UNIVERSITAS INDONESIA

RANCANG BANGUN SISTEM PEMANTAUAN DAN PENGENDALIAN SUHU TANGKI BERBASIS WEB SERVER MENGGUNAKAN MIKROKONTROLER ATMEGA8535

SKRIPSI

WISNU ANGGORO

0806366466

FAKULTAS TEKNIK

PROGRAM ELEKTRO EKSTENSI

DEPOK

JULI 2011

UNIVERSITAS INDONESIA

RANCANG BANGUN SISTEM PEMANTAUAN DAN PENGENDALIAN SUHU TANGKI BERBASIS WEB SERVER MENGGUNAKAN MIKROKONTROLER ATMEGA8535

SKRIPSI

Diajukan sebagai salah satu syarat untuk memperoleh gelar sarjana teknik

WISNU ANGGORO

0806366466

FAKULTAS TEKNIK

PROGRAM STUDI ELEKTRO EKSTENSI

DEPOK

JULI 2011

HALAMAN PERNYATAAN ORISINALITAS

Skripsi ini adalah hasil karya saya sendiri,

dan semua sumber baik yang dikutip maupun dirujuk

telah saya nyatakan dengan benar.

Nama	:	**Wisnu Anggoro**
NPM	:	**0806366466**
Tanda Tangan	:	
Tanggal	:	**01 Juli 2011**

HALAMAN PENGESAHAN

Skripsi ini diajukan oleh :

Nama	: Wisnu Anggoro
NPM	: 0806366466
Program Studi	: Elektro Ekstensi
Judul Skripsi	: Rancang Bangun Sistem Pemantauan dan Pengendalian Suhu Tangki Berbasis Web Server Menggunakan Mikrokontroler ATMega8535

Telah berhasil dipertahankan di hadapan Dewan Penguji dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Sarjana Teknik pada Program Studi Elektro Ekstensi, Fakultas Teknik Universitas Indonesia

DEWAN PENGUJI

Pembimbing : Ir. Amien Rahardjo MT. (.....)

Penguji : Dr. Abdul Muis ST., M.Eng. (.....)

Penguji : Dr. Abdul Halim M.Eng. (.....)

Ditetapkan di : Depok

Tanggal : 01 Juli 2011

KATA PENGANTAR

Puji syukur saya panjatkan kepada Tuhan Yang Maha Esa, karena atas berkat dan rahmat-Nya, saya dapat menyelesaikan skripsi ini. Penulisan skripsi ini dilakukan dalam rangka memenuhi salah satu syarat untuk mencapai gelar Sarjana Teknik Jurusan Elektro pada Fakultas Teknik Universitas Indonesia. Saya menyadari bahwa, tanpa bantuan dan bimbingan dari berbagai pihak, dari masa perkuliahan sampai pada penyusunan skripsi ini, sangatlah sulit bagi saya untuk menyelesaikan skripsi ini. Oleh karena itu, saya mengucapkan terima kasih kepada:

(1) Ir. Amien Rahardjo MT, selaku dosen pembimbing yang telah menyediakan waktu, tenaga, dan pikiran untuk mengarahkan saya dalam penyusunan skripsi ini;
(2) Orang tua dan keluarga yang telah memberikan bantuan dukungan baik material maupun moral;
(3) Vincentia Yuvenca Imramni, kekasih yang tak pernah lelah dan berhenti memberikan semangat dan dukungannya hingga selesainya skripsi ini; dan
(4) Semua rekan-rekan seangkatan yang turut memberikan saran dan informasinya.

Akhir kata, saya berharap semoga Tuhan Yang Maha Esa berkenan membalas segala kebaikan semua pihak yang telah membantu. Semoga skripsi ini membawa manfaat bagi pengembangan ilmu pengetahuan.

Depok, 01 Juli 2011

Penulis

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

Sebagai sivitas akademik Universitas Indonesia, saya yang bertanda tangan di bawah ini:

Nama	: Wisnu Anggoro
NPM	: 0806366466
Program Studi	: Elektro Ekstensi
Departemen	: Teknik Elektro
Fakultas	: Teknik
Jenis karya	: Skripsi

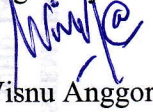
demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Indonesia **Hak Bebas Royalti Noneksklusif (Non-exclusive Royalty - Free Right)** atas karya ilmiah saya yang berjudul:

Rancang Bangun Sistem Pemantauan dan Pengendalian Suhu Tangki Berbasis Web Server Menggunakan Mikrokontroler ATMega8535

beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Noneksklusif ini Universitas Indonesia berhak menyimpan, mengalihmedia/formatkan, mengelola dalam bentuk pangkalan data (*database*), merawat, dan memublikasikan tugas akhir saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Depok
Pada tanggal : 01 Juli 2011
Yang menyatakan



(Wisnu Anggoro)

ABSTRAK

Nama : Wisnu Anggoro
Program Studi : Elektro Ekstensi
Judul : Rancang Bangun Sistem Pemantauan dan Pengendalian Suhu Tangki Berbasis Web Server Menggunakan Mikrokontroler ATMega8535

Tugas akhir ini membahas penggunaan internet dalam pengendalian dan pemantauan suhu di dalam tangki yang dilakukan dari jarak jauh dengan cara mengatur kondisi nyala atau mati pada pemanas dan pendingin. Metode yang dilakukan adalah perancangan dan pembuatan miniatur yang menyerupai keadaan pada proses industri dengan menggunakan mikrokontroler ATMega8535 sebagai pengendali utama dan ASP.net sebagai bahasa pemrograman dalam pembuatan *server* yang akan diakses oleh beberapa *client*. Hasilnya, sistem dapat berjalan dengan baik karena telah mampu merespon *input* dari *client* (*setting value*), membaca *output* dari sensor LM35 (*present value*), dan membandingkan kedua nilai *input* dan *output* tersebut. Suhu di dalam tangki dapat dikendalikan dengan toleransi sebesar ± 1°C dan untuk proses industri yang membutuhkan toleransi yang lebih kecil dapat mengubah kode program pada mikrokontroler agar pembacaan suhu yang terukur oleh sensor menjadi lebih presisi. Sistem juga dapat mendeteksi gangguan pada jalur komunikasi, baik komunikasi kabel maupun nirkabel sehingga saat terjadi gangguan pada jalur komunikasi, mikrokontroler akan menghentikan otomasi untuk mencegah terjadinya sistem yang tidak terkendali.

Kata kunci:
Internet, otomasi industri, mikrokontroler, ATMega8535, ASP.net, *Server*, *Client*

ABSTRACT

Name : Wisnu Anggoro
Study Program : Electrical Engineering Extention
Title : Design and Implementation of Controlling and Monitoring Tank's Temperature Web-based System By Using ATMega8535 Microcontroller

The focus of this final project is the using of internet in controlling and monitoring temperature inside the tank from long distance place by controlling on/off condition of heater and fan. The method of this study is designing and implementation the miniature like industrial process condition by using ATMega8535 microcontroller as the main controller and ASP.net as programming language for server which is going to be accessed by multiple client. The result is system can operate well because it is able to respond input from client (setting value), read output from LM35 sensor (present value), and compare both of setting value and present value. Temperature inside the tank can be controlled with a tolerance of ± 1°C and for industrial process that needs smaller tolerance, source code of microcontroller have to be changed in order that the reading of measured temperature from the sensor become more precise. The system can also detect disturbance on the line of communication, either wired or wireless communication so that when there is a disturbance on the line of communication, the microcontroller will stop the automation system to prevent uncontrolled condition.

Key words:
Internet, industrial automation, microcontroller, ATMega8535, ASP.net, server, client

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB 1

PENDAHULUAN

1.1. LATAR BELAKANG

Perkembangan dunia industri saat ini telah menghasilkan banyak pengaruh pada bidang otomasi. Pengendalian suatu proses industri tidak lagi harus dilakukan oleh seorang operator yang siap menjaganya 24 jam, namun kini dapat dilakukan oleh sebuah sistem otomatis yang akan menjaga kestabilan masukan dan keluaran pada proses tersebut. Operator hanya cukup menentukan nilai keluaran yang diharapkan (*setting point*) maka kemudian proses akan berjalan dan menghasilkan keluaran yang telah ditentukan sesuai dengan *setting point* tersebut. Dan selebihnya operator dapat melakukan pemantauan dari proses industri tersebut pada sebuah tempat yang biasa disebut ruangan kontrol (*control room*).

Namun ada kalanya terjadi sebuah kendala saat tidak adanya seorang pun yang berada di ruangan kontrol yang akan memasukkan nilai *set point* jika terjadi perubahan terhadap keluaran yang diinginkan dan melakukan pemantauan dari proses industri tersebut. Sebagai contoh adalah seorang pimpinan perusahaan yang ingin mengetahui keadaan dari proses industri yang sedang terjadi namun pada saat itu dia sedang berada jauh dari area industrinya.

Pengendalian dan pemantauan proses industri dari jarak jauh berbasis web merupakan sebuah solusi yang dapat diterapkan untuk memecahkan masalah di atas. Di mana pun operator berada, dia dapat terus mengendalikan dan memantau keadaan proses, tanpa ada batasan jarak sama sekali. Selain itu, proses industri akan dapat dikendalikan dan dipantau oleh sebanyak mungkin orang, kapanpun dan di manapun.

1.2. PERUMUSAN MASALAH

Berdasarkan latar belakang tersebut, dapat dirumuskan masalah yaitu bagaimana merancang sebuah sistem yang mampu melakukan pemantauan dan pengendalian peralatan industri dari jarak jauh yang tidak dibatasi oleh waktu dan tempat serta mudah dan handal dalam pengoperasian dan pemeliharaannya.

1.3. TUJUAN PENELITIAN

Tujuan dari dilakukannya skripsi ini adalah studi perancangan untuk menghasilkan sistem yang mampu mengendalikan dan memantau suhu tangki berbasis *web server* menggunakan mikrokontroler ATMega8535.

1.4. PEMBATASAN MASALAH

Masalah yang dikemukakan pada skripsi ini dibatasi pada perancangan rangkaian untuk mikrokontroler dan pemrogramannya agar dapat digunakan untuk melakukan pengendalian suhu di dalam tangki maupun untuk memperoleh data suhu pada tangki tersebut di mana pengendalian suhu yang dilakukan adalah dengan teknik *on/off* pemanas dan pendingin.

1.5. SISTEMATIKA PENULISAN

Sistematika penulisan skripsi ini dibagi atas lima bab. Pembagian babnya adalah sebagai berikut, bab satu berisi pendahuluan yang membahas tentang latar belakang penulisan, tujuan penulisan, batasan masalah dan sistematika penulisan untuk memberikan gambaran umum mengenai permasalahan yang dibahas dalam skripsi ini. Bab dua berisi teori dasar yang mendukung dan untuk selanjutnya digunakan pada bagian pembahasan. Teori yang dibahas antara lain tentang mikrokontroler, sensor suhu, dan *web server*. Bab tiga berisi rancang bangun sistem pemantauan dan pengendalian suhu tangki berbasis *web server*, meliputi

perancangan perangkat keras dan perangkat lunak. Bab empat berisi langkah-langkah ujicoba agar dapat dilakukan analisa pada sistem pemantauan dan pengendalian suhu tangki berbasis *web server*. Bab lima berisi kesimpulan dari rancang bangun sistem pemantauan dan pengendalian suhu tangki berbasis *web server*.

BAB 2

LANDASAN TEORI

2.1. Mikrokontroler

Mikrokontroler sering disebut dengan *Single Chip Computer* atau suatu kepingan IC di mana di dalamnya terdapat mikroprosesor dan memori program (ROM) beserta memori serba guna (RAM), *Input/Output* dan fasilitas pendukung lainnya. Mikrokontroler ini memiliki kemampuan untuk diprogram dan digunakan untuk suatu proses pengendalian. Terdapat beberapa perbedaan antara mikroprosesor dengan mikrokontroler antara lain adalah mikroprosesor hanya berupa *single chip* CPU (*Central Processing Unit*) tanpa memori dan *peripheral* lainnya sebagai pendukung sebuah komputer, sedangkan mikrokontroler adalah sebuah *chip* CPU yang cukup lengkap karena memiliki antara lain *Flash Memory*, RAM, antarmuka serial (USART), pewaktu (*timer*), dan penyela (*interrupt*).

Mikrokontroler muncul dengan dua alasan utama, yaitu kebutuhan pasar (*market need*) dan perkembangan teknologi baru (*expansion of technology*). Yang dimaksud dengan kebutuhan pasar adalah kebutuhan yang luas dari produk-produk elektronik akan perangkat pintar sebagai pengendali dan pemroses data. Sedangkan yang dimaksud dengan perkembangan teknologi baru adalah perkembangan teknologi semikonduktor yang memungkinkan pembuatan *chip* dengan kemampuan komputasi yang sangat cepat, bentuk yang semakin kecil, dan harga yang semakin murah. Karena kemampuannya yang tinggi, bentuknya yang kecil, konsumsi dayanya yang rendah, dan harga yang murah maka mikrokontroler begitu banyak digunakan, seperti pada mainan anak-anak, perangkat elektronik rumah tangga, peralatan telekomunikasi, peralatan medis dan kedokteran.

Salah satu mikrokontroler yang banyak digunakan saat ini adalah mikrokontroler AVR, yang merupakan mikrokontroler berjenis RISC (*Reduce*

Instruction Set Compute) 8 bit berdasarkan arsitektur Harvard, yang dibuat oleh Atmel pada tahun 1996. AVR merupakan singkatan dari *Advanced Versatile RISC* atau *Alf and Vegard's Risc processor* yang berasal dari nama dua mahasiswa Norwegian Institute of Technology, yaitu Alf-Egil Bogen dan Vegard Wollan.

AVR memiliki keunggulan dibandingkan mikrokontroler lainnya, yaitu memiliki kecepatan eksekusi program yang lebih cepat karena sebagian besar instruksi dieksekusi dalam satu siklus *clock*, lebih cepat dibandingkan dengan mikrokontroler keluarga MCS51 yang memiliki arsitektur CISC (*Complex Instruction Set Compute*) di mana membutuhkan 12 siklus *clock* untuk mengeksekusi satu instruksi. Selain itu mikrokontroler AVR memiliki fitur yang lengkap (ADC internal, EEPROM internal, *Timer/Counter*, *Watchdog Timer*, PWM, *Port Input – Output*, komunikasi serial, komparator, dan I2C) sehingga dengan fasilitas yang lengkap ini, mikrokontroler AVR dapat dengan mudah digunakan untuk berbagai aplikasi sistem elektronika, seperti robotika, otomasi industri, peralatan telekomunikasi, dan berbagai keperluan lainnya.

Secara umum mikrokontroler AVR dapat dikelompokkan menjadi tiga kelompok, yaitu keluarga ATtiny, keluarga AT90Sxx, dan keluarga ATMega. Perbedaan mendasar terletak pada jumlah pin dan ukuran *internal memory*. Tabel 2.1 menjelaskan perbedaan dari jenis-jenis mikrokontroler AVR.

Tabel 2.1. Jenis-jenis Mikrokontroler AVR[1]

Jenis	Jumlah Pin	Flash Memory	EEPROM	SRAM
ATTiny	8 - 32	1 – 2 KByte	64 - 128 Byte	0 - 128 Byte
AT90Sxx	20 - 44	1 – 8 KByte	128 - 512 Byte	0 - 1 KByte
ATMega	32 - 64	8 – 128 KByte	512 B - 4 KByte	512 B - 4 KByte

Dari Tabel 2.1 terlihat bahwa jenis ATMega lebih unggul bila dibandingkan dengan jenis ATTiny dan AT90Sxx, sehingga jenis AVR ATMega dipilih sebagai mikrokontroler untuk skripsi ini.

Terdapat empat macam mikrokontroler jenis AVR yang banyak ditemukan dipasaran, yaitu ATMega8, ATMega8535, ATMega16, dan ATMega32. Perbedaannya terletak pada kapasitas memori internal yang terdapat pada mikrokontroler tersebut, seperti terlihat pada Tabel 2.2.

Tabel 2.2. Kapasitas memori internal mikrokontroler AVR[2]

Jenis	Flash Memory	RAM	EEPROM
ATMega8	8 KByte	1 KByte	512 Byte
ATMega8535	8 KByte	512 Byte	512 Byte
ATMega16	16 KByte	1 KByte	512 Byte
ATMega32	32 KByte	2 KByte	1 KByte

Selain perbedaan kapasitas memori internal, keempat jenis mikrokontroler AVR di atas juga memiliki perbedaan pada jumlah pin port dan fungsi dari masing-masing pin tersebut. Tabel 2.3 memperlihatkan perbandingan fungsi pin pada keempat jenis mikrokontroler AVR.

Tabel 2.3. Perbandingan fungsi pin mikrokontroler AVR[3]

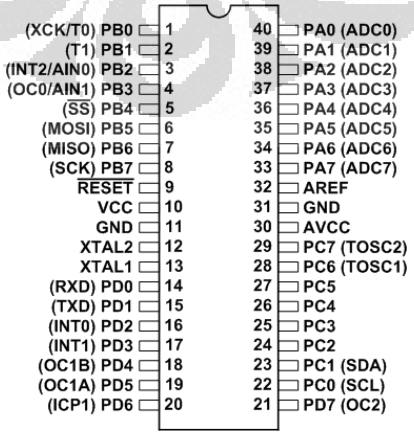
	ATMega8	ATMega8535	ATMega16	ATMega32
Port A	-	PA0 (ADC0)	PA0 (ADC0)	PA0 (ADC0)
	-	PA1 (ADC1)	PA1 (ADC1)	PA1 (ADC1)
	-	PA2 (ADC2)	PA2 (ADC2)	PA2 (ADC2)
	-	PA3 (ADC3)	PA3 (ADC3)	PA3 (ADC3)
	-	PA4 (ADC4)	PA4 (ADC4)	PA4 (ADC4)
	-	PA5 (ADC5)	PA5 (ADC5)	PA5 (ADC5)
	-	PA6 (ADC6)	PA6 (ADC6)	PA6 (ADC6)
	-	PA7 (ADC7)	PA7 (ADC7)	PA7 (ADC7)
Port B	PB0 (ICP1)	PB0 (XCK/T0)	PB0 (XCK/T0)	PB0 (XCK/T0)
	PB1 (OC1A)	PB1 (T1)	PB1 (T1)	PB1 (T1)
	PB2 (SS/OC1B)	PB2 (INT2/AIN0)	PB2 (INT2/AIN0)	PB2 (INT2/AIN0)
	PB3 (MOSI/OC2)	PB3 (OC0/AIN1)	PB3 (OC0/AIN1)	PB3 (OC0/AIN1)
	PB4 (MISO)	PB4 (SS^)	PB4 (SS^)	PB4 (SS^)
	PB5 (SCK)	PB5 (MOSI)	PB5 (MOSI)	PB5 (MOSI)
	PB6 (XTAL1/TOSC1)	PB6 (MISO)	PB6 (MISO)	PB6 (MISO)
	PB7 (XTAL2/TOSC2)	PB7 (SCK)	PB7 (SCK)	PB7 (SCK)
Port C	PC0 (ADC0)	PC0 (SCL)	PC0 (SCL)	PC0 (SCL)
	PC1 (ADC1)	PC1 (SDA)	PC1 (SDA)	PC1 (SDA)
	PC2 (ADC2)	PC2	PC2 (TCK)	PC2 (TCK)
	PC3 (ADC3)	PC3	PC3 (TMS)	PC3 (TMS)
	PC4 (ADC4/SDA)	PC4	PC4 (TDO)	PC4 (TDO)
	PC5 (ADC5/SCL)	PC5	PC5 (TDI)	PC5 (TDI)
	PC6 (RESET)	PC6 (TOSC1)	PC6 (TOSC1)	PC6 (TOSC1)
	-	PC7 (TOSC2)	PC7 (TOSC2)	PC7 (TOSC2)
Port D	PD0 (RXD)	PD0 (RXD)	PD0 (RXD)	PD0 (RXD)
	PD1 (TXD)	PD1 (TXD)	PD1 (TXD)	PD1 (TXD)
	PD2 (INT0)	PD2 (INT0)	PD2 (INT0)	PD2 (INT0)
	PD3 (INT1)	PD3 (INT1)	PD3 (INT1)	PD3 (INT1)
	PD4 (XCK/T0)	PD4 (OC1B)	PD4 (OC1B)	PD4 (OC1B)
	PD5 (T1)	PD5 (OC1A)	PD5 (OC1A)	PD5 (OC1A)
	PD6 (AIN0)	PD6 (ICP1)	PD6 (ICP1)	PD6 (ICP1)
	PD7 (AIN1)	PD7 (ICP2)	PD7 (ICP2)	PD7 (ICP2)

Dari Tabel 2.3 terlihat bahwa mikrokontroler AVR jenis ATTiny hanya memiliki 23 pin port (PB0 – PB7, PC0 – PC6, dan PD0 – PD7) sedangkan untuk ketiga jenis lainnya memiliki 32 pin port (PA0 – PA7, PB0 – PB7, PC0 – PC7, dan PD0 – PD7). Fungsi mikrokontroler jenis ATTiny menjadi terbatas karena keterbatasan jumlah pin port jika dibandingkan dengan jenis ATMega8535, ATMega16, maupun ATMega32. Tabel 2.3 juga menunjukkan bahwa walaupun memiliki jumlah pin port yang sama, ATMega8535 memiliki beberapa perbedaan pada fungsi khusus pin port jika dibandingkan dengan jenis ATMega16 dan ATMega32, yaitu pada pin port PC2 – PC5. ATMega8535 tidak memiliki fungsi TCK (*Test Clock*), TMS (*Test Mode Select*), TDO (*Test Data Out*), dan TDI (*Test Data In*).

Berdasarkan Tabel 2.2 dan Tabel 2.3, dapat disimpulkan bahwa ATMega8 tidak dapat digunakan untuk rancang bangun skripsi ini karena keterbatasan jumlah *pin port*. Dan karena dalam rancang bangun skripsi ini tidak menggunakan fungsi TCK, TMS, TDO, dan TDI serta tidak diperlukannya kapasitas memori yang besar, maka mikrokontroler AVR jenis ATMega8535 digunakan untuk rancang bangun skripsi ini.

2.1.1. Konfigurasi Pin ATMega8535[4]

Konfigurasai pin mikrokontroler AVR jenis ATMega8535 dengan kemasan 40 pin DIP (*Dual In-Line*) ditunjukkan pada gambar 2.1.



Gambar 2.1. Konfigurasi Pin ATMega8535

Fungsi dari masing-masing pin ATMega8535 berdasarkan gambar 2.1 adalah sebagai berikut:

1. VCC merupakan pin yang berfungsi sebagai pin masukan catu daya. Menurut *datasheet*, mikrokontroler ATMega8535 membutuhkan tegangan kerja sebesar 4.5 Volt – 5.5 Volt.

2. GND merupakan pin ground (0 Volt).

3. Port A (PA0 - PA7) merupakan pin I/O dua arah dan pin masukan *Analog to Digital Converter* (ADC0 – ADC7).

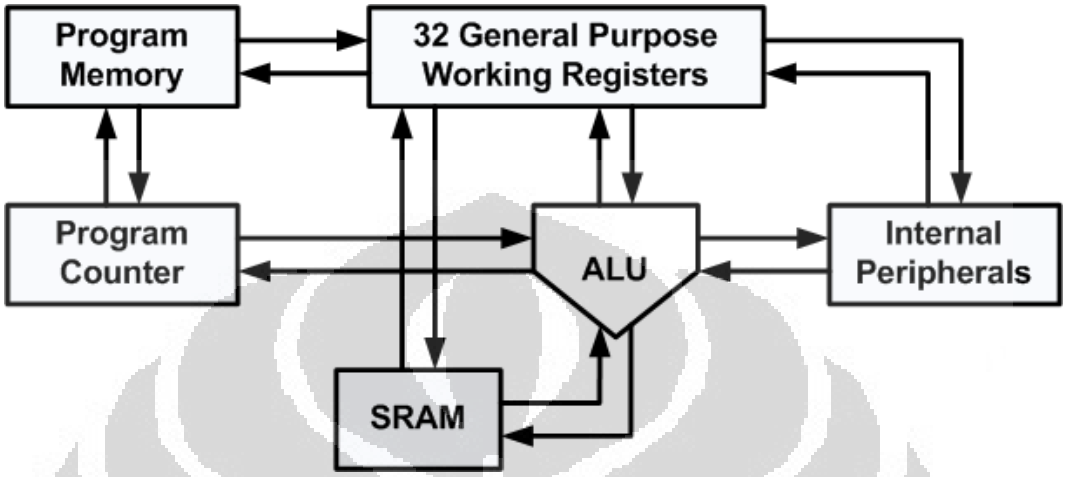
4. Port B (PB0 - PB7) merupakan pin I/O dua arah dan pin fungsi khusus, yaitu:

   - PB0 : T0 (*Timer/Counter 0 External Counter Input*) dan XCK (*USART External Clock Input/Output*)
   - PB1 : T1 (*Timer/Counter 1 External Counter Input*)
   - PB2 : AIN0 (*Analog Comparator Positive Input*) dan INT2 (*External Interrupt 2 Input*)
   - PB3 : AIN1 (*Analog Comparator Negative Input*) dan OC0 (*Timer/Counter 0 Output Compare Match Output*)
   - PB4 : SS – *active low* (SPI *Slave Select Input*)
   - PB5 : MOSI (SPI *Bus Master Input/ Slave Output*)
   - PB6 : MISO (SPI *Bus Master Output/ Slave Input*)
   - PB7 : SCK (*SPI Bus Serial Clock*)

5. Port D (PD0 - PD7) merupakan pin I/O dua arah dan pin fungsi khusus, yaitu:
   - PD0 : RXD (*USART Input Pin*)
   - PD1 : TXD (*USART Output Pin*)
   - PD2 : INT0 (*External Interupt 0 Input*)
   - PD3 : INT1 (*External Interupt 1 Input*)
   - PD4 : OC1B (*Timer/Counter 1 Output Compare B Match Output*)
   - PD5 : OC1A (*Timer/Counter 1 Output Compare A Match Output*)
   - PD6 : ICP (*Timer/Counter 1 Input Capture Pin*)
   - PD7 : OC2 (*Timer/Counter 2 Output Compare Match Output*)
6. Port C (PC0 - PC7) merupakan pin I/O dua arah dan pin fungsi khusus, yaitu:
   - PC0 : SCL (*Two-wire Serial Bus Clock Line*)
   - PC1 : SDA (*Two-wire Serial Bus Data Input/Output Line*)
   - PC6 : TOSC1 (*Timer Oscillator Pin 1*)
   - PC7 : TOSC2 (*Timer Oscillator Pin 1*)
7. RESET merupakan pin yang digunakan untuk me-*reset* mikrokontroler.
8. XTAL1 dan XTAL2 merupakan pin masukan *clock* eksternal.
9. AVCC merupakan pin masukan tegangan untuk ADC.
10. AREF merupakan pin masukan tegangan referensi ADC.

2.1.2. Arsitektur ATMega8535[5]

Secara umum, mikrokontroler jenis AVR memiliki arsitektur seperti terlihat pada Gambar 2.2.



Gambar 2.2. Arsitektur Sedehana Mikrokontroler AVR

Penjelasan dari gambar 2.2 adalah sebagai berikut:

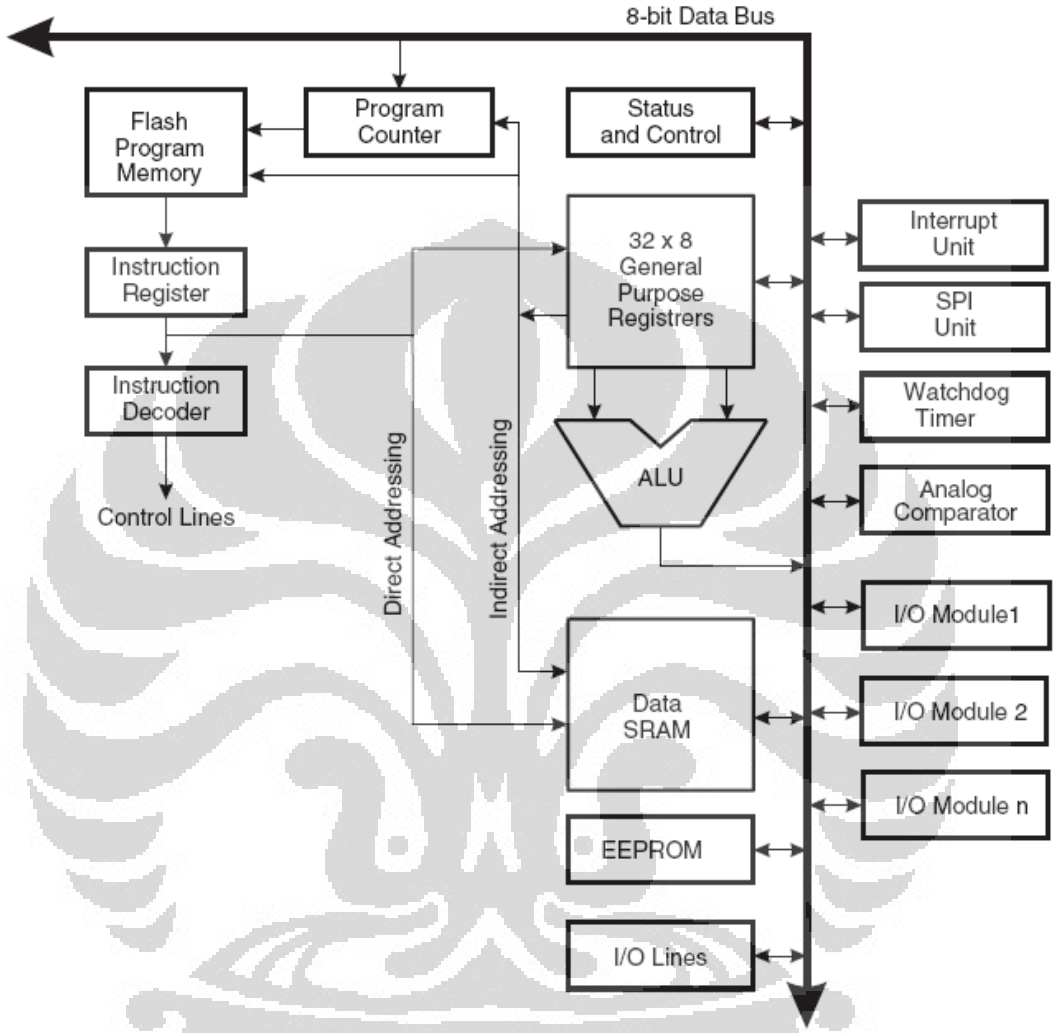
- ALU (*Arithmetic Logic Unit*) adalah *processor* yang bertugas mengeksekusi kode program yang ditunjuk oleh *program counter*.
- *Program Memory* adalah memori *flash* PEROM yang bertugas menyimpan program (*software*) yang dibuat dalam bentuk kode-kode program (berisi alamat memori beserta kode program dalam ruangan memori alamat tersebut) yang telah dikompilasi menjadi bilangan heksa desimal atau biner.
- *Program Counter* adalah komponen yang bertugas menunjuk alamat program memori ke ALU yang harus diterjemahkan kode programnya dan dieksekusi. Sifat dari *program counter* adalah linear yang berarti akan menghitung naik satu bilangan yang bergantung dari alamat awalnya.

- 32 *General Purpose Working Registers* (GPR) adalah register kerja yang bertugas sebagai tempat untuk ALU dalam mengeksekusi kode-kode program. GPR terbagi menjadi dua, yaitu kelompok atas (R16 – R31) dan kelompok bawah (R0 – R15). Kelompok bawah tidak dapat digunakan untuk mengakses data secara langsung seperti instruksi *assembly* LDI (hanya bisa digunakan antar register), SRAM, atau register I/O (register port). Sedangkan kelompok atas sama dengan kelompok bawah hanya saja terdapat kelebihan yaitu dapat mengakses data secara langsung.

- SRAM (*Static Random Access Memory*) adalah memori yang bertugas menyimpan data sementara dan sama seperti RAM pada umumnya yang memiliki alamat dan ruangan data. Alamat terakhir dari SRAM tergantung pada kapasitasnya.

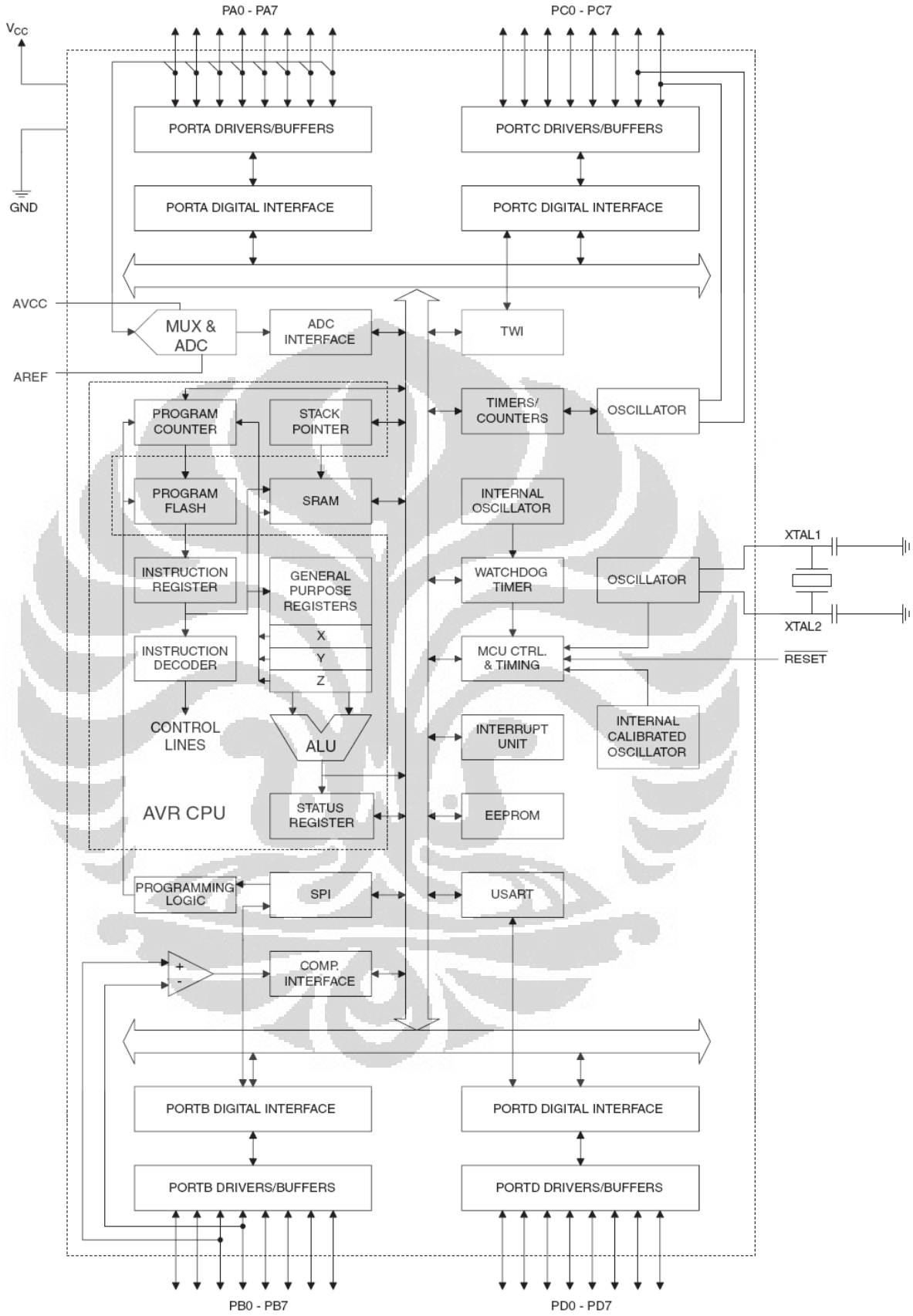
- *Internal Peripherals* adalah kumpulan peralatan atau modul internal yang ada dalam mikrokontroler seperti saluran I/O, interupsi eksternal, *Timer/Counter*, USART, EEPROM, dan lainnya. Tiap peralatan internal memiliki *register port* (*register* I/O) yang mengendalikan peralatan internal tersebut.

Gambar 2.3 di bawah ini memperlihatkan arsitektur dari mikrokontroler AVR ATMega8535. Cara kerjanya secara garis besar mirip dengan penjelasan untuk gambar 2.2.



Gambar 2.3. Arsitektur Lengkap Mikrokontroler AVR

Dari gambar 2.3 terlihat bahwa AVR menggunakan arsitektur Harvard dengan memisahkan antara memori dan *bus* untuk program dan data agar memaksimalkan kemampuan dan kecepatan. Instruksi dalam memori program dieksekusi dengan *pipelining single level*, yaitu ketika satu instruksi dieksekusi, instruksi berikutnya diambil dari memori program. Konsep ini mengakibatkan instruksi dieksekusi setiap *clock cycle*.



Gambar 2.4. Blok Diagram ATMega8535

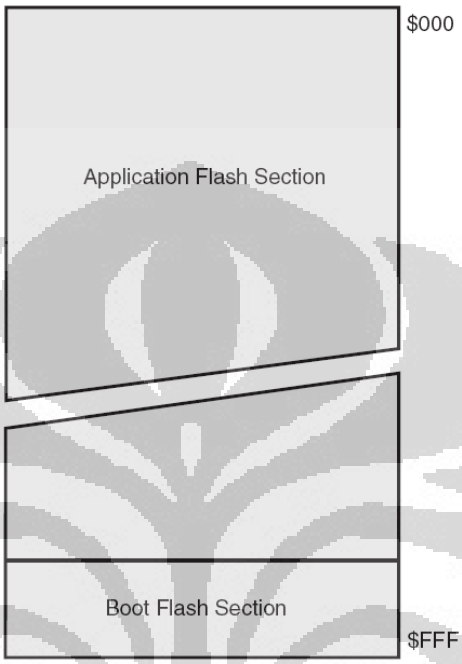
Dari gambar 2.4 terlihat blok diagram dari mikrokontroler ATMega8535 dan berdasarkan gambar tersebut dapat dilihat beberapa fitur dari ATMega8535, antara lain:

1. Saluran I/O ada 32 buah, yaitu Port A, Port B, Port C dan Port D.
2. ADC (*Analog Digital Converter*) 10 bit sebanyak 8 *channel*.
3. Tiga buah *timer/counter* dengan kemampuan pembandingan.
4. CPU yang terdiri dari 32 buah *register*.
5. *Watchdog Timer* dengan *osilator internal*.
6. Dua buah timer/counter 8 bit, satu buah *timer/counter* 16 bit.
7. SRAM internal 512 Byte.
8. Memori *flash* sebesar 8 KByte dengan kemampuan *Read While Write*.
9. Unit interupsi internal dan eksternal.
10. Port antar muka SPI.
11. EEPROM sebesar 512 byte yang dapat diprogram saat beroperasi.
12. Antarmuka komparator analog.
13. 4 Channel *Pulse Width Modulations* (PWM).
14. Hampir mencapai 16 MIPS pada kristal 16KHz.
15. Port USART *Programmable* untuk komunikasi serial.

2.1.3. Peta Memori ATMega8535[6]

Pada tabel 2.2 telah dijelaskan bahwa memori pada mikrokontroler AVR terbagi atas tiga jenis, yaitu *Flash Memory*, RAM (*Random Access Memory*), dan EEPROM (*Electrically Erasable Programmable Read Only Memory*). Masing-masing jenisnya memiliki kapasitas dan fungsi yang berbeda-beda. Penjelasan dari masing-masing fungsi memori yang terdapat pada mikrokontroler AVR ATMega8535 adalah sebagai berikut:

- *Flash Memory* adalah memori ROM (*read only memory*) tempat kode-kode program berada. Kata *flash* menunjukkan jenis ROM yang dapat ditulis dan dihapus secara elektrik. Memori flash terbagi menjadi dua bagian yaitu bagian aplikasi dan bagian *boot* seperti terlihat pada gambar 2.5.



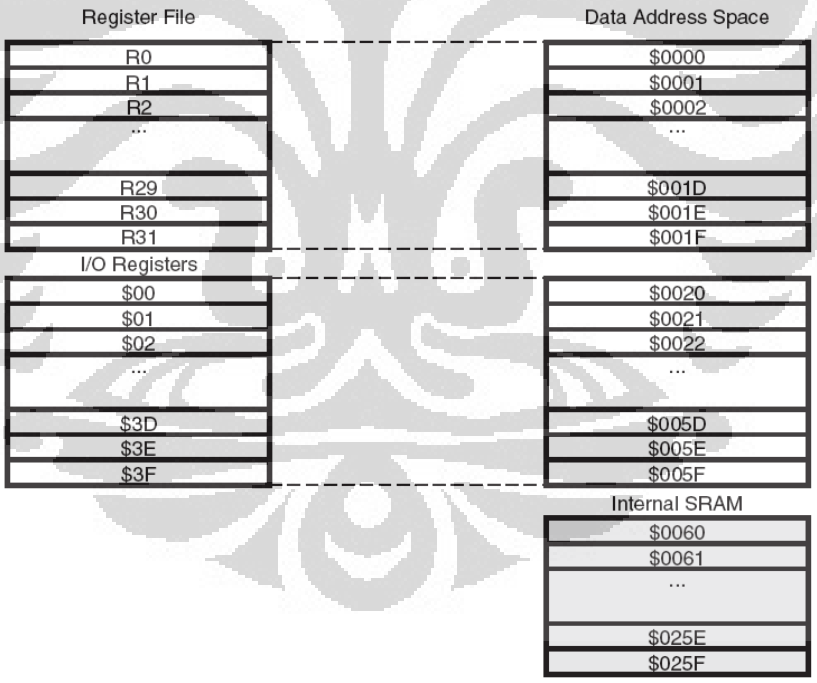
Gambar 2.5. Peta Memori *Flash* ATMega8535

Bagian *boot* adalah bagian yang digunakan khusus untuk *booting* awal yang dapat diprogram untuk menulis bagian aplikasi tanpa melalui *programmer* atau *downloader*, misalnya melalui USART (*serial port*). Pemakaian bagian *boot* akan secara otomatis mengurangi kapasitas memori *flash*.

ATMega8535 memiliki 8Kbyte memori *flash* untuk tempat penyimpanan program. Program yang terletak dalam memori *flash* tersebut tersusun dalam *word* atau 2 byte karena setiap instruksi memiliki lebar 16 bit atau 32 bit., sehingga pengorganisasian memorinya 4 KByte x 16 bit Flash PEROM dengan alamat mulai dari $000 sampai $FFF seperti yang ditunjukkan pada gambar 2.5.

- RAM adalah memori data yang digunakan untuk keperluan program. Memori data yang terdapat pada ATMega 8535 terdiri dari tiga bagian, yaitu 32 *General Purpose Register* (GPR), 64 buah I/O *Register*, dan 512 byte SRAM internal. GPR menempati ruang data pada alamat $00 sampai $1F. Sementara itu, I/O *Register* menempati 64 alamat berikutnya, yaitu mulai dari $20 hingga $5F. Alamat memori berikutnya digunakan untuk SRAM 512 byte, yaitu pada lokasi $60 sampai dengan $25F. Peta memori data ditunjukkan pada gambar 2.6.

  32 *General Purpose Register* (GPR) adalah register khusus yang betugas untuk membantu eksekusi program oleh ALU. Dalam instruksi *assembler*, setiap instruksi harus melibatkan GPR. Dan I/O register berfungsi khusus untuk mengendalikan berbagai *peripheral* dalam mikrokontroler seperti *pin port*, *timer/counter*, dan USART.



Gambar 2.6. Peta Memori Data ATMega8535

- EEPROM (*Electrically Erasable Programmable Read Only Memory*) adalah memori yang masih dapat menyimpan data walaupun catu daya dimatikan. Memori ini digunakan untuk keperluan menyimpan data yang tahan terhadap gangguan catu daya.

2.1.4. Interupsi pada Mikrokontroler ATMega8535

Interupsi adalah kondisi di mana pada saat program utama dieksekusi oleh CPU kemudian tiba-tiba berhenti untuk sementara waktu karena ada rutin lain yang harus ditangani terlebih dahulu oleh CPU, dan setelah selesai mengerjakan rutin tersebut maka CPU akan kembali mengerjakan instruksi pada program utama.[7] Tabel 2.4 memperlihatkan detail vektor interupsi yang menunjukkan 21 sumber interupsi yang terdapat pada mikrokontroler ATMega8535.

Tabel 2.4. Vektor Interupsi Pada Mikrokontroler ATMega8535

Nomor Vektor	Alamat Program	Sumber Interupsi	Definisi Interupsi
1	0x000	RESET	*External Pin, Power-on Reset, Brown-out Reset and Watchdog Reset*
2	0x001	INT0	*External Interrupt Request 0*
3	0x002	INT1	*External Interrupt Request 1*
4	0x003	TIMER2 COMP	*Timer/Counter2 Compare Match*
5	0x004	TIMER2 OVF	*Timer/Counter2 Overflow*
6	0x005	TIMER1 CAPT	*Timer/Counter1 Capture Event*
7	0x006	TIMER1 COMPA	*Timer/Counter1 Compare Match A*
8	0x007	TIMER1 COMPB	*Timer/Counter1 Compare Match B*
9	0x008	TIMER1 OVF	*Timer/Counter1 Overflow*
10	0x009	TIMER0 OVF	*Timer/Counter0 Overflow*
11	0x00A	SPI, STC	*Serial Transfer Complete*
12	0x00B	USART, RXC	*USART, Rx Complete*
13	0x00C	USART, UDRE	*USART Data Register Empty*
14	0x00D	USART, TXC	*USART, Tx Complete*
15	0x00E	ADC	*ADC Conversion Complete*
16	0x00F	EE_RDY	*EEPROM Ready*
17	0x010	ANA_COMP	*Analog Comparator*
18	0x011	TWI	*Two-wire Serial Interface*
19	0x012	INT2	*External Interrupt Request 2*
20	0x013	TIMER0 COMP	*Timer/Counter0 Compare Match*
21	0x014	SPM_RDY	*Store Program Memory Ready*

Sumber: *Datasheet* Atmel ATMega8535

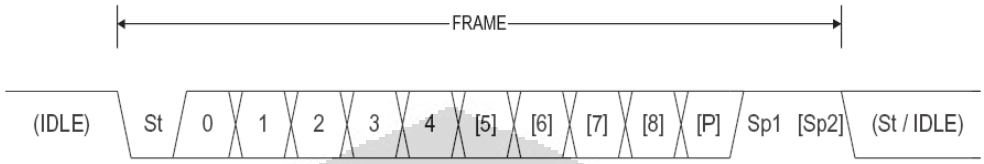
Interupsi digunakan untuk pembuatan tombol darurat (*emergency*) di mana tombol ini merupakan salah satu syarat dalam otomasi industri sebagai alat penjamin keselamatan *operator* dalam menjalankan peralatan-peralatan yang berjalan secara otomatis. Dengan menekan tombol ini, maka seluruh kegiatan otomasi akan berhenti sehingga hal-hal yang membahayakan operator dapat dicegah. Tombol ini juga digunakan saat perbaikan dan perawatan peralatan-peralatan otomasi untuk menjamin agar peralatan tersebut tidak akan berjalan secara otomatis saat dalam perbaikan.

Dari tabel 2.4 dapat dipilih urutan interupsi teratas agar tidak terganggu oleh aktivitas interupsi lainnya. *External Interrupt Request 0* digunakan sebagai sumber interupsi untuk membuat tombol *emergency*. Saat tombol *emergency* ditekan dan terkunci secara mekanik, interupsi mikrokontroler akan aktif sehingga jalannya program akan terhenti untuk kemudian akan menjalankan program yang terdapat pada fungsi *External Interrupt Request 0* di mana program tersebut akan memerintahkan mikrokontroler untuk mematikan semua aktivitas yang berbahaya seperti memutus sambungan listrik, menghentikan putaran mesin, menghentikan laju *conveyor*, menghentikan aliran udara panas, dan sebagainya.

2.1.5. Komunikasi Serial pada Mikrokontroler ATMega8535[8]

Transmisi data seri dibedakan menjadi dua macam, yaitu komunikasi data sinkron dan komunikasi data asinkron. Perbedaannya terletak pada *clock* pendorong data. Dalam komunikasi data sinkron, *clock* untuk *shift register* ikut dikirimkan bersama dengan data seri. Sebaliknya dalam komunikasi data asinkron, clock pendorong *shift register* tidak ikut dikirim sehingga rangkaian penerima data harus dilengkapi dengan rangkaian yang mampu membangkitkan *clock* yang bisa dipakai untuk mendorong shift register penerima. Untuk keperluan tersebut terlebih dulu ditentukan bahwa saat tidak ada pengiriman data, keadaan saluran adalah logika ‘1’, saat akan mulai mengirim data 1 byte saluran dibuat menjadi logika ‘0’ terlebih dahulu selama satu periode *clock* pendorong, kemudian dalam delapan periode *clock* berikutnya dikirim data dari bit pertama

sampai bit kedelapan (total menjadi 8 bit atau 1 byte), dan pada periode *clock* kesepuluh saluran dikembalikan menjadi logika ‘1’. Dengan demikian, data 8 bit yang dikirim memiliki format (*data frame*): diawali dengan bit start yang berlogika ‘0’, dan diakhiri dengan bit stop yang berlogika ‘1’. Gambar 2.7 menunjukkan data frame untuk komunikasi serial asinkron.



Gambar 2.7. Data Frame Komunikasi Serial Asinkron

Keterangan untuk gambar 2.7 adalah sebagai berikut:

St : *Start* bit (selalu berlogika ‘0’)

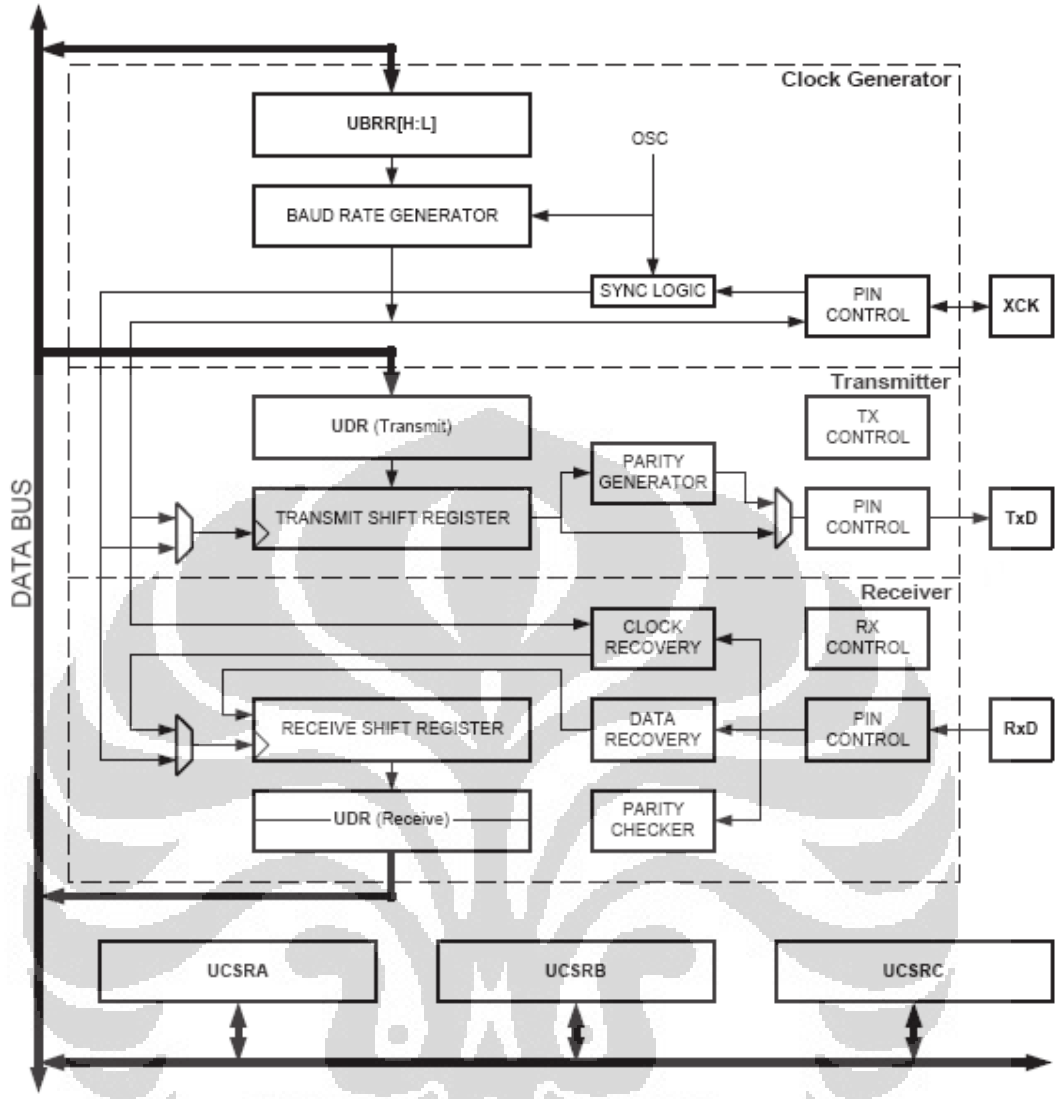
0 – 8 : Data bit (bit pertama sampai bit kedelapan)

P : *Parity* bit (tidak harus ada)

Sp : *Stop* bit (selalu berlogika ‘1’)

IDLE : Tidak ada transmisi data di dalam saluran (harus berlogika ‘1’)

Pada mikrokontroler ATMega8535, fungsi USART (*Universal Syncronous Asyncronous Receiver Transmitter*) digunakan untuk keperluan komunikasi data serial. USART terbagi dalam tiga blok, yaitu clock generator, transmitter, dan receiver seperti terlihat pada gambar 2.8.



Gambar 2.8. Blok Diagram USART ATMega8535

Clock generator berhubungan dengan kecepatan transfer data (*baud rate*) dan register yang bertugas menentukan *baud rate* adalah register UBRR (*USART Baud Rate Register*). UBRR merupakan register 16 bit yang dibagi menjadi dua yaitu UBRRH dan UBRRL. UBRRH menyimpan 4 bit tertinggi dan UBRRL menyimpan 8 bit sisanya.[9] Untuk menghitung nilai UBRR dari kecepatan transfernya (*baudrate*) atau menghitung *baudrate* dari nilai UBRR yang diketahui, dapat menggunakan rumus pada Tabel 2.5.

Tabel 2.5. Rumus Menghitung Nilai UBRR dan *Baudrate*

Mode Operasi	Rumus Menghitung Nilai UBRR	Rumus Menghitung Baud Rate
Mode Asinkron Kecepatan Normal (U2X = 0)	$UBRR = \frac{f_{osc}}{16xBaudRate} - 1$	$BAUD = \frac{f_{osc}}{16(UBRR+1)}$
Mode Asinkron Kecepatan Ganda (U2X = 1)	$UBRR = \frac{f_{osc}}{8xBaudRate} - 1$	$BAUD = \frac{f_{osc}}{8(UBRR+1)}$
Mode Sinkron	$UBRR = \frac{f_{osc}}{2xBaudRate} - 1$	$BAUD = \frac{f_{osc}}{2(UBRR+1)}$

Sumber: *Datasheet* Atmel ATMega8535

Keterangan Tabel 2.5:

BAUD : *Baud rate* [bits per second, bps]

f_{osc} : Frekuensi *clock* sumber untuk mikrokontroler [hertz, Hz]

UBRR : Nilai dari register UBRRH dan UBRRL [0 – 4095]

Contoh untuk menghitung nilai UBRR pada mode asinkron kecepatan normal jika diinginkan *baudrate* sebesar 9600 bps dengan frekuensi *clock* sebesar 4 MHz adalah:

$$UBRR = \frac{f_{osc}}{16xBaudRate} - 1 = \frac{4000000}{16x9600} - 1 = 25{,}041 \approx 25 \quad \text{.....(2.1)}$$

Sehingga nilai UBRRL = 25 dan UBRRH = 0.

Dari persamaan 2.1 terlihat bahwa nilai UBRR mengalami pembulatan angka karena register UBRRH dan UBRRL hanya dapat menampung bilangan *integer* (bilangan bulat). Dan jika nilai UBRR tersebut digunakan untuk menghitung *baud rate* untuk frekuensi *clock* yang sama yaitu 4 MHz, maka akan diperoleh *baud rate* sebesar:

$$BAUD = \frac{f_{osc}}{16(UBRR+1)} = \frac{4000000}{16(25+1)} = 9615.384615 \approx 9615\,\text{bps} \quad \text{.....(2.2)}$$

Dari persamaan 2.2 didapat nilai *baud rate* menjadi 9615 bps, berbeda dari *baud rate* yang diharapkan sebesar 9600 bps. Perbedaan hasil ini menyebabkan terjadinya nilai kesalahan (*error*) yang bisa dihitung dengan rumus:

$$Error[\%] = \left(\frac{BaudRate_{nilaibilanganbulat}}{BaudRate} - 1 \right) \cdot 100\% \quad \text{.....(2.3)}$$

sehingga dengan menggunakan persamaan 2.3, dapat dihitung *error* untuk frekuensi *clock* 4 MHz dengan *baud rate* 9600 bps yang menggunakan nilai UBRR = 25, yaitu:

$$Error[\%] = \left(\frac{9615}{9600} - 1 \right) \cdot 100\% = (1.00156 - 1) \cdot 100\% = 0.156\% \approx 0.2\% \quad \text{.....(2.4)}$$

Error sebesar 0.2% dapat ditoleransi karena nilainya yang masih sangat kecil. Namun keadaan akan berbeda jika *baud rate* yang diharapkan lebih besar dari 9600 bps. Untuk memperkecil nilai *error*, akan dipilih nilai frekuensi *clock* yang dapat menghasilkan nilai *error* paling kecil. Tabel 2.6 menunjukkan besarnya *error* dari perhitungan dengan rumus 2.3 dengan menggunakan frekuensi *clock* yang berbeda-beda. Sumber frekuensi *clock* diperoleh dari kristal yang banyak ditemukan di pasaran, yaitu kristal 8 MHz, 11.0592 MHz, dan 16 Mhz.

Tabel 2.6. Hubungan *Error*, UBRR, dan *Baudrate* pada Mode Asinkron Normal

Baud Rate	**f_{osc} = 8 MHz**		**f_{osc} = 11.0592 MHz**		**f_{osc} = 16 MHz**	
(bps)	**UBRR**	***Error***	**UBRR**	***Error***	**UBRR**	***Error***
2400	207	0.2%	287	0.0%	416	-0.1%
4800	103	0.2%	143	0.0%	207	0.2%
9600	51	0.2%	71	0.0%	103	0.2%
14.4k	34	-0.8%	47	0.0%	68	0.6%
19.2k	25	0.2%	35	0.0%	51	0.2%
28.8k	16	2.1%	23	0.0%	34	-0.8%
38.4k	12	0.2%	17	0.0%	25	0.2%
57.6k	8	-3.5%	11	0.0%	16	2.1%
76.8k	6	-7.0%	8	0.0%	12	0.2%
115.2k	3	8.5%	5	0.0%	8	-3.5%
230.4k	1	8.5%	2	0.0%	3	8.5%
250k	1	0.0%	2	-7.8%	3	0.0%

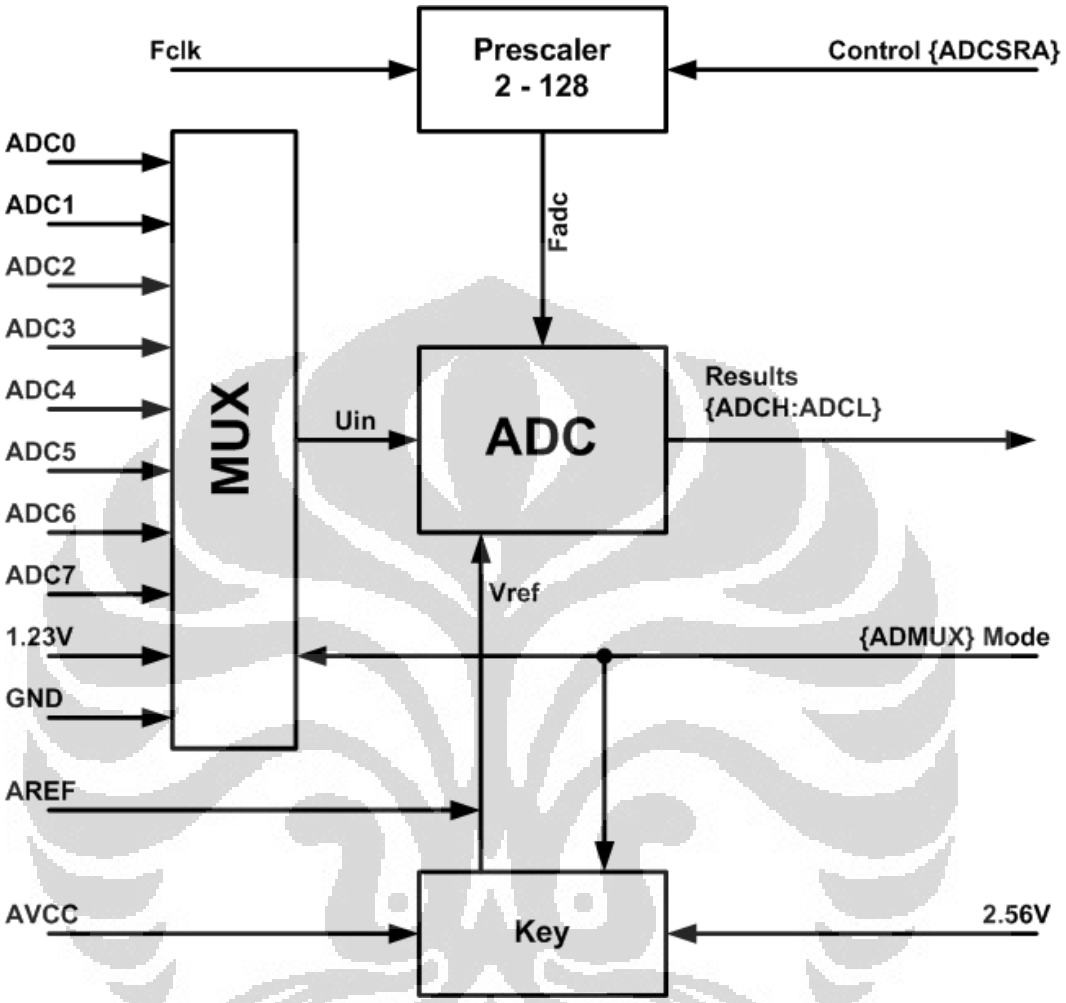
Sumber: *Datasheet* Atmel ATMega8535

Dari tabel 2.6 terlihat bahwa dengan menggunakan frekuensi *clock* sebesar 11.0592 MHz maka akan didapat *error* sebesar 0.0% untuk *baud rate* 2400 bps hingga 230.4kbps. Karena frekuensi *clock* sebesar 11.0592 MHz tidak memiliki *error* pada baud rate 2400 – 230.4 kbps, maka mikrokontroler ATMega8535 pada dalam rancang bangun skripsi ini akan menggunakan kristal sebesar 11.0592 MHz sebagai sumber frekuensi *clock*-nya.

2.1.6. *Analog to Digital Converter* (ADC) pada ATMega8535[10]

Analog to Digital Converter (ADC) adalah sebuah piranti yang dirancang untuk mengubah data analog menjadi data digital. Contoh penerapannya seperti pada *multimeter* digital. Masukan yang berupa data analog (tegangan atau arus) diubah menjadi data digital untuk dapat ditampilkan pada *display*.

Mikrokontroler ATMega8535 telah memiliki rangkaian ADC di dalamnya. Gambar 2.9 merupakan blok diagram ADC pada ATMega8535.



Gambar 2.9. Blok Diagram ADC ATMega8535

Terdapat hanya satu buah ADC sedangkan ada delapan saluran masukkannya (ADC0 – ADC7). Maka, sinyal input dari pin masukan ADC (ADC0 – ADC7) akan dipilih oleh multiplexer (register ADMUX) secara bergantian. ADC memiliki rangkaian untuk mengambil *sample* dan *hold* (menahan) tegangan input ADC sehingga dalam keadaan konstan selama proses konversi. ADC memiliki catu daya yang terpisah, yaitu pin AVCC dan AGND dan tegangan AVCC tidak boleh berbeda ± 0.3V dari VCC.

Operasi ADC membutuhkan tegangan referensi Vref dan *clock* Fadc (register ADCSRA). Tegangan referensi eksternal pada pin AREF tidak boleh melebihi AVCC. Tegangan referensi eksternal dapat di-*decouple* pada pin AREF dengan kapasitor untuk mengurangi *noise*. Selain menggunakan tegangan referensi eksternal, ADC ATMega8535 telah menyediakan tegangan referensi internal sebesar 2.56V dan agar tegangan referensi internal tetap stabil, pin AREF diberi kapasitor eksternal.

ADC mengonversi tegangan input analog menjadi bilangan digital selebar 10 bit. GND (0 Volt) adalah nilai minimum yang mewakili ADC dan nilai maksimum ADC diwakili oleh tegangan pada pin AREF - 1. Sinyal input ADC tidak boleh melebihi tegangan referensi dan setelah proses konversi, hasilnya bisa ditemukan pada register pasangan ADCH:ADCL. Untuk hasil ADC dengan resolusi 10 bit digunakan rumus pada persamaan 2.5 dan untuk resolusi 8 bit digunakan rumus pada persamaan 2.6.

$$ADC = \left(\frac{V_{IN} \cdot 1024}{V_{REF}} \right) \quad (2.5)$$

$$ADC = \left(\frac{V_{IN} \cdot 256}{V_{REF}} \right) \quad (2.6)$$

Misalnya input pada pin ADC dengan resolusi 8 bit adalah 2.5 Volt dan tegangan referensi yang digunakan adalah tegangan referensi internal sebesar 2.56 Volt, maka nilai digitalnya adalah:

$$ADC = \left(\frac{V_{IN} \cdot 256}{V_{REF}} \right) = \left(\frac{2500mV \cdot 256}{2560mV} \right) = 250 = 0xFA \quad (2.7)$$

ADCH akan bernilai F_{16} (15_{10}) dan ADCL akan bernilai A_{16} (10_{10}).

2.2. Komunikasi Port Serial Komputer[11]

Komunikasi serial adalah pengiriman data secara seri (data dikirim satu per satu secara berurutan). Port serial lebih sulit ditangani karena peralatan yang dihubungkan ke port ini harus berkomunikasi menggunakan transmisi serial sedangkan data di komputer diolah secara paralel. Oleh karena itu data dari/ke port serial harus dikonversikan ke/dari bentuk parallel untuk bisa digunakan. Jika menggunakan perangkat keras, hal ini bisa dilakukan oleh *Universal Asyncronous Receiver Transmitter* (UART).

Kelebihan komunikasi serial adalah jangkauan panjang kabel yang lebih jauh dibandingkan komunikasi paralel karena port serial mengirimkan logika '1' dengan kisaran tegangan -3 volt hingga -25 volt dan logika '0' sebagai +3 volt hingga +25 volt sehingga kehilangan daya karena panjangnya kabel bukan masalah utama.

Perangkat keras pada komunikasi serial dibagi menjadi dua kelompok, yaitu *Data Communication Equipment* (DCE) dan *Data Terminal Equipment* (DTE). Contoh DCE adalah modem, scanner, atau peralatan lainnya, sedangkan contoh DTE adalah terminal di komputer. Spesifikasi elektris port serial merujuk pada *Electronic Industry Association* (EIA) yaitu:

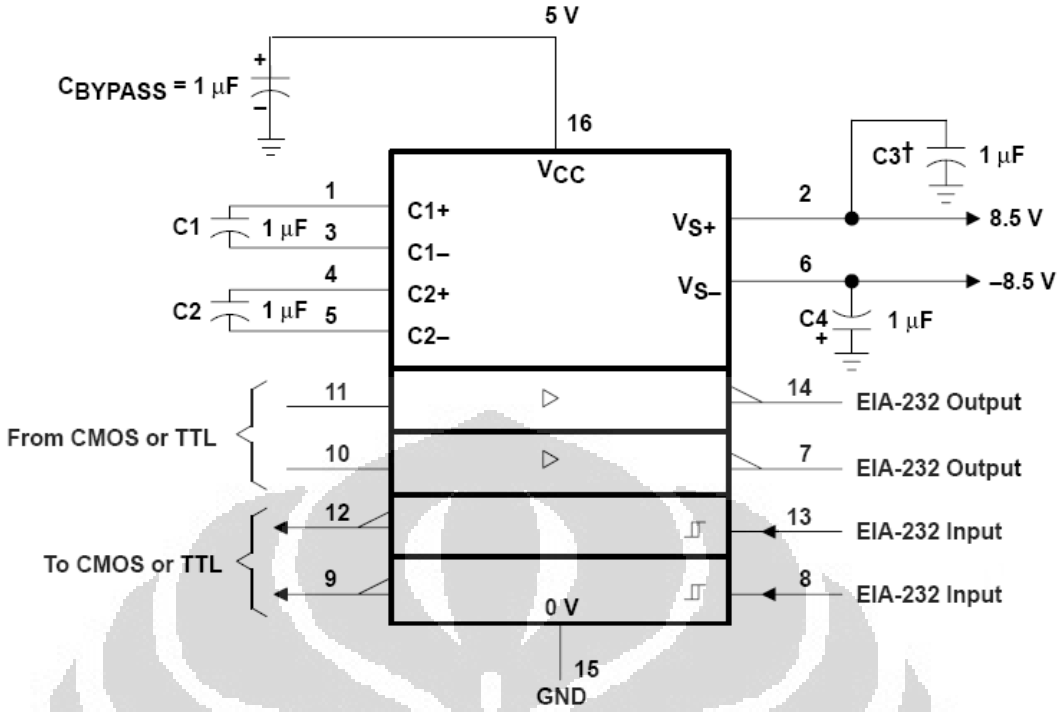
- "Space" (logika '0') adalah tegangan +3 hingga +25 volt.
- "Mark" (logika '1') adalah tegangan antara -3 hingga -25 volt.
- Daerah antara +3 volt hingga -3 volt tidak didefinisikan atau tidak terpakai.
- Tegangan *open circuit* tidak boleh melebihi 25 volt.
- Arus hubungan singkat tidak boleh melebihi 500 mA.

Konektor port serial memiliki 9 pin, yang berjenis DB9 dengan masing-masing pin memiliki fungsi yang berbeda-beda seperti ditunjukkan pada tabel 2.7.

Tabel 2.7. Fungsi Pin Port Serial DB9

Pin	Fungsi
1	*Data Carrier Detect*
2	*Received Data*
3	*Transmitter Data*
4	*Data Terminal Ready*
5	*Sinyal Ground*
6	*Data Set Ready*
7	*Request to Send*
8	*Clear to Send*
9	*Ring Indicator*

Jika peralatan yang akan berkomunikasi dengan port serial komputer menggunakan logika TTL, sinyal dari port serial komputer harus dikonversi ke pulsa TTL. Sebaliknya, sinyal dari peralatan yang menggunakan logika TTL harus dikonversikan ke logika RS232 sebelum dimasukkan ke port serial komputer. Contoh penerapan hal ini adalah saat mikrokontroler akan dihubungkan dengan port serial komputer karena mikrokontroler berlogika TTL sedangkan serial port komputer berlogika RS232. Untuk melakukan konversi dapat digunakan IC MAX232 karena di dalamnya telah terdapat *Charge Pump* yang akan membangkitkan tegangan sebesar +8.5 volt dan -8.5 volt dari sumber tunggal tegangan +5 volt. Terdapat dua transmitter dan dua receiver di dalam IC MAX232 yang dapat digunakan secara bersamaan. Gambar 2.10 menunjukkan blok diagram IC MAX232.



Gambar 2.10. Blok Diagram MAX232

Dari gambar 2.10 terlihat IC MAX232 memiliki 16 pin di mana terdapat konfigurasi kapasitor C1 sampai C4 agar menghasilkan tegangan sebesar +8.5 volt dan -8.5 volt dan juga terdapat dua masukan TTL untuk dikeluarkan menjadi sinyal RS232 serta sebaliknya, terdapat dua masukan RS232 untuk dikeluarkan menjadi sinyal TTL.

2.3. *Liquid Crystal Display* (LCD)

LCD tidak hanya mampu untuk menampilkan angka-angka, tetapi juga huruf-huruf, kata-kata, dan semua sarana simbol, serta lebih bagus dan serbaguna daripada penampil-penampil yang menggunakan *seven segment* LED (*Light Emiting Diode*). Modul LCD memiliki basis *interface* yang sesuai dengan mikrokontroler AVR pada khususnya dan semua mikrokontroler pada umumnya.

Akses pin yang tersedia mempunyai delapan jalur hubungan data, tiga jalur hubungan kontrol, tiga jalur catu daya dan pada modul LCD dengan fasilitas *back*

lighting terdapat dua jalur catu untuk mengatur *back lighting* sehingga pencahayaan LCD dapat diubah sesuai kebutuhan.

Ketika *power* pada LCD dinyalakan, *display* menampilkan sederet persegi gelap. Sel-sel karakter ini sebenarnya merupakan bagian yang mati. Modul *display* me-*reset* sendiri pada bagian awal ketika power dinyalakan, di mana layar akan menjadi kosong sehingga karakter-karakter tidak terlihat. Dengan demikian perlu memberikan perintah untuk menampilkan karakater pada LCD.

Modul LCD yang digunakan pada rancang bangun skripsi ini memiliki ukuran 16 karakter x 2 baris dengan fasilitas *back lighting*, memiliki 16 pin yang terdiri dari 8 jalur data, 3 jalur kontrol dan jalur-jalur catu daya. Tabel 2.8 menjelaskan fungsi dari masing-masing pin pada LCD 2x16 karakter.

Tabel 2.8. Fungsi Pin LCD 2x16 Karakter

Pin	Simbol	Fungsi
1	Vss	Power Supply (GND)
2	Vdd/Vcc	Power Supply (+5V)
3	Vee/Vo	Contrast Adjust
4	RS	0 = Instruction Input, 1 = Data Input
5	R/W	0 = Write to LCD Module, 1 = Read from LCD Module
6	E	Enable Signal
7	DB0	Data Pin 0
8	DB1	Data Pin 1
9	DB2	Data Pin 2
10	DB3	Data Pin 3
11	DB4	Data Pin 4
12	DB5	Data Pin 5
13	DB6	Data Pin 6
14	DB7	Data Pin 7
15	VB+	Back Light (+5V)
16	VB-	Back Light (GND)

Pada umumnya, modul LCD sudah dilengkapi perangkat pengontrol sendiri yang didesain untuk mengendalikan karakter yang akan ditampilkan oleh LCD. Semua bentuk *display* dari karakter, disimpan didalam memori. Baik memori permanen maupun memori sementara. Terdapat 3 jenis memori yang digunakan untuk mengatur tampilan LCD yaitu:[13]

1. DDRAM (*Display Data Random Access Memory*), adalah memori RAM sekaligus mewakili tampilan karakter LCD. DDRAM menyimpan data *display* yang tiap karakternya berukuran 8 bit dan mampu menyimpan hingga 80 karakter (80 x 8 bit). Gambar 2.11 menunjukkan alamat untuk mengakses DDRAM pada LCD dengan ukuran 16 karakter x 2 baris.

Display position	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
DDRAM address	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
	40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F

Gambar 2.11. Alamat DDRAM LCD Dengan Ukuran 2x16 Karakter

Untuk mengakses masing-masing lokasi DDRAM di LCD, nilai DDRAM *address* harus ditambah dengan 0x80. Contohnya, untuk menulis huruf ‘A’ di baris pertama kolom 9, maka alamatnya menjadi 0x88 (0x80 ditambah dengan 08).

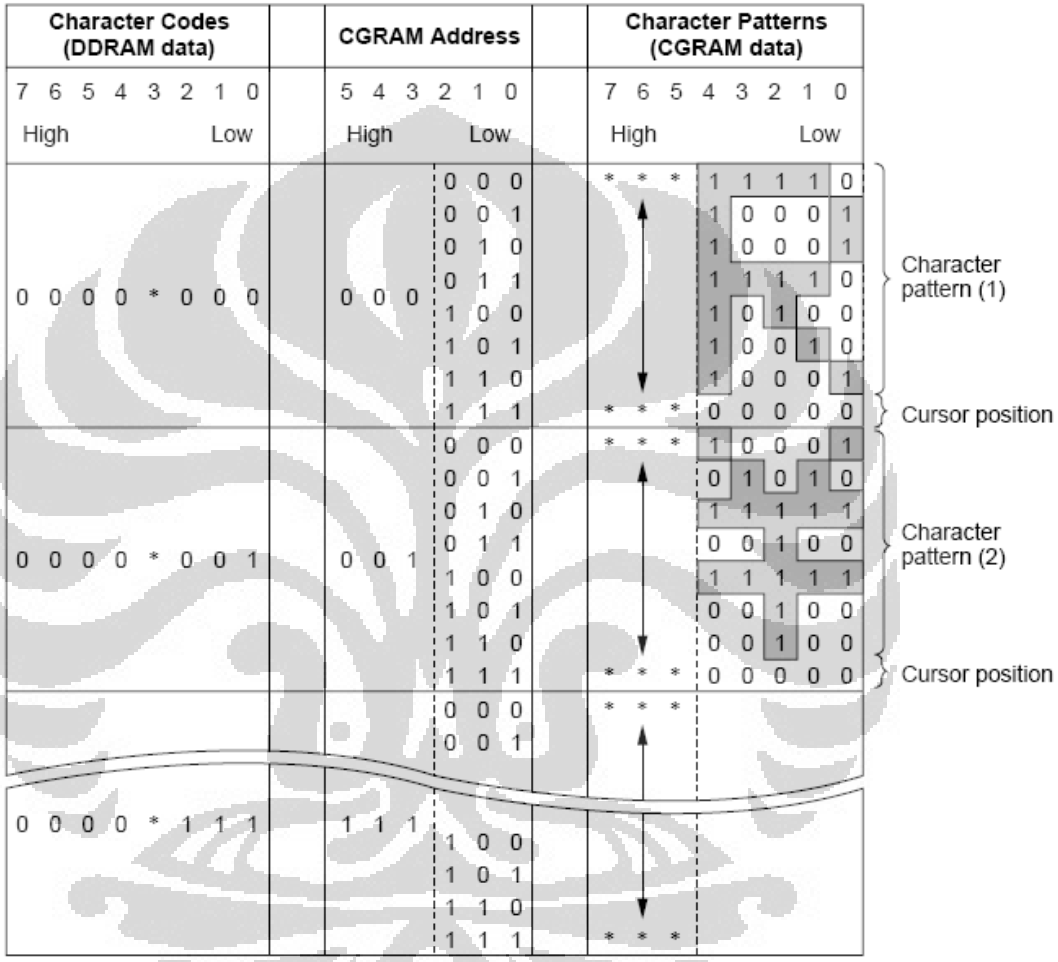
2. CGROM (*Character Generator Read Only Memory*), adalah memori ROM yang telah diisi pola karakter kode ASCII dari pabriknya. Tabel 2.9 merupakan kode ASCII untuk tiap-tiap karakter yang dapat ditampilkan oleh LCD dengan ukuran 16 karakter x 2 baris.

Tabel 2.9. Kode ASCII LCD Hitachi HD44780U

Sumber: *Datasheet* Hitachi HD44780U

Jika akan menampilkan huruf ‘M’ maka kode yang harus diberikan adalah 0x4D karena huruf M ada di kolom ke-5 dengan kode bit 0100 (4) dan baris ke-14 dengan kode bit1101 (D).

3. CGRAM (Character *Generator Random Access Memory*), adalah memori RAM yang dapat digunakan untuk membuat pola karakter sendiri yang tidak terdapat pada CGROM dan akan terhapus jika tegangan catu mati. Ukuran karakter yang digunakan adalah 5 x 8 titik. Gambar 2.12 menunjukkan contoh penggunaan CGRAM.



Gambar 2.12. Penggunaan CGRAM

Pola karakter yang terbentuk akan menghasil CGRAM data yang ditempatkan pada memori CGRAM yang dapat diakses alamatnya pada CGRAM *address*. Untuk menampilkan karakter tesebut, alamat dari CGRAM *address* yang harus diakses.

2.4. Sensor Suhu (*Temperature Sensor*)

Untuk mengukur suhu suatu benda atau zat diperlukan elemen perasa yang akan mendeteksi perubahan suhu. Elemen perasa ini dikenal sebagai sensor suhu. Terdapat empat jenis sensor suhu yaitu *Thermocouple*, RTD, *Thermistor*, dan IC Sensor. Tabel 2.10 memperlihatkan kelebihan dan kekurangan masing-masing sensor suhu tersebut.

Tabel 2.10. Jenis-jenis Sensor Suhu[14]

Jenis	Kelebihan	Kekurangan
Thermocouple	▪ Tidak membutuhkan catu daya dari luar ▪ Penggunaannya mudah ▪ Harga murah ▪ Banyak jenisnya ▪ Jangkauan suhu yang luas	▪ Hasilnya tidak linier ▪ Tegangan rendah ▪ Butuh tegangan referensi ▪ Kurang stabil ▪ Kurang sensitif terhadap perubahan suhu
RTD	▪ Paling stabil ▪ Hasilnya paling akurat ▪ Hasilnya lebih linier daripada *thermocouple*	▪ Harga mahal ▪ Butuh sumber arus ▪ Perubahan resistansinya kecil ▪ Menghasilkan panas
Thermistor	▪ Hasil keluarannya tinggi ▪ Waktu pengukurannya cepat ▪ Pengukurannya hanya menggunakan 2 kawat	▪ Hasilnya tidak linier ▪ Jangkauan suhunya terbatas ▪ Mudah rusak ▪ Butuh sumber arus ▪ Menghasilkan panas
IC Sensor	▪ Hasilnya paling linier ▪ Keluarannya paling tinggi ▪ Harga murah	▪ Suhu yang dapat diukur kurang dari 200 °C ▪ Butuh catu daya dari luar ▪ Waktu pengukurannya lambat ▪ Menghasilkan panas ▪ Konfigurasinya terbatas

Dari tabel 2.10 dipilih IC Sensor untuk pengukuran suhu pada rancang bangun skripsi ini karena ditinjau dari hasil yang paling linier agar mudah dilakukan kalibrasi. Dan IC Sensor dalam skala derajat celcius (°C) yang banyak ditemukan dipasaran adalah jenis LM35.

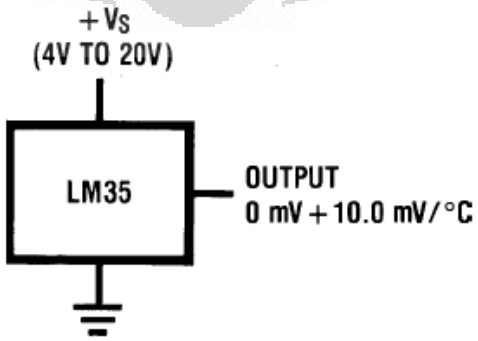
Sensor LM35 merupakan sensor suhu yang mempunyai keluaran berupa tegangan yang berubah secara linier dan proporsional terhadap perubahan suhu dalam skala derajat celcius yaitu sebesar:

$$V_{out} = \frac{10mV}{°C} \quad \text{.....(2.8)}$$

Pada suhu 0°C output LM35 adalah 0 volt dan setiap perubahan suhu 1°C, keluarannya akan bertambah sebesar 10 mV. Sensor LM35 tidak memerlukan kalibrasi eksternal untuk menghasilkan akurasi ±0.25°C. Dengan menggunakan persamaan 2.8, misalnya suhu yang terukur adalah 40°C maka tegangan keluaran dari sensor LM35 adalah:

$$V_{out} = \frac{10mV}{°C} \times 40°C = 400mV \quad \text{.....(2.9)}$$

Agar dapat bekerja, sensor LM35 harus mendapatkan tegangan catu daya dari luar sebesar 4 volt sampai 20 volt. IC sensor LM35 hanya memiliki 3 pin, satu pin untuk tegangan catu daya positif (4 volt – 20 volt), satu pin untuk tegangan *ground*, dan lainnya adalah pin keluaran yang bisa langung diaplikasikan dengan menggunakan persamaan 2.8 di atas. Gambar 2.13 menunjukkan blok rangkaian IC sensor LM35.

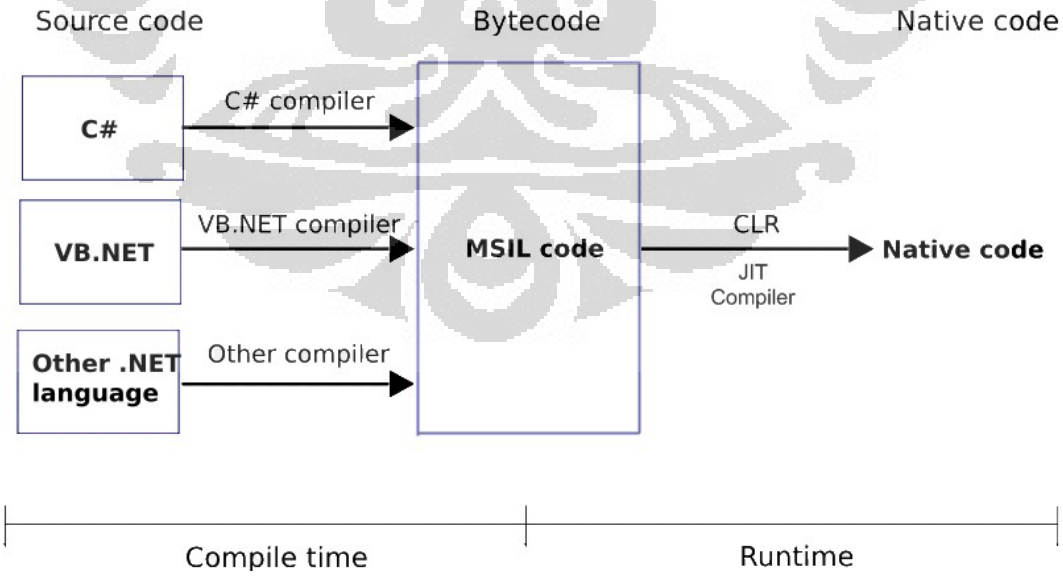


Gambar 2.13. Blok Rangkaian IC Sensor LM35

2.5. Pemrograman *Web* dengan ASP.NET[15]

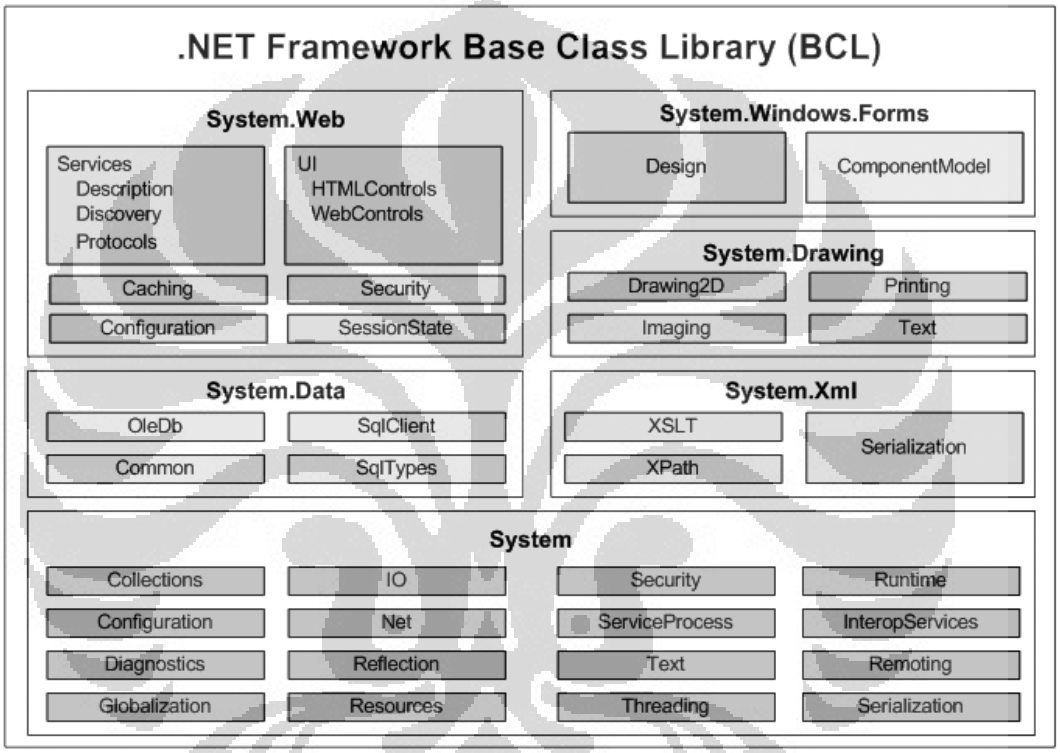
ASP.NET merupakan teknologi Microsoft yang dikhususkan untuk pengembangan aplikasi berbasis *web* dinamis dengan *platform* .NET (dibaca: dotnet) *framework*. Sementara .NET *framework* adalah satu set kumpulan teknologi dari Microsoft yang ditujukan untuk membantu pengembang untuk mengembangkan aplikasi secara aman, mudah, efisien dan produktif.

.NET *framework* terdiri dari dua komponen utama, yaitu *Common Language Runtime* (CLR) dan *Base Class Library* (BCL). CLR merupakan pondasi utama dari .NET *framework* yang antara lain bertanggung jawab untuk melakukan manajemen memori, melakukan eksekusi kode, melakukan verifikasi kode terhadap keamanan kode, menentukan hak akses dari kode, dan melakukan kompilasi kode. Dalam mengompilasi kode, CLR mengubah kode-kode dalam bahasa ASP menjadi bahasa *assembly* MSIL (Microsoft *Intermediate Language*). Proses kompilasi ini dilakukan oleh komponen yang bernama *Just in Time* (JIT). Gambar 2.14 menjelaskan cara kerja CLR mengompilasi kode menjadi bahasa *assembly*.



Gambar 2.14. Blok Diagram MSIL

BCL atau sering disebut .NET *Framework Class Library* adalah koleksi dari *reusable types* yang sangat banyak dan terintegrasi secara melekat dengan CLR. Kumpulan *Class Library* ini sangat berguna untuk pengembangan aplikasi karena pembuat kode program tidak perlu membuat semuanya dari awal, misalnya *class* untuk membuat aplikasi berbasis Windows, *class* untuk membuat objek-objek koleksi, *class* untuk koneksi dengan database, dan masih banyak lagi lainnya. Gambar 2.15 merupakan kumpulan dari *class* yang ada dalam BCL.



Gambar 2.15. *Base Class Library*

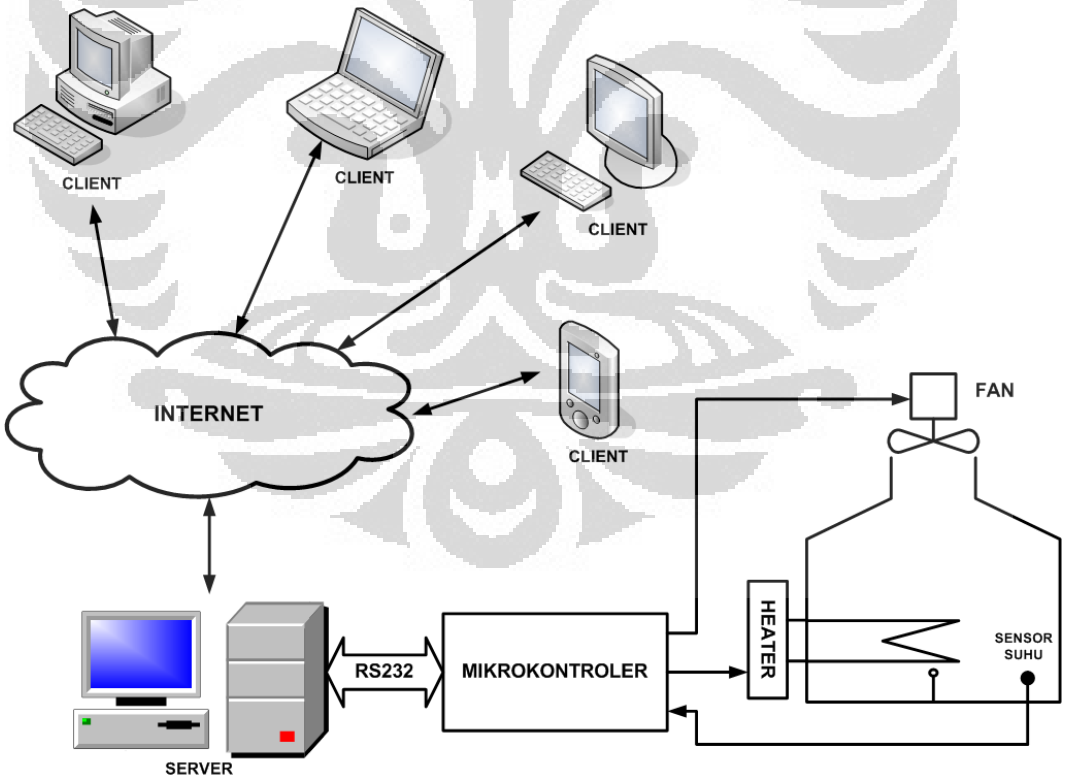
Untuk menjalankan aplikasi ASP.NET dibutuhkan program yang bernama *web server*. *Web server* adalah *server* internet yang mampu melayani koneksi transfer data dalam protocol HTTP (*Hypertext Transfer Protocol*). Web server berfungsi sebagai tempat aplikasi atau *software* beroperasi dalam mendistribusikan halaman *web* ke *user*.

BAB 3

PERANCANGAN

3.1. Perancangan Sistem

Sistem ini dirancang untuk mengendalikan masukan (*input*) dan memantau keluaran (*output*) sensor suhu yang diletakkan di dalam sebuah tangki dari jarak jauh dengan menggunakan teknologi internet. Sistem ini akan menjaga suhu di dalam tangki sesuai dengan yang diinginkan oleh para pengguna (*client*) dengan cara mengatur kondisi nyala atau mati (*on/off*) pada pemanas dan pendingin dan dapat dikendalikan oleh banyak *client* selama terdapat koneksi internet. Gambar 3.1 menjelaskan gambaran sistem secara keseluruhan.

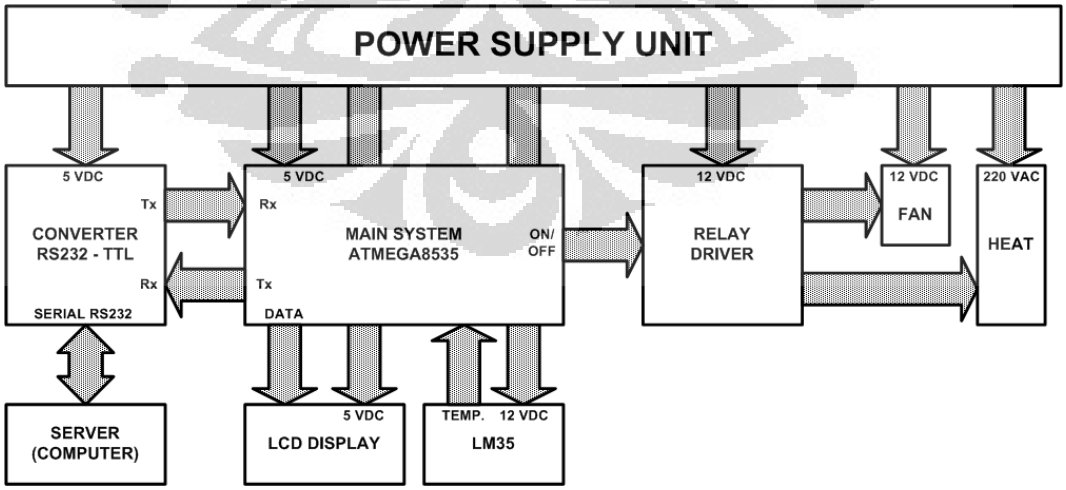


Gambar 3.1. Blok Diagram Sistem

Client memberikan nilai pengaturan (*setting point*) berupa besarnya suhu yang diinginkan (dalam satuan derajat celcius) ke *server* dan kemudian *server* akan memberikan nilai tersebut ke sistem mikrokontroler ATMega8535. Mikrokontroler akan mengendalikan pemanas (*heater*) dan pendingin (*fan*) pada tangki agar kondisi suhu di dalam tangki sesuai dengan keinginan *client* dan tetap terjaga suhunya hingga *client* mengubah *setting point*. Selain itu, mikrokontroler mengirimkan informasi ke *server* tentang kondisi besarnya suhu yang ada pada tangki untuk kemudian oleh *server* dikirimkan ke *client* agar suhu di dalam tangki dapat dipantau secara *real-time* oleh *client*.

3.2. Perancangan Perangkat Keras

Perangkat keras yang akan dirancang untuk skripsi ini terdiri dari lima subsistem, yaitu *power supply* (catu daya), sistem utama mikrokontroler ATMega8535, *RS232 to TTL converter* (pengubah sinyal RS232 dari/ke TTL), LCD *Display* (peraga LCD), dan *relay driver* untuk fungsi nyala dan mati pada pemanas dan pendingin melalui *relay*. Gambar 3.2 menunjukan hubungan antar subsistem yang menciptakan sebuah sistem yang utuh.

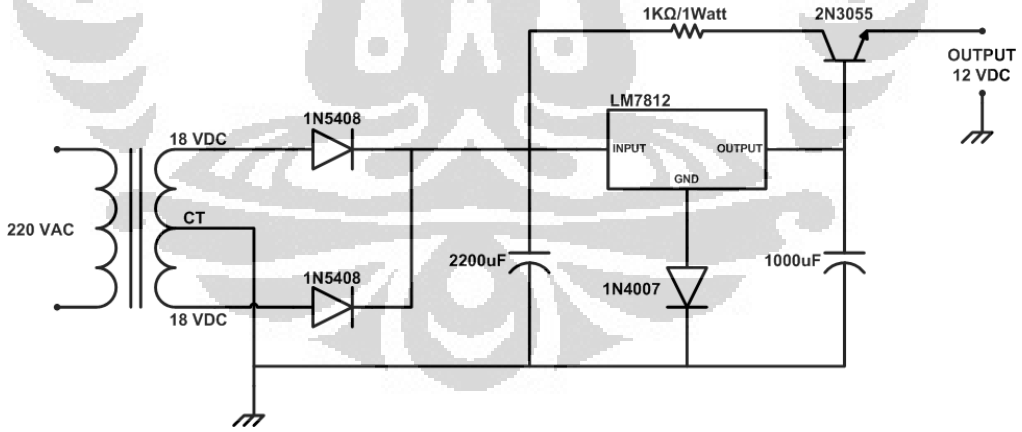


Gambar 3.2. Blok diagram perangkat keras sistem

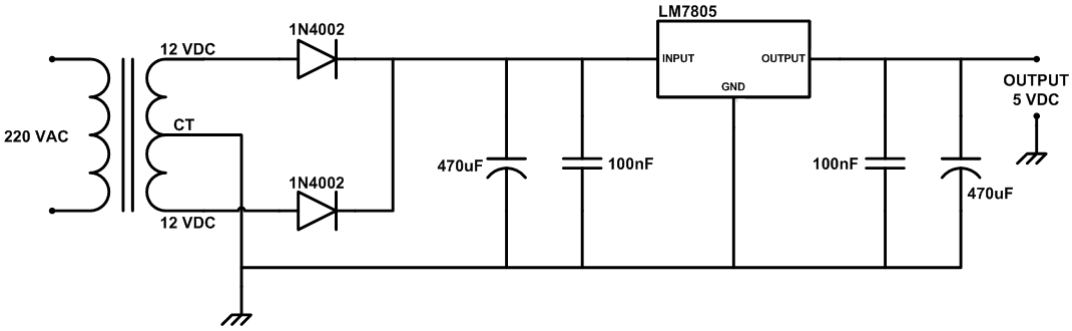
Perangkat keras ini dikendalikan oleh mikrokontroler ATMega8535. Pengguna (*client*) memberikan nilai *setting point* melalui komputer (*server*) yang akan dikirimkan ke mikrokontroler melalui *RS232 to TTL converter*. Mikorokontroler mengolah nilai *setting point* dan membandingkan dengan data suhu yang diperoleh dari LM35. Dari hasil perbandingan tersebut, mikrokontroler mengendalikan *relay driver* untuk menyalakan atau mematikan pemanas maupun pendingin agar diperoleh suhu yang diinginkan oleh *client*. Besarnya suhu yang terukur oleh LM35 juga akan ditampilkan pada LCD *display* dan akan dikirimkan ke *server* melalui *RS232 to TTL converter* agar dapat diketahui pula oleh *client*.

3.2.1. Catu Daya

Catu daya yang dirancang dalam skripsi ini mampu menghasilkan tegangan keluaran 12 Volt DC dan 5 Volt DC. IC Regulator LM7812 digunakan untuk menghasilkan tegangan 12 Volt DC dan IC Regulator LM7805 digunakan untuk menghasilkan tegangan 5 Volt DC. Rangkaian lengkapnya seperti terlihat pada gambar 3.3 dan gambar 3.4.

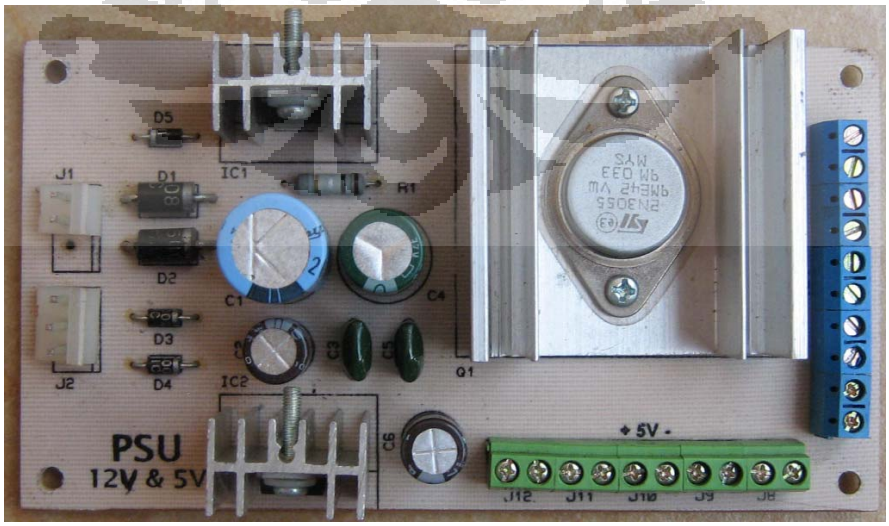


Gambar 3.3. Rangkaian Catu Daya 12 Volt DC

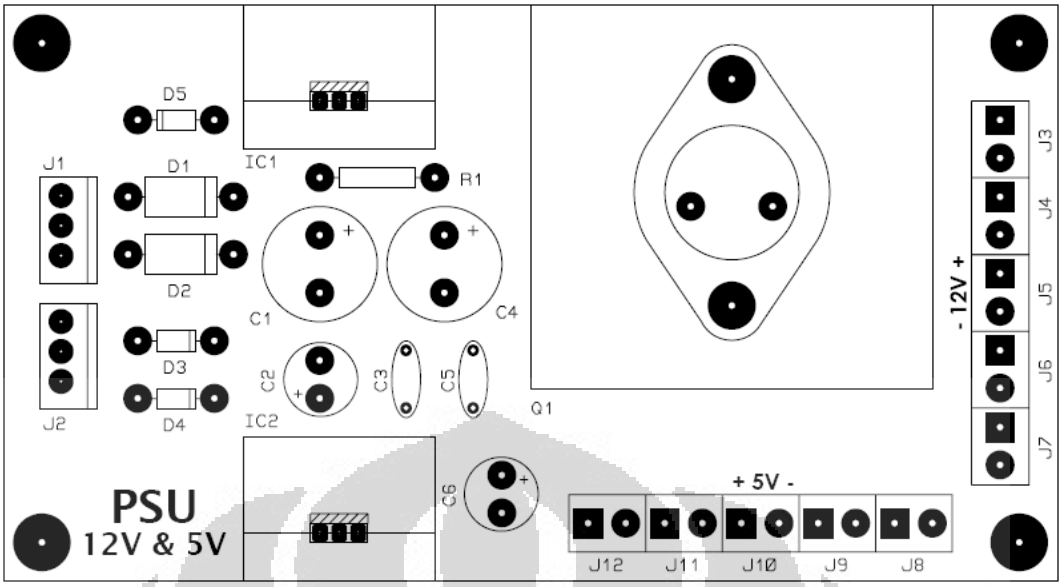


Gambar 3.4. Rangkaian Catu Daya 5 Volt DC

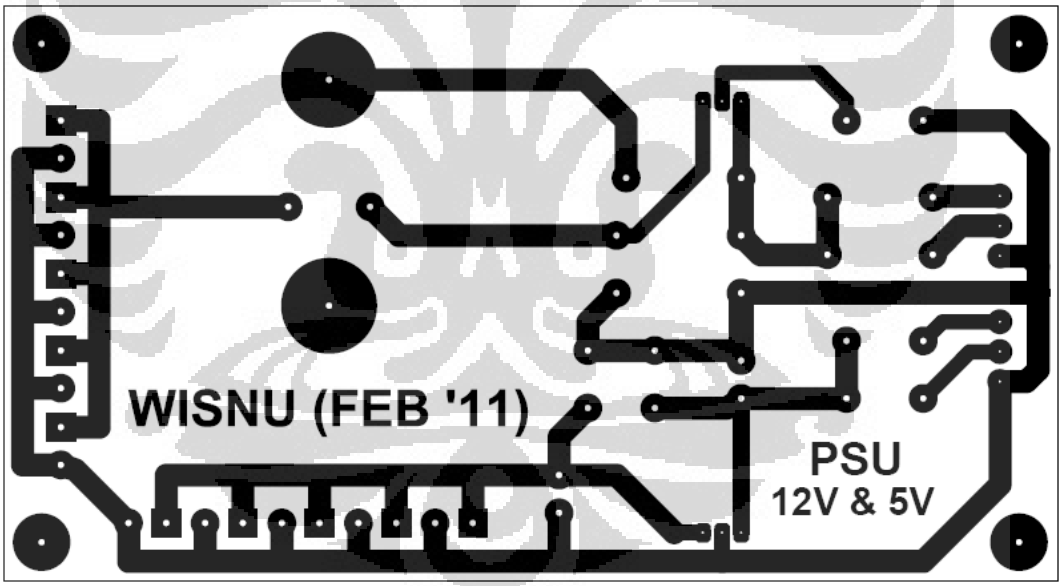
IC LM7812 dan LM7805 (dalam keadaan diberi *heat sink*) hanya mampu menghasilkan arus maksimal sebesar 1 Ampere. Pada gambar 3.2 terlihat bahwa tegangan 5 volt digunakan tiga subsistem, yaitu system utama mikrokontroler, pengubah sinyal RS232, dan peraga LCD. Ketiga subsistem tersebut menghasilkan daya listrik yang relatif kecil dan total arus yang dihasilkan oleh tegangan 5 volt tidak melebihi 1 Ampere, sehingga penggunaan *heat sink* untuk IC LM7805 sudah mencukupi. Lain halnya dengan tegangan 12 volt, yang digunakan oleh subsistem pengendali relay, IC LM35, dan dua buah kipas (*fan*). Kebutuhan akan arus jauh lebih besar dibandingkan tegangan 5 volt. Oleh sebab itu, ditambahkan penguat arus dengan menggunakan transistor 2N3005 yang mampu dialiri arus sampai 15A (dalam keadaan diberi *heat sink*).



Gambar 3.5. Modul Catu Daya



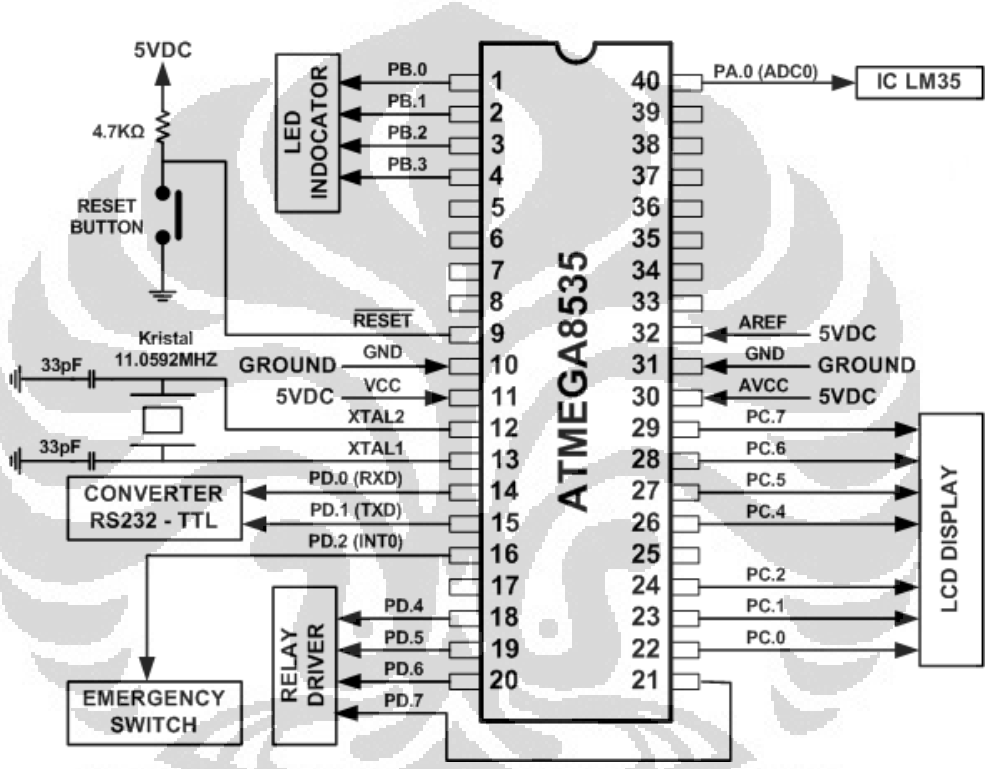
Gambar 3.6. Tata Letak Komponen Modul Catu Daya



Gambar 3.7. Jalur PCB Modul Catu Daya

3.2.2. Sistem Mikrokontroler ATMega8535

Mikrokontroler ini merupakan sistem utama yang mengendalikan semua perangkat yang terdapat pada seluruh sistem. Diprogram sesuai dengan alur progam menggunakan bahasa C dengan *compiler* Codevision AVR. Alur program dan pemrograman ATMega8535 akan dibahas lebih detail pada bagian perangkat lunak. Rangkaian sistem utama mikrokontroler ATMega8535 ditunjukan pada gambar 3.8.

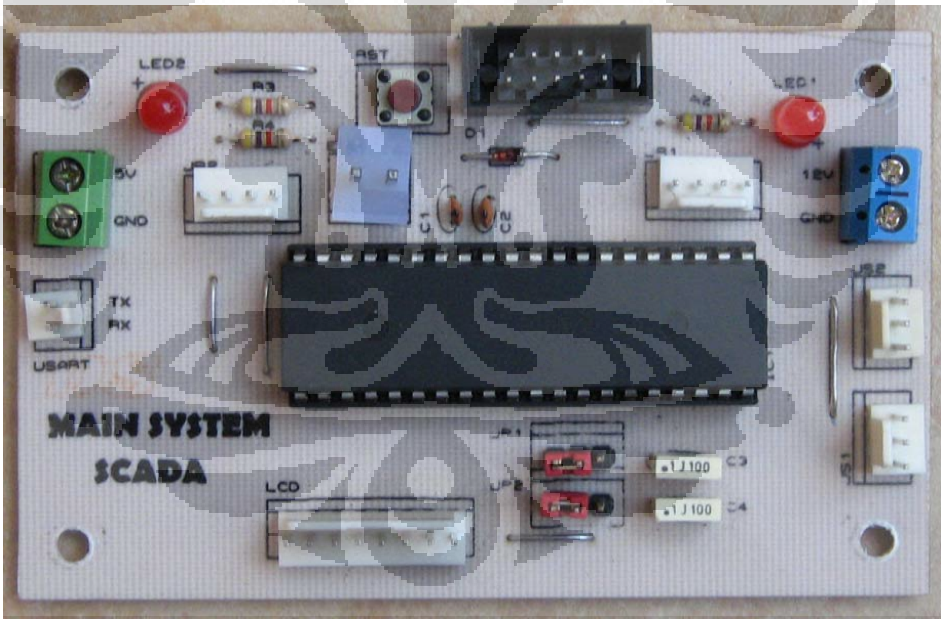


Gambar 3.8. Rangkaian Sistem Mikrokontroler ATMega8535

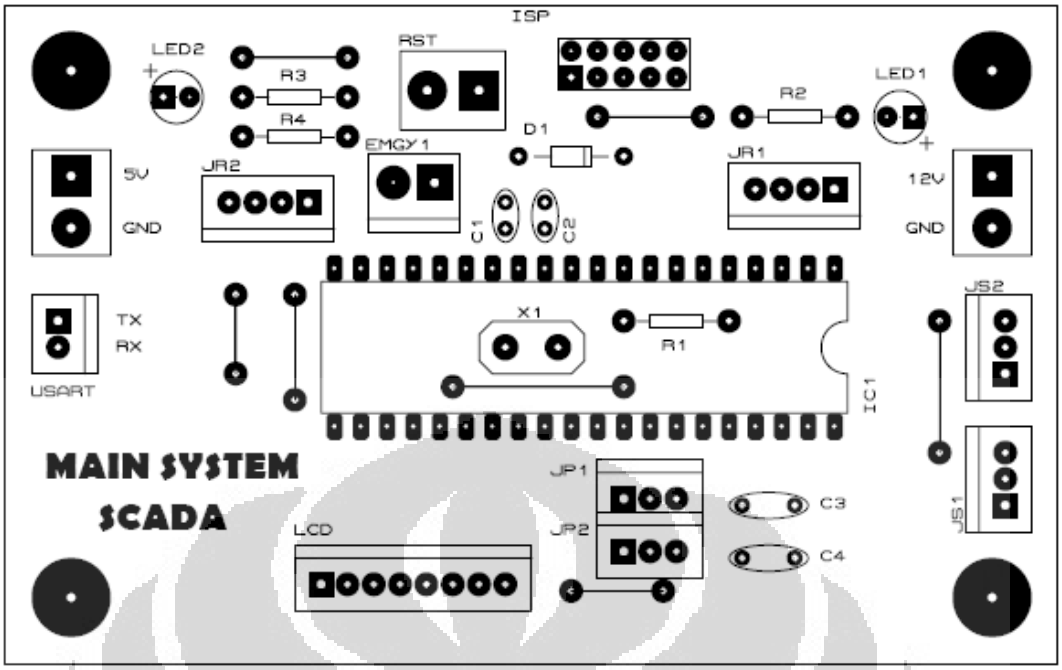
ATMega8535 pada rangkaian ini menggunakan *external oscillator* dari sebuah kristal dengan denyut (*clock*) sebesar 11,0592 MHz dan dua buah kapasitor 33 pF sesuai *datasheet*. Pin RESET (pin 9) dihubungkan ke tombol tekan untuk memudahkan jika terjadi kekacauan jalannya program pada mikrokontroler sehingga dengan menekan tombol tersebut maka fungsi RESET pada mikrokontroler akan aktif dan menyebabkan program berjalan dari awal.

ADC (*analog to digital converter*) yang sudah terdapat pada rangkaian internal IC, digunakan untuk membaca nilai keluaran dari IC LM35 (sensor suhu) sehingga tidak diperlukan lagi rangkaian penguat maupun pengkondisi sinyal untuk IC LM35. Pengaturan untuk untuk IC LM35 dilakukan sepenuhnya oleh program yang telah ditanam pada mikrokontroler.

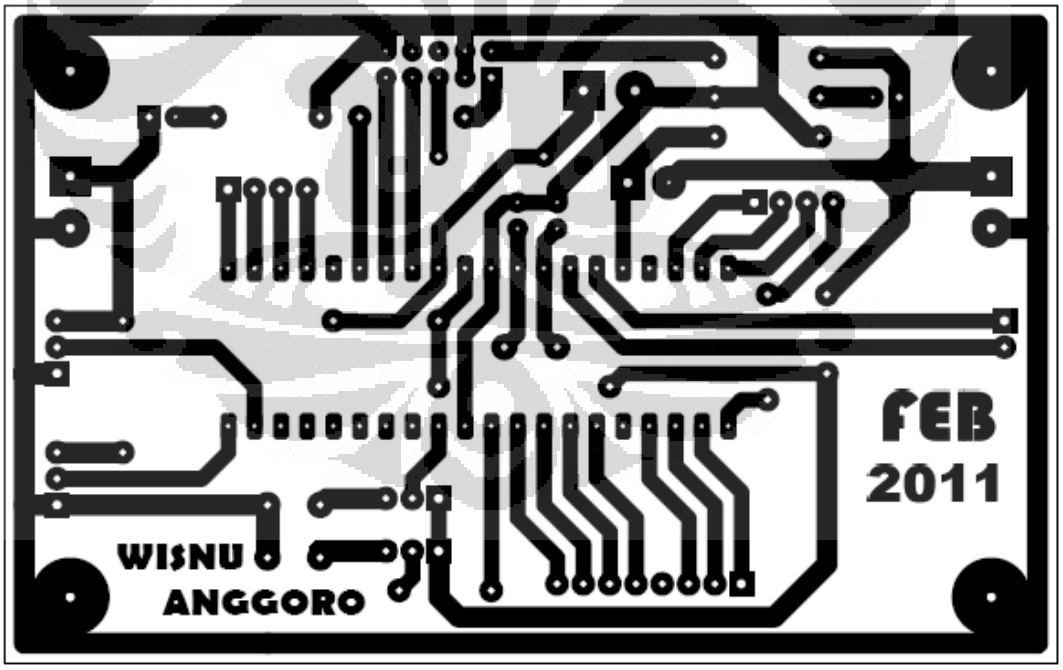
Untuk melakukan hubungan dengan komputer atau mikrokontroler lain, digunakan fungsi Rx (*receiver*/ penerima) dan Tx (*transmitter*/ pengirim) pada pin 14 dan pin 15. Saat hendak dihubungkan ke komputer, sinyal TTL (5 volt untuk logika "1" dan 0 volt untuk logika "0") tidak dapat langsung dihubungkan ke port serial pada komputer. Karena alasan itulah, digunakan pengubah sinyal RS232 ke/ dari TTL. Pin-pin berikutnya dihubungkan ke subsistem lainnya dan pemrograman disesuaikan dengan rangkaian di atas.



Gambar 3.9. Modul Mikrokontroler ATMega8535



Gambar 3.10. Tata Letak Komponen Modul Mikrokontroler



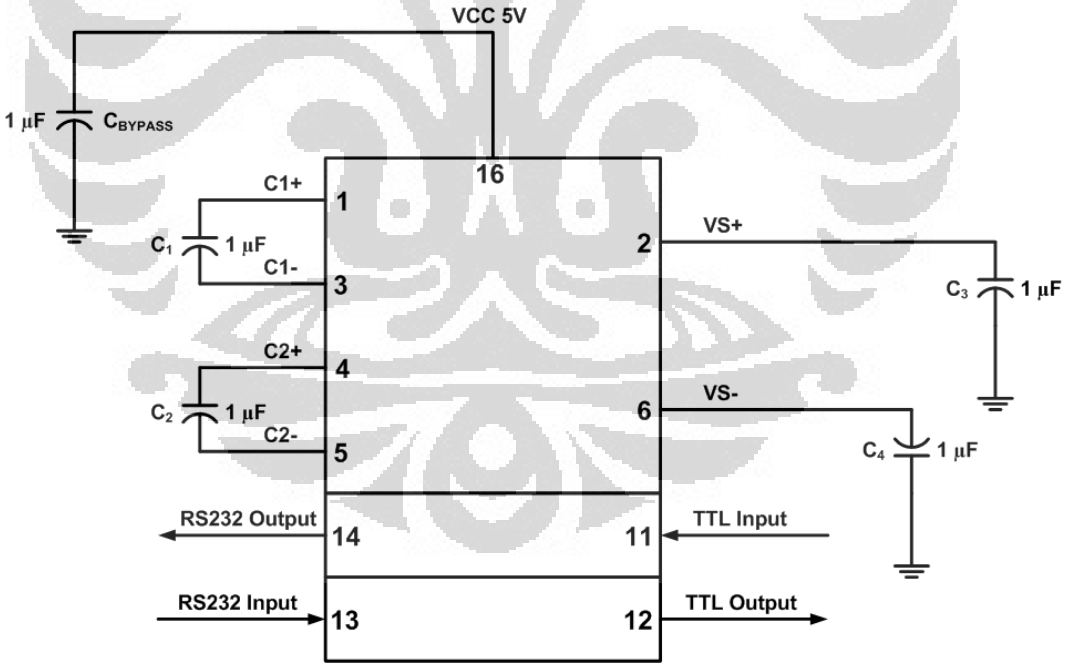
Gambar 3.11. Jalur PCB Modul Mikrokontroler

3.2.3. Pengubah sinyal RS232 ke TTL (*RS232 to TTL Converter*)

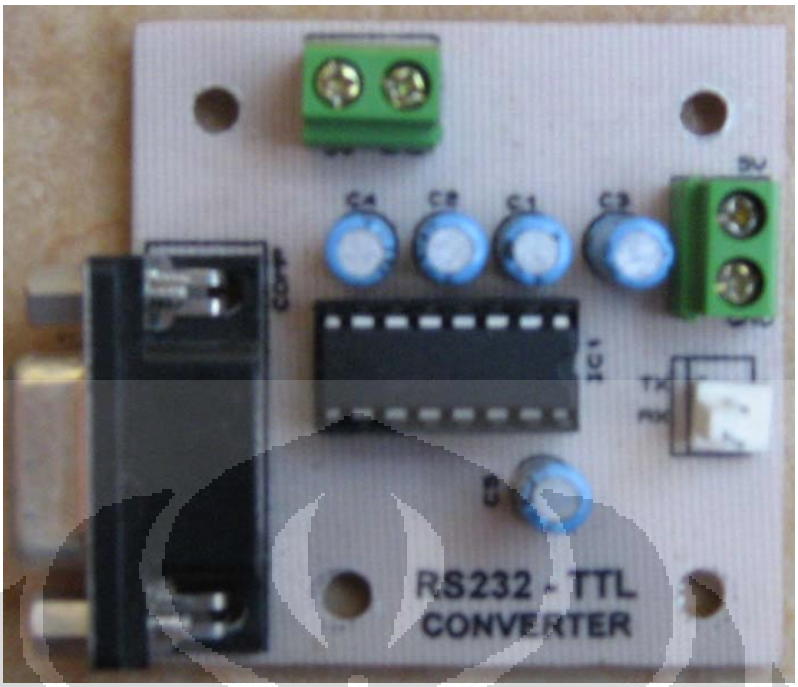
Fungsi dari rangkaian ini adalah mengubah sinyal dari mikrokontroler agar sesuai dengan sinyal pada *serial port* komputer. Perbedaan sinyal antara kedua sinyal tersebut adalah sebagai berikut:

- Tegangan logika "0" pada mikrokontroler (LOW) adalah 0 hingga +0,8 volt sedangkan pada *serial port* (SPACE) adalah +3 hingga +25 volt.
- Tegangan logika "1" pada mikrokontroler (HIGH) adalah +3,7 hingga +5 volt sedangkan pada *serial port* (MARK) adalah -3 hingga -25 volt.

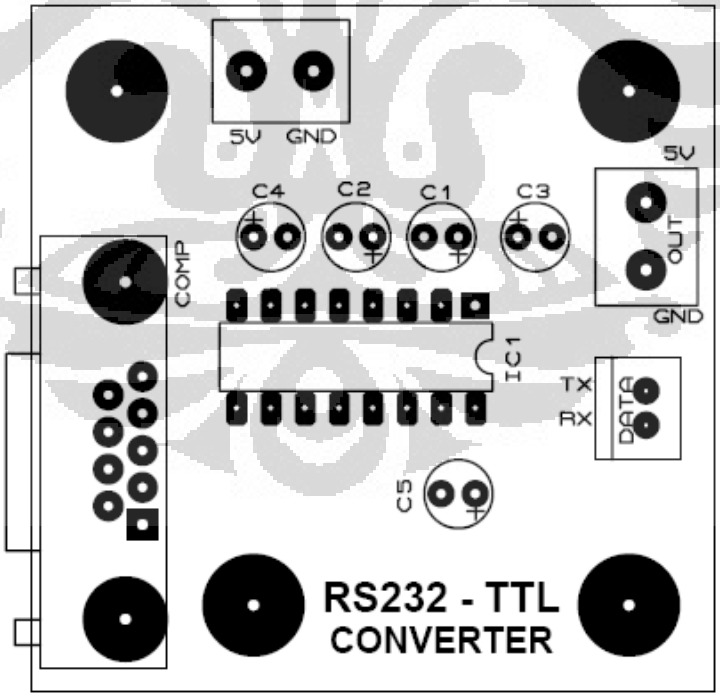
Karena perbedaan nilai tegangan itulah, maka diperlukan *RS232 to TTL converter*. Rangkaian lengkapnya seperti terlihat pada gambar 3.12. Rangkaian yang dibuat sesuai dengan *datasheet* yang dikeluarkan oleh pabrik pembuat IC MAX232.



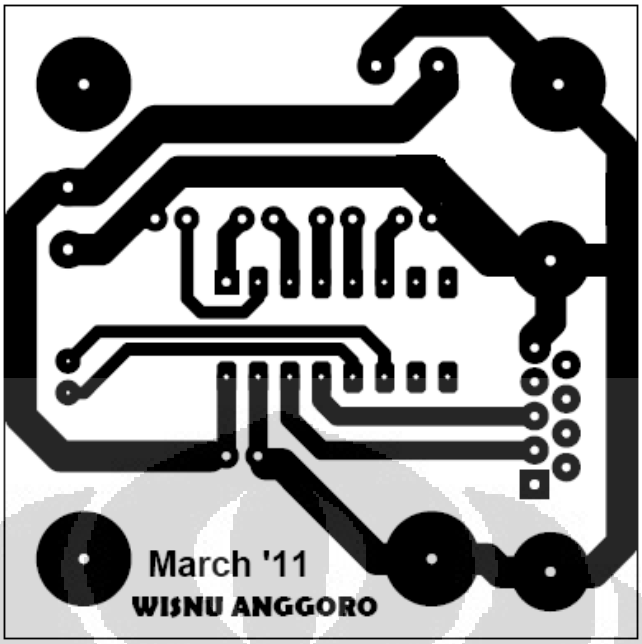
Gambar 3.12. Rangkaian pengubah sinyal RS232 dengan IC MAX232



Gambar 3.13. Modul Mikrokontroler RS232 *Converter*



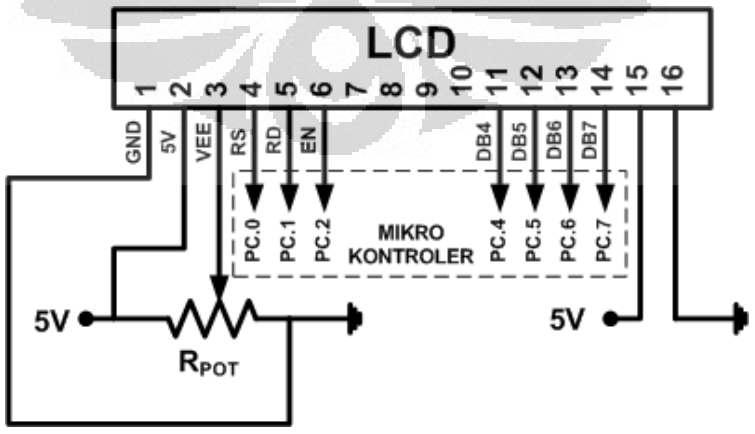
Gambar 3.14. Tata Letak Komponen Modul RS232 *Converter*



Gambar 3.15. Jalur PCB Modul RS232 *Converter*

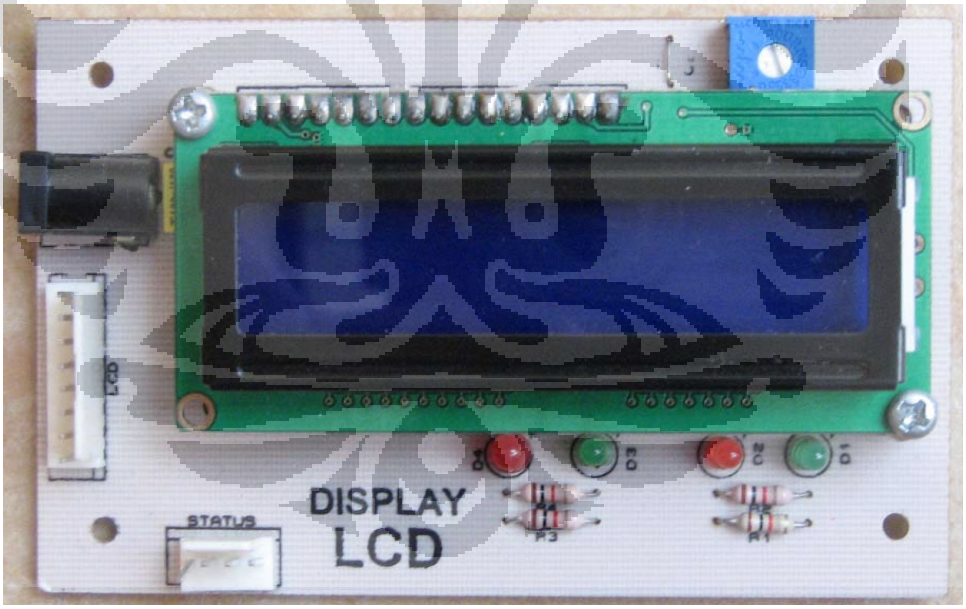
3.2.4. Peraga LCD (LCD *Display*)

Peraga LCD digunakan sebagai informasi bagi *user* yang sedang berada di *plant* agar dapat mengetahui kondisi yang sedang terjadi di dalam *plant.* Dalam kasus ini, data yang akan di tampilkan adalah nilai *setting point* (suhu yang diinginkan) dan nilai *present value* (suhu yang terjadi di dalam tangki). Rangkaian lengkap dari peraga LCD dapat dilihat pada gambar 3.16.

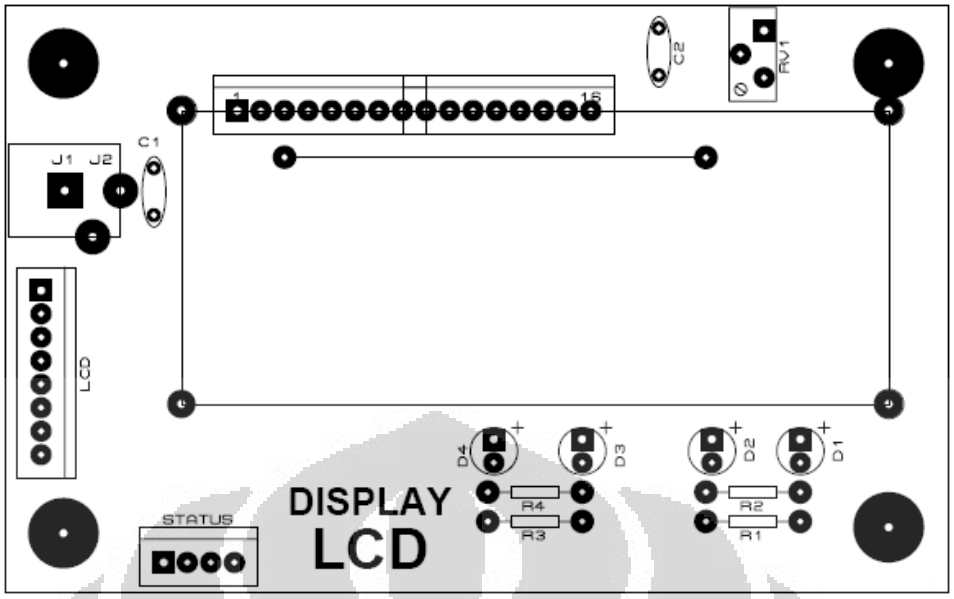


Gambar 3.16. Rangkaian Peraga LCD

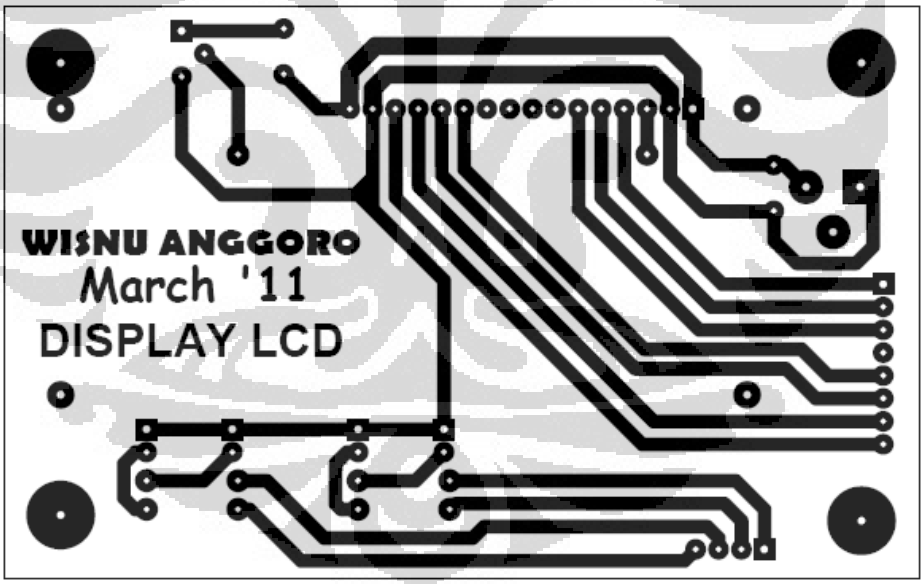
LCD yang digunakan memiliki spesifikasi 2x16, yang terdiri dari dua baris dan masing-masing barisnya dapat menampilkan 16 karakter, sehingga total karakter yang dapat ditampilkan sebanyak 32 karakter. Agar LCD ini dapat bekerja, diberi tegangan catu +5 volt untuk pin 2 dan 0 volt untuk pin 1 serta pin 16. Tegangan dari resistor variable diberikan ke pin 3 agar tingkat kecerahan tulisan pada LCD dapat diatur dengan cara mengatur nilai hambatan pada resistor *variable*. Pin 4 (*Register Select*) digunakan untuk memberi perintah ataupun membaca data pada LCD. *Read and Write* pada pin 5 digunakan agar LCD dapat menentukan akan melakukan tindakan baca atau tulis data. Pin 6 pada LCD untuk mengaktifkan (*enable*) atau menonaktifkan (*disable*) kegiatan baca dan tulis yang dilakukan oleh mikrokontroler. Pin 11 hingga pin 14 digunakan untuk jalur data *4 bit interface*. Dan pin 15 (dihubungkan ke tegangan +5 volt) berfungsi sebagai *backlight* yang akan menyinari LCD agar tulisan dapat terlihat dengan jelas.



Gambar 3.17. Modul LCD *Display*



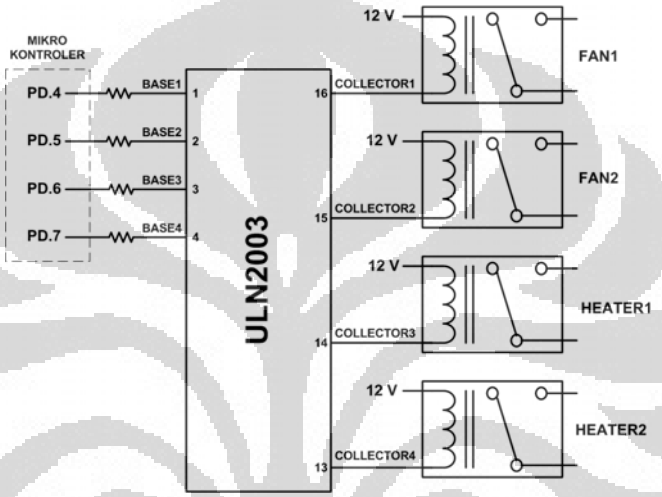
Gambar 3.18. Tata Letak Komponen Modul LCD *Display*



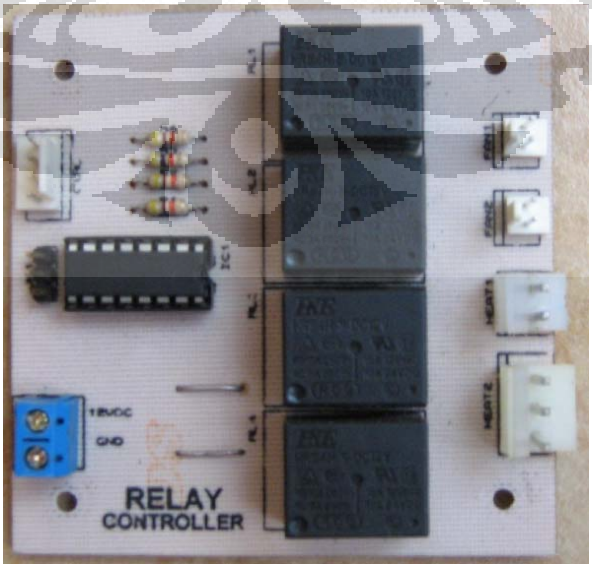
Gambar 3.19. Jalur PCB Modul LCD *Display*

3.2.5. Pengendali relay (*relay driver*)

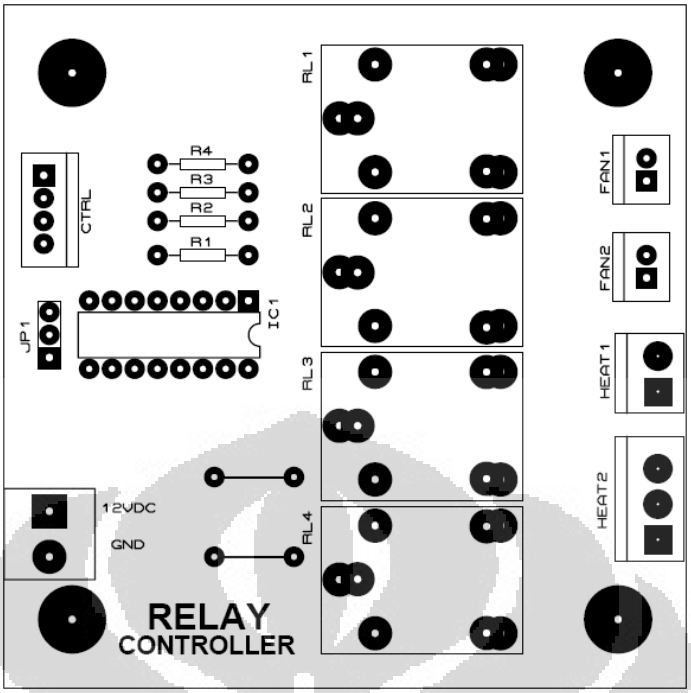
Kebanyakan beban yang digunakan pada proses industri memiliki tegangan dan daya yang sangat besar. Pin-pin keluaran pada mikrokontroler tidak mampu untuk dihubungkan secara langsung ke peralatan bertegangan tinggi. Karena alasan itulah, pengendali relay digunakan untuk menghubungkan mikrokontroler dengan peralatan industris yang ingin dikendalikan. Gambar 3.20 menunjukan rangkaian pengendali relay yang digunakan pada sistem ini.



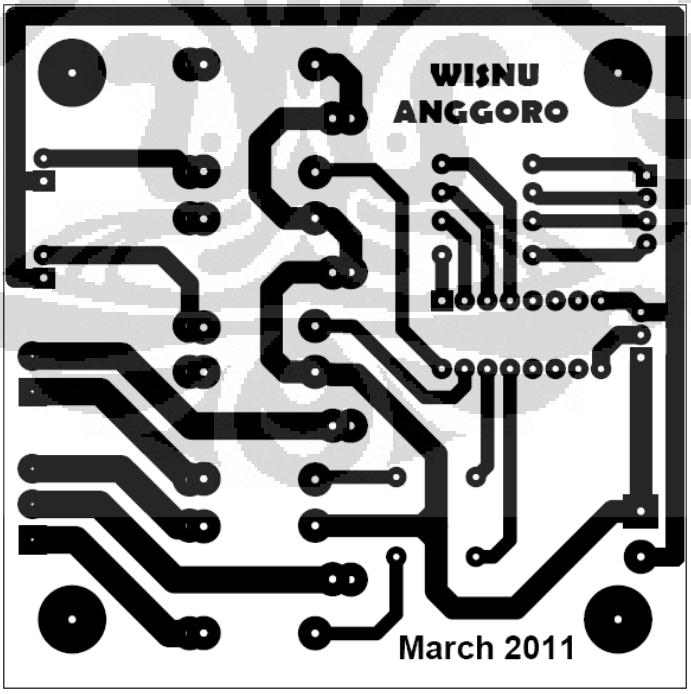
Gambar 3.20. Rangkaian Pengendali *Relay*



Gambar 3.21. Modul Pengendali *Relay*



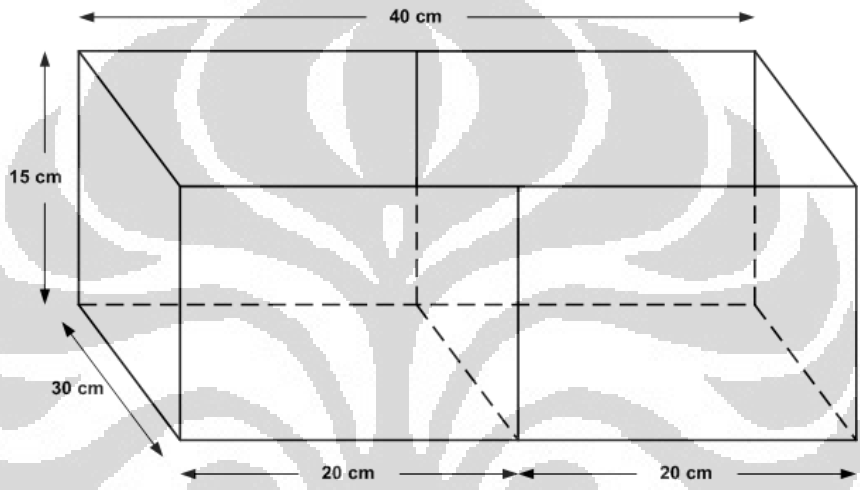
Gambar 3.22. Tata Letak Komponen Modul Pengendali *Relay*



Gambar 3.23. Jalur PCB Modul Pengendali *Relay*

3.3. Perancangan *Casing*

Kelima modul yang telah dibuat pada perancangan perangkat keras disusun pada sebuah *casing* yang terbuat dari kerangka *stainless* yang pada bagian bawahnya diberi akrilik tanpa warna (bening) dengan ketebalan 3 mm serta pada sisi depan dan belakang diberi akrilik bening dengan ketebalan 2 mm. Gambar 3.24 menunjukkan bentuk serta ukuran kerangka *casing*.



Gambar 3.24. Ukuran dan Bentuk Kerangka *Casing*

Gambar 3.25. Tampak Depan *Casing*

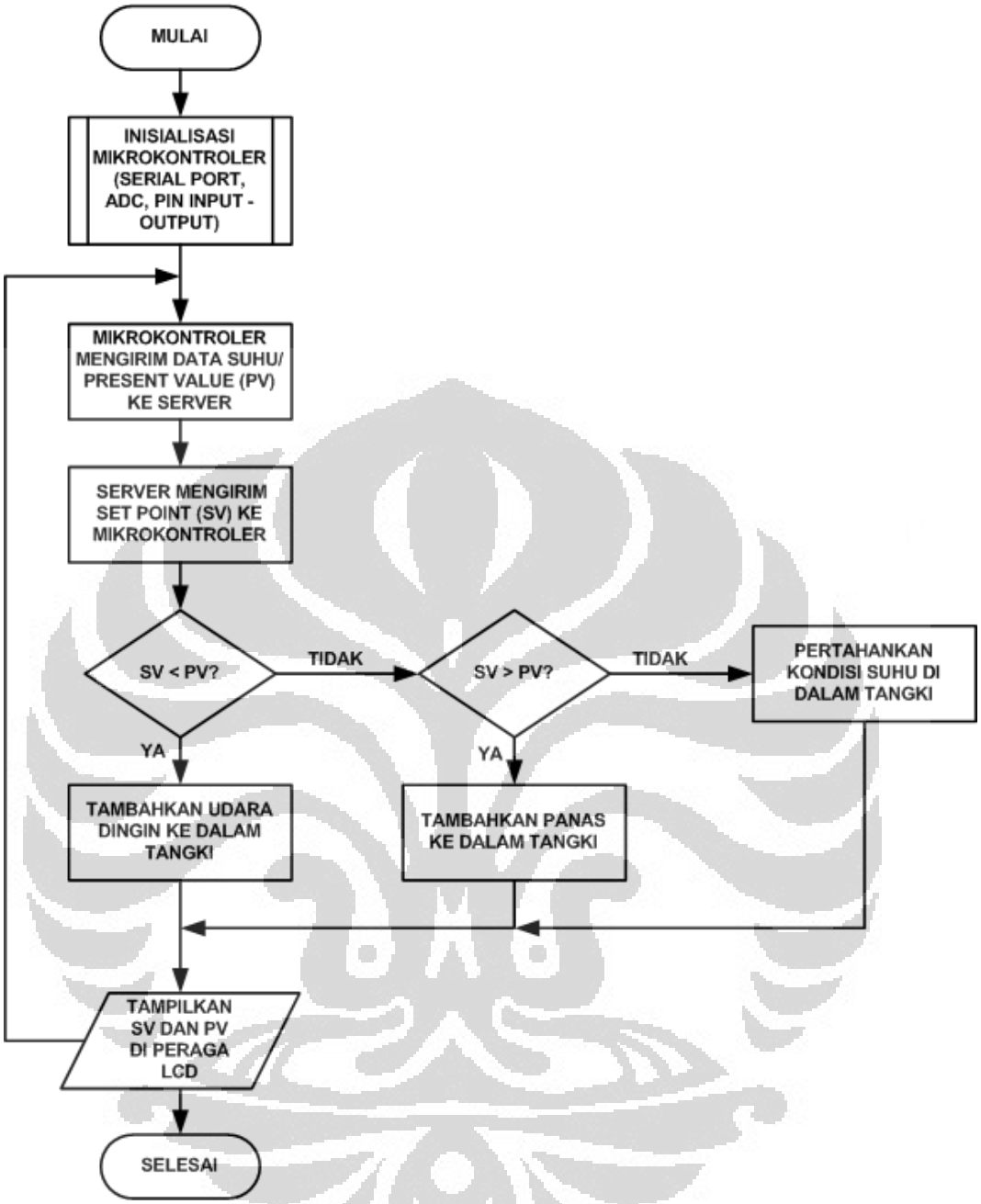
3.4. Perancangan Perangkat Lunak

Perancangan perangkat lunak untuk sistem ini dibagi menjadi dua bagian, yaitu perangkat lunak pada mikrokontroler dan *server*. Perangkat lunak pada mikrokontroler bertugas untuk mengendalikan kipas dan pemanas yang digunakan untuk pengendalian suhu sedangkan perangkat lunak pada *server* bertugas sebagai penerima masukan (*input*) dari pengguna dan memberikan keluaran (*output*) berupa tampilan data suhu yang telah diukur oleh mikrokontroler melalui sensor suhu. Secara keseluruhan, jalannya program pada sistem dapat dijelaskan dengan algoritma pada gambar 3.26.

Program di mikrokontroler melakukan inisialisasi pada serial port, ADC, dan pin input - output. Kemudian mikrokontroler mengirimkan data suhu yang diperoleh dari pembacaan sensor LM35 ke *server* (komputer) melalui serial port. Komputer mengirimkan data *setting point* dari *user* ke mikrokontroler.

Setelah mikrokontroler memiliki data suhu di dalam tangki (*present value*) dan data suhu yang diinginkan oleh user (*setting point*), maka akan dibandingkan kedua data tersebut dan data perbandingan inilah yang akan dijadikan acuan untuk menggerakkan relay. Jika SV lebih kecil daripada PV, berarti *user* ingin menurunkan suhu yang ada di dalam tangki. Maka kipas (*fan*) akan berputar dengan lebih kencang sedangkan kondisi *heater* tidak aktif. Sebaliknya, jika SV lebih besar daripada PV, berarti *user* ingin menaikkan suhu di dalam tangki, sehingga *heater* akan menyala dengan lebih kencang untuk menghasilkan panas yang lebih besar.

Namun jika bukan kedua hal tersebut yang terjadi, maka program akan menyimpulkan bahwa suhu di dalam tangki sama dengan suhu yang diinginkan oleh *user* (SV = PV) sehingga sistem akan mempertahankan suhu yang ada di dalam tangki dengan cara mengaktifkan kipas dan pemanas secara bersamaan namun dalam tingkat yang cukup rendah.



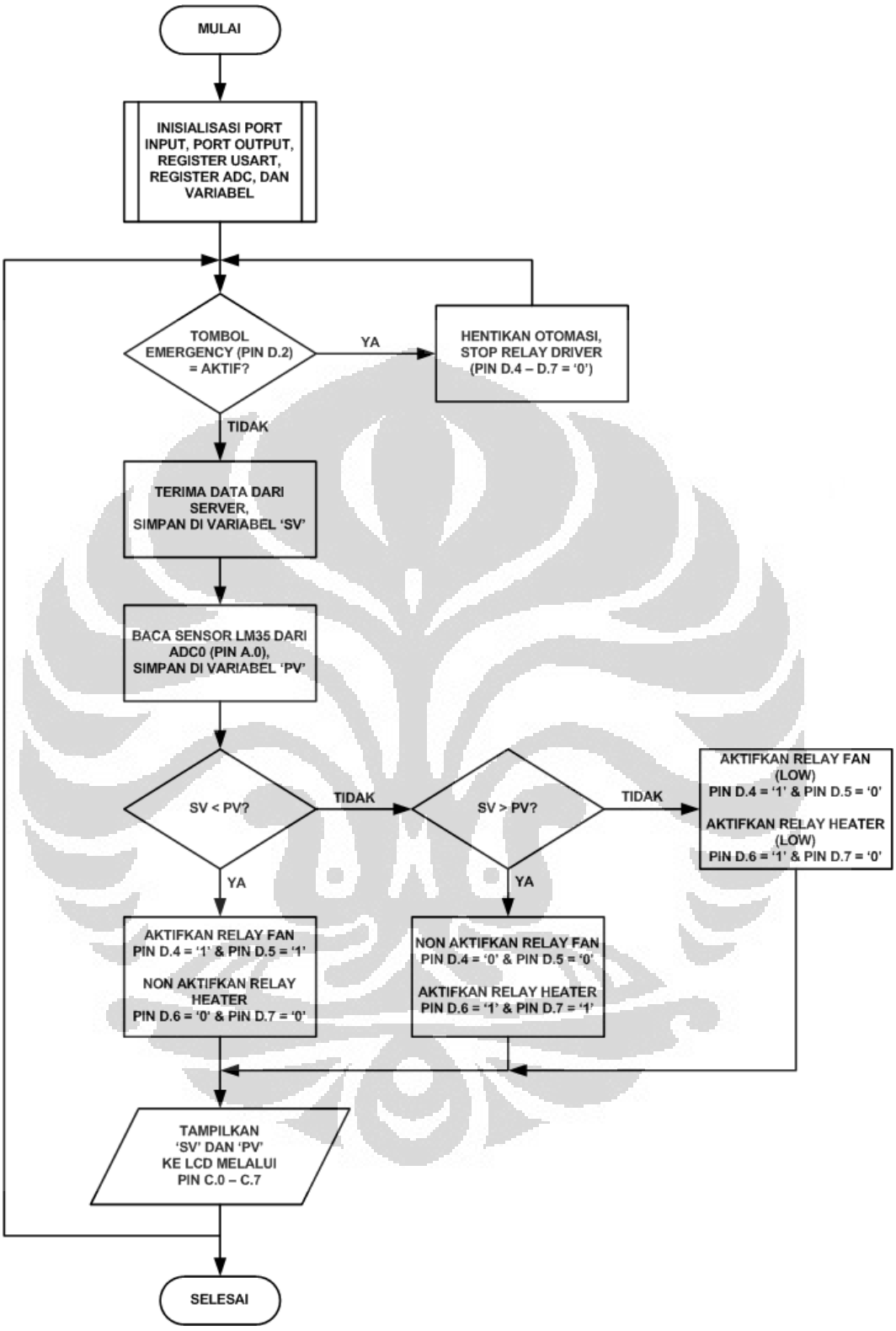
Gambar 3.26. Diagram Alir Keseluruhan Sistem

3.4.1. Perancangan Perangkat Lunak Pada Mikrokontroler

Pembuatan program untuk mikrokontroler ATMega8535 menggunakan bahasa C dengan bantuan *software* Code Vision AVR C Compiler yang merupakan perangkat lunak pemrograman mikrokontroler untuk keluarga AVR berbasis bahasa C. Ada tiga komponen utama yang telah diintegrasikan dalam *compiler* ini: *text editor* bahasa pemrograman C, *assembler* (mengubah file *.c menjadi *.hex), dan *program downloader* (untuk mengunduh file *.hex ke dalam *chip* mikrokontroler). Ketiga komponen tersebut sudah cukup lengkap untuk menghasilkan sebuah program bagi chip mikrokontroler.

Gambar 3.27. Code Vision AVR

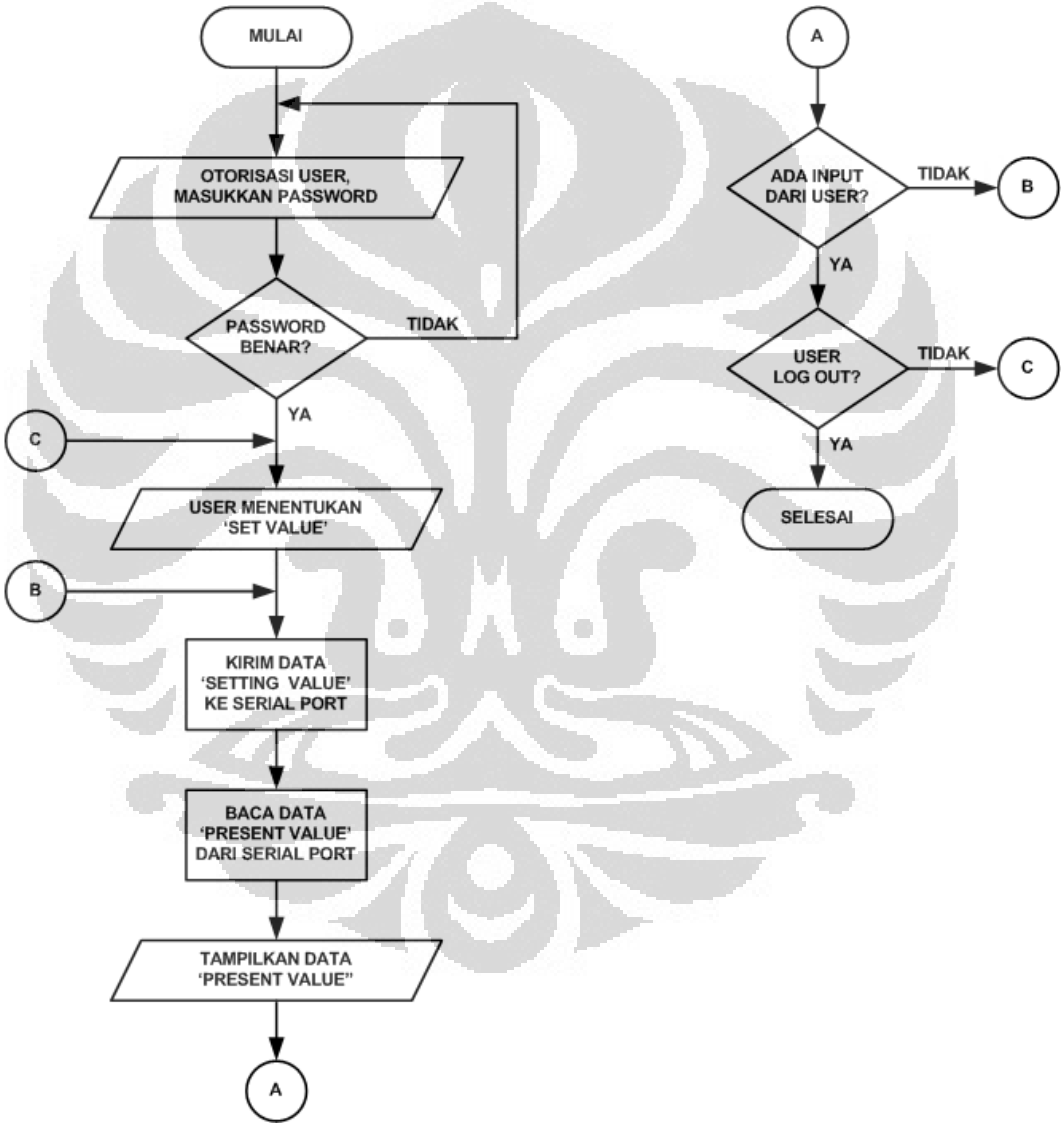
Gambar 3.28 menjelaskan diagram alir jalannya program yang akan dilakukan oleh mikrokontroler. Setelah memperoleh nilai *setting value* (SV) dari server, mikrokontroler membandingkan nilai tersebut dengan nilai *present value* (PV) yang telah didapat dari pengukuran yang telah dilakukan oleh sensor suhu. Hanya akan ada tiga kemungkinan dari hasil perbandingan tersebut, yaitu SV lebih kecil dari PV yang berarti pengguna ingin menurunkan suhu pada tangki, SV lebih besar dari PV yang berarti pengguna ingin menaikkan suhu pada tangki, dan jika bukan salah satu dari dua keadaan diatas yang terjadi, maka mikrokontroler akan menyimpulkan bahwa suhu pada tangki (PV) sama dengan permintaan pengguna (SV).



Gambar 3.28. Diagram Alir Perangkat Lunak Pada Mikrokontroler

3.4.2. Perancangan Perangkat Lunak Pada *Server*

Perangkat lunak pada *server* dibuat dengan bahasa pemrograman ASP.NET karena kemudahan dalam pemrogramannya dan kompabilitas ke semua versi sistem operasi Microsoft Windows® terbaru. Gambar 3.29 menunjukkan diagram alir jalannya program pada sisi *server*.



Gambar 3.29. Diagram Alir Perangkat Lunak Pada *Server*

Saat *user* mengakses ke *server*, halaman pertama yang muncul adalah permintaan untuk memasukkan *username* dan *password* sebagai bagian dari pengamanan agar tidak semua orang bisa melakukan pengendalian dan pemantauan. Halaman tidak akan berpindah ke menu berikutnya jika *username* dan *password* yang dimasukkan salah.

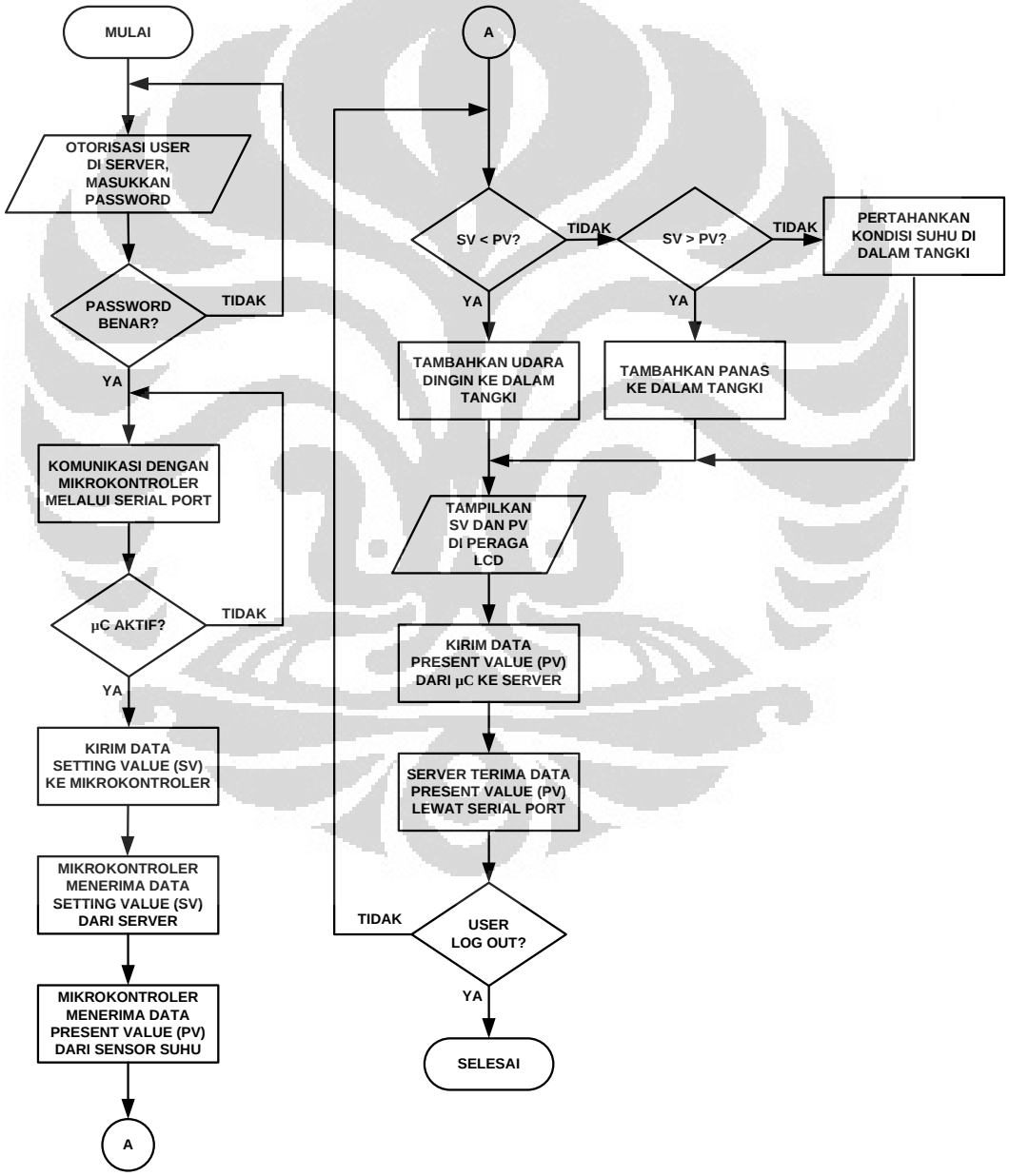
Pada halaman selanjutnya terdapat menu untuk *user* agar bisa memberikan nilai *setting value* (SV). Nilai SV tersebut akan langsung dikirim ke mikrokontroler melalu serial port dan setelah pengiriman data tersebut *server* membaca data yang ada pada *serial port* dan ditandai sebagai *present value* (PV). Dan nilai PV tersebut akan ditampilkan di layar untuk menginformasikan kepada *user*. Halaman ini akan terus tampil sampai user melakukan *log out* (keluar).

BAB 4

PENGUJIAN DAN ANALISIS

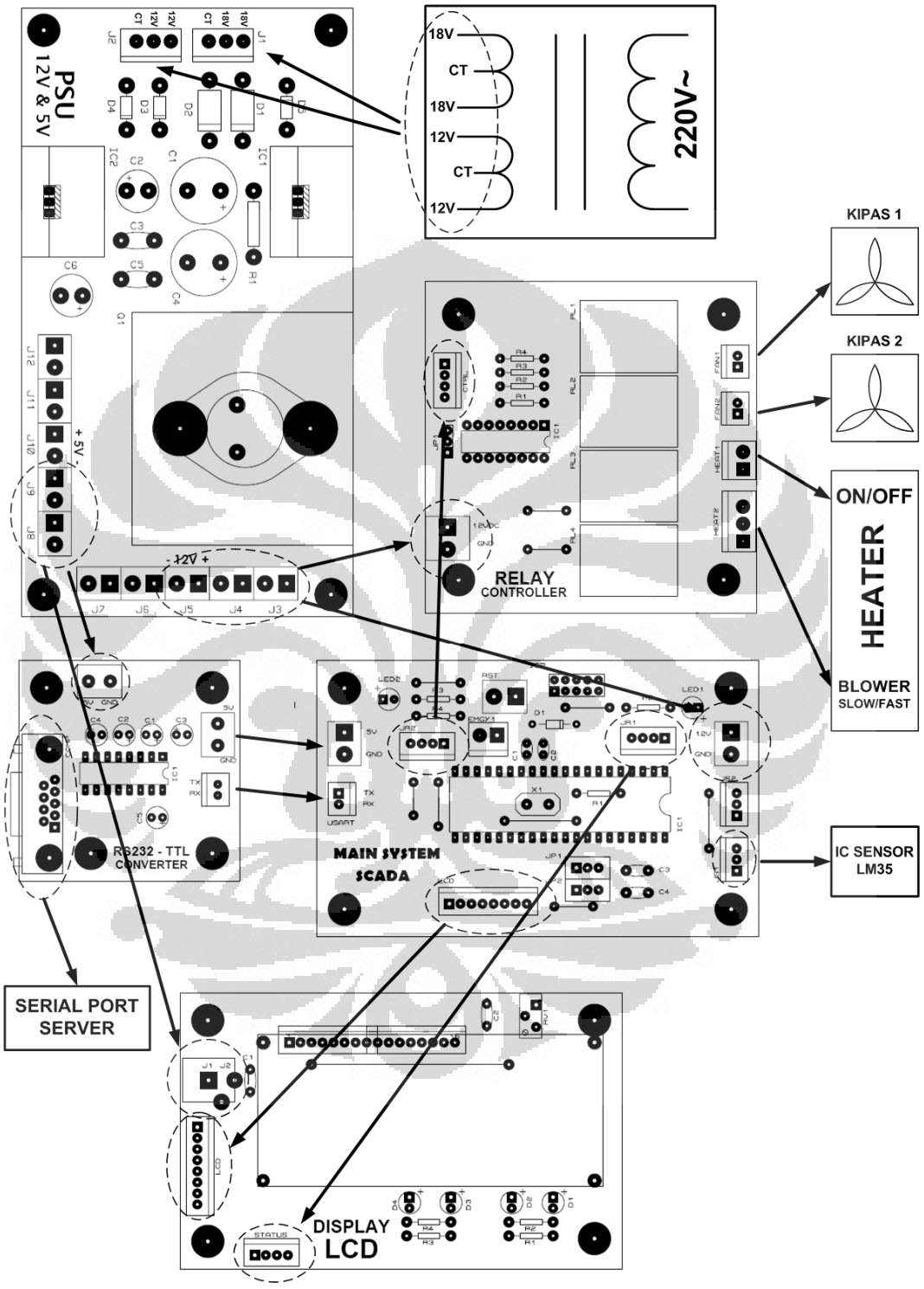
4.1. Pengujian Alat

Alat yang telah dibuat akan diuji dengan alur program seperti terlihat pada gambar 4.1.

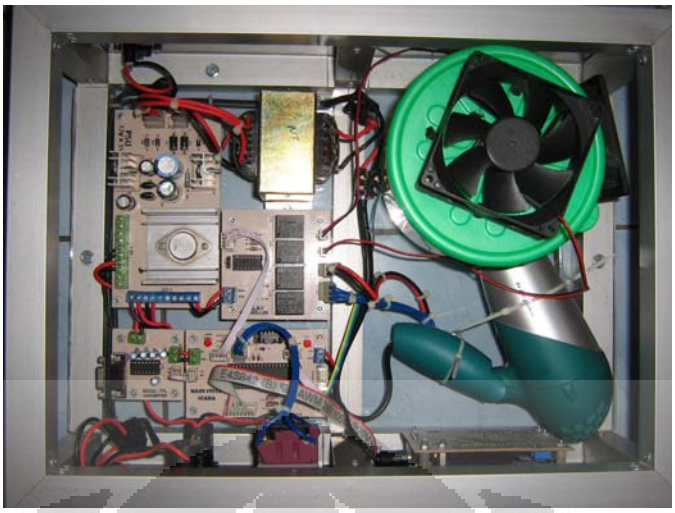


Gambar 4.1. Diagram Alir Pengujian Alat

Sedangkan untuk pengkabelan (*wiring*) pada keseluruhan subsistem yang diuji terlihat pada gambar 4.2 di bawah ini.



Gambar 4.2. Hubungan Antar Subsistem



Gambar 4.3. Tata Letak Keseluruhan Subsistem

Langkah-langkah yang dilakukan dalam pengujian dan pengambilan data adalah sebagai berikut:

1. Menjalankan subsistem catu daya dengan memberikan tegangan 220 V~ ke trafo *Center Tap* kemudian menghubungkan *output* trafo (18V dan 12V) ke modul catu daya,
2. Menghubungkan *output* +5 volt dari modul catu daya ke modul mikrokontroler, RS232 *converter*, dan LCD. Sedangkan *output* +12 volt dari modul catu daya dihubungkan ke *relay controller* dan sensor LM35 yang ada pada modul sistem utama mikrokontroler,
3. Menghubungkan kipas 1, kipas 2, dan pemanas ke modul *relay controller*,
4. Menghubungkan konektor DB9 pada RS232 *converter* ke *port* serial yang ada di *server*,
5. Menghubungkan jalur data 8 pin dari mikrokontroler ke modul LCD serta dua buah jalur 4 pin dihubungkan ke modul LCD dan *relay controller* sesuai gambar 4.2,
6. Menjalankan program pada *server* menggunakan *localhost* (*virtual server*),
7. Melakukan koneksi ke mikrokontroler dan set *baud rate* sebesar 9600 bps,
8. Saat nilai *Present Value* adalah 32°C , ubah nilai *Setting Value* ke suhu 47°C dan mencatat hasilnya seperti tabel 4.1 dan tabel 4.2,
9. Saat nilai *Present Value* telah mencapai 47°C, ubah nilai *Setting Value* ke suhu 32°C dan mencatat hasilnya seperti tabel 4.3 dan tabel 4.4.

Tabel 4.1. Hasil Pengujian SV Lebih Besar Daripada PV

SV (°C)	PV (°C)	FAN 1	FAN 2	HEATER	WAKTU TEMPUH	TOTAL WAKTU
47	33	OFF	OFF	MAX	06.52	00:06.52
47	34	OFF	OFF	MAX	11.77	00:18.29
47	35	OFF	OFF	MAX	03.99	00:22.28
47	36	OFF	OFF	MAX	09.10	00:31.38
47	37	OFF	OFF	MAX	04.27	00:35.65
47	38	OFF	OFF	MAX	08.09	00:43.74
47	39	OFF	OFF	MAX	04.14	00:47.88
47	40	OFF	OFF	MAX	08.87	00:56.75
47	41	OFF	OFF	MAX	12.74	01:09.49
47	42	OFF	OFF	MAX	05.91	01:15.40
47	43	OFF	OFF	MAX	13.89	01:29.29
47	44	OFF	OFF	MAX	04.21	01:33.50
47	45	OFF	OFF	MAX	14.02	01:47.52
47	46	OFF	OFF	MAX	26.55	02:14.07
47	47	ON	OFF	MIN	07.40	02:21.47

Tabel 4.2. *Fan* dan *Heater* Mempertahankan Suhu Sesuai SV = 47°C

FAN 1	FAN 2	HEATER	WAKTU TEMPUH	TOTAL WAKTU
ON	OFF	MIN	01.52	00:01.52
OFF	OFF	MAX	00.68	00:02.20
ON	OFF	MIN	01.25	00:03.45
OFF	OFF	MAX	01.65	00:05.10
ON	OFF	MIN	01.10	00:06.20
OFF	OFF	MAX	00.93	00:07.13
ON	OFF	MIN	01.32	00:08.44
OFF	OFF	MAX	01.41	00:09.85
ON	OFF	MIN	00.17	00:10.02

Tabel 4.3. Hasil Pengujian SV Lebih Kecil Daripada PV

SV (°C)	PV (°C)	FAN 1	FAN 2	HEATER	WAKTU TEMPUH	TOTAL WAKTU
32	46	ON	ON	OFF	03.29	00:03.29
32	45	ON	ON	OFF	02.99	00:06.28
32	44	ON	ON	OFF	08.17	00:14.45
32	43	ON	ON	OFF	04.20	00:18.65
32	42	ON	ON	OFF	01.77	00:20.42
32	41	ON	ON	OFF	06.55	00:26.97
32	40	ON	ON	OFF	02.62	00:29.59
32	39	ON	ON	OFF	03.94	00:33.53
32	38	ON	ON	OFF	10.71	00:44.24
32	37	ON	ON	OFF	04.24	00:48.48
32	36	ON	ON	OFF	06.59	00:55.07
32	35	ON	ON	OFF	04.97	01:00.04
32	34	ON	ON	OFF	11.63	01:11.67
32	33	ON	ON	OFF	07.44	01:19.11
32	32	ON	OFF	MIN	09.07	01:28.18

Tabel 4.4. *Fan* dan *Heater* Mempertahankan Suhu Sesuai SV = 32°C

FAN 1	FAN 2	HEATER	WAKTU TEMPUH	TOTAL WAKTU
ON	OFF	MIN	01.34	00:01.34
ON	ON	OFF	00.77	00:02.11
ON	OFF	MIN	12.71	00:14.82
ON	ON	OFF	06.58	00:21.40
ON	OFF	MIN	01.19	00:22.59
ON	ON	OFF	00.79	00:23.38
ON	OFF	MIN	03.89	00:27.27
ON	ON	OFF	01.67	00:28.94
ON	OFF	MIN	01.54	00:30.48

Pada tahap pengujian ini juga dilakukan percobaan untuk mengetahui tanggapan sistem terhadap gangguan yang terjadi, yaitu gangguan putusnya kabel komunikasi RS232 antara mikrokontroler dengan *server* dan gangguan yang diakibatkan karena kegagalan pengiriman data dari *client* ke mikrokontroler.

Langkah-langkah yang dilakukan dalam pengujian tanggapan sistem terhadap gangguan adalah sebagai berikut:

1. Menjalankan alat sesuai dengan langkah-langkah pengujian sebelumnya dan tentukan *setting value* sebesar 40°C
2. Saat alat telah berjalan untuk mencapai suhu yang diinginkan, lepaskan kabel RS232 antara mikrokontroler dengan *server*. Setelah 10 detik, hubungkan kembali kabel RS232 antara mikrokontroler dengan *server*.
3. Dalam kondisi kabel RS232 masih terhubung, hentikan *server* dari IIS *control panel* (*server virtual* untuk ASP.net) untuk membuat seolah-olah telah terjadi kegagalan pengiriman data. Setelah 10 detik, nyalakan kembali *server* dari IIS *control panel*.

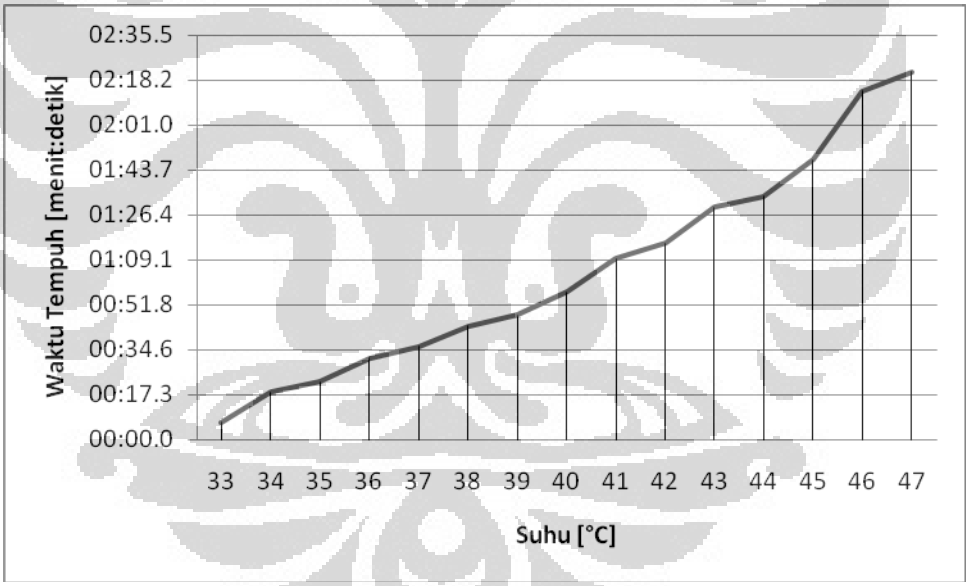
Hasil pengujian sistem terhadap gangguan jalur komunikasi terlihat pada tabel 4.5 di bawah ini.

Tabel 4.5. Hasil Pengujian Sistem Terhadap Gangguan Jalur Komunikasi

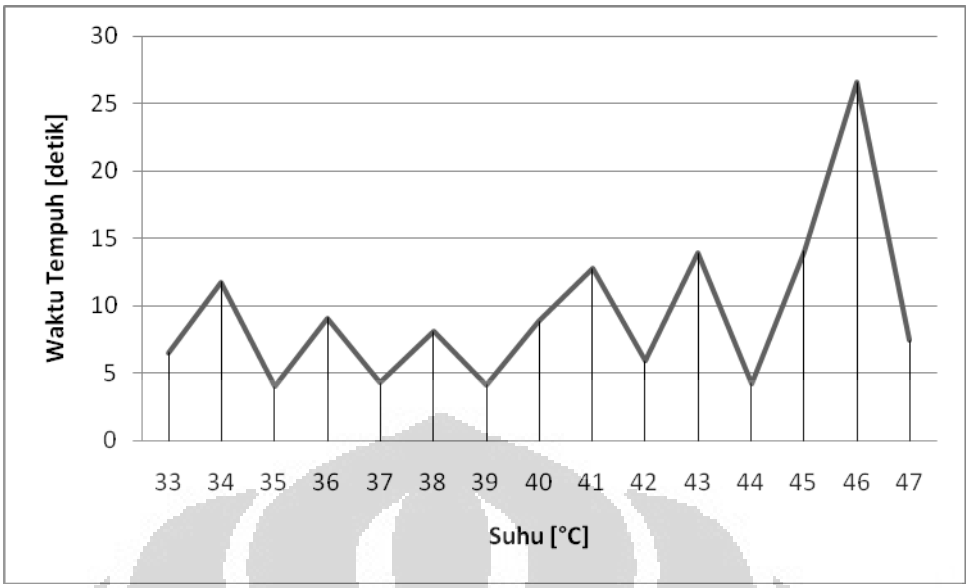
GANGGUAN YANG TERJADI		**KONDISI SISTEM OTOMASI**
KABEL RS232	TERPUTUS	BERHENTI
	TERHUBUNG	BERJALAN
SERVER	OFF	BERHENTI
	RUNNING	BERJALAN

4.2. Analisis Data Hasil Pengujian

Dari tabel 4.1 terlihat bahwa waktu yang diperlukan untuk memperoleh suhu sebesar 47°C dari suhu awal sebesar 32°C adalah selama 2 menit 21 detik 47 milidetik (02:21.47). Pada gambar 4.4 terlihat bahwa peningkatan suhu di dalam tangki berbanding lurus dengan waktu tempuh yang diperlukan. Gambar 4.5 menunjukkan grafik waktu tempuh untuk masing-masing perubahan suhu (dari dingin ke panas) yang terjadi di dalam tangki. Terlihat bahwa waktu tempuh terlama adalah saat kondisi suhu 45°C dan akan menginjak ke suhu 46°C yaitu sebesar 26 detik 55 milidetik. Sedangkan waktu tempuh tercepat adalah pada saat kondisi suhu 34°C dan akan menginjak ke suhu 35°C yaitu sebesar 3 detik 99 milidetik.



Gambar 4.4. Grafik Peningkatan Suhu Berbanding Dengan Waktu Tempuh

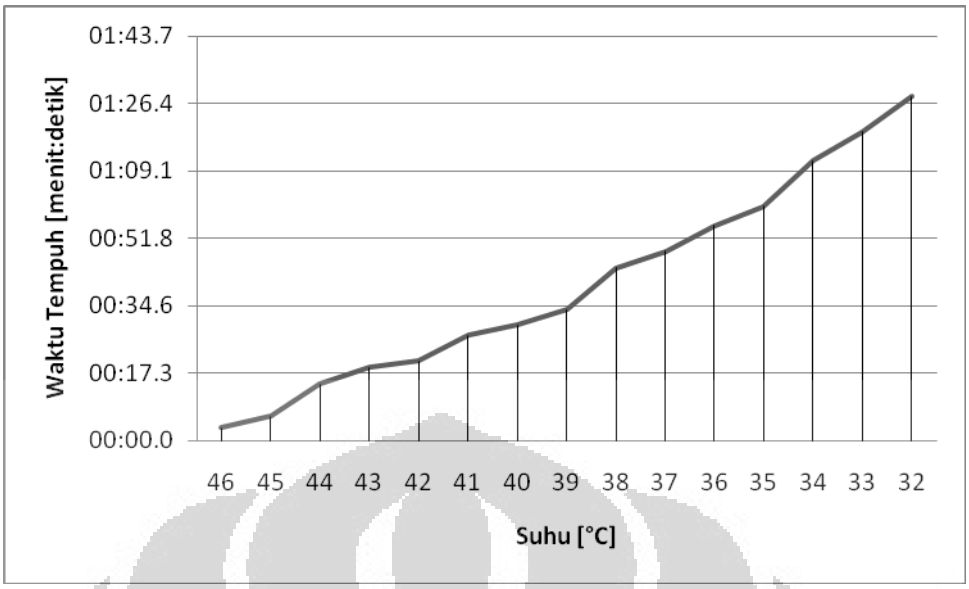


Gambar 4.5. Grafik Waktu Tempuh dari Suhu Dingin ke Panas

Ketika suhu yang diinginkan telah tercapai (*present value* menjadi sama dengan *setting value*), maka kondisi di dalam tangki akan dipertahankan dengan cara *heater* bekerja dengan putaran rendah (agar menghasilkan panas yang kecil) dan *fan* yang bekerja hanya satu buah agar tidak banyak udara dingin yang mengalir ke tangki. Tabel 4.2 menunjukkan kondisi *heater* dan *fan* untuk menjaga suhu tetap sesuai dengan nilai SV, yaitu 47°C.

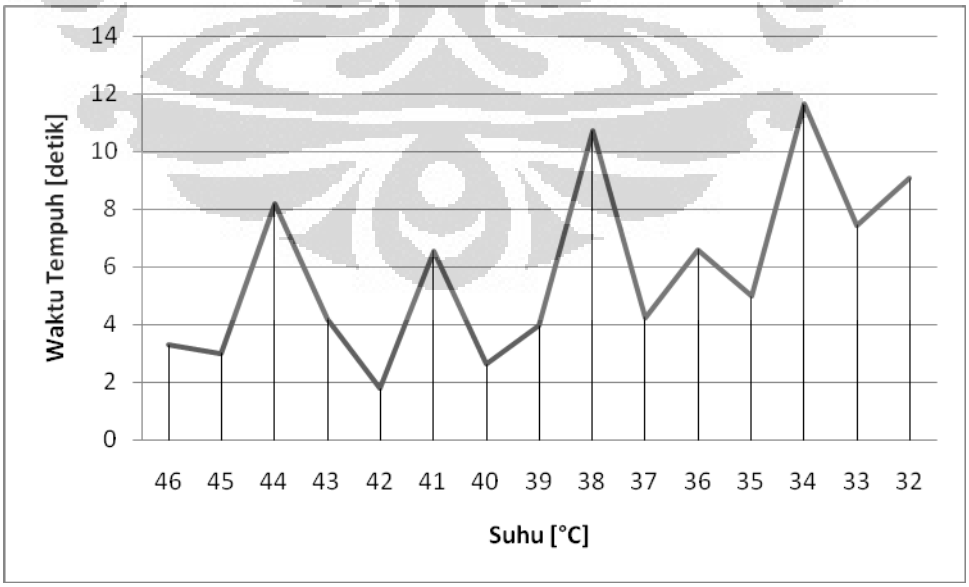
Ketika suhu tangki sudah mencapai nilai *setting value,* maka *fan* dan *heater* akan berputar secara perlahan. Selang sekitar 2 menit pada kondisi tersebut, suhu di dalam tangki turun satu derajat celcius, sehingga putaran *heater* menjadi kencang kembali hingga *setting value* tercapai. Kejadian tersebut terus berulang dan suhu pada tangki dapat dijaga dengan toleransi ± 1°C dari nilai *setting value*.

Tabel 4.3 menunjukkan bahwa waktu yang diperlukan untuk memperoleh suhu sebesar 32°C dari suhu awal sebesar 47°C adalah selama 1 menit 28 detik 18 milidetik (01:28.18). Pada gambar 4.6 terlihat penurunan suhu di dalam tangki berbanding lurus dengan waktu tempuh yang diperlukan. Dibandingkan dengan grafik pada gambar 4.4 terlihat bahwa waktu tempuh untuk menurunkan suhu di dalam tangki lebih cepat daripada menaikkan suhu di dalam tangki dengan ΔT yang sama yaitu 15°C.



Gambar 4.6. Grafik Penurunan Suhu Berbanding Dengan Waktu Tempuh

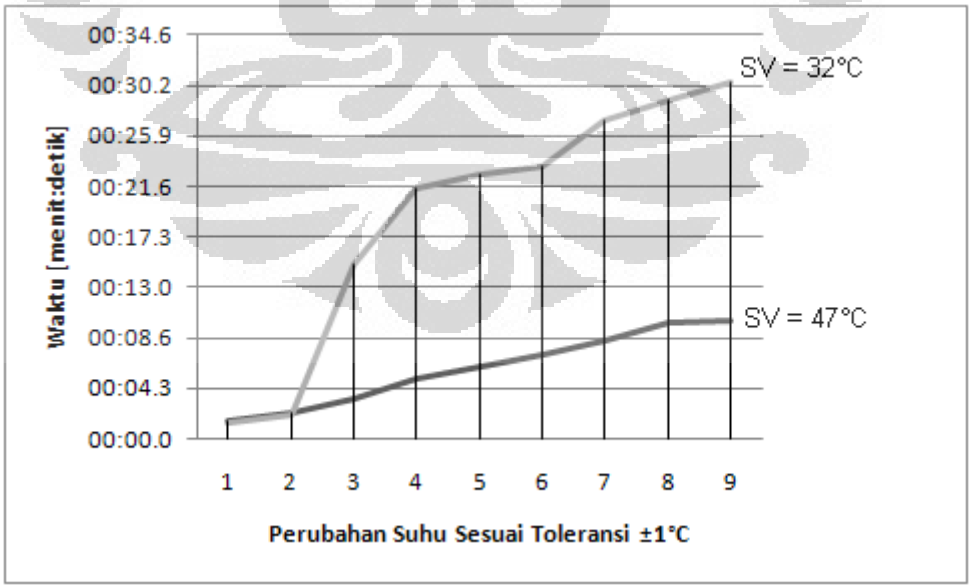
Gambar 4.7 menunjukkan grafik waktu tempuh untuk masing-masing perubahan suhu (dari panas ke dingin) yang terjadi di dalam tangki. Terlihat bahwa waktu tempuh terlama adalah saat kondisi suhu 35°C dan akan menuju ke suhu 34°C yaitu sebesar 11 detik 63 milidetik. Sedangkan waktu tempuh paling cepat adalah pada saat kondisi suhu 43°C dan akan menuju ke suhu 42°C yaitu sebesar 1 detik 77 milidetik.



Gambar 4.7. Grafik Waktu Tempuh dari Suhu Panas ke Dingin

Ketika suhu yang diinginkan telah tercapai (*present value* menjadi sama dengan *setting value*), maka kondisi di dalam tangki akan dipertahankan dengan cara *heater* bekerja dengan putaran rendah (agar menghasilkan panas yang kecil) dan *fan* yang bekerja hanya satu buah agar tidak banyak udara dingin yang mengalir ke tangki. Tabel 4.4 menunjukkan kondisi *heater* dan *fan* untuk menjaga suhu tetap sesuai dengan nilai SV, yaitu 32°C. Ketika suhu tangki sudah mencapai nilai *setting value,* maka *fan* dan *heater* akan berputar secara perlahan. Selang sekitar 1.5 menit pada kondisi tersebut, suhu di dalam tangki naik satu derajat celcius, sehingga putaran *fan* menjadi kencang kembali hingga *setting value* tercapai. Kejadian tersebut terus berulang dan suhu pada tangki dapat dijaga dengan toleransi ± 1°C dari nilai *setting value*.

Untuk data hasil percobaan di tabel 4.2 dan tabel 4.4 dapat dibuat grafik seperti pada gambar 4.8 untuk menunjukkan kondisi sistem dalam keadaan mempertahankan suhu di dalam tangki. Saat nilai SV diatur menjadi 47°C dan PV telah sama dengan SV, suhu di dalam tangki berubah-ubah dengan cepat (di bawah 2 detik) namun tetap berada dalam toleransi ± 1°C. Sedangkan saat nilai SV diatur menjadi 32°C dan PV telah sama dengan SV, suhu di dalam tangki cenderung lebih stabil karena sistem dapat menjaga suhu sesuai PV lebih lama.



Gambar 4.8. Grafik Kestabilan Suhu

Tabel 4.5 menunjukkan kondisi sistem saat terjadi gangguan, di mana otomasi akan berhenti untuk menghindari sistem yang tidak terkendali. Ketika kabel RS232 terputus, maka tidak terjadi proses *handshaking* antara mikrokontroler dengan komputer *server* dan dengan demikian maka mikrokontroler akan menjalankan rutin lainnya yang dalam hal ini adalah mematikan seluruh relay sehingga otomasi akan berhenti. Tabel 4.6 menunjukkan kode program untuk menangani masalah tersebut.

Tabel 4.6. Kode Program Penanganan Gangguan Jalur Komunikasi Kabel

KODE PROGRAM

stopAll=getchar();
data_ser=getchar();
if (stopAll==0){
// Jalankan rutin otomasi
if (data_ser > PV) { // Panaskan tangki }
else if (data_ser < PV) { // Dinginkan tangki }
else { // Jaga kondisi suhi di dalam tangki }
// Kembalikan nilai stopAll menjadi “1” sehingga jika tidak diubah oleh
// server, maka dianggap tidak ada permintaan dari client atau karena
// terjadi gangguan komunikasi sehingga rutin “Hentikan Otomasi” yang
// akan dijalankan
stopAll=1;
}
else {
// Hentikan Otomasi
}

Sementara saat IIS *Server* dihentikan untuk menguji kegagalan pengiriman data dari *client* ke mikrokontroler, *handshaking* antara mikrokontroler dengan komputer *server* tetap berlangsung namun data yang terkirim tidak akan berubah walaupun *client* telah mengubah data *setting value*. Mikrokontroler mendeteksi kegagalan ini dari ada atau tidaknya perubahan nilai *handshaking* karena saat data gagal terkirim maka nilai *handshaking* tidak berubah. Dan jika hal itu terjadi, maka mikrokontroler akan menjalankan rutin lainnya yaitu mematikan seluruh relay sehingga otomasi akan berhenti. Tabel 4.7 menunjukkan kode program untuk menangani kegagalan pengiriman data dari *client* ke mikrokontroler.

Tabel 4.7. Kode Program Penanganan Kegagalan Pengiriman Data

KODE PROGRAM

handShaking=getchar();
// Jika data handshaking saat ini berbeda dari data handshaking sebelumnya maka
// pengiriman data berhasil dan otomasi dijalankan
if (tempHandShaking=! handShaking){
// Jalankan rutin otomasi
}
// Jika data handshaking saat ini sama dengan data handshaking sebelumnya maka
// pengiriman data dianggap gagal dan otomasi dihentikan
else {
// Hentikan Otomasi
}

BAB 5

KESIMPULAN

Dari hasil perancangan dan pengujian didapat kesimpulan sebagai berikut:

1. Sistem ini dapat berjalan dengan baik karena telah mampu merespon *input* dari *client* (*setting value*), membaca *output* dari sensor LM35 (*present value*), dan membandingkan kedua nilai *input* dan *output* tersebut.

2. Saat nilai *present value* telah sama dengan *setting value*, maka mikrokontroler akan mempertahankan kondisi suhu dengan toleransi ± 1°C (yang berarti bahwa suhu yang terukur tidak akan lebih 1°C dari *setting value* maupun kurang 1°C dari *setting value*) dengan cara mengatur kombinasi *on/off* pada pemanas dan pendingin.

3. Perangkat lunak yang ditanam pada mikrokontroler dapat mendeteksi gangguan pada jalur komunikasi kabel RS232 antara mikrokontroler dengan *server* sehingga saat terjadi gangguan pada jalur komunikasi kabel (pada pengujian dilakukan pemutusan kabel) maka mikrokontroler akan menghentikan otomasi untuk mencegah terjadinya sistem yang tidak terkendali.

4. Gangguan komunikasi nirkabel (*wireless*) antara mikrokontroler dengan *client* (seperti *delay* yang terlalu lama atau data yang dikirim tidak sampai ke tujuan) dapat dideteksi oleh mikrokontroler dengan melihat perubahan nilai variable *handshaking* pada kode program mikrokontroler karena saat data gagal terkirim maka nilai *handshaking* tidak berubah sehingga menyebabkan terhentinya otomasi.

DAFTAR REFERENSI

[1] Heri Andrianto, *Pemrograman Mikrokontroler AVR ATMEGA16 Menggunakan Bahasa C (Code Vision AVR)* (Bandung: Informatika Bandung, 2008), hal. 2

[2] Ardi Winoto, *Mikrokontroler AVR ATmega8/32/16/8535 dan Pemrogramannya dengan Bahasa C pada WinAVR* (Bandung: Informatika Bandung, 2008), hal. 52

[3] Ardi Winoto, *Mikrokontroler AVR ATmega8/32/16/8535 dan Pemrogramannya dengan Bahasa C pada WinAVR* (Bandung: Informatika Bandung, 2008), hal. 43

[4] Heri Andrianto, *Pemrograman Mikrokontroler AVR ATMEGA16 Menggunakan Bahasa C (Code Vision AVR)* (Bandung: Informatika Bandung, 2008), hal. 5

[5] Ardi Winoto, *Mikrokontroler AVR ATmega8/32/16/8535 dan Pemrogramannya dengan Bahasa C pada WinAVR* (Bandung: Informatika Bandung, 2008), hal. 45

[6] Ardi Winoto, *Mikrokontroler AVR ATmega8/32/16/8535 dan Pemrogramannya dengan Bahasa C pada WinAVR* (Bandung: Informatika Bandung, 2008), hal. 51

[7] Heri Andrianto, *Pemrograman Mikrokontroler AVR ATMEGA16 Menggunakan Bahasa C (Code Vision AVR)* (Bandung: Informatika Bandung, 2008), hal. 103

[8] Heri Andrianto, *Pemrograman Mikrokontroler AVR ATMEGA16 Menggunakan Bahasa C (Code Vision AVR)* (Bandung: Informatika Bandung, 2008), hal. 113

[9] Ardi Winoto, *Mikrokontroler AVR ATmega8/32/16/8535 dan Pemrogramannya dengan Bahasa C pada WinAVR* (Bandung: Informatika Bandung, 2008), hal. 146

[10] Ardi Winoto, *Mikrokontroler AVR ATmega8/32/16/8535 dan Pemrogramannya dengan Bahasa C pada WinAVR* (Bandung: Informatika Bandung, 2008), hal. 158

[11] Widodo Budiharto, *Interfacing Komputer dan Mikrokontroler* (Jakarta: PT Elex Media Komputindo, 2004), hal. 97

[12] Iswanto, *Belajar Sendiri Mikrokontroler AT90S2313 dengan Basic Compiler* (Yogyakarta: CV Andi Offset, 2009), hal. 85

[13] Ardi Winoto, *Mikrokontroler AVR ATmega8/32/16/8535 dan Pemrogramannya dengan Bahasa C pada WinAVR* (Bandung: Informatika Bandung, 2008), hal. 195

[14] J. Michael Jacob, *Industrial Control Electronics – Application and Design* (New Jersey: Prentice-Hall Inc., 1988), hal. 165

[15] Erick Kurniawan, *Cepat Mahir ASP.NET 3.5 Untuk Aplikasi Web Interaktif* (Yogyakarta: CV Andi Offset, 2010), hal. 1

DAFTAR PUSTAKA

Andrianto, Heri. (2008). *Pemrograman Mikrokontroler AVR ATMEGA16 Menggunakan Bahasa C (Code Vision AVR)*. Bandung: Informatika Bandung.

Budiharto, Widodo. (2004). *Interfacing Komputer dan Mikrokontroler*. Jakarta: PT Elex Media Komputindo.

Iswanto. (2009). *Belajar Sendiri Mikrokontroler AT90S2313 dengan Basic Compiler*. Yogyakarta: CV Andi Offset.

Jacob, J. Michael. (1988). *Industrial Control Electronics – Application and Design*. New Jersey: Prentice-Hall Inc.

Kurniawan, Erick. (2010). *Cepat Mahir ASP.NET 3.5 Untuk Aplikasi Web Interaktif*. Yogyakarta: CV Andi Offset.

Winarno, Edi & Zaki, Ali. (2010). *Web Programming dengan Visual Basic 2010*. Jakarta: PT Elex Media Komputindo.

Winoto, Ardi. (2008). *Mikrokontroler AVR ATmega8/32/16/8535 dan Pemrogramannya dengan Bahasa C pada WinAVR*. Bandung: Informatika Bandung.

Lampiran 1: Kode Program Code Vision AVR

```
// Standard Input/Output functions
#include <mega8535.h>
#include <stdio.h>
#include <stdlib.h>
#include <delay.h>

// Alphanumeric LCD Module functions
#asm
   .equ __lcd_port=0x15 ;PORTC
#endasm
#include <lcd.h>

// Deklarasi konstanta global
// digunakan juga oleh interupsi
#define STATFANLO PORTB.0
#define STATFANHI PORTB.1
#define STATHEATLO PORTB.2
#define STATHEATHI PORTB.3
#define FAN1 PORTD.4
#define FAN2 PORTD.5
#define HEATER1 PORTD.6
#define HEATER2 PORTD.7

// Jika tombol Emergency Stop ditekan
// dan terkunci secara mekanik,
// Mikrokontroler akan menjalankan
// program pada sub di bawah
interrupt [EXT_INT0] void
ext_int0_isr(void)
{
   // Tampilkan pesan di LCD
   lcd_gotoxy(0,0);
   lcd_putsf("-Emergency Stop-");
   lcd_gotoxy(0,1);
   lcd_putsf("Otomasi Berhenti");
   // Keadaan Status
   STATFANLO=!STATFANLO;
   STATFANHI=!STATFANHI;
  STATHEATLO=!STATHEATLO;
   STATHEATHI=!STATHEATHI;

// Matikan semua Relay
   FAN1=0;
   FAN2=0;
   HEATER1=0;
   HEATER2=0;

   // Delay agar indikator
   // status berkedip
   delay_ms(500);
}

// Deklarasi konstanta untuk USART
#define RXB8 1
#define TXB8 0
```

Lampiran 1: Kode Program Code Vision AVR (lanjutan)

```
#define UPE 2
#define OVR 3
#define FE 4
#define UDRE 5
#define RXC 7
#define FRAMING_ERROR (1<<FE)
#define PARITY_ERROR (1<<UPE)
#define DATA_OVERRUN (1<<OVR)
#define DATA_REGISTER_EMPTY (1<<UDRE)
#define RX_COMPLETE (1<<RXC)
// USART Receiver Buffer
#define RX_BUFFER_SIZE 8
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
// Flag rx_buffer_overflow akan di-set
// jika USART Receiver Buffer mengalami 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 ((status & (FRAMING_ERROR | PARITY_ERROR | DATA_OVERRUN))==0)
  {
  rx_buffer[rx_wr_index]=data;
  if (++rx_wr_index == RX_BUFFER_SIZE) rx_wr_index=0;
  if (++rx_counter == RX_BUFFER_SIZE)
    {
    rx_counter=0;
    rx_buffer_overflow=1;
    };
};
}
#ifndef _DEBUG_TERMINAL_IO_
// Ambil karakter dari USART Receiver Buffer
#define _ALTERNATE_GETCHAR_
#pragma used+
char getchar(void)
{
char data;
while (rx_counter==0);
data=rx_buffer[rx_rd_index];
```

Lampiran 1: Kode Program Code Vision AVR (lanjutan)

```
if (++rx_rd_index == RX_BUFFER_SIZE) rx_rd_index=0;
#asm("cli")
--rx_counter;
#asm("sei")
return data;
}
#pragma used-
#endif

// USART Transmitter Buffer
#define TX_BUFFER_SIZE 8
char tx_buffer[TX_BUFFER_SIZE];
#if TX_BUFFER_SIZE<256
unsigned char tx_wr_index,tx_rd_index,tx_counter;
#else
unsigned int tx_wr_index,tx_rd_index,tx_counter;
#endif

// USART Transmitter Interrupt Service Routine
interrupt [USART_TXC] void usart_tx_isr(void)
{
if (tx_counter)
  {
  --tx_counter;
  UDR=tx_buffer[tx_rd_index];
  if (++tx_rd_index == TX_BUFFER_SIZE) tx_rd_index=0;
  };
}
#ifndef _DEBUG_TERMINAL_IO_

// Kirim karakter ke USART Tx Buffer
#define _ALTERNATE_PUTCHAR_
#pragma used+
void putchar(char c)
{
while (tx_counter == TX_BUFFER_SIZE);
#asm("cli")
if (tx_counter || ((UCSRA & DATA_REGISTER_EMPTY)==0))
  {
  tx_buffer[tx_wr_index]=c;
  if (++tx_wr_index == TX_BUFFER_SIZE) tx_wr_index=0;
  ++tx_counter;
  }
else
  UDR=c;
#asm("sei")
}
#pragma used-
#endif
```

Lampiran 1: Kode Program Code Vision AVR (lanjutan)

```
// Deklarasi konstanta ADC
#define ADC_VREF_TYPE 0x00
#define VREF 4.5

// Baca Hasil Konversi ADC
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;
}

// Deklarasi variabel global
unsigned char stopAll, data_ser;
unsigned int data_adc;
float data_suhu;
unsigned char sbuff[16];

void main(void)
{
// Input/Output Ports initialization
PORTA=0x00;
DDRA=0x00;
PORTB=0x00;
DDRB=0xFF;
PORTC=0x00;
DDRC=0x00;
PORTD=0x00;
DDRD=0xFF;

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

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: Timer 1 Stopped
// Mode: Normal top=FFFFh
// OC1A output: Discon.
// OC1B output: Discon.
```

Lampiran 1: Kode Program Code Vision AVR (lanjutan)

```
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer 1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=0x00;
TCCR1B=0x00;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

// Timer/Counter 2 initialization
ASSR=0x00;
TCCR2=0x00;
TCNT2=0x00;
OCR2=0x00;

// External Interrupt(s) initialization
// INT0: On
// INT0 Mode: Low level
// INT1: Off
// INT2: Off
GICR|=0x40;
MCUCR=0x00;
MCUCSR=0x00;
GIFR=0x40;

// Inisialisasi Interupsi Timer/Counter
TIMSK=0x00;

// Inisialisasi USART
// Parameters: 8 Data, 1 Stop, No Parity
// USART Receiver: Aktif
// USART Transmitter: Aktif
// USART Mode: Asynchronous
// USART Baud Rate: 9600 bps
UCSRA=0x00;
UCSRB=0xD8;
UCSRC=0x86;
UBRRH=0x00;
UBRRL=0x47;

// Inisialisasi Analog Comparator
ACSR=0x80;
SFIOR=0x00;

// Inisialisasi ADC
// ADC Clock frequency: 691.200 kHz
```

Lampiran 1: Kode Program Code Vision AVR (lanjutan)

```
// ADC Voltage Reference: AREF pin
// ADC High Speed Mode: Off
// ADC Auto Trigger Source: None
ADMUX=ADC_VREF_TYPE & 0xff;
ADCSRA=0x83;
SFIOR&=0xEF;

// LCD module initialization
lcd_init(16);
lcd_gotoxy(0,0);
lcd_putsf("Smart Controller");
lcd_gotoxy(0,1);
lcd_putsf(" oleh: Wisnu A. ");

// Semua Lampu Indikator Mati
STATHEATLO=1;
STATHEATHI=1;
STATFANLO=1;
STATFANHI=1;

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

while (1)
    {
    unsigned char Tampilan[33];
    unsigned int PV;
    // Terima nilai SV dari Server
    // melalui serial port
    stopAll=getchar();
    data_ser=getchar();
   // Baca sensor suhu dengan ADC
    data_adc=read_adc(0);

data_suhu=((float)data_adc/1023)*VRE
F*100;
    PV=(int)data_suhu;
    ftoa(data_suhu,0,sbuff);
    // Kirim nilai PV ke Server
    // melalui serial port
    printf("%s",sbuff);
    // Tampilkan data SV dan PV
    // di layar LCD
    lcd_gotoxy(0,0);
    lcd_putsf("Smart Controller");
    // Jika terjadi koneksi by wired
    if (stopAll==0){
    // Bandingkan nilai SV dengan PV
    // SV adalah nilai data_ser dan PV
adalah data_suhu
      if (data_ser > PV) {
         HEATER1=1; //Heater On
         HEATER2=1; //Putaran
Maksimal
         STATHEATLO=1;
         STATHEATHI=0;
```

Lampiran 1: Kode Program Code Vision AVR (lanjutan)

```
    FAN1=0; //Kipas1 Off
    FAN2=0; //Kipas2 Off
    STATFANLO=1;
    STATFANHI=1;
}
else if (data_ser < PV) {
    HEATER1=0; //Heater Off
    HEATER2=0;
    STATHEATLO=1;
    STATHEATHI=1;
    FAN1=1; //Kipas1 On
    FAN2=1; //Kipas2 On
    STATFANLO=1;
    STATFANHI=0;
}
else {
    HEATER1=1; //Heater Off
    HEATER2=0; //Putaran Minimal
    STATHEATLO=0;
    STATHEATHI=1;
    FAN1=1; //Kipas1 On
    FAN2=0; //Kipas2 Off
    STATFANLO=0;
    STATFANHI=1;
    };
    lcd_gotoxy(0,1);
    sprintf(Tampilan, "SV=%i\xdfC
PV=%s\xdfC",data_ser,sbuff);
    lcd_puts(Tampilan);
}
    else {
        // Hentikan Otomasi
        HEATER1=0;
        HEATER2=0;
        STATHEATLO=1;
        STATHEATHI=1;
        FAN1=0;
        FAN2=0;
        STATFANLO=1;
        STATFANHI=1;
        lcd_gotoxy(0,1);
        sprintf(Tampilan,
"SV=??\xdfC  PV=??\xdfC");
        lcd_puts(Tampilan);
    };
}
```

Lampiran 2: Kode Program ASP.net

```
<%@ Page Language="VB" AutoEventWireup="false"
CodeFile="halamanutama.aspx.vb" Inherits="halamanutama" %>


<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">


<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Halaman Utama</title>
    <style type="text/css">
        .style1
        {
            text-align: center;
        }
        .style2
        {
            font-family: Arial, Helvetica, sans-serif;
        }
        .style3
        {           width: 132px;
        }
        .style11
        {
            font-size: xx-large;
            font-weight: bold;
            font-family: sans-serif;
        }
```

```
        .style12
        {
            font-family: sans-serif;
        }
        .style15
        {
            font-weight: bold;
            text-align: center;
            width: 132px;
        }
        .style24
        {
            font-weight: bold;
            text-align: center;
            width: 138px;
        }
        .style25
        {
            width: 138px;
            text-align: center;
        }
    </style>
</head>
```

Lampiran 2: Kode Program ASP.net (lanjutan)

```
<body>
<form id="form1" runat="server">
   <div>
        <div>
        <asp:ScriptManager ID="ScriptManager1" runat="server">
        </asp:ScriptManager>
        <asp:UpdatePanel ID="UpdatePanel1" runat="server">
        </asp:UpdatePanel>
      </div>
      <hr />
      </div>
   <p class="style1">
      <b><span class="style2">SISTEM PEMANTAUAN DAN PENGENDALIAN
SUHU TANGKI<br />
      BERBASIS WEB SERVER MENGGUNAKAN MIKROKONTROLER
ATMEGA8535</span></b></p>
   <div>
      <hr />
   </div>
   <div>
      <p>
          </p>
      <p style="text-align: center">
            <b>Pilih Port: </b><asp:DropDownList ID="cboPort" runat="server">
            </asp:DropDownList>
          <asp:Button ID="btnRun" runat="server" Text="Start" />
         </p>
   </div>
```

Lampiran 2: Kode Program ASP.net (lanjutan)

```
<table align="center" border="1" style="width: 267px; height: 49px">
  <tr>
    <td class="style24">Setting Value</td>
    <td class="style15"> Present Value</td>
  </tr>
  <tr>
    <td class="style25">
      <asp:UpdatePanel ID="UpdatePanel3" runat="server">
        <ContentTemplate>
          <span class="style12">
          <asp:Label ID="lblSV" runat="server"
          style="font-weight: 700; font-size: xx-large; font-family: sans-serif"
          Text="00"></asp:Label>
          <span class="style11">°C<br />
          <asp:Button ID="btnDown" runat="server" Text="-" Width="26px" />
          <asp:Button ID="btnUp" runat="server" Text="+" Width="26px" />
          </span></span>
        </ContentTemplate>
      </asp:UpdatePanel>
    </td>
    <td class="style3">
      <asp:UpdatePanel ID="UpdatePanel2" runat="server">
        <ContentTemplate>
          <div class="style1">
            <asp:Label ID="lblPV" runat="server"
              style="font-family: sans-serif; font-size: xx-large; font-weight: 700"
              Text="??"></asp:Label>
```

Lampiran 2: Kode Program ASP.net (lanjutan)

```
                    <span class="style11">°C</span><asp:Timer ID="Timer1"
runat="server"
                      Interval="1000" OnTick="Timer1_Tick" Enabled="False">
                    </asp:Timer>
                    <br />
                    <asp:Label ID="lblOff" runat="server" Text="21"
Visible="False"></asp:Label>
                    <br />
                  </div>
                </ContentTemplate>
              </asp:UpdatePanel>
            </td>
          </tr>
        </table>
<div>
  <p>
     </p>
  <hr />
</div>
    </form>
</body>
</html>
```

Lampiran 3: Kode Program VB.net

```
Imports System.IO.Ports

Partial Class halamanutama

    Inherits System.Web.UI.Page

    Dim WithEvents serialPort1 As New SerialPort

    Dim comPorts As Array   'Com ports

    Dim rxBuff As String    'Buffer untuk terima data

    Dim nilaiSV As Integer

    Dim statusSend As String

    Dim dataSend As String


    Protected Sub Page_Init(ByVal sender As Object, ByVal e As System.EventArgs)
Handles Me.Init

        'Inisialisasi Variabel

        nilaiSV = 38

        lblSV.Text = nilaiSV

    End Sub


    Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs)
Handles Me.Load

        'When the form loads

        'Enumerate available Com ports and add to ComboBox1

        comPorts = IO.Ports.SerialPort.GetPortNames()

        'Hapus CboPort sebelum mulai

        cboPort.Items.Clear()

        For i = 0 To UBound(comPorts)

            cboPort.Items.Add(comPorts(i))

        Next
```

Lampiran 3: Kode Program VB.net (lanjutan)

```
        'Set text ComboBox1 di urutan pertama
        cboPort.SelectedIndex = 0
        'Set nama SerialPort1 di urutan pertama
        serialPort1.PortName = cboPort.SelectedItem.Text
        'Set atribut serial port
        serialPort1.BaudRate = 9600
        serialPort1.Parity = IO.Ports.Parity.None
        serialPort1.StopBits = IO.Ports.StopBits.One
        serialPort1.DataBits = 8
    End Sub

    Protected Sub cboPort_SelectedIndexChanged(ByVal sender As Object, ByVal e As
System.EventArgs) Handles cboPort.SelectedIndexChanged
        If serialPort1.IsOpen = False Then
            serialPort1.PortName = cboPort.SelectedItem.Text
        Else : MsgBox("Operation only valid when port is closed.",
MsgBoxStyle.Exclamation, "Error")
        End If
    End Sub

    Protected Sub btnUp_Click(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnUp.Click
        If Not ViewState("nilaiSV") Is Nothing Then
            nilaiSV = CInt(ViewState("nilaiSV"))
        End If
        nilaiSV += 1
        ViewState("nilaiSV") = nilaiSV
        lblSV.Text = nilaiSV
```

```
        btnDown.Enabled = True
        If nilaiSV = 60 Then
            btnUp.Enabled = False
        End If
    End Sub

    Protected Sub btnDown_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles btnDown.Click
        If Not ViewState("nilaiSV") Is Nothing Then
            nilaiSV = CInt(ViewState("nilaiSV"))
        End If
        nilaiSV -= 1
        ViewState("nilaiSV") = nilaiSV
        lblSV.Text = nilaiSV
        btnUp.Enabled = True
        If nilaiSV = 32 Then
            btnDown.Enabled = False
        End If
    End Sub

    Protected Sub btnRun_Click(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnRun.Click
        If btnRun.Text = "Start" Then
            btnRun.Text = "Stop"
            statusSend = Chr("0")
            CommPort()
            Timer1.Enabled = True
        Else
```

Lampiran 3: Kode Program VB.net (lanjutan)

```
            btnRun.Text = "Start"
            statusSend = Chr("1")
            Timer1.Enabled = False
            CommPortB()
        End If
    End Sub

    Protected Sub Timer1_Tick(ByVal sender As Object, ByVal e As System.EventArgs)
Handles Timer1.Tick
        CommPort()
    End Sub

    Private Sub SerialPort1_DataReceived(ByVal sender As Object, ByVal e As
System.IO.Ports.SerialDataReceivedEventArgs) Handles serialPort1.DataReceived
        'Pause selama pembacaan data
        System.Threading.Thread.Sleep(300)
        Try
            'Move recieved data into the buffer
            rxBuff = (serialPort1.ReadExisting)
            'If the buffer is still empty then no data. End sub
            If rxBuff = "" Then GoTo ends
            'Else display the recieved data in the RichTextBox
            lblPV.Text = rxBuff
        Catch ex As Exception
            'MsgBox(ex.Message)
        End Try
ends:
    End Sub
```

Lampiran 3: Kode Program VB.net (lanjutan)

```
    Private Sub CommPort()
        Try
            If serialPort1.IsOpen = False Then serialPort1.Open()
            'Tulis data ke port
             serialPort1.Write(statusSend)
            serialPort1.Write(Chr(lblSV.Text))
            'Pause for 800ms
            System.Threading.Thread.Sleep(800)

            If serialPort1.IsOpen = True Then serialPort1.Close()
        Catch ex As Exception
        End Try
    End Sub

    Private Sub CommPortB()
        Try
            If serialPort1.IsOpen = False Then serialPort1.Open()
            System.Threading.Thread.Sleep(50)
            serialPort1.Write(statusSend & Chr(lblOff.Text))
            System.Threading.Thread.Sleep(800)
            If serialPort1.IsOpen = True Then serialPort1.Close()
        Catch ex As Exception
        End Try
    End Sub
End Class
```