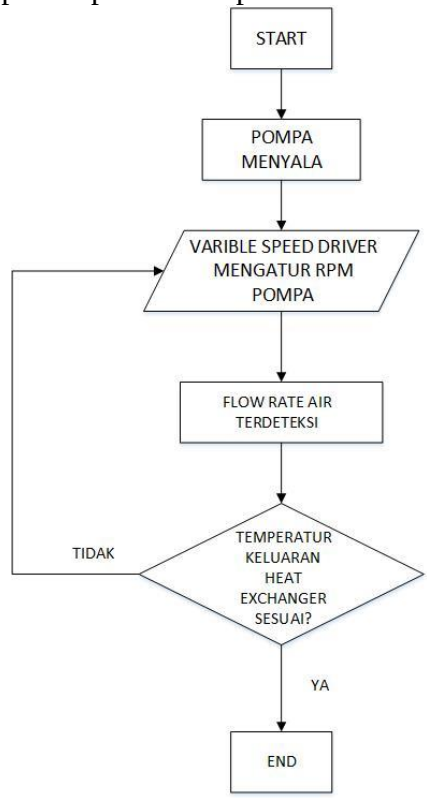
### 3.2.1 Diagaram Alir Pengerjaan



**Gambar 3.1** Diagram Alir Pengerjaan

Untuk mencapai tujuan penyelesaian tugas akhir yang direncanakan, maka perlu dilakukan suatu langkah-langkah dalam menyelesaikan tugas akhir ini. Adapun langkah- langkahnya adalah sebagai berikut :

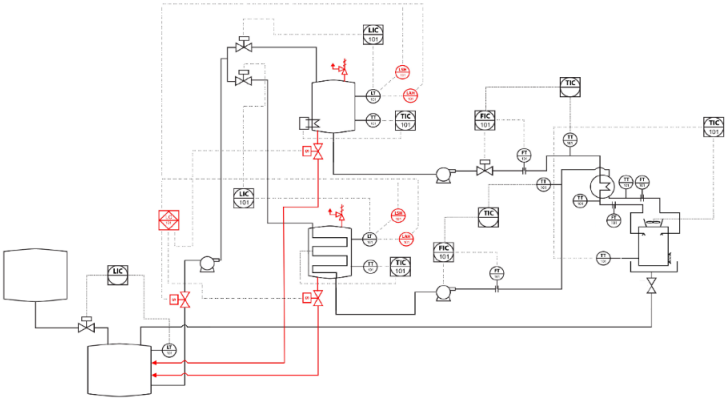
a. Studi Pustaka, mengumpulkan jurnal nasional, Internasioanl dan makalah yang berkaitan dengan pengendalian pompa dan variable speed drive yang sudah ada.
b. Studi Literatur,
c. Perancangan pengendalian laju aliran pada supply untuk kebutuhan proses pertukaran panas




**Gambar 3.2** Diagram Alir Pengendalian

d. Pembuatan alat, membuat rancang bangun variable speed driver pada pompa supply air dingin ke alat penukar panas
e. Pengujian Masing-masing komponen, dilakukan pengujian apakah kinerja tiap komponen sesuai
f. Pengujian secara keseluruhan
g. Analisa kerja
h. Pembuatan Laporan

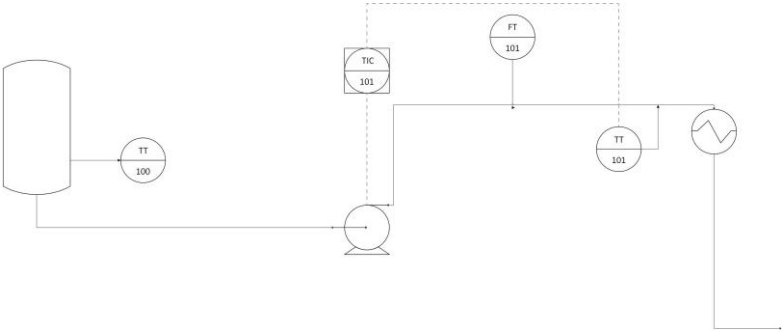
### 3.3 Filosofi Proses *Mini Plant Heat Exchanger*



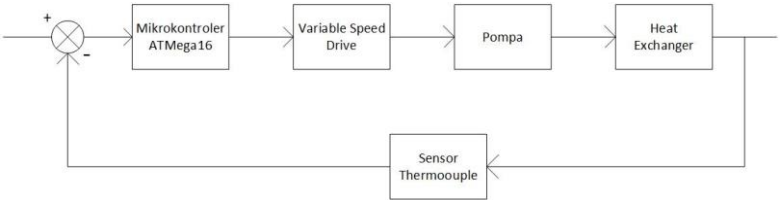
**Gambar 3.3** *Piping and Instrument Diagram Full Plant*

Pada *Mini Plant Heat Exchanger* terdapat dua tangki yaitu tangki pemanas dan juga tangki pendingin, kedua tangki tersebut mendapatkan *supply* air dari Tangki *Storage*. Kedua tangki tersebut akan dikontrol keluarannya oleh dua aktuator yang berbeda yaitu Motoriezed Operated Valve dan juga Pompa yang dikendalikan oleh Variable Speed Drive. Keluaran kedua tangki tersebut akan mengahsilkan air hangat dari ketika keluar dari *Heat Exchanger*.

## 3.3 Perancangan Sistem Pengendalian




**Gambar 3.4** *Piping and Instrument Diagram* Pengendalian Flowrate




**Gambar 3.5** Diagram Blok Pengendalian

Pada *Mini Plant Heat Exchanger* ini terdapat pengendalian pompa menggunakan *Variable Speed Drive,* pengendalian ini berguna agar masukan dari *Heat Exchanger* tidak lebih dari set point dari tangki pendingin.

## 3.3 Perancangan Sistem Elektrik

Setelah melakukan perancangan pembuatan alat dengan membuat P&ID plant keseluruhan dan juga plant pengendalian pompa. Maka, tahap selanjutnya adalah perancangang *software* atau perangkat lunak dan *hardware* atau perangkat keras. Untuk merancang simulasi pengendalian digunakan *software* Proteus 7

Professional dan CodeVisionAVR sebagai *software* pemrograman untuk ATMega16.

**Tabel 3.1** Penentuan Port Pada ATMega16

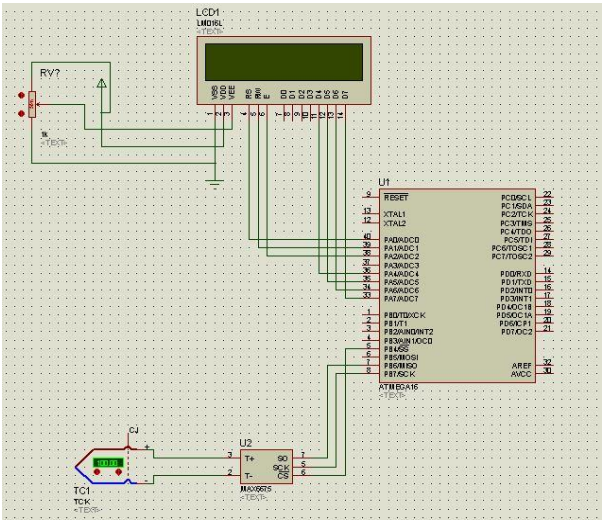
| Pin/Port | A | state | B | state | C | state | D | state |
|---|---|---|---|---|---|---|---|---|
| 0 | adc Temp set | T | Sdin (6675) /in | T | LCD 0 | 0 | rx4 | T |
| 1 | adc Flow set | T | cs (6675) /out | 0 | LCD 1 | 0 | tx4 | 0 |
| 2 | adc NTC | T | sclk (6675) /in | 0 | LCD 2 | 0 | Flow 3 /int0 | P |
| 3 | | T | sda | T | LCD 3 | 0 | run/stop/int1 | P |
| 4 | | T | scl | 0 | LCD 4 | 0 | inverter/PWM | 0 |
| 5 | | T | mosi 4 | T | LCD 5 | 0 | Led oveer | 0 |
| 6 | | T | miso 4 | T | LCD 6 | 0 | ledrun | 0 |
| 7 | | T | sck 4 | T | LCD 7 | 0 | p1 | 0 |

Fungsi dari penentuan port ini agar memudahkan konfigurasi untuk pemrograman yang akan dilakukan pada *software* CodeVisionAVR. Adapun penjelasan dari tabel diatas adalah:

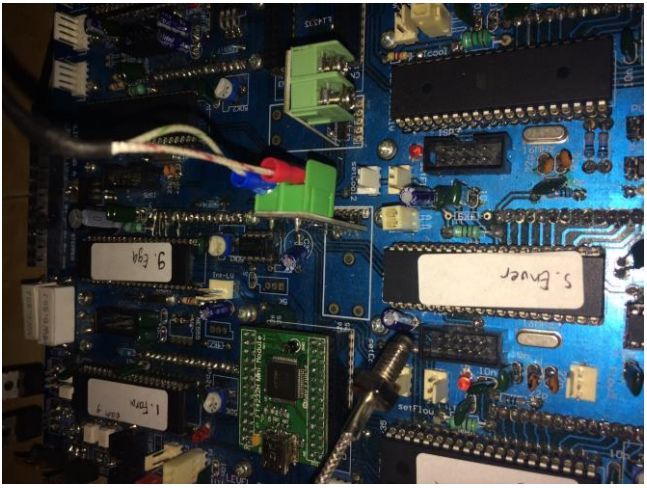
a. Port A, digunakan sebagai *analog digital converter* untuk *Thermocouple*.
b. Port B, digunakan sebagai port input dari sensor yang akan digunakan yaitu sensor Flow dan Thermocouple.
c. Port C, digunakan untuk *Liquid Crystal Display* (LCD)
d. Port D, digunakan sebagai port untuk aktuator atau Variable Speed Drive.

### 3.2.1 Sensor Thermocouple

Sensor Thermocople membutuhkan modul MAX6675 sebagai pengondisian sinyal yang akan dikirimkan mikrokontroler ATMega16 yang kemudian ditampilkan ke LCD dan juga dikirimkan HMI. Pada plant ini fungsi thermocouple adalah untuk mendeteksi temperatur dan akan mengirimkan sinyal kepada ATMega16.




**Gambar 3.6** Simulasi Proteus Thermocouple tipe K dan MAX6675 ke ATMega

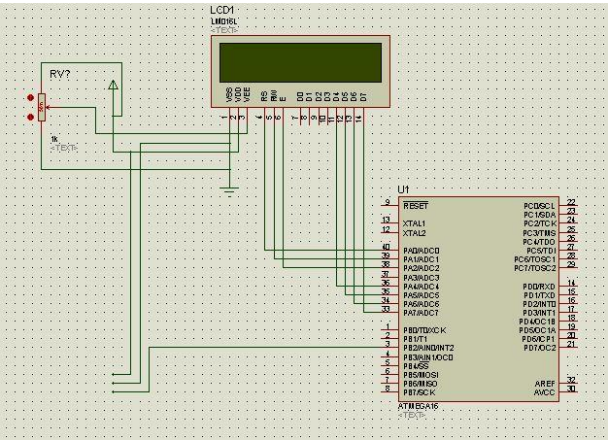


**Gambar 3.7** Wiring Thermocouple tipe K dan MAX6675 ke ATMega16

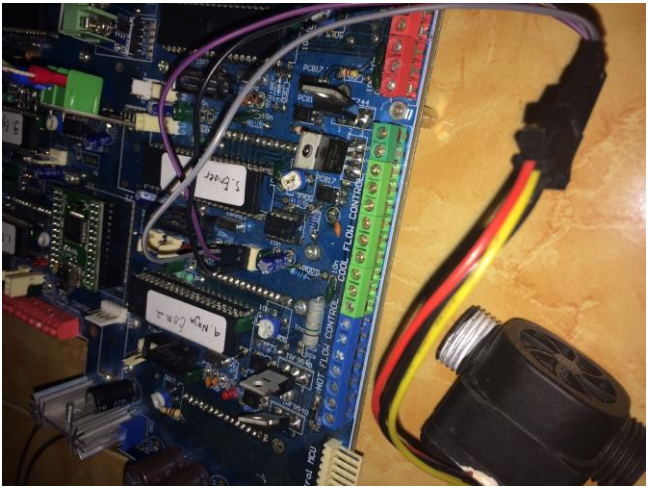
### 3.2.2 Water Flow Sensor

Kegunaan dari sensor Flow ini adalah untuk mengetahui kecepatan aliran fluida dari keluaran pompa. Cara kerja dari sensor ini adalah

menggunakan medan magnet yang terdapat pada rotor akan memberikan efek pada sensor hall effect hal tersebut akan menghasilkan sebuah sinyal pulsa yang berupa tenggangan atau *Pulse Width Modulation.* Lalu, pulsa tersbut akan diproses di mikrokontoroler ATMega16 dan akan menampilkan nya pada LCD.



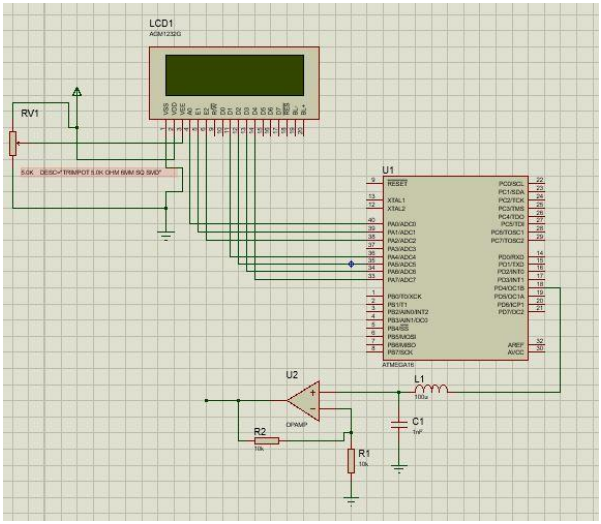

**Gambar 3.8** Simulasi Proteus Water Flow Sensor ke ATMega16



**Gambar 3.9** Wiring Water Flow Sensor ke ATMega16

### 3.2.3 *Buck Converter* dan Op-Amp

*Buck Converter* digunakan untuk mengubah *Pulse Width Modulation* keluaran dari Mikrokontroler menjadi *Duty Cycle*, tetapi tegangan yang keluar dari *Buck* tidak mencukupi masukan dari *Variable Speed Drive* jadi dinaikan dengan OPAMP untuk memenuhi tegangan VSD.



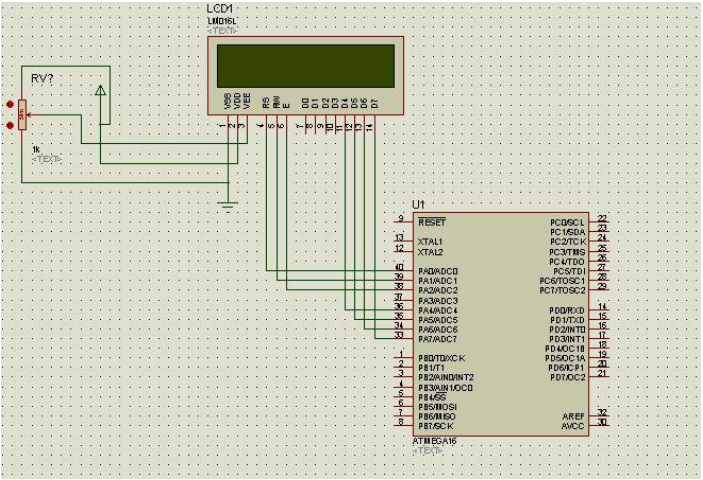

**Gambar 3.10** Rangkaian *Buck Converter* dan OPAMP ke ATMega16

### 3.2.4 *Liquid Crystal Display*

LCD yang digunakan adalah 2 baris x 16 kolom. LCD memiliki memori internal yang berisi definisi karakter sesuai dengan standart ASCII ( CGROM- Character Generator ROM ) dan memori sementara (RAM) yang biasa digunakan bila memerlukan karakter khusus (kapasitas 8 karakter). RAM ini juga berfungsi untuk menyimapan karakter yang ingin ditampilkan di LCD.

Pin yang digunakan di LCD menggunakan semua di port C. untuk data karakter yang dikirm dari ATmega16 ke LCD semua menggunakan port di LCD D0-D7 serta juga port RS, RW dan E. bagian VSS disambungkan ke ground dan bagian VDD disambungkan ke sumber 0-5 Volt, bagian VEE ke potensio untuk

mengatur kontras dari karakter LCD dan port A dan K untuk mngaktifkan lampu LCD. Berikut adalah gambar simulasi proteus LCD dan gambar aslinya di plant sesuai dengan port yang sudah ditentukan pada tabel di atas.




**Gambar 3.11** Simulasi Proteus Liquid Crystal Display ke ATMega16

# BAB IV
# ANALISA DATA

## 4.1. Validasi Sensor

Validasi sensor digunakan untuk mengetahui apakah data *real* sensor sesuai dengan data uji.

### 4.1.1. Validasi *Water Flow Sensor*

Untuk mendapatkan perhitungan debit air dengan mengetahui spesifikasi tangki yang memilik tinggi tangki 49 cm, lebar alas 20x20 cm, dengan volume air yang terisi penuh. Adapun perhitungan laju debit air adalah sebagai berikut:

$$Volume\ tanki = p\ x\ l\ x\ t$$

$$Volume\ tanki = 20\ x\ 20\ x\ 49$$

$$Volume\ tanki = 19600\ cm^3$$

$$Debit = \frac{Volume}{Waktu}$$

$$Debit = \frac{19{,}6\ L}{2\ m}$$

$$Debit = 7{,}8\ LPM$$

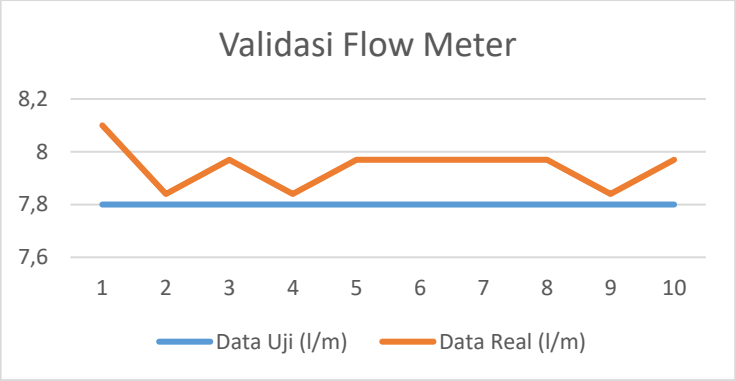
5.

Untuk mendapatkan data diatas dilakukan validasi dengan cara menggunakan tangki berukuran 20x20x49 yang diisi penuh oleh air. Setelah itu, flow meter dipasang pada keluaran tangki untuk mengukur berapa flowrate yang keluar pada tangki tersebut.

Adapun cara untuk melakukan validasi untuk mendapatkan data sesuai tabel diatas adalah dengan cara mengisi penuh air lalu mencatat laju aliran yang terukur oleh sensor, metode tersebut dilakukan secara terus menerus selama 2 menit dan mencatat hasil dari sensor setiap 12 detik sekali.

**Tabel 4.1** Data Validasi Flow

| No | Data Uji (l/m) | Data Real (l/m) |
|---|---|---|
| 1 | 7,8 | 8,10 |
| 2 | 7,8 | 7,84 |
| 3 | 7,8 | 7,97 |
| 4 | 7,8 | 7,84 |
| 5 | 7,8 | 7,97 |
| 6 | 7,8 | 7,97 |
| 7 | 7,8 | 7,97 |
| 8 | 7,8 | 7,97 |
| 9 | 7,8 | 7,84 |
| 10 | 7,8 | 7,97 |



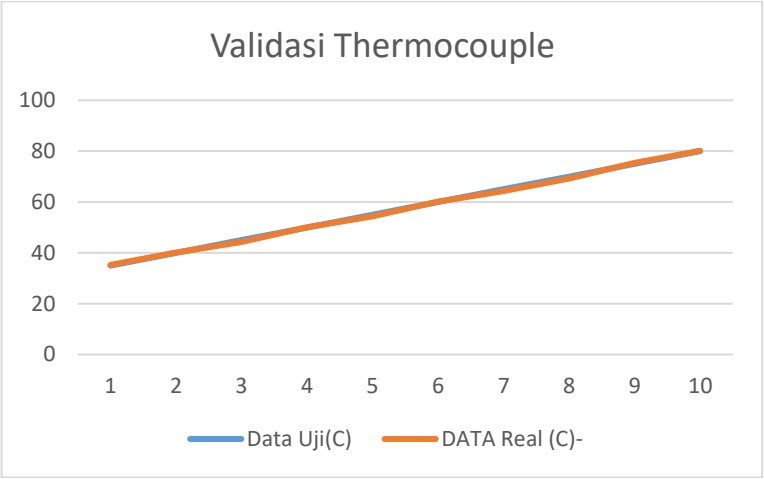

**Gambar 4.1** Grafik Validasi Flow Meter

### 4.1.2 Validasi *Thermocouple type K*

**Tabel 4.2** Data Validasi Thermocouple

| Data Uji(C) | DATA Real (C)- |
|---|---|
| 35 | 35,25 |
| 40 | 40,1 |
| 45 | 44,27 |
| 50 | 50,1 |

| | |
|---|---|
| 55 | 54,37 |
| 60 | 60,12 |
| 65 | 64,32 |
| 70 | 69,23 |
| 75 | 75,37 |
| 80 | 80,12 |




**Gambar 4.2** Grafik Validasi Thermocouple

Data diatas didapatkan dengan cara membandingkan antara sensor Thermocouple dengan sensor tempeartur yang dimiliki oleh Departemen Teknik Instrumentasi. Perbandingan dilakukan dengan cara memanaskan air lalu dilakukan sensing oleh kedua sensor, pemanasan dilakukan mulai dari suhu 35 derajat celcius heingga 80 derajat celcius.

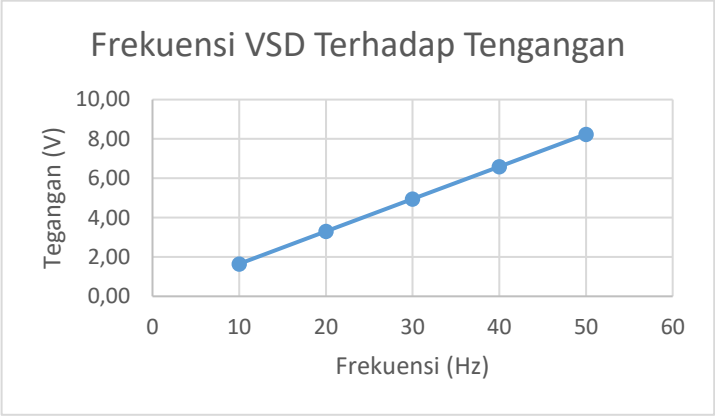
## 4.2 Pengambilan Data

Pengambilan data dilakukan agar dapat mengetahui perbedaan keluaran antara objek yang diukur, hal tersebut untuk mengetahui perbandingan nilai antara dua variable yang berbeda.

### 4.2.1 Data Frekuensi Terhadap Tegangan

**Tabel 4.3** Data Frekuensi *Variable Speed Drive* Terhadap Tegangan

| No | Frequency | Voltage |
|---|---|---|
| 1 | 10 | 1,65 |
| 2 | 20 | 3,30 |
| 3 | 30 | 4,95 |
| 4 | 40 | 6,59 |
| 5 | 50 | 8,24 |




**Gambar 4.3** Grafik Frekuensi *Variable Speed Drive* Terhadap Tegangan

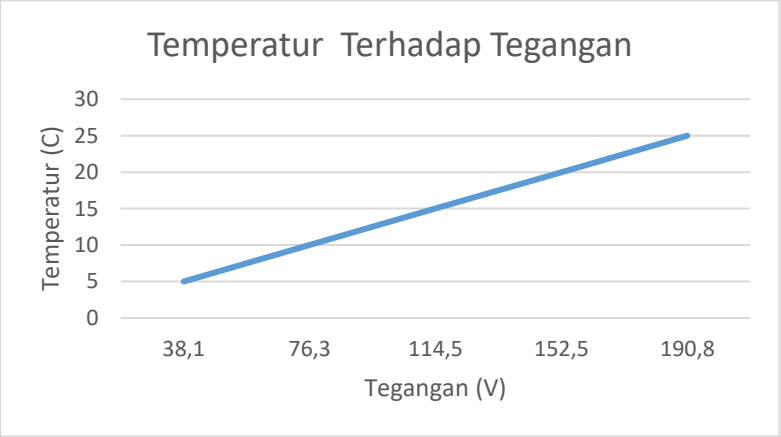
Pengambilan data diatas dilakukan untuk mendapatkan perbandingan antara frekuensi yang dikeluarkan oleh *Variable Speed Drive* dan tegangan yang dikeluarkan *Variable Speed Drive* untuk mengahsilkan frekuensi tersebut.

Untuk mendapatkan data tersebut *Variable Speed Drive* dihubungkan dengan potensio untuk merubah-rubah frekuensi dari VSD. Bukaan dari potensio lalu diukur oleh *Volt Meter* untuk mengetahui tegangan yang keluar untuk mengubah

### 4.2.2 Data Temperatur Terhadap Tegangan

**Tabel 4.4** Data Temperatur Terhadap Tegangan

| No | Tempaeratur (c) | Tegangan (miliVolt) |
|---|---|---|
| 1 | 25 | 1,2 |
| 2 | 30 | 1,4 |
| 3 | 35 | 1,7 |
| 4 | 40 | 1,9 |
| 5 | 45 | 2,1 |




**Gambar 4.4** Grafik Temperatur Terhadap Tegangan

Data diatas diambil untuk dapat mengetahui masukan tegangan yang akan diproses oleh modul Thermocouple atau max6678 yang akan masuk ke mikrokontroller ATMega16.

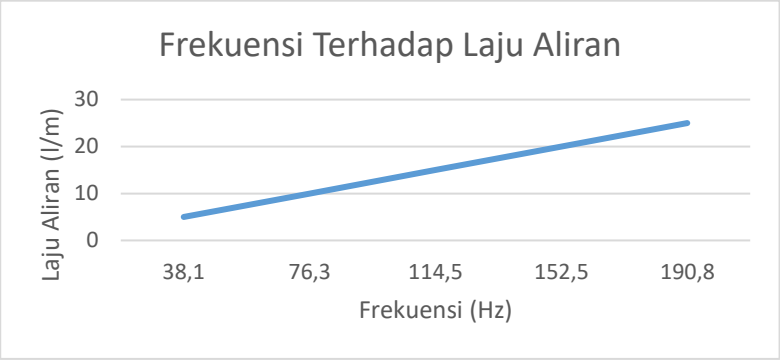
Dapat

### 4.2.3 Data Laju Aliran Terhadap Frekuensi

**Tabel 4.5** Data Frekuensi Putaran Flow Meter Terhadap Laju Aliran

| No | Flow (l/m) | Frekuensi (Hz) |
|---|---|---|

| | | |
|---|---|---|
| 1 | 5 | 38,1 |
| 2 | 10 | 76,3 |
| 3 | 15 | 114,5 |
| 4 | 20 | 152,5 |
| 5 | 25 | 190,8 |



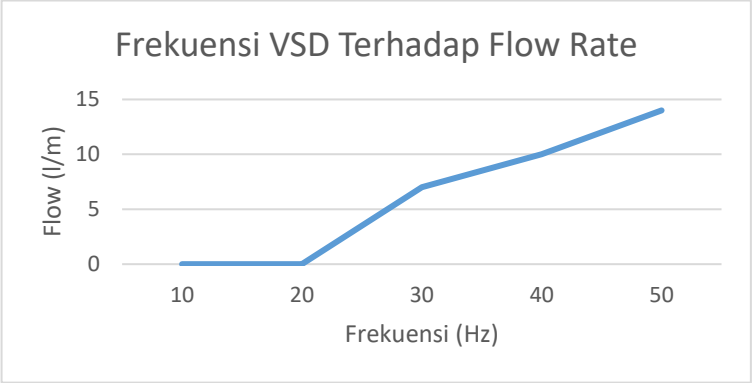

**Gambar 4.5** Grafik Frekuensi Putaran Flow Meter Terhadap Laju Aliran

Data diatas diambil untuk mengetahui berapa frekuensi keluaran Flow Meter yang digunakan saat masuk keinput mikrokontroler ATMega16 dan diproses untuk mendapatkan hasil laju aliran di *Liquid Crystal Display* atau LCD.

### 4.2.4 Frekuensi Terhadap Laju Aliran

**Tabel 4.6** Data Frekuensi Terhadap Laju Aliran

| No | Frequency | Flow |
|---|---|---|
| 1 | 10 | 0 |
| 2 | 20 | 0 |
| 3 | 30 | 7 |
| 4 | 40 | 10 |
| 5 | 50 | 14 |




**Gambar 4.6** Grafik Frekuensi Impeler Pompa Terhadap Flow

Data diatas diambil untuk dapat mengetahui berapa laju aliran yang dihasilkan pompa pada frekuensi Impeler tertentu, pada saat frekuensi 10 dan 20 didapatkan laju aliran 0 karena ketika *Variable Speed Driver* pada menunjukkan frekuensi tersebut tidak dapat menyalakan pompa karena pompa sendiri baru menyala pada frekuensi 29.

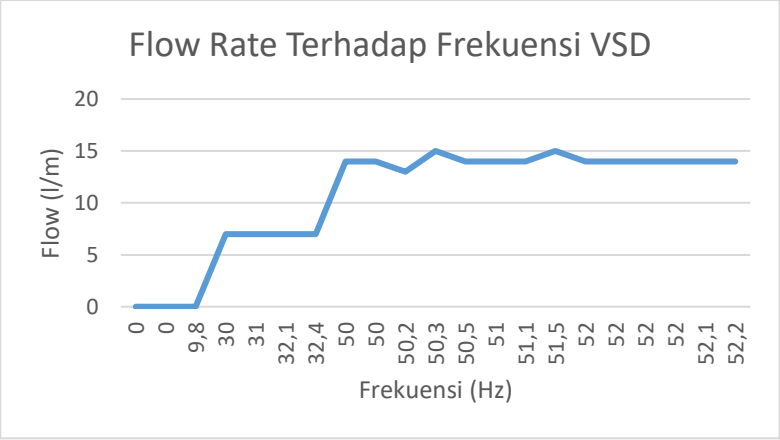
Untuk mendapatkan data tersebut dilakukan perubahan frekuensi dari 10 sampai dengan 50 dengan melihat Water Flow Sensor yang ada untuk mengetahui berapa laju aliran yang terdeteksi.

### 4.2.5 **Pengambilan Data Pengendalian**

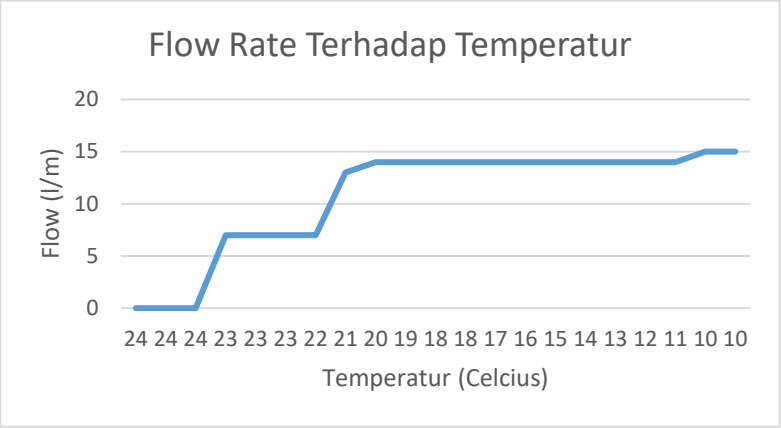
**Tabel 4.7** Pengambilan Data Pengendalian

| No | Flow Rate (l/m) | Frekuensi VSD (Hz) | Temperatur (C) |
|---|---|---|---|
| 1 | 0 | 9,8 | 24 |
| 2 | 7 | 32,1 | 24 |
| 3 | 7 | 32,4 | 24 |
| 4 | 7 | 31 | 23 |
| 5 | 7 | 30 | 23 |
| 6 | 0 | 0 | 23 |
| 7 | 14 | 50 | 22 |
| 8 | 14 | 50 | 21 |

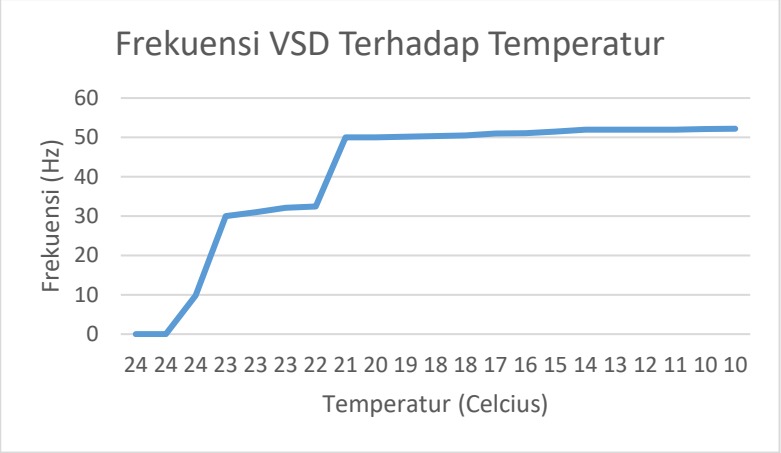
| | | | |
|---|---|---|---|
| 9 | 14 | 50,5 | 20 |
| 10 | 14 | 52,2 | 19 |
| 11 | 14 | 52 | 18 |
| 12 | 14 | 52 | 18 |
| 13 | 13 | 50,2 | 17 |
| 14 | 14 | 52 | 16 |
| 15 | 14 | 52,1 | 15 |
| 16 | 15 | 51,5 | 14 |
| 17 | 14 | 52 | 13 |
| 18 | 15 | 50,3 | 12 |
| 19 | 14 | 51,1 | 11 |
| 20 | 14 | 51 | 10 |
| 21 | 0 | 0 | 10 |




**Gambar 4.7** Grafik Frekuensi Putaran Variable Speed Drive Terhadap Flow




**Gambar 4.8** Grafik Perubahan Temperatur Terhadap Flow




**Gambar 4.9** Grafik Perubahan Frekuensi Variable Speed Drive Terhadap Temperatur

Data diatas menunjukkan performa impeler pompa dan juga perubahan antar laju aliran dan juga temeperatur. Pada data diatas set point temperatur diset 10 derajat celcius, sedangkan pada pengukuran pertama didapatkan hasil pengukuran 25.

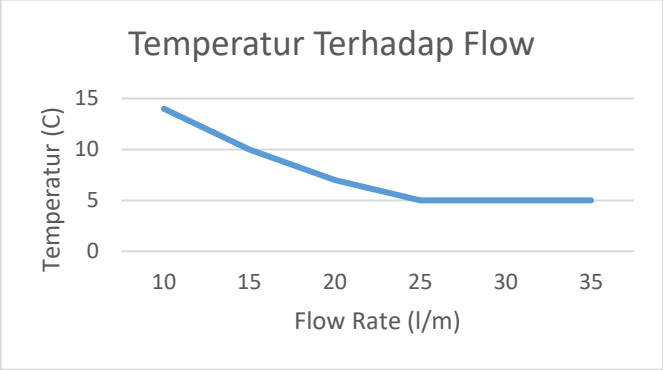
Dapat dilihat pada diatas apa bila temperatur yang terukur tidak sesuai dengan set point Mikrokontroler akan mendrive VSD untuk

mempercepat putaran impeler pompa untuk mendapatkan suhu yang sesuai.

### 4.2.6 Pengambilan Data

**Tabel 4.9** Data Pengaruh Flow Terhadap Temeperatur

| Temperatur | Flow |
|---|---|
| 10 | 14 |
| 15 | 10 |
| 20 | 7 |
| 25 | 5 |




**Gambar 4.9** Grafik Pengaruh Flow Terhadap Temperatur

Data diatas diambil untuk mengetahui flow yang dihasilkan apabila dirubah pada set point tertentu. Data diatas set point yang digunakan adalah antara 10 sampai 35 derajat celcius.

Dapat dilihat pada flow maksimal yang dapat di drive oleh aktuator adalah 14 liter per menit dan menunjukkan temperatur 10 derajat, sedangkan pada flow paling rendah 5-7 liter per menit temperatur terus drop hingga 25 derajat atau mendekati suhu normal air.

# BAB V
# PENUTUP

## 5.1. Kesimpulan

Adapun kesimpulan yang dicapai dalam tugas akhir ini adalah :

a. Perancangan pengendalian *flow rate* dilakukan dengan cara memanipulasi putaran pompa dan didapatkan ketika flow maksimal 15 liter/menit didapatkan temperatur terukur 10 derajat celcius.
b. Pompa sentrifugal dapat dikendalikan dengan Variable Speed Drive menggunakan mode kontrol *Up and Down.* Mode kontrol tersebut dipilih karena dapat mencari flow yang sesuai untuk mendapatkan temperatur tertentu.

## 5.2. Saran

Selama proses pengerjaan Tugas Akhir ini dari awal sampai akhir, adapun saran yang diperlukan untuk keberlangsungan Tugas Akhir ini apabila ada orang lain yang ingin mengembangkan Tugas Akhir ini di kemudian hari:

1. Perlu ditetapkannya *timeline* dari awal pengerjaan sampai akhir dan kesepakatan dari masing-masing anggota kelompok mengenai *timeline* tersebut sehingga pengerjaan simulasi alat penukar panas bisa berjalan sesuai dengan *timeline*.
2. Pengujian sensor bisa dilakukan sebelum menyatukan plant agar dapat menghemat waktu untuk mengerjakan uji data lainnya ketika plant telah disatukan

## DAFTAR PUSTAKA


1. Ahmad Wafi B, “Rancang Bangun Heat Exchanger Shell And Tube Single Phase”, Tugas Akhir D3 Teknik Mesin Universitas Diponegoro, 2011
2. Mismail, Budiono “Rangkaian Listrik”, Universitas Brawijaya, Malang, 1981.
3. “Water Flow Sensor Datasheet”
4. “Variable Speed Drive dan Inverter”, Unversitas Negeri Sriwijaya, Palembang, 2016.
5. “Manual Book LS is SV004IG5”
6. Anggraini, Dian “Aplikasi Mikrokontroller ATMega 16 Sebagai Sistem Pengontrol Sistem Emergency”
7. “Oprational Amplifier (Op-Amp)” Electrical Engineering, Telkom University.

**LAMPIRAN**

```
/*****************************************************
*
This program was produced by the
CodeWizardAVR V2.05.3 Standard
Automatic Program Generator
© Copyright 1998-2011 Pavel Haiduc, HP InfoTech s.r.l.
http://www.hpinfotech.com

Project :
Version :
Date    : 6/1/2018
Author  : Enver Rafid Sadida
Company : ITS
Comments:


Chip type               : ATmega16A
Program type            : Application
AVR Core Clock frequency: 11.059200 MHz
Memory model            : Small
External RAM size       : 0
Data Stack size         : 256
*****************************************************
/

#include <mega16a.h>

// Alphanumeric LCD functions
#include <alcd.h>
// I2C Bus functions
#include <i2c.h>

#define ledrun PORTD.6
#define ledover PORTD.5
#define p1 PORTD.7
#define CS PORTB.1
```

```
#define sclk PORTB.2
#define sdin PINB.0
#define maxVSD 255
//#define EEPROM_BUS_ADDRESS 0xa0
#define backlight PORTC.7


bit run,second;

int periode,minperiode,frek,remotevsd;
int count, gain,  LPM, tduty,override;
unsigned pulse;
// External Interrupt 0 service routine
interrupt [EXT_INT0] void ext_int0_isr(void)
{
// Place your code here
 periode=pulse;
 minperiode=TCNT2;
 TCNT2=0;
 pulse=0;
}

// External Interrupt 1 service routine
interrupt [EXT_INT1] void ext_int1_isr(void)
{
// Place your code here
run=~run;
}

#ifndef RXB8
#define RXB8 1
#endif

#ifndef TXB8
#define TXB8 0
#endif
```

```
#ifndef UPE
#define UPE 2
#endif

#ifndef DOR
#define DOR 3
#endif

#ifndef FE
#define FE 4
#endif

#ifndef UDRE
#define UDRE 5
#endif

#ifndef RXC
#define RXC 7
#endif

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

// USART Receiver buffer
#define RX_BUFFER_SIZE 7
char rx_buffer[RX_BUFFER_SIZE];

#if RX_BUFFER_SIZE <= 256
unsigned char rx_wr_index,rx_rd_index,rx_counter;
#else
unsigned int rx_wr_index,rx_rd_index,rx_counter;
#endif

// This flag is set on USART Receiver buffer overflow
```

```
bit rx_buffer_overflow;

// USART Receiver interrupt service routine
interrupt [USART_RXC] void usart_rx_isr(void)
{
char status,data;
status=UCSRA;
data=UDR;
if (data==0x23)
{
   rx_counter=0;
   rx_wr_index=0;
}
if ((status & (FRAMING_ERROR | PARITY_ERROR | DATA_OVERRUN))==0)
  {
  rx_buffer[rx_wr_index++]=data;
#if RX_BUFFER_SIZE == 256
  // special case for receiver buffer size=256
  if (++rx_counter == 0) rx_buffer_overflow=1;
#else
  if (rx_wr_index == RX_BUFFER_SIZE) rx_wr_index=0;
  if (++rx_counter == RX_BUFFER_SIZE)
    {
    rx_counter=0;
    rx_buffer_overflow=1;
    }
#endif
  }
}

#ifndef _DEBUG_TERMINAL_IO_
// Get a character from the USART Receiver buffer
#define _ALTERNATE_GETCHAR_
#pragma used+
char getchar(void)
{
```

```
char data;
while (rx_counter==0);
data=rx_buffer[rx_rd_index++];
#if RX_BUFFER_SIZE != 256
if (rx_rd_index == RX_BUFFER_SIZE) rx_rd_index=0;
#endif
#asm("cli")
--rx_counter;
#asm("sei")
return data;
}
#pragma used-
#endif

// Standard Input/Output functions
#include <stdio.h>

// Timer 0 overflow interrupt service routine
// For Display
interrupt [TIM0_OVF] void timer0_ovf_isr(void)
{
// Place your code here
count++;
   if (count>=2700)
   {
   second = 1;
   count = 0;
   }
}

// Timer1 overflow interrupt service routine
interrupt [TIM1_OVF] void timer1_ovf_isr(void)
{
// Place your code here

}
```

```
// Timer2 overflow interrupt service routine
interrupt [TIM2_OVF] void timer2_ovf_isr(void)
{
// Place your code here
pulse++;
}

int read_Temp(int Temp)
{int i;

 CS=1;
 CS=1;
 sclk=0;
 sclk=0;
 CS=0;
 CS=0;
 sclk=1;
 sclk=1;
 //if(sdin==0)
 //  {
    Temp=0;
      for (i=1;i<=12;i++)
      {
      sclk=0;
      sclk=0;
      sclk=1;
      sclk=1;
      switch(i)
      {case 1 :{Temp = Temp + sdin * 2048; break;}
       case 2 :{Temp = Temp + sdin * 1024; break;}
       case 3 :{Temp = Temp + sdin * 512; break;}
       case 4 :{Temp = Temp + sdin * 256; break;}
       case 5 :{Temp = Temp + sdin * 128; break;}
       case 6 :{Temp = Temp + sdin * 64; break;}
       case 7 :{Temp = Temp + sdin * 32; break;}
       case 8 :{Temp = Temp + sdin *  16; break;}
       case 9 :{Temp = Temp + sdin * 8; break;}
```

```
        case 10 :{Temp = Temp + sdin * 4; break;}
        case 11 :{Temp = Temp + sdin * 2; break;}
        case 12 :{Temp = Temp + sdin * 1; break;}
       }
       }
      Temp = Temp/4;
//  }
   sclk=0;
   sclk=0;
   sclk=1;
   sclk=1;
   sclk=0;
   sclk=0;

   CS=1;
   CS=1;

  return Temp;

}

#define ADC_VREF_TYPE 0x40

// Read the AD conversion result
unsigned int read_adc(unsigned char adc_input)
{
ADMUX=adc_input | (ADC_VREF_TYPE & 0xff);
// Delay needed for the stabilization of the ADC input voltage
delay_us(10);
// Start the AD conversion
ADCSRA|=0x40;
// Wait for the AD conversion to complete
while ((ADCSRA & 0x10)==0);
ADCSRA|=0x10;
return ADCW;
}
```

```
// Declare your global variables here

void main(void)
{
// Declare your local variables here
   int Target_Temp, error, Tout, maxFlow, duty,suhuawal,optimal,MsbLPM,refreshcount,setawal,flowawal;
   unsigned char buff[16];
   unsigned char skip[3];
   /* read a byte from the EEPROM */

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

// Port B initialization
// Func7=In Func6=In Func5=In Func4=Out Func3=Out Func2=In Func1=In Func0=Out
// State7=T State6=T State5=T State4=0 State3=0 State2=P State1=T State0=0
PORTB=0x00;
DDRB=0x06;

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

// Port D initialization
```

```
// Func7=Out Func6=Out Func5=Out Func4=Out Func3=Out Func2=In Func1=Out Func0=In
// State7=0 State6=0 State5=0 State4=0 State3=0 State2=P State1=0 State0=T
PORTD=0x04;
DDRD=0xFA;

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: 1382.400 kHz
// Mode: Normal top=0xFF
// OC0 output: Disconnected
TCCR0=0x02;
TCNT0=0x00;
OCR0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: 11059,200 kHz
// Mode: Fast PWM top=0x00FF
// OC1A output: Discon.
// OC1B output: Non-Inv.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=0x21;
TCCR1B=0x09;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
```

```
OCR1BL=0x00;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: 172.800 kHz
// Mode: Normal top=0xFF
// OC2 output: Disconnected
ASSR=0x00;
TCCR2=0x02;
TCNT2=0x00;
OCR2=0x00;

// External Interrupt(s) initialization
// INT0: On
// INT0 Mode: Falling Edge
// INT1: On
// INT1 Mode: Falling Edge
// INT2: Off
GICR|=0xC0;
MCUCR=0x0A;
MCUCSR=0x00;
GIFR=0xC0;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x45;

// USART initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART Receiver: On
// USART Transmitter: On
// USART Mode: Asynchronous
// USART Baud Rate: 9600
UCSRA=0x00;
UCSRB=0x98;
UCSRC=0x86;
UBRRH=0x00;
UBRRL=0x47;
```

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

// ADC initialization
// ADC Clock frequency: 86.400 kHz
// ADC Voltage Reference: AVCC pin
// ADC Auto Trigger Source: ADC Stopped
ADMUX=ADC_VREF_TYPE & 0xff;
ADCSRA=0x87;

// SPI initialization
// SPI disabled
SPCR=0x00;

// TWI initialization
// TWI disabled
TWCR=0x00;

// I2C Bus initialization
// I2C Port: PORTD
// I2C SDA bit: 6
// I2C SCL bit: 7
// Bit Rate: 100 kHz
// Note: I2C settings are specified in the
// Project|Configure|C Compiler|Libraries|I2C menu.
i2c_init();

// Alphanumeric LCD initialization
// Connections are specified in the
// Project|Configure|C Compiler|Libraries|Alphanumeric LCD menu:
// RS - PORTC Bit 0
// RD - PORTC Bit 1
```

```
// EN - PORTC Bit 2
// D4 - PORTC Bit 3
// D5 - PORTC Bit 4
// D6 - PORTC Bit 5
// D7 - PORTC Bit 6
// Characters/line: 8
lcd_init(16);

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

lcd_clear();
p1 = 0;
backlight=1;
lcd_gotoxy(0,0);
lcd_putsf("10511500000019");
lcd_gotoxy(6,1);
lcd_putsf("---");
delay_ms(1000);
lcd_clear();
p1 = 0;
ledrun=1;
ledover=0;
run=1;
override=0;
duty=0;
refreshcount=700;
while (1)
     {
/*-------------------------------------------------------------Komunikasi
Data (Exsa)--------------*/
     if (rx_buffer_overflow==1)
     {
     if ((rx_buffer[0]==0x23) && (rx_buffer[6]==0x25))
     {
       if (rx_buffer[1]==0x35)
       {
```

```
    switch (rx_buffer[2])
    {
    //read
    case 97:{    printf ("05;");
              break;}
    case 98:{    printf ("%d;", LPM);
              break;}
    case 99:{    printf ("%d;", Tout);
              break;}
    case 100:{  tduty=duty/2.55;
              printf ("%d;", tduty);
              break;}
    case 101:{   printf ("%d;", override);
              break;}
    case 102:{   printf ("%d;", run);
              break;}
    case 103:{   printf ("%d;\r\n", backlight);
              break;}

    //write
    case 109:{   skip[0]=rx_buffer[3];
              skip[1]=rx_buffer[4];
              skip[2]=rx_buffer[5];
              remotevsd=atoi(skip);}
    case 110:{   if (rx_buffer[5]=='1')override=1;
              else if (rx_buffer[5]=='0')override=0;
              break;}
    case 111:{   if (rx_buffer[5]=='1')run=1;
              else if (rx_buffer[5]=='0')run=0;
              break;}
    case 112:{   if (rx_buffer[5]=='1')backlight=1;
              else if (rx_buffer[5]=='0')backlight=0;
              break;}
       }
     }
   }
  else
```

```
    {
    printf ("error");
    }
    rx_buffer_overflow=0;
    }
/*-------------------------------------------------------------------------
-------------------*/


     if (second == 1)
       {
        maxFlow = read_adc(1)/10;
        Target_Temp= 10+(read_adc(0)/50);
        Tout=read_Temp(Tout);

        if((periode==0)&&(minperiode==0))frek=0;
        else
frek=1/((periode*0.000185)+(minperiode*0.000000723));
        LPM=frek*0.130985915;


        if(run==1)
        {
         ledrun=1;

        if (run==1 )
        {
         ledrun=1;
         if ((Tout>=Target_Temp+1)&&(optimal==0))
             {
              if (LPM<maxFlow)
              {
             if(duty<=255)duty++;
              if  (suhuawal>Tout)optimal=2;  //start  find  maksimum
flow when T down
              else if (suhuawal<Tout)optimal=3; // start find maksimu
flow T rise
```

```
            }
            else if (LPM>1)
                {
                if (duty>=1)duty--;
                }

           }
        else if ((Tout<=Target_Temp-1)&&(optimal==0))
           {
            if(LPM<maxFlow)
            {
            if(duty<=255)duty++;
            //if (suhuawal<Tout)optimal=1; //  T start find flow when
rise
            if(suhuawal<Tout)optimal=3;
            }
            else if (LPM>=maxFlow)
             {
             if (duty>=1)duty--;
             //    if (suhuawal>Tout)optimal=3;
             }
           }

        if ((Tout>=Target_Temp-1)&&(optimal==2)) // maksimum
flow T down when speedup
           {if (LPM<maxFlow)
            {
             if (duty<=255)duty++;
             if (suhuawal<Tout)optimal=1;
            }
            else if (LPM>maxFlow)
              {
              if(duty>=1)duty--;
              }
           }
        else    if    ((Tout<=Target_Temp+1)&&(optimal==2))    //
maksimum flow at T down
```

```
            {
            if (duty>=1)duty--;
            }

          if  ((Tout>=Target_Temp-1)&&(optimal==3))  //minimum
flow at valve close
            {
              if(duty>=1)duty--;
            }
          else if ((Tout<=Target_Temp-1)&&(optimal==3))
            {
             if (LPM<maxFlow)
             {
              if (duty<=255)duty++;

             }
             else if (LPM>maxFlow)
               {
               if(duty>=1)duty--;
               }
            }




//-------------------------------------------------------------Override
(Exsa)----------------------
        if(override==1)
        {
        duty=remotevsd*2.55;
        }
//-------------------------------------------------------------------------
-----------------
         p1=1;
         OCR1BL=duty;
        }
        else
```

```
        {
        p1=0;
        OCR1BL=0;
        ledrun=0;
        ledover=0;

        }



        if
((refreshcount>=600)||(setawal=!Target_Temp)||(flowawal!=max
Flow))
            {

            if (flowawal!=maxFlow)
               { lcd_gotoxy(0,0);
                 lcd_putsf("Qset=");
                 sprintf(buff,"%2dL",maxFlow/10);
                 lcd_puts(buff);
               }
            else if ((refreshcount>=600)||(setawal=!Target_Temp))
              {   lcd_gotoxy(0,0);
                  lcd_putsf("Tset=");
                  sprintf(buff,"%2dC",Target_Temp);
                  lcd_puts(buff);
                  }
            refreshcount=0;optimal=0;
            }

        refreshcount++;
        setawal=Target_Temp;
        suhuawal=Tout;
        flowawal=maxFlow;
        lcd_gotoxy(9,0);
        lcd_putsf("T= ");
        sprintf(buff,"%4d",Tout);
```

```
        lcd_puts(buff);


        MsbLPM=LPM/10;
        lcd_gotoxy(0,1);
        lcd_putsf("l/m= ");
        sprintf(buff,"%d,%d ",MsbLPM,LPM-(MsbLPM*10));
        lcd_puts(buff);


        lcd_gotoxy(10,1);
        lcd_putsf("D= ");
        sprintf(buff,"%3d",duty);
        lcd_puts(buff);
        //backlight=~backlight;
        second = 0;

       }
      } }
}
```