SKRIPSI

PERANCANGAN DAN PEMBUATAN ALAT PENGUKUR SUHU, KELEMBAPAN DAN TEKANAN UDARA BERBASIS PROTOKOL TCP/IP

Disusun Oleh :

RENALDO ARIAI PANAUHAN

NIM: 03.17.084

KONSENTRASI TEKNIK ELEKTRONIKA

JURUSAN TEKNIK ELEKTRO S-1

FAKULTAS TEKNOLOGI INDUSTRI

INSTITUT TEKNOLOGI NASIONAL MALANG

MARET 2008

LEMBAR PERSETUJUAN

PERANCANGAN DAN PEMBUATAN ALAT PENGUKUR SUHU, KELEMBAPAN DAN TEKANAN UDARA BERBASIS PROTOKOL TCP/IP

SKRIPSI

Disusun dan Diajukan Sebagai Salah Satu Syarat Untuk Memperoleh Gelar Sarjana Teknik Elektro Strata Satu (S-1)

Disusun Oleh :

RENALDO ARIAI PANAUHAN
Nim : 03.17.084

Diperiksa dan Disetujui

Dosen Pembimbing I	Dosen Pembimbing II
Ir. Sidik Noertjahjono,MT	**Dr. Cahyo Crysdian, MSc**
NIP.Y 1028700167	**NIP. 1030400412**

Mengetahui,
Ketua Jurusan Teknik Elektro S-1

Ir. F. Yudi Limpraptono, MT
NIP.Y 1039500274

KONSENTRASI TEKNIK ELEKTRONIKA
JURUSAN TEKNIK ELEKTRO S-1
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI NASIONAL MALANG
2008

INSTITUT TEKNOLOGI NASIONAL
FAKULTAS TEKNIK INDUSTRI
JURUSAN TEKNIK ELEKTRO

BERITA ACARA UJIAN SKRIPSI

FAKULTAS TEKNOLOGI INDUSTRI

Nama Mahasiswa	:	Renaldo Ariai Panauhan
NIM	:	0317084
Jurusan	:	Teknik Elektro S1
Konsentrasi	:	Teknik Elektronika
Judul Skripsi	:	Perencanaan dan Pembuatan Alat Pengukur Suhu, Kelembapan, dan Tekanan Udara Berbasis Protokol TCP/IP

Dipertahankan dihadapan tim penguji Skripsi jenjang Sarjana (S1) pada :

Hari : Senin

Tanggal : 17 Maret 2008

Dengan Nilai : 90,245 (A)

PANITIA UJIAN SKRIPSI

KETUA

Ir. Mochtar Asroni, MSME
NIP.Y.1018100036

SEKRETARIS

Ir. F. Yudi Limpraptono,MT
NIP.Y.1039500274

ANGGOTA PENGUJI

PENGUJI I

Ir. F. Yudi Limpraptono,MT
NIP.Y.1039500274

PENGUJI II

Ir. M. Abdul Hamid, MT
NIP.Y. 1018800188

ABSTRAKSI

Perancangan dan Pembuatan Alat Pengukur Suhu, Kelembapan dan Tekanan Udara Berbasis Protokol TCP/IP

Renaldo Ariai Panauhan

03.17.084

Jurusan teknik Elektronika – Institut Teknologi Nasional Malang

Jalan Raya Karanglo Km 2 Malang

r3n4ldo@linuxmail.org

Dosen Pembimbing : I. Ir. Sidik Noertjahjono, MT

II. Dr. Cahyo Crysdian, MSc

Kata Kunci :TCP/IP, Sensor, Mikrokontoller

Pengukuran Suhu, kelembapan, dan tekanan udara yang digunakan saat ini masih jarang yang menggunakan metode pengukuran jarak jauh dan harga perangkatnya pun mahal. Skripsi ini bertujuan untuk merancang dan membuat alat pengukur suhu, kelembapan dan tekanan udara dimana perangkatnya dapat diletakkan jauh dari kantor pusat dengan biaya yang lebih rendah dari perangkat yang ada saat ini.

Perangkat terdiri dari sensor tekanan udara MPX4100, sensor suhu dan kelembapan udara SHT11, mikrokontoller Atmega16, dan modul wiznet yang berfungsi untuk mengubah format data serial dari mikrokontoller ke format data TCP/IP. Sebagai media transmisi antara perangkat dengan komputer bisa menggunakan kabel atau gelombang elektromagnetik.

Menurut data sheet MPX4100 memiliki error maksimum sebesar 1,8%, SHT11 memiliki error sebesar 0.4% untuk pengukuran suhu dan error sebesar 3% untuk pengukuran kelembapan sehingga sensor-sensor tersebut baik untuk untuk digunakan. Penggunaan Penggunaan protokol TCP/IP memungkinkan perangkat diakses dari internet. Pada komputer data tidak hanya ditampilkan tetapi disimapan untuk keperluan analisa.

KATA PENGANTAR

Puji syukur kehadirat Tuhan Yang Maha Esa karena senantiasa memberikan karunia-Nya sehingga penyusun dapat menyelesaikan skripsi yang berjudul:

"Perancangan dan Pembuatan Alat Pengukur Suhu, Kelembapan dan Tekanan Udara Berbasis Protokol TCP/IP"

Pembuatan Skripsi ini disusun guna memenuhi syarat akhir kelulusan pendidikan jenjang Strata-1 di Institut Teknologi Nasional Malang. Laporan Skripsi ini merupakan tanggung jawab tertulis atas ilmu pengetahuan yang didapat selama penyusun mengikuti kuliah.

Atas terselesaikannya Skripsi ini, penulis mengucapkan terima kasih kepada :

- Bapak Ir. Sidik Noertjahjono, MT dan Bapak Dr. Cahyo Crysdian, MSc selaku Dosen Pembimbing yang telah memberikan bimbingan, pengarahan, serta ilmu-ilmu yang sangat berharga sehingga Skripsi ini dapat terselesaikan.
- Bapak Dr. Ir. Abraham Lomi, MSEE selaku Rektor Institut Teknologi Nasional Malang.
- Bapak Ir. Mochtar Asroni, MSME selaku Dekan Fakultas Teknologi Industri Institut Teknologi Nasional Malang
- Bapak Ir. Yudi Limpraptono, MT selaku Ketua Jurusan Teknik Elektro S1 / Elektronika.
- Teman-teman yang telah membantu dalam penyelesaian Skripsi ini.

Penulis menyadari bahwa laporan ini masih banyak yang perlu disempurnakan. Oleh sebab itu kritik dan saran yang membangun sangat diharapkan.

Akhir kata, penulis mohan maaf kepada semua pihak bilamana selama penyusunan Skripsi ini penyusun membuat kesalahan secara tidak sengaja dan semoga Skripsi ini dapat bermanfaat bagi kita semua.

Malang, Maret 2008

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1 Latar Belakang

Kemampuan kita dalam mengamati dan mengobservasi keadaan cuaca merupakan hal yang penting, hal ini karena keadaan cuaca dapat mempengaruhi berbagai kegiatan yang dilakukan oleh manusia. Dengan memiliki data-data cuaca yang cukup akurat akan dapat mencegah dan meminimalkan kerusakan yang diakibatkan perubahan cuaca. Data-data cuaca atau iklim digunakan pada berbagai bidang dalam kehidupan, pada bidang pertanian kita dapat menentukan komoditas apa yang cocok untuk dikembangkan pada daerah dengan keadaan cuaca tertentu, pada bidang konstruksi para perancang dapat menentukan rancangan bangunan yang tepat agar dapat bertahan pada perubahan cuaca yang ekstrim, bahkan sampai pada bidang perdagangan dengan melihat kondisi cuaca para pemasar dapat menentukan produk apa yang cocok agar nilai penjualannya meningkat.

Pada kesempatan ini penulis akan melakukan perancangan dan pembuatan alat pengukur suhu, kelembapan dan tekanan udara berbasis protokol TCP/IP. TCP/IP merupakan kumpulan dari beberapa protokol komunikasi data yang digunkanan untuk menghubungkan satu *host* dengan yang lain pada suatu jaringan lokal atau internet.

Pemakaian dari perangkat ini nantinya dapat digunakan didalam atau diluar ruangan. Contoh aplikasi pemakaian dalam ruangan adalah pada gudang penyimpanan kertas, penyimpanan tembakau, dan percetakan yang membutuhkan suhu dan kelembapan tertentu agar tidak merusak bahan produksi. Untuk pemakaian

diluar ruangan apabila digabungkan dengan perangakat pengamatan cuaca yang lain seperti penakar hujan dan pengukur kecepatan angin dapat digunakan dalam proses peramalan cuaca.

1.2 Tujuan

Tujuan pembuatan alat ini adalah perancangan sebuah sistem pengukuran suhu, kelembapan, dan tekanan udara yang data hasil pengukurannya dapa didistribusikan dengan memanfaatkan protokol komunikasi TCP/IP.

1.3 Rumusan Masalah

Dalam perencanaan dan pembuatan stasiun cuaca dapat dirumuskan beberapa masalah yang akan dibahas :

1. Bagaimana melakukan komunikasi dengan modul sensor SHT11 untuk mendapatkan nilai suhu dan kelembapan udara. SHT11 adalah sensor suhu dan kelembapan yang keluarannya merupakan data digital dan sudah terkalibrasi.
2. Bagaimana merancang rangakaian pengkondisi sinyal sensor tekanan udara
3. Bagaimana melakukan komunikasi antara Mikrokontroller dengan modul Wiznet EG-SR-7150 MJ. Wiznet EG-SR-7150 MJ merupakan perangkat yang berfungsi merubah format data serial menjadi format data TCP/IP.
4. Bagaimana melakukan komunikasi antara modul Wiznet EG-SR-7150 MJ dengan komputer.

1.4 Batasan Masalah

Agar pembahasan dari perancangan dan pembuatan perangkat tidak meluas maka perlu adanya pembatasan permasalahan yang meliputi :

1. Tidak membahas catu daya

2. Tidak membahas sistem operasi dan konfigurasi perangkat jaringannya
3. Membahas proses komunikasi serial mikrokontoller Atmega16
4. Membahas ADC mikrokontoller Atmega16
5. Membahas proses komunikasi serial *synchronous* Sensor SHT11

1.5 Metodologi Penulisan

Langkah-langkah yang diambil untuk penyelesaian perancangan dan pembuatan perangkat adalah adalah:

1. Studi literatur tentang iklim.
2. Studi literatur tentang perencanaan dan pembuatan perangkat
3. Perancangan dan pembuatan perangkat.
4. Pengujian terhadap peralatan serta pengukuran data hasil pemantauan peralatan.
5. Penyusunan laporan

1.6 Sistematika Penulisan

Sistematika pembahasan dari skripsi ini terdiri dari pokok pembahasan yang saling berkaitan antara satu dengan yang lain, yaitu :

BAB I PENDAHULUAN

Pada bab ini dibahas tentang latar belakang,tujuan, permasalahan, batasan masalah,metodologi dan sistematika penulisan.

BAB II LANDASAN TEORI

Pada bab ini dibahas tentang teori-teori yang mendukung dalam perancangan dan pembuatan alat.

BAB III PERANCANGAN DAN PEMBUATAN ALAT

Pada bab ini dibahas tentang perancangan dan pembuatan keseluruhan sistem perangkat keras dan sistem perangkat lunak.

BAB IV PENGUJIAN DAN ANALISA

Pada bab ini dibahas tentang proses serta hasil dari pengujian alat, yang didasarkan oleh pengukuran-pengukuran yang diperlukan.

BAB V PENUTUP

Pada bab ini akan disampaikan kesimpulan dari perancangan dan pembuatan sistem ini.

BAB II

LANDASAN TEORI

2.1 Unsur Iklim

Bagian ini akan menyajikan beberapa unsur iklim yang dibahas dalam pembuatan tugas akhir penulis, yaitu suhu , kelembapan, dan tekanan udara.

2.1.1 Suhu Udara

Suhu udara yang diukur dengan termometer air raksa atau perangkat semikonduktor, merupakan unsur iklim yang sangat penting. Suhu adalah unsur iklim yang sulit didefinisikan. Tapi secara fisis suhu udara dapat didefinisikan sebagai tingkat gerakan molekul benda, makin cepat gerakan molekul, makin tinggi suhunya. Suhu dapat juga didefinisikan sebagai tingkat panas suatu benda. Panas bergerak dari sebuah benda yang mempunyai suhu tinggi ke benda dengan suhu rendah. Untuk menyatakan suhu udara dipakai berbagai skala. Dua skala yang sering dipakai adalah skala Fahrenheit dan skala Celcius.

2.1.2 kelembapan Udara

Besaran yang sering dipakai untuk menyatakan kelembapan udara adalah kelembapan nisbi (RH) yang didapat diukur oleh psikometer, hygrometer, dan sensor kelembapan udara.

Kelembapan udara dapat di definisikan dengan persamaan dibawah :

$$RH = \frac{r}{rs} \times 100\,\%$$

dimana :

$$r \approx \epsilon \frac{e}{p} \quad , \quad rs \approx \epsilon \frac{es}{p}$$

r = rasio percampuran, yaitu perbandingan antara massa uap air dengan massa udara kering.

rs = rasio percampuran jenuh, yaitu perbandingan antara massa uap air jenuh dengan massa udara kering

e = tekanan parsial uap air

es = tekanan uap air jenuh

p = tekanan udara

$\epsilon \approx 0.623$

2.1.3 Tekanan Udara

Berat kolom udara persatuan luas diatas sebuah titik menunjukan tekanan udara pada titik tersebut. Dipermukaan laut tekanan udara adalah 101,32 kPa atau 1.013,2 mb. Distribusi tekanan horizontal dinyatakan oleh isobar, yaitu garis yang menghubungkan tempat yang mempunyai tekanan udara sama pada ketinggian tertentu. Tekanan udara berubah sesuai tempat dan waktu.

Karena atmosfer mengikuti hukum gas dan atmosfer bersifat mampat, maka massa jenis atmosfer paling besar berada dilapisan bawah karena pada atmosfer ini tertekan oleh massa udara diatasnya. Tekanan udara selalu berkurang dengan bertambahnya ketinggian .

2.2 Mikrokontroller ATmega16

AVR ATmega16 adalah mikrokontroler 8-bit CMOS, *low-power* yang berdasarkan pada bentuk arsitektur AVR RISC (*Reduced Instruction Set Computer*), yang hampir semua instruksinya selesai dikerjakan dalam satu siklus *clock*. AVR

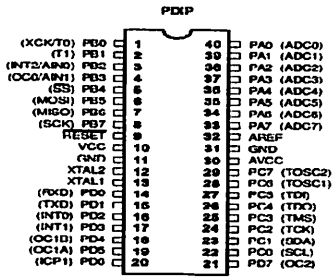
ATmega16 menggunakan instruksi tunggal (*Single Clock Cycle*), yaitu sistem mikrokontroler yang frekuensi kerja dalam chip sama dengan frekuensi kristal untuk osilator tanpa memerlukan rangkaian pembagi frekuensi setelah osilator yang diperlukan untuk memperoleh perbedaan fase dari *clock*, sehingga AVR 12 kali lebih cepat dibanding MCS51.

Berbagai karakteristik yang tersedia dalam IC ATmega16 adalah sebagai berikut:

- *In-Sistem Programable Flash* sebesar 16 KB
- EEPROM (*Electrical Erasable Programable Read Only Memory*) sebesar 512 byte
- SRAM (*Static Random Access Memory*) sebesar 512 byte
- Jalur I/O *general-purposes* sebanyak 32
- *Timer/ Counter* yang fleksibel dengan *mode* pembandin
- *Serial Port* SPI (*Serial Peripheral Interface*)

2.2.1 Konfigurasi Kaki Mikrokontroller ATmega16

Mikrokontroller ATmega16 mempunyai 40 pin seperti pada gambar 2.1



***Gambar 2.1** Pin Atmega16*

Fungsi tiap kaki adalah sebagai berikut:

1. VCC merupakan kaki IC yang berfungsi sebagai pin masukan catu data.
2. GND merupakan pin *ground.*
3. Port A (PA0...PA7) merupakan kaki IC I/O dua arah dan pin masukan ADC.
4. Port B (PB0...PB7) merupakan kaki IC I/O dua arah dan pin fungsi khusus, yaitu Timer/Counter dan SPI.
5. Port C (PC0...PC7) merupana kaki IC I/O dua arah dan pin fungsi khusus, yaitu TWI dan *Timer Oscilator*
6. Port D (PD0...PD7) merupakan pin I/O dua arah dan pin fungsi khusus, yaitu intrupsi eksternal dan komunikasi serial.
7. RESET merupakan pin yang digunakan untuk mereset mikrokontroler.
8. AVCC merupakan pin masukan tegangan untuk ADC.
9. AREF merupakan pin masukan tegangan referensi ADC
10. XTAL1 DAN XTAL2 merupakan pin masukan *clock* eksternal

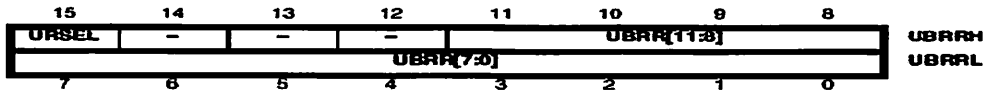
2.2.2 USART Mikrokontroller Atmega16

2.2.2.1 Inisialisasi USART

Dalam proses inisialisasi ada beberapa buah register yang perlu ditentukan nilainya yaitu:

1. UBRR (USART Baut Rate Register)
2. UCSRB (USART Control and Status Register B)
3. UCSRC (USART Control and Status Register C)

UBRR merupakan register 16 bit yang berfungsi melakukan penentuan kecepatan transmisi data yang digunakan. UBRR dibagi menjadi dua, yaitu UBRRH dan UBRRL seperti gambar berikut



Gambar 2.2 Register UBRR

Bit penyusunnya dapat dijelaskan sebagai berikut :

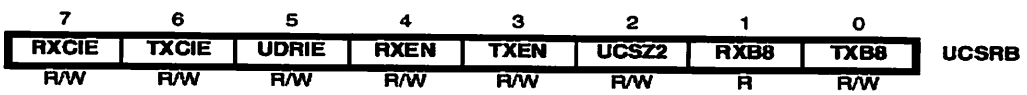
a) URSEL merupakan bit pemilih antara UBRR dan UCSRC. Hal itu disebabkan keduanya menempati lokasi yang sama. Untuk akses UBRR, bit ini bernilai 0.

b) UBRR [11...0] merupakan bit penyimpanan konstanta kecepatan komunikasi serial. UBRRH menyimpan 4 bit tertinggi data seting *baut rate* dan UBBRL menyimpan 8 bit sisanya. Data yang dimasukan ke UBRRH dan UBRRL dihitung menggunakan rumus sesuai Tabel 2.1. U2X merupakan bit pada register UCSRA.

***Tabel 2.1** Rumus Perhitungan Nilai UBRR untuk Berbagai Mode Operasi*

Mode Operasi	Nilai UBRR
Asinkrom mode kecepatan normal(U2X=0)	$UBRR = \frac{f_{osc}}{16 \times baut\ rate} - 1$
Asinkrom mode kecepatan ganda(U2X=1)	$UBRR = \frac{f_{osc}}{8 \times baut\ rate} - 1$
Sinkron	$UBRR = \frac{f_{osc}}{2 \times baut\ rate} - 1$

UCSRB merupakan register 8 bit pengatur aktivitas penerima dan pengirim USART. Komposisinya seperti gambar 2.4.



Gambar 2.3 Register UCSRB

Bit penyusunya dapat dijelaskan sebagai berikut :

a) RXCIE mengatur aktivitas intrupsi penerimaan data serial. Bernilao 0

sehingga proses penerimaan data berdasar pada sistem *pooling*. Jika bernilai 1 dan bit RXC pada UCSRA bernilai 1, interupsi penerimaan data serial akan dieksekusi.

b) TXCIE mengatur aktivitas interupsi penerimaan dta serial. Bernilai awal 0. Jika bernilai 1 dan jika bit TXC pada UCSRA bernilai 1, interupsi pengiriman data serial akan dieksekusi.

c) UDRIE mengatur aktivitas interupsi yang berhubungan dengan kondisi bit UDRE pada UCSRA. Bernilai awal 0. Jika bernilai 1, maka intrupsi akan terjadi hanya jika bit UDRE bernilai 1.

d) RXEN merupakan bit aktivasi penerima serial ATmega16. Bernilao awal 0. Jika bernilai 1, maka penerima data serial diaktifkan.

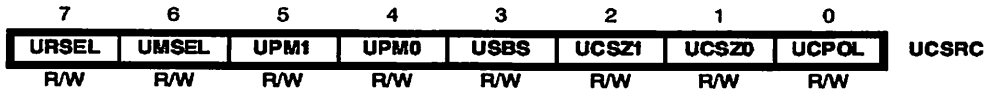
e) TXEN merupakan bit aktivasi pengirim serial Atmega16. Berniali awal 0. Jika bernilai 1, maka pengirim data serial diaktifkan.

f) UCSZ2 bersama dengan bit UCSZ1 dan UCSZ0 di register UCSRC menentukan ukuran karakter serial yang dikirimkan. Pada saat awal, ukuran karakter diset 8 bit. Detail nilai bit ini seperti Tabel 2.2. UCSRC merupakan register 8 bit yang dipergunakan untuk mengatur mode dan kecepatan komunikasi serial yang dilakukan.

***Tabel 2.2** Penentuan Ukuran Karakter*

UCSZ[2...0]	Ukuran Karakter dalam Bit
000	5
001	6
010	7
011	8
111	9

UCSRC merupakan register 8 bit yang dipergunakan untuk mengatur mode dan kecepatan komunikasi serial yang dilakukan. Komposisinya seperti Gambar 2.5.



Gambar 2.4 Register UCSRC

Bit penyusunnya dapat dijelaskan sebagai berikut:

a) URSEL merupakan bit pemilih akses antara UCSRC dan UBRR. Bernilai awal 1 sehingga secara normal akan selalu mengakses register UCSRC.

b) UMSEL merupakan bit pemilih mode komunikasi serial antara sinkrin dan asinkron. Bernilai awal 0 sehingga modenya asinkron. Jika bernilai 1, maka modenya sinkron.

c) UPM[1...0] merupakan bit pengatur paritas. Bernilai awal 00 sehingga paritas tidak dipergunakan.

d) USBS merupakan bit pemilih unuran bit stop. Bernilai awal 0 sehingga jumlah bit stop yaitu 2 bit.

e) UCSZ1 dan UCSZ0 merupakan bit pengatur jumlah karakter serial.

f) UCPOL merupakan bit pengatur hubungan antara perubahan data keluaran dan data masukan serial dengan clock sinkronisasi. Hanya berlaku untuk mode asinkron, bit ini diset 0.

2.2.2.2 Pengiriman Data

Proses pengiriman data serial dilakukan per byte data dengan menunggu register UDR yang merupakan tempat data serial akan disimpan menjadi kosong sehingga siap ditulis dengan data yang baru. Proses tersebut menggunakan bit yang ada pada register UCSRA, yaitu bit UDRE (*USART Data Register Empty*). Bit UDRE merupakan indikator kondisi register UDR. Jika UDRE bernilai 1, maka register UDR telah kosong dan siap diisi dengan data yang baru.

2.2.2.3 Penerimaan Data

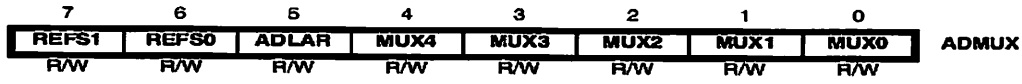
Proses penerimaan data serial dilakukan dengan mengecek nilai RXC (*USART Receive Complete*) pada regiser UCSRA. RXC akan benilai satu jika ada data yang siap dibaca di *buffer* penerima, dan bernilai nol jika tidak ada data pada buffer penerima. Jika penerima USART dinonaktifkan, maka bit akan selalu bernilai nol.

2.2.3 ADC Mikrokontroller Atmega16

2.2.3.1 Inisialisasi ADC

Proses inisialisasi ADC meliputi proses penentuan clock, teganga referensi, format output data, dan mode pembacaan. Register yang perlu diset nilainya adalah ADMUX (*ADC Multiplexer Selection Register*), dan ADCSRA (*ADC Control Status Register*).

ADMUX merupakan register 8 bit yang berfungsi menentukan tegangan referensi ADC, format data output, dan saluran ADC yang digunakan. Konfigurasi seperti gambar di bawah ini.



***Gambar 2.5** Register ADMUX*

Bit penyusunya dapat dijelaskan sebagai berikut :

a) REF[1...0] merupakan bit pengatur tegangan referensi ADC ATmega16. Memiliki nilai awal 00 sehingga referensi tegangan berasal dari AREF. Detail nilai yang lain terdapat pada tabel dibawah ini.

***Tabel 2.3** Pemilihan Mode Tegangan Referensi ADC*

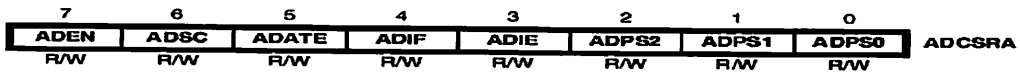
REF[1...0]	Mode tegangan referensi
00	Berasal dari pin AREF

01	Bersal dari AVCC
10	Tidak Dipergunakan
11	Berasal dari tegangan referensi ineternal 2,56 V

b) ADLAR merupakan bit pemilih mode data keluaran ADC. Bernilai awal 0 sehingga 2 bit tertinggi data hasil konversinya berada diregister ADCH dan sisanya terdapat pada ADCL.

c) MUX[4...0] merupakan bit pemilih saluran pembacaan ADC. Bernilai awal 0000. Untuk mode *single ended* input, MUX[4...0] bernilai dari 00000-00001.

ADCSRA merupakan register 8 bit yang berfungsi melakukan menajemen sinyal kontrol dan status dari ADC. Memiliki susunan seperti gambar dibawah ini.



***Gambar 2.6** Register ADCSRA*

Bit Penyusunnya dapat dijelaskan sebagai berikut :

a) ADEN merupakan bit pengatur aktivasi ADC. Bernilai awal 0. Jika bernilai 1, maka ADC aktif.

b) ADCS merupakan bit penanda mulainya konversi ADC. Bernilai awal 0 selama konversi ADC akan bernilai 1, sedangkan jika konversi telah selesai akan bernilai 0.

c) ADATE merupakan bit pengatur aktivitas picu otomatis operasi ADC. Bernilai awal 0. Jika bernilai 1, operasi konversi ADC akan dimulai pada saat transisi positif dari sinyal picu yang dipilih. Pemilihan sinyal picu menggunakan bit ADTS pada register SFIOR.

d) ADIF merupakan bit penanda akhir suatu konversi ADC. Bernilai awal 0. Jika bernilai 1, maka konversi ADC pada suatu saluran telah selesai dan data

siap diakses.

e) ADIE merupakan bit pengatur aktivasi interupsi yang berhubungan dengan konversi ADC. Bernilai awal 0. Jika bernilai 1 dan jika sebuah konversi ADC telah selesai, sebuah interupsi akan dieksekusi.

f) ADPS[2...0] merupakan bit pengatur clock ADC.

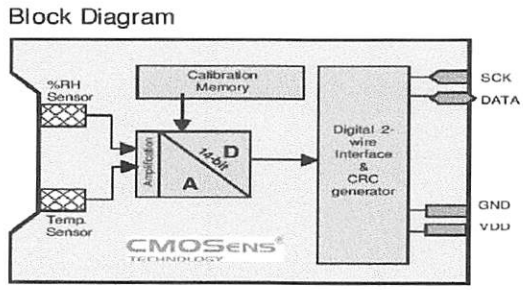
2.2.3.2 Pembacaan ADC

Dalam proses pembacaan hasil konversi ADC, dilakukan pengecekan terhadap bit ADIF (*ADC Interupt Flag*) pada register ADCRA. ADIF akan bernilai satu jika konversi sebuah saluran ADC telah selesai dilakukan dan data hasil konversi siap untuk diambil, dan demikian sebaliknya. Data disimpan dalam dua buah register, yaitu ADCH dan ADCL.

2.3 Sensor Suhu dan Kelembapan Udara SHT11

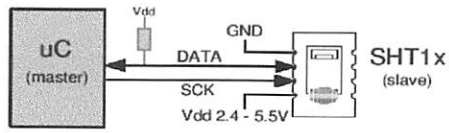
SHT11 merupakan chip tunggal untuk mengukur kelembapan dan suhu dengan keluaran digital yang telah terkalibrasi. Sensor ini menggunakan polimer kapasitif sebagai sensor kelembapan dan sensor suhu semikonduktor yang serupa dengan LM35. Keduanya kemudian diproses oleh ADC 12 bit dan 14 bit sehingga sinyal yang dihasilkan bebas dari ganguan luar dan proses pengambilan data yang cepat serta dengan akurasi yang tinggi. Disamping itu sensor ini memiliki dimensi yang kecil (7,47 mm x 4,93 mm) dan hemat dalam pemakaian energi.

Sensor SHT11 biasanya digunakan pada stasiun cuaca, monitoring gudang penyimpanan, dan bidang otomotif. Fungsionalitas SHT11 dapat dilihat pada blok diagram dibawah.



Gambar 2.7 *Blok Diagram SHT11*

2.3.1 Konfigurasi Pin SHT11



Gambar 2.8 *Konfigurasi Pin*

Fungsi tiap pin :

a) GND sebagai *ground*

b) DATA digunakan untuk melakukan transfer informasi antara MCU dengan SHT11, perubahan data terjadi pada saat sinyal clock beralih dari satu(*high*) ke nol(*low*), data valid berada pada saat sinyal clock berubah dari nol(*low*) ke satu(*high*). Pada saat proses transmisi data stabil harus berada pada saat SCK high.

c) SCK digunakan untuk mensinkronkan komunikasi antara MCU dengan SHT11, Karena fungsionalitas antarmuka sudah full digital maka tidak ada batasan untuk frekuensi untuk SCK.

d) Vdd sebagai masukan tegangan yang nilainya antara 2.4 sampai 5 V

2.3.2 Komunikasi Serial SHT11

Komunikasi serial pada SHT11 adalah komunikasi sinkron yang aturan dan format datanya ditentukan oleh perusahan pembuat sensor tersebut.

2.3.2.1 Mengirim Perintah

Gambar 2.9 Kondisi Start

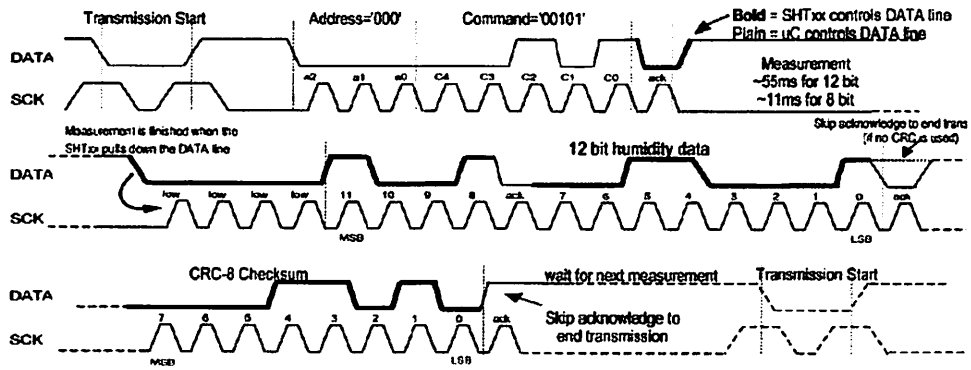
Untuk memulai proses transmisi, terlebih dahulu kita harus membuat kondisi Start. Start kondisi terdiri dari, data dalam keadaan low pada jalur DATA, Selama SCK high, diikuti pulsa pada SCK low dan naik kembali dan SCK dalam keadaan high terus.Baris perintah terdiri dari 3bit alamat dan 5bit perintah, bit ke9 adalah ACK.

Tabel 2.4 Daftar Perintah SHT11

Command	Kode
Mengukur suhu	00011
Mengukur kelembapan	00101
Membaca register status	00111
Menulis register status	00110
Soft Reset	11110

2.3.2.2 Mengukur Suhu dan Kelembapan

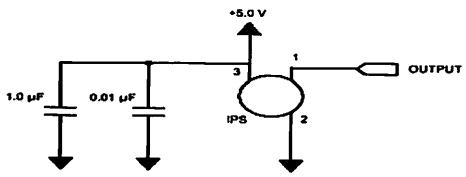
Setelah mengirimkan Perintah untuk melakukan pengukuran('00000101' untuk RH, atau 00000011 untuk Suhu) MCU menunggu sampai proses pengukuran selesai. Kira-kira 11/55/210 ms untuk pengukuran 8/12/14bit. Waktu tepat untuk proses pengukuran berpariasi sampai +-15% kecepatan osilator internal. Untuk sinyal penandaan bahwa pengukuran sudah selesai, SHT11 menjadi Low dan memasuki keadaan idle. MCU harus menunggu sampai sinyal data siap sudah diterima, Data hasil pengukuran akan disimpan sampai ada proses pembacaan. 2 byte data hasil pengukuran dan 1 byte CRC akan ditransmisikan. MCU harus merespon setiap byte dengan me-low-kan jalut DATA.



Gambar 2.10 Proses Pembacaan Kelembapan

2.4 Sensor Tekanan Udara MPX4100

Sensor ini digunakan untuk mengukur tekanan udara, Sensor ini memiliki sensitifitas 54mV/kPa. Keluran dari sensor ini kemudian dimasukkan ke ADC internal dari AVR, keluaran yang dihasilkan linear pada range 20kPa – 105kPa, Konfigurasi untuk sensor ini juga tidak terlalu rumit seperti yang ditunjukan pada gambar.



Gambar 2.11 Rangkaian Sensor MPX4100

Transfer fungsi untuk sensor MPX4100 adalah :

$$P=\frac{((\frac{(V_{MPX}+error)}{V_s})+0.1518)}{0.01059}$$

error = Pressure Error x Temperature Error x 0.01059 x Vs
Vs = 4,94 V
Pressure Error = 1,5
Temperature Error Factor =1
error = 1,5 x 1 x 0.01059 x 4,94 =0.0784719

2.5 Protokol TCP/IP

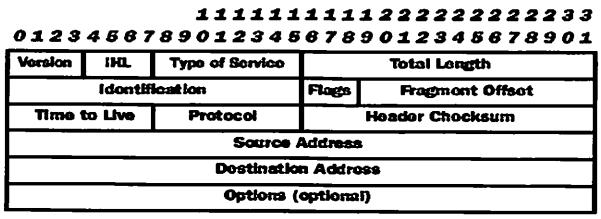
TCP/IP terdiri atas sekumpulan protokol yang masing-masing bertanggung jawab atas bagian-bagian tertentu dari komunikasi data. TCP/IP menjadi protokol komunikasi data yang fleksibel, dapat diterapkan dengan modah disetiap jenis komputer dan interface jaringan. Kalau terjadi perubahan, hanya perlu perubahan pada protokol yang berhubungan dengan interface jaringan saja.

Pada protokol model TCP/IP standar, protokol dibagi menjadi 4 lapisan, yaitu *Application Layer, Host-to-host Layer, Internetworking Layer, Network Interface Layer.* Dalam TCP/IP, terjadi penyampaian data dari satu protokol ke protokol lainnya, setiap protokol memperlakukan semua informasi sebagai data, jika suatu protokol menerima data dari protokol lain diatasnya, maka protokol tersebut akan menambahkan informasi tambahan miliknya ke data tersebut, informasi ini memiliki fungsi yang sesuai dengan fungsi protokol tersebut, setelah itu data dikirim kembali ke bawahnya, peristiwa tersebut masuk dalam proses pengiriman data.

Fungsi dari masing-masing lapisan dapat dijelaskan sebagai berikut :

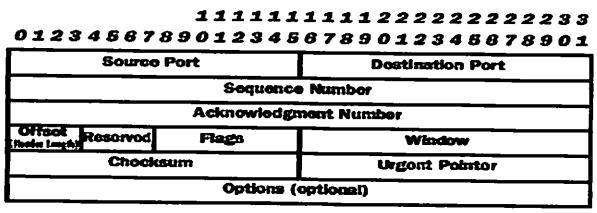
1. *Network Interface Layer* bertanggung jawab mengirim dan menerima data dari media fisik. Media fisik bisa berupa kabel, serat optik, atau gelombang radio, sehingga protokol ini harus mampu menerjemahkan sinyal listrik menjadi data digital yang dimengerti komputer, yang berasal dari peralatan lain.
2. *Internetworking Layer* bertanggung jawab dalam proses pengiriman paket ke alamat yang tepat. Pada lapisan ini terdapat tiga macam protokol, yaitu IP, ARP, dan ICMP. IP(*Internet Protocol*) berfungsi menyampaikan data kealamat yang tepat, ARP(*Address Resolution Protocol*) berfungsi sebagai

penentu alamat perangkat keras dari host yang terletak pada jaringan yang sama, dan ICMP (*Internet Control Message Protocol*) berfungsi untuk melaporkan kegagalan pengiriman data.



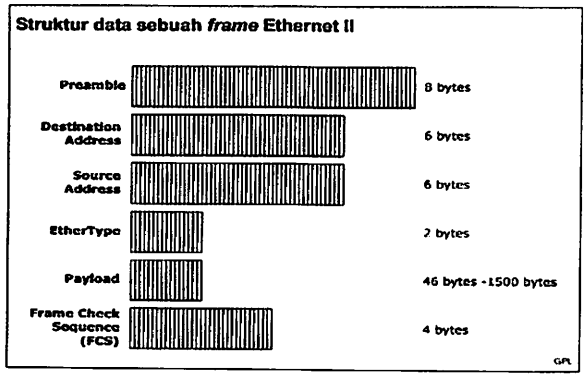
Gambar 2.12 Format Data IP

3. *Host-to-Host Layer* berisi protokol yang bertanggung jawab untuk mengadakan komunikasi antara dua host. protokol tersebut adalah TCP (*Transmission Control Protocol*) dan UDP (*User Datagram Protocol*). TCP merupakan protokol yang berorientasi pada hubungan yang handal yang mengijikan sebuah aliran data yang berasal dari satu mesin dikirimkan tanpa kesalahan ke mesin lainnya. TCP memecah aliran byte data menjadi pesan-pesan diskret dan meneruskannya ke *Internetworking Layer* . Proses TCP penerima merakit kembali pesan-pesan yang diterimanya menjadi aliran output. TCP juga menangani pengirim yang cepat tidak akan membanjiri penerima yang lambat. UDP merupakan protokol yang tidak andal dan tidak menjamin keneksi yang dilakukan tanpa kesalahan atau tidak.



***Gambar 2.12** Format Data TCP*

4. *Application Layer* lapisan ini bertanggung jawab dalam rangka menyediakan akses kepada aplikasi terhadap jaringan TCP/IP. Protokol-protokol yang berjalan pada lapisan ini adalah protokol *Dynamic Host Configuration Protocol* (DHCP), *Domain Name System* (DNS), *Hypertext Transfer Protocol* (HTTP), *File Transfer Protocol* (FTP), Telnet, *Simple Mail Transfer Protocol* (SMTP), *Simple Network Management Protocol* (SNMP), dan lain-lain.

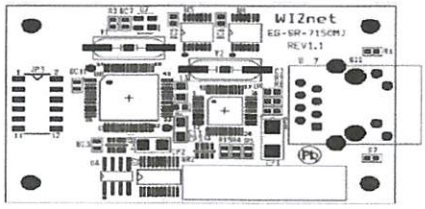


***Gambar 2.14** Format Data Ethernet*

2.6 Modul Wiznet EG-SR 7150MJ

Pada sistem, modul wiznet ini berfungasi untuk mengubah format data serial menjadi format data TCP/IP. Mode operasi yang menggunakan protokol TCP adalah sebagai TCP server dan TCP client. Apabila bekerja sebagai TCP server maka modul ini akan menunggu permintaan dari client dan setelah ada client terkoneksi maka terjadilah komunikasi antara keduanya. Apabila bekerja sebagai TCP client maka proses sebaliknya yang terjadi, yaitu modul akan melakukan permintaan sambungan dengan server.

Dengan penggunaan modul ini kita dengan mudah membuat aplikasi client/server dengan perangkat selain PC, seperti misalnya mikrokontoller, pembaca RFID, dan berbagai perangkat yang menggunakan jalur komunikasi serial.



Gambar 2.15 *Modul Wiznet EG-SR 7150MJ*

Tabel 2.5 *Spesifikasi Wiznet EG-SR 7150MJ*

Kategori	Spesifikasi
Dimensi	40mm x 62mm x 17mm
LAN interface	Konektor RJ-45, 10/100 Mbps Auto Sensing
protokol	TCP, UDP, IP, ARP, ICMP, MAC, (IGMP, PPPoE)
mikrokontoller	AT89C51RC2
Serial Interface	RS232 (LVTTL)
Sinyal Serial	TXD, RXD, CTS, RTS, GND
Parameter Serial	Parity : None, Even, Odd Data bit : 7,8 Flow Control : RTS/CTS, XON/XOF
Menajemen	Perangkat lunak konfigurasi pada Windows
Suhu	0 – 70 °C (Operasi), -40 – 80 °C (Penyimpanan)
Kelembapan	10-90%
Power	150mA pada 3,3V

BAB III

PERANCANGAN DAN PEMBUATAN ALAT

Bab ini akan membahas tentang perencanaan dan pembuatan alat yang meliputi perencanaan perangkat keras (*Hardware*) dan perangkat lunak (*Software*). Perancangan secara keseluruhan dapat dibagi menjadi dua bagian, yaitu :

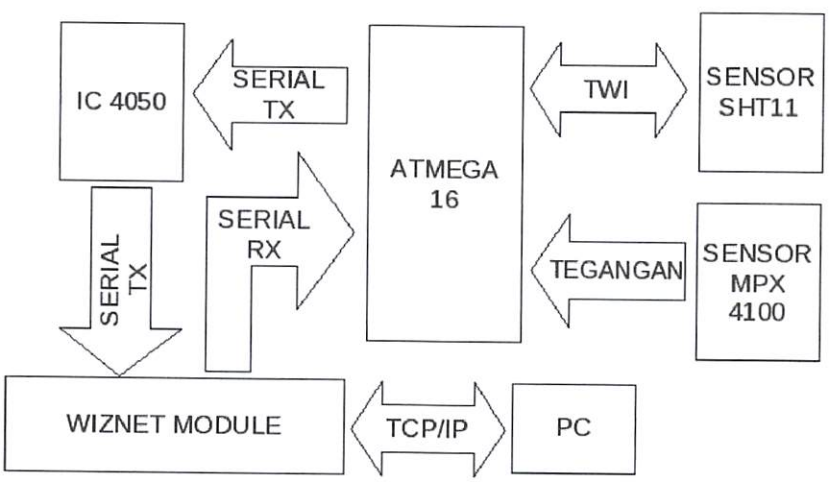
1. Perancangan Perangkat Keras
2. Perancangan Perangkat Lunak

Pada perancangan perangkat keras akan meliputi seluruh peripheral yang digunakan pada sistem ini. Pada perancangan perangkat lunak akan meliputi perancangan perangkat lunak pada mikrokontoller dan perangkat lunak pada PC.

3.1 Perancangan Perangkat Keras

3.1.1 Diagram Blok Sistem

Diagram blok akan memberikan gambaran umum sistem yang akan dirancang seperti yang ditunjukan pada gambar 3.1.



***Gambar 3.1** Diagram Blok Sistem*

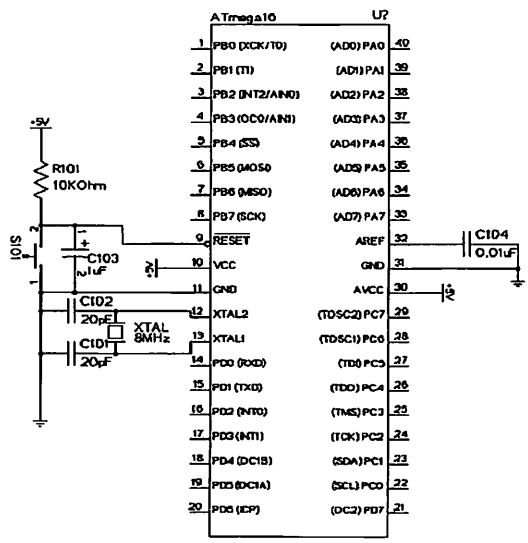
Fungsi tiap pada blok diagram adalah :

1. Mikrokontoller ATmega16 berfungsi untuk melakukan pengambilan data pada sensor SHT11 dan MPX4100 kemudian memproses data mentah hasil pengukuran, setelah data diproses akan dikirimkan melalui secara serial ke modul WIZNET EG-SR 7150 MJ.
2. SHT11 merupakan sensor yang berfungsi untuk mengukur suhu dan kelembapan.
3. MPX4100 merupakan sensor yang berfungsi mengukur tekanan udara, data keluaran dari sensor ini berupa tegangan. Keluaran sensor ini dimasukan ke pin ADC mikrokontoller.
4. WIZNET EG-SR 7150 MJ berfungsi merubah format data serial pada mikrokontoller ke format data TCP/IP.
5. IC CMOS 4050 berfungsi merubah level *UART* TX mikrokontoller yang bernilai 5V menjadi 3.3V agar dapat dibaca oleh modul wiznet.
6. PC berfungsi menampilkan dan mengumpulkan data hasil pengukuran.

3.1.2 Perancangan Rangkaian Mikrokontoller ATmega16

Mikrokontoller ATmega16 memegang peranan penting dalam sistem ini. Dimana mikrokontoller ini yang bertugas untuk melakukan pengambilan dan pemprosesan data dari kedua buah sensor dan mengirimkan data olahan yang sudah jadi dan mudah untuk dibaca.

Bagian yang cukup penting pada mikrokontoller adalah power suply. Pada ATmega16 terdapat berbagai variasi tegangan yang dapat dipakai tergantung dari seri ATmega16 yang digunakan, pada perancangan tegangan yang digunakan sebesar 5V. Gambar 3.2 merupakan rangkaian ATmega16.



***Gambar 3.2** Rangakaian ATmega16*

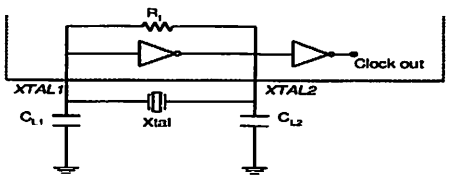
Dengan rangkaian yang ada pada gambar diatas ATmega16 sudah dapat bekerja melaksanakan fungsi-fungsi yang diinginkan oleh sistem. ATmega16 dapat bekerja tanpa ada rangkaian tambahan dan hanya menggunakan power suply saja, karena pada ATmega16 sudah ada kristal internal sebesar 1Mhz. Tetapi untuk keperluan pemprosesan yang lebih cepat kita memerlukan kristal eksternal sebesar 8Mhz dan rangkaian reset apabila terjadi error.

3.1.2.1 Osilator Kristal

Pada mikrokontoller ATMEL AVR dalam hal ini termasuk ATmega16 terdapat beberapa macam sumber pewaktu eksternal diantaranya resonator keramik dan resonator kristal. Pada perancangan kali ini penulis menggunakan resonator kristal sebagai sumber pewaktu. Hal ini disebabkan resonator kristal memiliki kelebihan yang lebih baik dari resonator keramik. Keuntungan pada resonator kristal adalah nilai Q atau faktor kualitas yang tinggi, stabilitas frequensi yang baik, dan kurang sensitif terhadap perubahan suhu.

Pada ATmega16 kristal dipasang secara paralel metode ini disebut paralel

resonant kristal. Pada resonator paralel dibutuhkan komponen reaktif kapasitor agar osilator dapat berosilasi dengan baik. Gambar dibawah menunjukan rangkaian ekuivalen osilator kristal pada ATmega16 untuk frequensi diatas 400KHz.



***Gambar 3.3** Rangkaian ekuivalent osilator kristal untuk frequensi diatas 400KHz*

Penambahan kapasitor C_{L1} dan C_{L2} pada rangkaian diperlukan untuk memberikan beban kapasitif pada osilator, tanpa adanya beban kapasitif osilator berosilasi secara tidak stabil. Tetapi pemasangan kapasitor ini perlu diperhatikan juga karena apabila menghasilkan beban kapasitif yang terlalu besar maka osilator akan kesulitan untuk memulai osilasi. Oleh sebab itu kita perlu untuk memperkirakan berapa nilai kapasitor yang sesuai untuk dipasang, persamaan dibawah dapat membantu untuk menentukan nilai kapasitor yang dipasang.

$$C_L=\frac{C_{L1}.C_{L2}}{C_{L1}+C_{L2}}+C_S$$

karena $C_{L1} = C_{L2}$ maka persamaan menjadi,

$$C_{L1}=C_{L2}=2\times(C_L-C_S)$$

Pada rangkaian osilator, nilai kapasitif stray C_S antara 5-10pF, ini merupakan nilai kapasitif yang disebabkan oleh PCB, dan C_L antara 16-21pF. Dari nilai-nilai estimasi kapasitor akan kita dapatkan perkiraan nilai kapasitor yang dipasang.

Kita gunakan nilai C_S = 10pF dan C_L = 20pF

$$C_{L1}=C_{L2}=2\times(C_L-C_S)$$

$$C_{L1}=C_{L2}=2\times(20\text{pF}-10\text{pF})$$

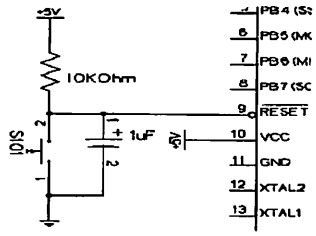
$$C_{L1}=C_{L2}=20\text{pF}$$

Nilai kapasitor ini sesuai dengan yang ditentukan oleh data sheet yaitu antara 12-22pF.

3.1.2.2 Rangakaian Reset

Reset pada ATmega16 dan keluarga AVR yang lain adalah aktif *low*, secara sederhana apabila kita hubungkan pin reset dengan ground maka ATmega16 akan mengalami proses reset.

Pada pin reset sebenarnya telah memiliki internal pull-resistor, sehingga secara sederhana rangkaian reset hanya terdiri dari sebuah push button saja untuk menghubungkan jalur ke ground tetapi apabila ada derau maka bisa terjadi reset yang tidak diinginkan. Oleh sebab itu kita memerlukan tambahan pull-up resistor. Disamping itu penambahan kapasitor untuk mengurangi derau, tetapi ini bersifat optional saja karena pada pin reset juga sudah terdapat LPF.

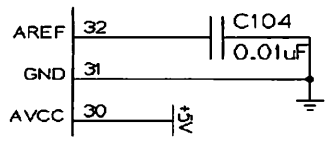


***Gambar 3.4** Rangkaian Reset*

3.1.2.3 Internal ADC ATmega16

Pada ATmega16 terdapat 8 chanel masukan ADC sehingga kita tidak memerlukan lagi IC ADC. Sama seperti IC ADC yang memerlukan tegangan referensi demikian juga pada ATmega16. Pada ATmega16 tegangnan referensi yang dipakai adalah 5V dan referensi internal 2,56V. Pada perancangan kali ini tegangan referensi yang dipakai adalah tegangan referensi internal 2,56V. Gambar dibawah

merupakan koneksi pin yang diperlukan agar ADC dapat berfungsi.



***Gambar 3.5** Rangakaian ADC Internal ATmega16*

Seperti dilihat pada gamabr pin GND dihubungkan ke ground dan AVCC dihubungkan ke Vcc. Pada AREF dipasang kapasitor untuk yang digunakan sebagai *noice canceler,* besarnya nilai kapasitor antara 0.1uF dan 0.01uF. Tegangan keluaran yang dihasilkan dapat dilihat pada persamaan dibawah.

$$Vout = \frac{((ADCH) \times Vref)}{255}$$

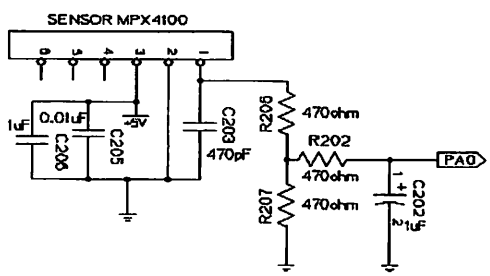
ADCH = register penyimpanan hasil konversi ADC

Vout = Tegangan Keluaran(V)

Vref = Tegangan Referensi(V)

3.1.3 Perancangan Rangkaian Sensor Tekanan Udara MPX4100

Sensor tekanan udara MPX4100 merupakan sensor yang digunakan untuk mengukur perubahan tekanan udara. Pada sensor ini keluaran sudah terkalibrasi sehingga relatif mudah untuk digunakan, fitur yang juga penting dari sensor ini adalah sudah terdapat unit kompensasi suhu sehingga menjamin keluaran yang linear. Gambar dibawah merupakan rengkaian sensor tekanan udara.



***Gambar 3.6** Rangakaian Sensor Tekanan Udara MPX4100*

Dengan rangkaian diatas maka data yang dilekuarkan siap untuk dimasukkan pada mikrokontoller. C206 dan C205 digunakan untuk mengurangi noise yang mungkin timbul pada tegangan supply.

Tegangan keluaran maksimum dari sensor adalah 4,9V, sedangkan tegangan maksimum yang diijinkan untuk masuk kedalam ADC mikrokontoller adalah 2,56V. Untuk mengatasi hal tersebut maka R207 dan R206 digunakan sebagai pembagi tegangan. Persamaan untuk pembagi tegangan dapat dijelaskan pada persamaan dibawah.

$$Vout = \frac{R207}{R207 + R06} \times Vin$$

Karena R207=R206=470ohm

$$Vout = \frac{1}{2} \times Vin$$

Vout = Tegangan keluaran setelah pembagi tegangan

Vin = Tegangan keluaran sensor

R202 dan C202 merupakan LPF (*Low Pass Filter*), menurut AN1646 sensor MPX4100 frequensi *cutt off* yang direkomendasikan adalah 650Hz. Sedangkan frequensi *cutt off* yang dihasilkan dapat dilihat pada persamaan dibawah.

$$f_C = \frac{1}{2 \Pi RC}$$

$$f_C = \frac{1}{2 \times 3{,}14 \times 470 \times 1.10^{-6}}$$

$$f_C = 338{,}79\ Hz$$

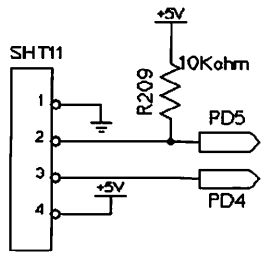
f_C = frequensi *cutt off*

Melihat hasil dari perhitungan diatas, rangkaian LPF menghasilkan frequensi *cutt off* yang lebih rendah dari yang telah direkomendasikan sehingga kemampuan untuk

menapis noise menjadi lebih baik dan memastikan tegangan DC yang masuk ke ADC.

3.1.4 Perancangan Rangkaian Sensor Suhu dan Kelembapan Udara SHT11

Perancangan rangkaian untuk sensor ini relatif mudah karena keluarannya sudah merupakan data digital sehingga hanya perlu menambahkan resistor pull-up saja pada pin data, hal ini dimaksudkan agar mikrokontoller hanya mendrive data low saja pada sensor ini. Untuk lebih jelas dapat dilihat pada gambar 3.7.



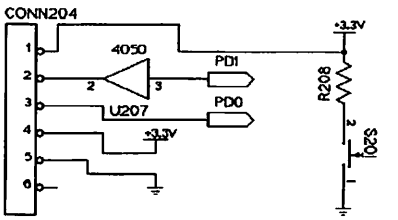
***Gambar 3.7** Rangakaian Sensor Suhu dan Kelembapan*

3.1.5 Perancangan Rangkaian Konverter Tegangan 5V ke 3,3V

Komunikasi antara Mikrokontoller dan modul wiznet adalah komunikasi serial . Rangkaian konverter tegangan 5V ke 3,3V diperlukan karena level high pada mikrokontoller 5V sedangkan pada modul wiznet untuk level high 3,3V. Menurut spesifikasi yang ada pada data sheet WIZNET EG-SR 7150 MJ, maksimum tegangan masukan adalah 3,6V.

Pada perancangan kali ini penulis menggunakan IC CMOS 4050 yang merupakan buffer tak membalik (*non inverting*). Pada data sheet IC ini terdapat keterangan $V_{DD} = 3V - 5V$, $V_{INPUT} = 0V - 15V$, $V_{OUT} = 0 - V_{DD}$. Dari data yang ada tersebut maka apabila kita menggunakan tegangan supply sebesar 3,3V maka keluarannya kondisi high adalah 3,3V juga walupun tegangan masukan sebesar 5V. Buffer ini hanya dipasang pada kaki Tx mikrokontoller saja sedangkan pada kaki

Rx tidak perlu karena tegangan 3,3V sudah dianggap kondisi high oleh mikrokontoller. Gambar 3.8 menunjukan skematik dari rangkaian konverter tegangan.



***Gambar 3.8** Rangkaian Konverter tegangan 5V ke 3,3V*

3.2 Perancangan Perangkat Lunak

Perancangan perangkat lunak akan dibagi menjadi dua bagian, yaitu perancangan perangkat lunak pada Mikrokontoller dan perancangan perangkat lunak pada PC. Pada bagian perancangan perangkat lunak ini tidak akan dijelaskan secara rinci tentang kode program yang ditulis, melaikan hanya alur programnya saja.

3.2.1 Perancangan Perangkat Lunak Pada Mikrokontoller

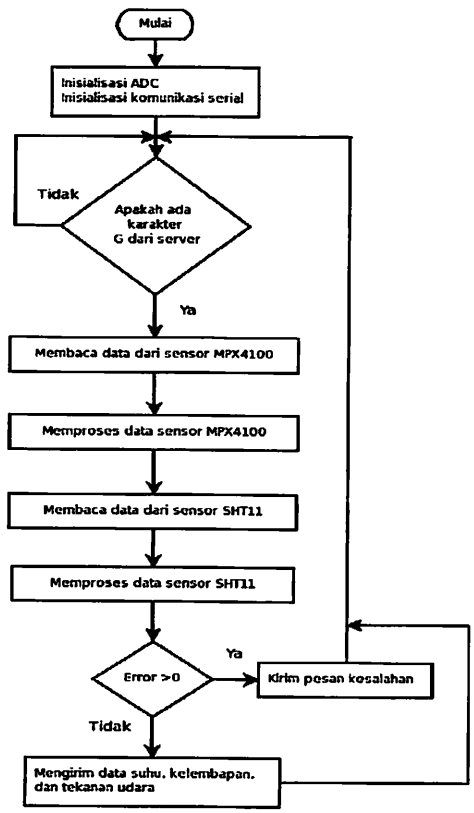
Pada mikrokontoller kita akan merancang perangkat lunak agar dapat melakukan pengambilan data mentah dari kedua buah sensor, memproses data pengukuran sensor agar menjadi besaran yang kita inginkan, dan mengirimkan semua besaran yang telah diukur melalui jalur komunikasi serial. Bahasa pemograman yang dipakai adalah C dan kompiler bahasa C yang digunakan adalah AVR-GCC.

Proses pengambilan data Sensor tekanan udara MPX4100 yang akan dilakukan oleh mikrokontoller dengan cara mengaktifkan ADC internalnya dan melakukan konversi data. Pengambilan data pada Sensor SHT11 sedikit lebih rumit karena kita akan membuat perangkat lunak yang dapat melakukan komunikasi serial

sinkron dengan sensor, dimana selain dapat berkomunikasi kita juga harus menjamin integritas data yang masuk dengan menggunakan metode CRC(Cyclic Redudancy Check).

Pada tahap pemrosesan data yang akan kita lakukan adalah melakukan perhitungan pada data yang masuk sesuai dengan ketentuan sensor-sensor tersebut. Pada pengiriman data kita akan mengaktifkan fungsi komunikasi serial pada mikrokontoller. Gambar 3.9 dibawah merupakan diagram alir program utama.

Pada diagram alir terdapat proses pengambilan keputusan terhadap adanya kesalahan, yang apabila Error>0 akan dikirimkan pesan kesalahan. Hal ini karena pada perancangan penulis menandai adanya kesalahan dengan membuat Error>0 dan apabila tidak terjadi kesalahan maka Error=0.



***Gambar 3.9** Diagram Alir Program Utama*

3.2.1.1 Pengambilan Data Pada Sensor Suhu dan Tekanan Udara SHT11

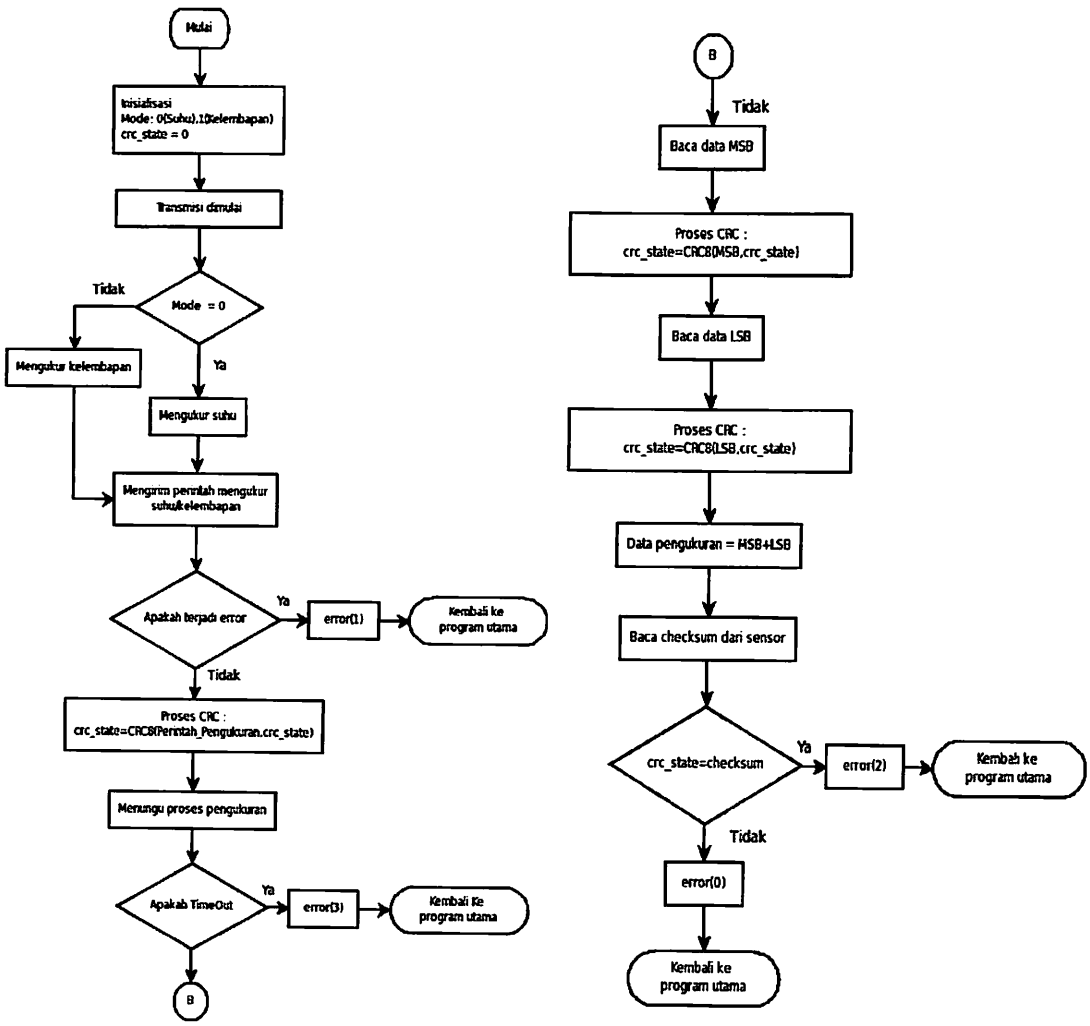
Data yang dapat diambil dari pengukuran sensor SHT11 adalah suhu dan kelembapan, karena data yang dikeluarkan oleh sensor ini sudah digital fokus kita adalah bagaimana mikrokontoller dapat berkomunikasi dengan sensor ini. Secara singkat proses pengambilan data dari sensor ini adalah :

1. Mikrokontoller harus memulai komunikasi dengan membuat jalur data dalam kondisi *start*, kondisi *start* pada sensor SHT11 sudah dijesakan pada BAB II.
2. Mengirimkan Perintah pengukuran suhu atau kelembapan udara.
3. Menunggu proses pengukuran selesai.
4. Menerima data suhu atau kelembapan.

Dengan mengikuti alur diatas kita sudah dapat melakukan pengambilan data suhu dan kelembapan. Ada keuntungan yang dapat kita peroleh dalam penggunaan sensor ini, selain karena tidak membutuhkan penaganan perangkat keras yang rumit, kita juga dapat melakukan pengukuran unsur iklim lain yaitu *Dew Point*(titik embun) secara tidak langsung dengan melakukan perhitungan dari data suhu dan kelembapan udara. Unsur iklim titik embun ini tidak akan dibahas pada penulisan tugas akhir ini tetapi akan tetap ditambahkan pada perangkat lunak. Diagram alir dapat dilihat pada gambar 3.10.

Seperti yang sudah dijelaskan pada bagian sebelumnya bahwa setiap kode *Error* akan menunjukan dimana letak kesalahan pada sistem terjadi. Pada diagram alir diperlihatkan letak *Error* tersebut pada program. Sebagai contoh apabila *Error*=3 maka terjadi T*imeout* atau data tidak tiba pada mikrokontoller sesuai dengan waktu yang telah ditetapkan. Dari diagram alir juga dapat dilihat bahwa proses pengambilan data suhu dan kelembapan udara tidak dapat dilaukan secara bersamaan. Resolusi

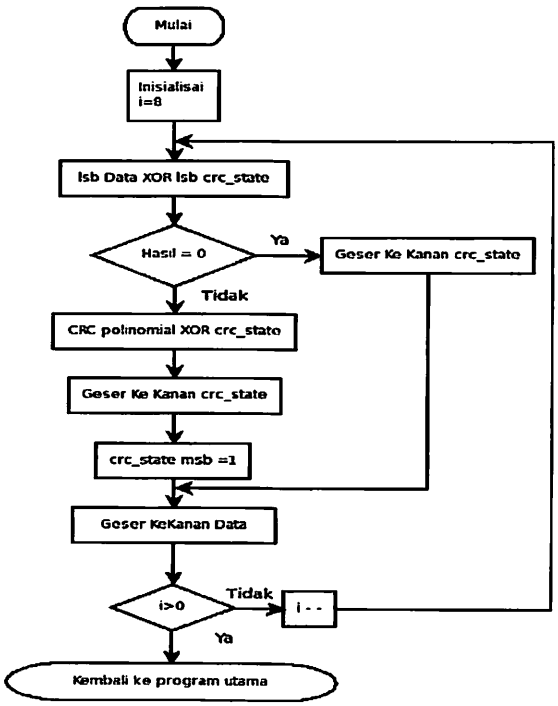
untuk data suhu yang dikirim adalah 14bit sedangkan untuk kelembapan 12bit, dimana pada saat proses pengiriman data MSB terlebih dahulu dikirim dan LSB dikirim kemudian.



***Gambar 3.10** Diagram Alir Proses Pengambilan Data Pada Sensor SHT11*

Untuk menjamin integritas data digunakan CRC. Karena setiap data yang dikirim besarnya 1 byte, CRC yang digunakan adalah CRC8. Secara sederhana kita dapat memandang CRC sebagai hubungan antara kunci dan gembok, dimana sensor sht11 menyediakan gemboknya dan mikrokontoller yang membuat kuncinya, yang

apabila cocok akan menunjukan data terkirim dengan benar tanpa adanya kerusakan pada saat proses transmisi. Diagram alir penerapan CRC8 dapat dilihat pada gambar 3.11.



***Gambar 3.11** Diagram Alir CRC8*

Pada diagram alir program utama, data hasil pengukuran tidak langsung dikirimkan, melaikan harus diproses terlebih dahulu. Persamaan dibawah menunjukan proses perhitungan data sensor sht11.

Perhitungan kelembapan relativ(RH):

$$RH_{Linier} = C_1 + C_2 \times SO_{RH} + C_3 \times SO_{RH}^{\ 2}$$

SO_{RH}	C_1	C_2	C_3
12 bit	-4	0.0405	$-2.8 * 10^{-6}$
8 bit	-4	0.648	$-7.2 * 10^{-4}$

Apabila suhu berbeda jauh dengan $25^{\circ}C$ maka kita melakukan proses sekali lagi yaitu menambahkan persamaan dibawah :

$$RH_{True}=(T_{\circ C}-25)\times(t_1+t_2\times SO_{RH})+RH_{Linier}$$

SO_{RH}	t_1	t_2
12 bit	0.01	0.00008
8 bit	0.01	0.00128

Semua variabel yang dipakai adalah untuk perhitungan 12 bit.

Perhitungan suhu :

$$Suhu=d_1+d_2\times SO_T$$

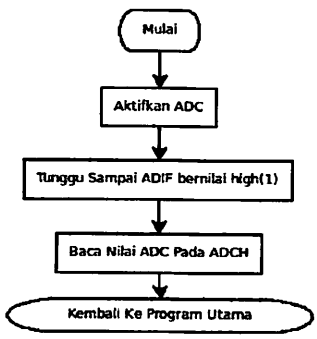
VDD	d_1 [°C]	d_1 [°F]
5V	-40.00	-40.00
4V	-39.75	-39.55
3.5V(3)	-39.66	-39.39
3V(3)	-39.60	-39.28
2.5V(3)	-39.55	-39.19

SO_T	d_2 [°C]	d_2 [°F]
14bit	0.01	0.018
12bit	0.04	0.072

SO_{RH}, SO_T merupakan data hasil pembacaan dari sensor.

3.2.1.2 Pengambilan Data Pada Sensor Tekanan Udara MPX4100

Keluaran dari MPX4100 adalah tegangan, oleh karena itu untuk dapat mengambil data dari sensor ini kita harus menggunakan ADC internal dari ATmega16. Gambar 3.12 menunjukan diagram alir proses pengambilan data pada MPX4100. ADIF terdapat pada register ADCSRA, apabila nilainya 1 maka proses konversi telah selesai dan hasil pembacaan dapat dibaca pada register ADCH.



***Gambar 3.12** Diagram Alir Pengambilan data Pada MPX4100*

Setelah proses konversi selesai mengubah nilai pada register ADCH menjadi tegangan dan mengkalkulasi tegangan tersebut menjadi tekanan udara, semua proses dapat dilihat pada persamaan dibawah.

1. Merubah nilai ADCH menjadi tegangan

$$Vout = \frac{ADCH \times V_{REF}}{255}$$

2. Karena tegangan masukan dibagi 2, maka untuk mendapatkan tegangan sebenarnya sensor kita kali 2 lagi nilai keluaran dari ADC

$$Vout = Vout \times 2$$

3. Untuk mendapatkan tekanan udara dapat diselesaikan dengan persamaan yang sudah disediakan oleh data sheet.

$$P = \frac{\frac{Vout + 0.079425}{5} + 0.1518}{0.01059}$$

3.2.1.3 Komunikasi Serial Pada ATmega16

Untuk dapat berkomunikas dengan modul wiznet, maka kita harus mengaktifkan komunikasi serial pada ATmega16. Sebelum dapat melaksanakan komunikasi serial kita terlebih dahulu melakukan inisialisasi. Pada proses inisialisasi ini kita menentukan kecepatan transfer data aktivasi TX/RX serial pada register UCSRB. Untuk kecepatan transfer kita dapat mengisi register UBRR dengan nilai yang ditentukan oleh persamaan dibawah.

$$UBRR = \frac{F_{OSC}}{16\,x\,baut\,rate} - 1$$

Pada modul wiznet kecepatan transfer sudah ditentukan yaitu 9600 bps, maka nilai register UBRR adalah :

$$UBRR = \frac{8000000}{16x9600} - 1$$

$UBRR = 51{,}0833$, dibulatkan sehingga $UBRR = 51$

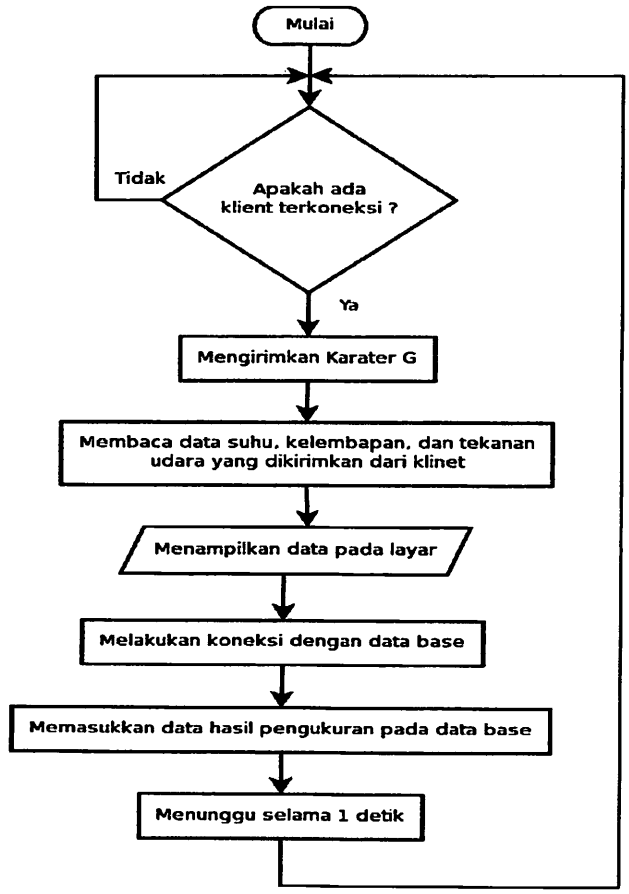
Proses penerimaan melibatkan register UCSRA, pada register ini terdapat bit RXC. RXC bernilai awal 0 tetapi apabila proses penerimaan telah selesai maka nilai ini berubah menjadi 1 dan data yang masuk dapat dibaca pada register UDR. Pada proses pengiriman kita menunggu bit UDRE bernilai 1, kemudian kita masukkan data yang kita kirim ke register UDR setelah itu maka data tersebut akan dikirimkan oleh mikrokontoller.

3.2.2 Perancangan Perangkat Lunak Pada PC

Pada PC bahasa pemograman yang digunakan adalah java, untuk kompiler digunakan JDK 6, untuk membangun GUI serta editor listing program digunakan NetBeans 6.0, dan Database PostgreSQL 8.1.4. Perangkat lunak yang dibuat akan menjadikan PC sebagai TCP server yang bertugas mengumpulkan data dari mikrokontoller melalui modul wiznet, menampilkan data, dan membuat koneksi dengan database PostgreSQL untuk proses penyimpanan data. Algoritma dari program java yang dibuat adalah :

1. Pertama kita membuat kelas dengan nama DataAccuisition, pada kelas ini nanti kita dapat menambahkan metod sesuai dengan keperluan. Persiapkan port yang dipakai untuk komunikasi, secara default wiznet menggunakan port 5000, maka pada PC kita menyiapkan port 5000 juga. Bind alamat lokal pada port tersebut, untuk itu kita memerlukan kelas ServerSocket. Pada kelas ServerSocket terdapat konstruktor yang dapat melakukan proses tersebut, disamping itu terdapat dua method yang penting yaitu accept() dan close().

2. Pada program akan dibuat objek dengan nama WiznetServer dari kelas ServerSocket. Untuk dapat menjadikan server pada mode listen, maka method yang kita gunakan adalah accept(). Setelah penggunaan method accept ini server akan menuggu terus sampai adanya permintaan dari client.
3. Apabila terjadi ada client yang terkoneksi, akan dihasilkan sebuah objek dari kelas Socket. Pada program objek baru ini akan dinamakan WiznetInterface.
4. Agar dapat melakukan pertukaran data diperlukan adanya I/O stream, sebagai tempat pertukaran data. Untuk input stream akan digunakan dua kelas yaitu BufferedReader dan InputStreamReader. Kelas InputStreamReader berfungsi mengubah data dari input stream socket ke bentuk deretan karakter yang akan dimanipulasi oleh BufferedReader. Pada BufferedReader data diperlakukan sebagai string atau karakter tergantung dari method yang digunakan. Dari kedua kelas diatas akan dihasilkan sebuah objek yang diberi nama FromWiznet.
5. Untuk output stream, digunakan kelas DataOutputStream yang akan mengirimkan data ke socket sesuai tipe data reguler Java. Objek yang dihasilkan oleh kelas ini diberi nama ToWiznet.
6. Server akan mengirimkan karakter ke client untuk memerintahkan pengambilan data dan mengirimkan data hasil pengukuran ke server. Proses ini dilakukan secara terus-menerus dengan selang waktu tertentu sesuai yang diinginkan.
7. Data tersebut kemudian dimasukkan ke dalam database dengan terlebih dahulu melakukan koneksi ke database PostgreSQL.



***Gambar 3.13** Diagram Aktivitas program pada PC*

BAB IV

PENGUJIAN DAN ANALISA

4.1 Pengujian Komunikasi Serial *synchronous* Sensor SHT11

4.1.1 Tujuan

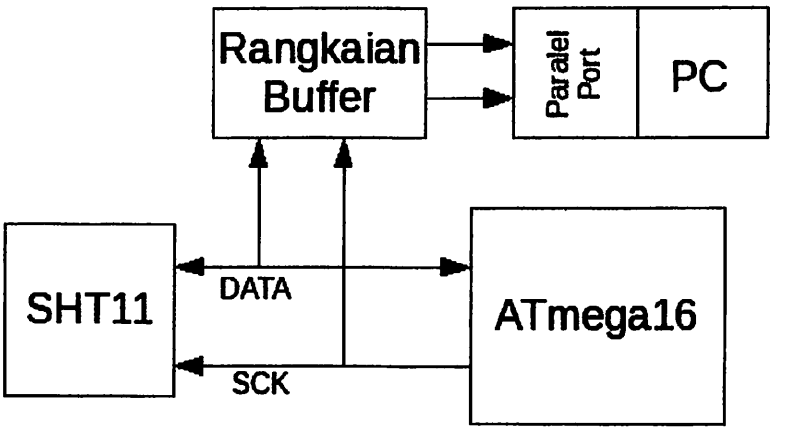
Pengujian ini bertujuan untuk mengamati bentuk sinyal yang dikeluarkan oleh sensor SHT11 serta melakukan analisa terhadap sinyal tersebut. Dari hasil analisa kita dapat melihat karakteristik sinyal yang terjadi dan akan dibandingkan dengan pembacaan oleh mikrokontoller untuk mengetahui apakah proses pembacaan pada mikrokontoller sudah benar.

4.1.2 Peralatan Yang Digunakan

1. Rangkaian MAX232
2. Konverter USB ke Serial
3. Power Supply +5 V
4. Perangkat lunak Minicom 2.1
5. Konektor DB25
6. IC 4050
7. Rangkaian mikrokontroller dan sensor SHT11
8. Perangkat lunak TFLA 0.1.3 (*tfla-01.berlios.de*)

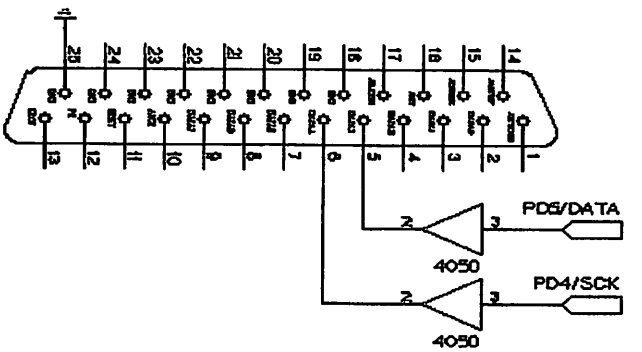
4.1.3 Langkah – langkah Percobaan

1. Hubungkan rangkaian mikrokontoller, MAX232, Power Supply, rangkaian IC 4050 yang merupakan rangkaian buffer.

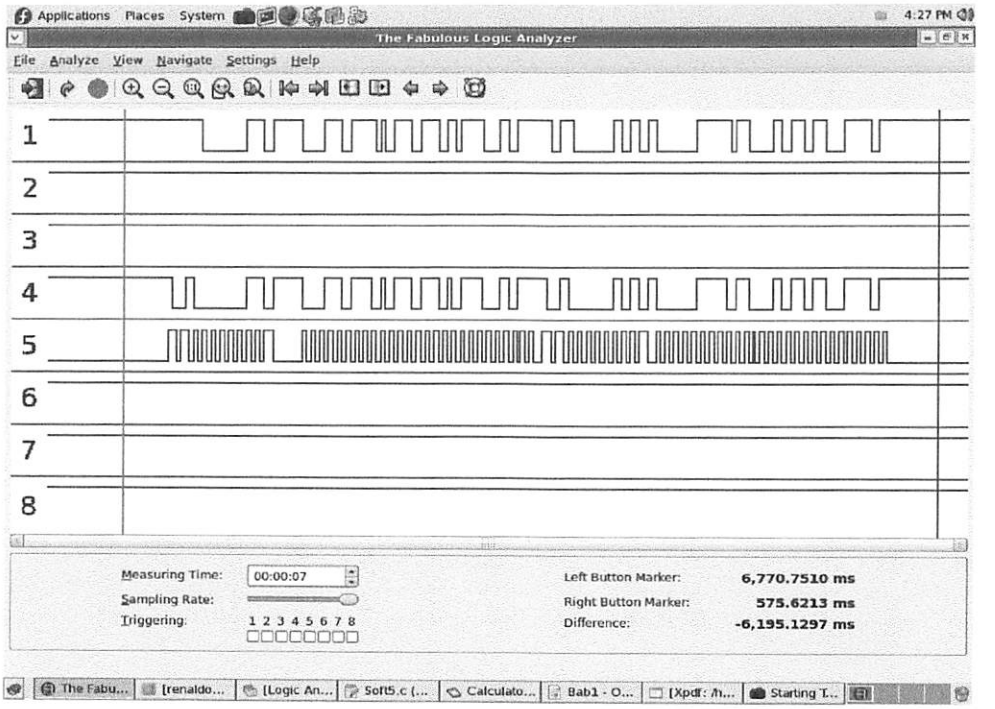


***Gambar 4.1** Diagram Blok Pengujian Sensor SHT11*

2. Jalankan perangkat lunak minicom dan tfla, tfla merupakan perangkat lunak yang dapat menjadikan komputer berfungsi sebagai *logic analyzer* dengan menggunakan port paralel sebagai masukkan data dari jalur komunikasi.
3. Setalah semua telah terpasang klik *start* pada menu *analyze* dan lakukan proses pengambilan data.



***Gambar 4.2** Rangkaian Buffer*



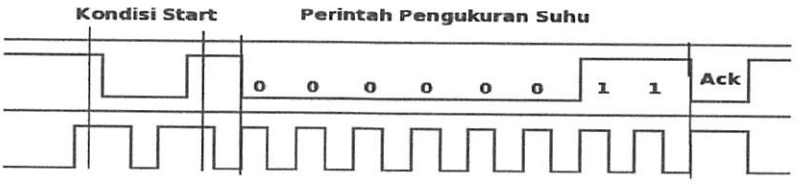
***Gambar 4.3** Hasil Pengambilan Data pada SHT11*

4.1.4 Analisa

Hasil dari proses pembacaan akan berupa sinyal-sinyal digital 1 atau 0 yang akan dibagi-bagi menjadi beberapa bagian sesuai fungsinya masing-masing. Gambar 4.6 menunjukan hasil pengambilan data pada sensor SHT11. Data suhu yang terbaca adalah 34,94 ^{o}C .

Pada baris ke-4 di gambar menunjukan jalur data dan pada baris ke-5 menunjukkan jalur clock. Sinyal tersebut akan dibagi menjadi beberapa bagian sesuai dengan fungsinya.

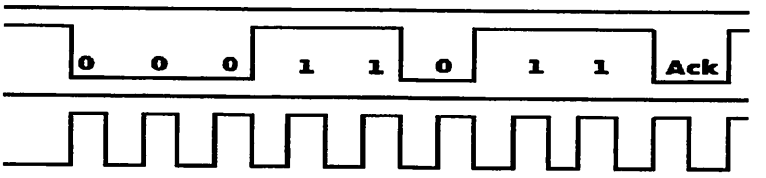
1. Kondisi Start dan pengiriman perintah untuk melaksanakan pengukuran suhu.



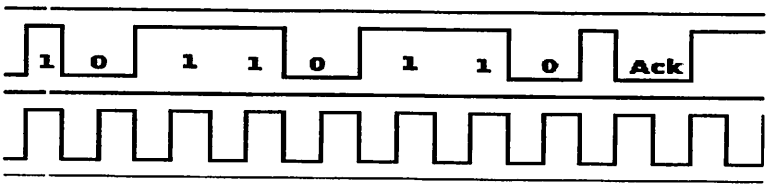
***Gambar 4.4** Kondisi Start dan Perintah Pengukuran Suhu*

Seperti yang sudah ditunjukan dijelaskan pada bagian sebelumnya kondisi start terdiri dari, data dalam keadaan low pada jalur DATA, Selama SCK high, diikuti pulsa pada SCK low dan naik kembali dan SCK dalam keadaan high terus.Baris perintah terdiri dari 3bit alamat dan 5bit perintah, bit ke9 adalah ACK. Menurut keterangan yang ada pada datasheet alamat yang dikenali hanyalah 000 atau dengan lain kata jalur data ini hanya dapat digunakan oleh satu perangkat saja, 00011 merupakan perintah melakukan pengukuran suhu.

2. Setelah proses pengiriman perintah maka akan mikrokontoller akan menunggu beberapa saat sebelum data siap untuk dikirimkan, data yang dikirim sebesar 16 bit, yang pertama dikirim adalah data MSB dan kemudian data LSB.



***Gambar 4.5** Pengiriman Data MSB dari Sensor SHT11*



***Gambar 4.6** Pengiriman Data LSB dari Sensor SHT11*

Data pada gambar 4.4, 4.5, dan 4.6 merupakan sebagian dari data yang dikirimkan oleh sensor SHT11 masih ada data pengukuran kelembapan udara, akan tetapi tidak dijabarkan karena prosesnya sama seperti proses pengukuran suhu.

Data hasil pengukuran suhu adalah 00011011 10110110 = 7094

maka, $T = d1 + d2 \times 7094$

$T = -40 + 0.01 \times 7094$

$T = 30.94^{\circ}C$

Hasil analisa terhadap sinyal menghasilkan nilai suhu yang sama dengan pembacaan oleh mikrokontoller dan kita dapat mengetahui lebih jelas bentuk sinyal yang komunikasi antara sensor SHT11 dan mikrokontoller.

4.2 Pengujian ADC

4.2.1 Tujuan

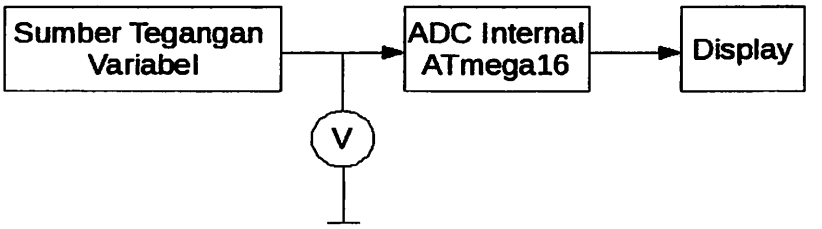
Pengujian ADC ini bertujuan utnuk mengetahui apakah konfigurasi dari ADC internal mikrokontroller sudah benar dan melihat hasil konversi yang diperoleh. Dari hasil yang diperoleh kita dapat melakukan revisi terhadap nilai keluaran agar dapat memberikan hasil yang optimal.

4.2.2 Peralatan Yang Digunakan

1. Rangkaian MAX232
2. Konverter USB ke Serial
3. Power Supply +5 V
4. Perangkat lunak Minicom 2.1
5. Voltmeter
6. Sumber tegangan DC variabel antara 0-2.5V
7. Rangkaian mikrokontroller yang sudah dikonfigurasi untuk ADC

4.2.3 Langkah – langkah Percobaan

1. Hubungkan rangakaian mikrokontroller, MAX232, dan Power Supply.



***Gambar 4.7** Blok Diagram Pengujian ADC*

2. Masukkan perangkat lunak pada mikrokontroller. Perangkat lunak ini berfungsi untuk melakukan proses konversi dan mengirimkan hasil konversi melalui komunikasi serial. Potongan program dibawah merupakan program yang akan dimasukkan ke mikrokontroller.

```
main(){
      //Inisialisasi
      Init_ADC();
      InitUART(51);
      char tombol,ADCout[6];
      while(1){
            tombol=ReceiveByte();
            if(tombol=='D'){
            inADC=Read_ADC();
            Tegangan=inADC*0.01;
            dtostrf(Tegangan,5,2,ADCout);
            KirimSerial(Tegangan);
            KirimSerial("\n\r");
            }
      }
      return 0;
}
```

3. Hubungkan rangakaian MAX232 ke PC dengan menggunakan konverter USB ke serial.
4. Jalankan Minicom dengan mengetikkan minicom pada terminal

   [root@localhost renaldo]# minicom
5. Masukkan tegangan pada pin PA.0 dari sumber tegangan DC variabel
6. Mencatat hasil konversi ADC yang ada pada minicom.

4.2.4 Analisa

Hasil dari konversi ADC mikrokontoller ATmega16 dapat diselesasikan dengan persamaan dibawah :

$$Vout = \frac{((ADCH) \times Vref)}{255}$$

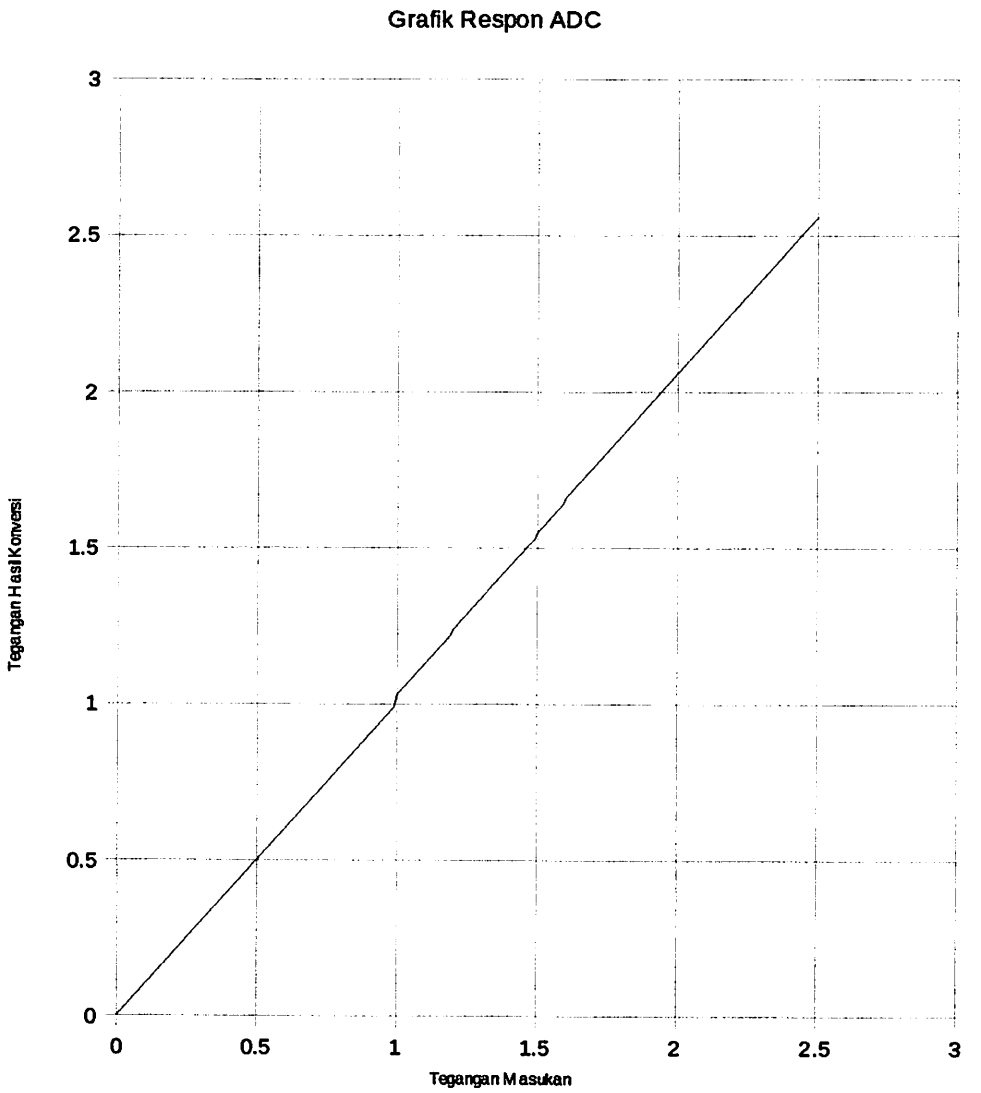
Apabila Vin=2,33V maka hasil perhitungan Vout=2,33V

***Tabel 4.1** Hasil Konversi ADC*

No	Vin (V)	Vadc (V)	Vadc-Vin (V)	% Kesalahan	No	Vin (V)	Vadc (V)	Vadc-Vin (V)	% Kesalahan
1	0	0	0	0	127	1.26	1.3	0.04	3.17
2	0.01	0.01	0	0	128	1.27	1.31	0.04	3.15
3	0.02	0.02	0	0	129	1.28	1.32	0.04	3.13
4	0.03	0.03	0	0	130	1.29	1.33	0.04	3.1
5	0.04	0.04	0	0	131	1.3	1.34	0.04	3.08
6	0.05	0.05	0	0	132	1.31	1.35	0.04	3.05
7	0.06	0.06	0	0	133	1.32	1.36	0.04	3.03
8	0.07	0.07	0	0	134	1.33	1.37	0.04	3.01
9	0.08	0.08	0	0	135	1.34	1.38	0.04	2.99
10	0.09	0.09	0	0	136	1.35	1.39	0.04	2.96
11	0.1	0.1	0	0	137	1.36	1.4	0.04	2.94
12	0.11	0.11	0	0	138	1.37	1.41	0.04	2.92
13	0.12	0.12	0	0	139	1.38	1.42	0.04	2.9
14	0.13	0.13	0	0	140	1.39	1.43	0.04	2.88
15	0.14	0.14	0	0	141	1.4	1.44	0.04	2.86
16	0.15	0.15	0	0	142	1.41	1.45	0.04	2.84
17	0.16	0.16	0	0	143	1.42	1.46	0.04	2.82
18	0.17	0.17	0	0	144	1.43	1.47	0.04	2.8
19	0.18	0.18	0	0	145	1.44	1.48	0.04	2.78
20	0.19	0.19	0	0	146	1.45	1.49	0.04	2.76
21	0.2	0.2	0	0	147	1.46	1.5	0.04	2.74
22	0.21	0.21	0	0	148	1.47	1.51	0.04	2.72
23	0.22	0.22	0	0	149	1.48	1.52	0.04	2.7
24	0.23	0.23	0	0	150	1.49	1.53	0.04	2.68
25	0.24	0.24	0	0	151	1.5	1.55	0.05	3.33
26	0.25	0.25	0	0	152	1.51	1.56	0.05	3.31
27	0.26	0.26	0	0	153	1.52	1.57	0.05	3.29
28	0.27	0.27	0	0	154	1.53	1.58	0.05	3.27
29	0.28	0.28	0	0	155	1.54	1.59	0.05	3.25
30	0.29	0.29	0	0	156	1.55	1.6	0.05	3.23
31	0.3	0.3	0	0	157	1.56	1.61	0.05	3.21
32	0.31	0.31	0	0	158	1.57	1.62	0.05	3.18
33	0.32	0.32	0	0	159	1.58	1.63	0.05	3.16
34	0.33	0.33	0	0	160	1.59	1.64	0.05	3.14
35	0.34	0.34	0	0	161	1.6	1.66	0.06	3.75
36	0.35	0.35	0	0	162	1.61	1.67	0.06	3.73

37	0.36	0.36	0	0	163	1.62	1.68	0.06	3.7
38	0.37	0.37	0	0	164	1.63	1.69	0.06	3.68
39	0.38	0.38	0	0	165	1.64	1.7	0.06	3.66
40	0.39	0.39	0	0	166	1.65	1.71	0.06	3.64
41	0.4	0.4	0	0	167	1.66	1.72	0.06	3.61
42	0.41	0.41	0	0	168	1.67	1.73	0.06	3.59
43	0.42	0.42	0	0	169	1.68	1.74	0.06	3.57
44	0.43	0.43	0	0	170	1.69	1.75	0.06	3.55
45	0.44	0.44	0	0	171	1.7	1.76	0.06	3.53
46	0.45	0.45	0	0	172	1.71	1.77	0.06	3.51
47	0.46	0.46	0	0	173	1.72	1.78	0.06	3.49
48	0.47	0.47	0	0	174	1.73	1.79	0.06	3.47
49	0.48	0.48	0	0	175	1.74	1.8	0.06	3.45
50	0.49	0.49	0	0	176	1.75	1.81	0.06	3.43
51	0.5	0.5	0	0	177	1.76	1.82	0.06	3.41
52	0.51	0.51	0	0	178	1.77	1.83	0.06	3.39
53	0.52	0.52	0	0	179	1.78	1.84	0.06	3.37
54	0.53	0.53	0	0	180	1.79	1.85	0.06	3.35
55	0.54	0.54	0	0	181	1.8	1.86	0.06	3.33
56	0.55	0.55	0	0	182	1.81	1.87	0.06	3.31
57	0.56	0.56	0	0	183	1.82	1.88	0.06	3.3
58	0.57	0.57	0	0	184	1.83	1.89	0.06	3.28
59	0.58	0.58	0	0	185	1.84	1.9	0.06	3.26
60	0.59	0.59	0	0	186	1.85	1.91	0.06	3.24
61	0.6	0.6	0	0	187	1.86	1.92	0.06	3.23
62	0.61	0.61	0	0	188	1.87	1.93	0.06	3.21
63	0.62	0.62	0	0	189	1.88	1.94	0.06	3.19
64	0.63	0.63	0	0	190	1.89	1.95	0.06	3.17
65	0.64	0.64	0	0	191	1.9	1.96	0.06	3.16
66	0.65	0.65	0	0	192	1.91	1.97	0.06	3.14
67	0.66	0.66	0	0	193	1.92	1.98	0.06	3.13
68	0.67	0.67	0	0	194	1.93	1.99	0.06	3.11
69	0.68	0.68	0	0	195	1.94	2	0.06	3.09
70	0.69	0.69	0	0	196	1.95	2.01	0.06	3.08
71	0.7	0.7	0	0	197	1.96	2.02	0.06	3.06
72	0.71	0.71	0	0	198	1.97	2.03	0.06	3.05
73	0.72	0.72	0	0	199	1.98	2.04	0.06	3.03
74	0.73	0.73	0	0	200	1.99	2.05	0.06	3.02
75	0.74	0.74	0	0	201	2	2.06	0.06	3
76	0.75	0.75	0	0	202	2.01	2.07	0.06	2.99
77	0.76	0.76	0	0	203	2.02	2.08	0.06	2.97
78	0.77	0.77	0	0	204	2.03	2.09	0.06	2.96
79	0.78	0.78	0	0	205	2.04	2.1	0.06	2.94
80	0.79	0.79	0	0	206	2.05	2.11	0.06	2.93
81	0.8	0.8	0	0	207	2.06	2.12	0.06	2.91
82	0.81	0.81	0	0	208	2.07	2.13	0.06	2.9
83	0.82	0.82	0	0	209	2.08	2.14	0.06	2.88
84	0.83	0.83	0	0	210	2.09	2.15	0.06	2.87
85	0.84	0.84	0	0	211	2.1	2.16	0.06	2.86
86	0.85	0.85	0	0	212	2.11	2.17	0.06	2.84
87	0.86	0.86	0	0	213	2.12	2.18	0.06	2.83
88	0.87	0.87	0	0	214	2.13	2.19	0.06	2.82
89	0.88	0.88	0	0	215	2.14	2.2	0.06	2.8

90	0.89	0.89	0	0	216	2.15	2.21	0.06	2.79
91	0.9	0.9	0	0	217	2.16	2.22	0.06	2.78
92	0.91	0.91	0	0	218	2.17	2.23	0.06	2.76
93	0.92	0.92	0	0	219	2.18	2.24	0.06	2.75
94	0.93	0.93	0	0	220	2.19	2.25	0.06	2.74
95	0.94	0.94	0	0	221	2.2	2.26	0.06	2.73
96	0.95	0.95	0	0	222	2.21	2.27	0.06	2.71
97	0.96	0.96	0	0	223	2.22	2.28	0.06	2.7
98	0.97	0.97	0	0	224	2.23	2.29	0.06	2.69
99	0.98	0.98	0	0	225	2.24	2.3	0.06	2.68
100	0.99	0.99	0	0	226	2.25	2.31	0.06	2.67
101	1	1.03	0.03	3	227	2.26	2.32	0.06	2.65
102	1.01	1.04	0.03	2.97	228	2.27	2.33	0.06	2.64
103	1.02	1.05	0.03	2.94	229	2.28	2.34	0.06	2.63
104	1.03	1.06	0.03	2.91	230	2.29	2.35	0.06	2.62
105	1.04	1.07	0.03	2.88	231	2.3	2.36	0.06	2.61
106	1.05	1.08	0.03	2.86	232	2.31	2.37	0.06	2.6
107	1.06	1.09	0.03	2.83	233	2.32	2.38	0.06	2.59
108	1.07	1.1	0.03	2.8	234	2.33	2.39	0.06	2.58
109	1.08	1.11	0.03	2.78	235	2.34	2.4	0.06	2.56
110	1.09	1.12	0.03	2.75	236	2.35	2.41	0.06	2.55
111	1.1	1.13	0.03	2.73	237	2.36	2.42	0.06	2.54
112	1.11	1.14	0.03	2.7	238	2.37	2.43	0.06	2.53
113	1.12	1.15	0.03	2.68	239	2.38	2.44	0.06	2.52
114	1.13	1.16	0.03	2.65	240	2.39	2.45	0.06	2.51
115	1.14	1.17	0.03	2.63	241	2.4	2.46	0.06	2.5
116	1.15	1.18	0.03	2.61	242	2.41	2.47	0.06	2.49
117	1.16	1.19	0.03	2.59	243	2.42	2.48	0.06	2.48
118	1.17	1.2	0.03	2.56	244	2.43	2.49	0.06	2.47
119	1.18	1.21	0.03	2.54	245	2.44	2.5	0.06	2.46
120	1.19	1.22	0.03	2.52	246	2.45	2.51	0.06	2.45
121	1.2	1.24	0.04	3.33	247	2.46	2.52	0.06	2.44
122	1.21	1.25	0.04	3.31	248	2.47	2.53	0.06	2.43
123	1.22	1.26	0.04	3.28	249	2.48	2.54	0.06	2.42
124	1.23	1.27	0.04	3.25	250	2.49	2.55	0.06	2.41
125	1.24	1.28	0.04	3.23	251	2.5	2.56	0.06	2.4
126	1.25	1.29	0.04	3.2	252				



***Gambar 4.8** Grafik Hasil Konversi Analog Ke Digital*

4.3 Pengujian Sensor MPX4100

4.3.1 Tujuan

Pengujian ini bertujuan untuk mengetahui apakah tindakan pemprosesan data pada sensor MPX4100 sudah benar. Proses yang akan dilakukan adalah melakukan pengukuran tegangan keluaran dari sensor kemudian melakukan perhitungan, dengan

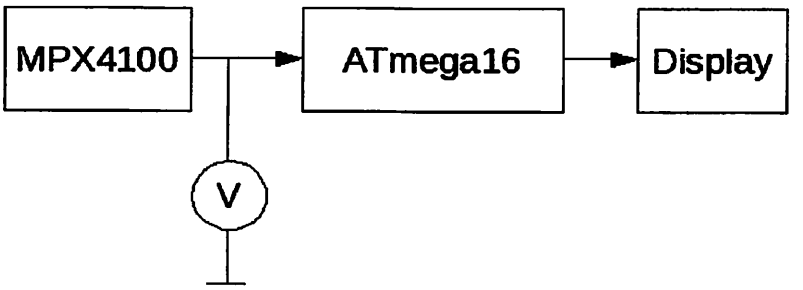
persamaan yang sama pada mikrokontoller.

4.3.2 Peralatan Yang Digunakan

1. Rangakaian mikrokontoller beserta semua sensor
2. Modul Wiznet EG-SR 7150MJ
3. Kabel UTP dan konektor RJ-45
4. Voltmeter
5. Perangkat lunak yang sudah dirancang pada BAB III

4.3.3 Langkah – langkah Percobaan

1. Hubungkan rangkaian pada power supply



Gambar 4.9 Diagram Blok Pengujian Sensor MPX4100

2. Pasang voltmeter pada pin 1 sensor MPX4100
3. Jalankan program di PC
4. Melakukan proses pengambilan data
5. Pada saat proses pengambilan data berlangsung catat nilai tegangan yang dibaca oleh voltmeter.

4.3.4 Analisa

Transfer fungsi untuk sensor MPX4100 adalah :

$$P = \frac{((\frac{(V_{MPX} + error)}{V_s}) + 0.1518)}{0.01059}$$

error = Pressure Error x Temperature Error x 0.01059 x Vs

Vs = 4,94 V

Pressure Error = 1,5

Temperature Error Factor =1

error = 1,5 x 1 x 0.01059 x 4,94 =0.0784719

Tabel 4.2 Hasil Pengujian Sensor MPX4100

Waktu	Vout (Volt)	Hasil Pengukuran (kPa)	Hasil Perhitungan (kPa)
31/1/2008 10:15:18	4,26	97,26	97,26
1/2/2008 08:38:26	4,24	96,88	96,88
6/2/2008 21:40:12	4,16	95,35	95,35
6/2/2008 21:40:42	4.18	95,74	95,74

4.4 Pengujian Komunikasi Serial

4.4.1 Tujuan

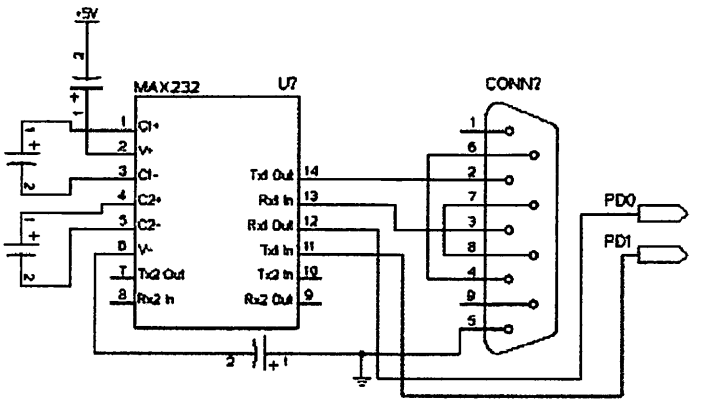
Pengujian ini bertujuan untuk mengetahui apakah komunikasi serial dari mikrokontroller berjalan dengan baik. Proses pengujian ini akan melihat fungsi penerima dan pengirim dengan cara mengirimkan karakter ke mikrokontroller dan apabila mikrokontroller menerima karakter yang dikirim akan mengirimkan kembali suatu pesan ke PC.

4.4.2 Peralatan Yang Digunakan

8. Rangkaian MAX232
9. Konverter USB ke Serial
10. Power Supply +5 V
11. Perangkat Lunak Minicom 2.1

4.4.3 Langkah – langkah pengujian

1. Hubungkan rangakaian mikrokontroller, MAX232, dan Power Supply.



***Gambar 4.10** Rangkaian Max232*

2. Masukkan perangkat lunak pada mikrokontroller. Perangkat lunak ini berfungsi untuk melakukan pengujian fungsi kirim dan terima dari mikrokontroller. Potongan program dibawah merupakan program yang akan dimasukkan ke mikrokontroller.

```
main(){
    InitUART(51); //9600 8-N-1
    char tombol;
    while(1){
        tombol=ReceiveByte();
        if(tombol=='a'){
                KirimSerial("Anda mengirim Karakter (a) \n\r");
        }
   if(tombol=='b'){
                KirimSerial("Anda mengirim Karakter (b) \n\r");
        }
        if(tombol=='c'){
                KirimSerial("Anda mengirim Karakter (c) \n\r");
        }
        if(tombol=='d'){
                KirimSerial("Anda mengirim Karakter (d) \n\r");
        }
        if(tombol=='e'){
                KirimSerial("Anda mengirim Karakter (e) \n\r");
        }
 }
        return 0;
}
```

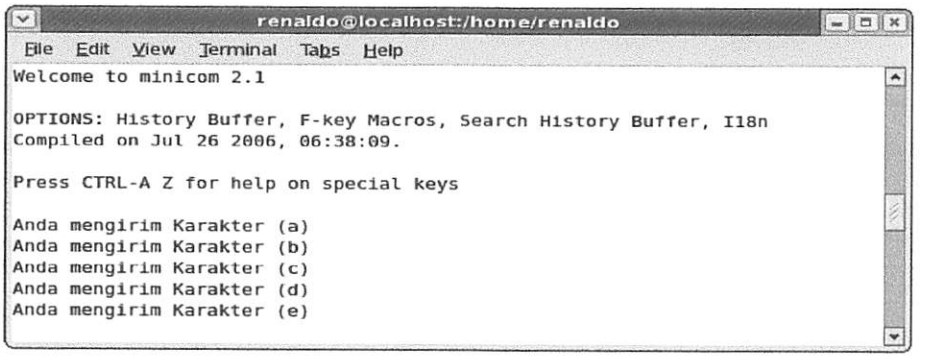
3. Hubungkan rangakaian MAX232 ke PC dengan menggunakan konverter USB ke serial.

4. Jalankan Minicom dengan mengetikkan minicom pada terminal

   [root@localhost renaldo]# minicom
5. Masukkan karakter sesuai dengan yang ditulis pada program untuk melihat respon dari mikrokontroller.
6. Pada langkah ke-6 ini akan dilakukan pengiriman data hasil pengukuran, untuk program pada mikrokontroller akan dihapus dan diisi dengan program yang sudah dijelaskan pada BAB III.

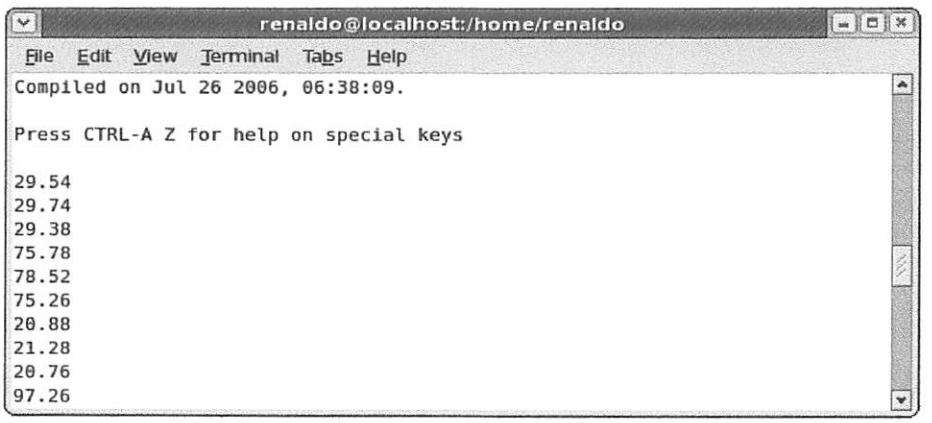
4.4.4 Analisa

Pada saat pengiriman karakter 'a' maka mikrokontroller akan membalas dengan pernyataan 'Anda mengirim Karakter (a)', hal yang sama apabila kita kirimkan karakter 'b','c','d', dan 'e'. Apabila kita mengirimkan karakter selain yang telah disebutkan diatas maka mikrokontroller tidak akan memberikan balasan apa-apa.



***Gambar 4.11** Hasil Pengujian Komunikasi Serial Pada Minicom*

Hasil percobaan yang telah dilakukan menunjukkan fungsi komunikasi serial dari mikrokontroller sudah dapat berjalan dengan baik. Oleh karena itu akan dilakukan pengiriman data hasil pengukuran oleh sensor. Gambar 4.2 menunjukan hasil dari pengiriman serial pada data hasil pengukuran dari sensor.



Gambar 4.12 *Hasil Pengujian Pengiriman Data Hasil Pengukuran*

Penjelasan data yang dikirim :

29,54 (Suhu Sekarang)

29,74 (Suhu Maksimum)

29,38 (Suhu Minimum)

75,78 (Kelembapan Sekarang)

78,52 (Kelembapan Maksimum)

75,26 (Kelembapan Minimum)

20,88 (Dew Point Sekarang)

21,28 (Dew Point Maksimum)

20,76 (Dew Point Minimum)

97,26 (Tekanan Udara)

Percobaan diatas dilaksanakan pada 31-01-2008 jam 12:10 selama 10 menit, oleh sebab itu perbadaan antara nilai sekarang, Maksimum, dan Minimum tidak terlalu jauh karena hanya mengambil nilai Maksimum dan Minimum pada rentang waktu yang singkat.

4.5 Pengujian Komunikasi TCP/IP

4.5.1 Tujuan

Pengujian ini dilakukan untuk mengetahui proses pengiriman data sudah berjalan dengan benar dan melihat apakah data yang dikirimkan sudah sesuai dengan data yang ditampilkan pada PC. Yang akan dilakukan merekam setiap data yang keluar dan masuk pada kartu jaringan dengan menggunakan perangkat lunak Wireshark 0.99.3a. Wireshark merekam semua protokol yang melewati kartu jaringan, oleh karena itu untuk mempermudah analisa kita batasi saja protokol yang direkam adalah TCP.

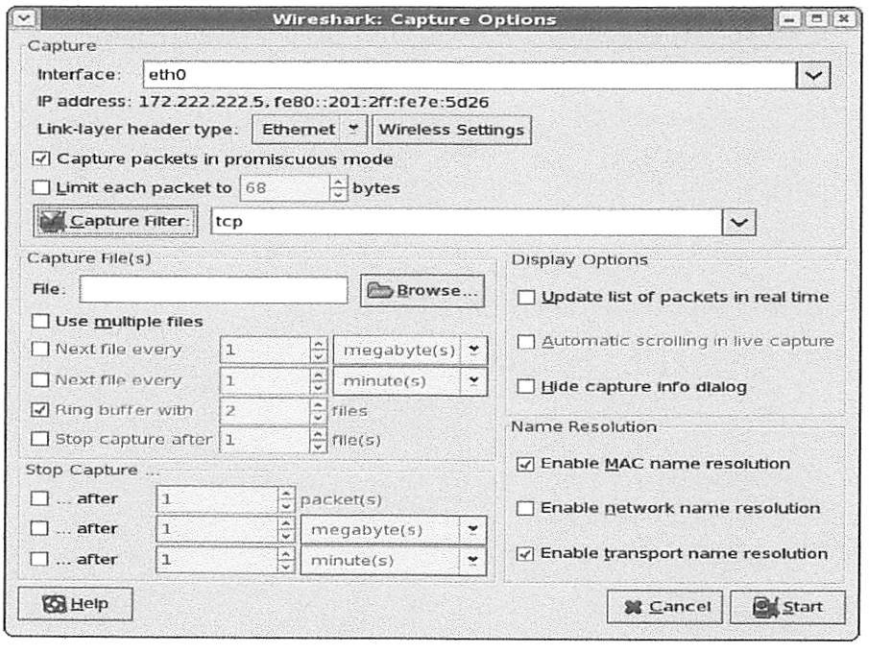
4.5.2 Peralatan Yang Digunakan

6. Rangakaian mikrokontoller beserta semua sensor
7. Modul Wiznet EG-SR 7150MJ
8. Kabel UTP dan konektor RJ-45
9. Perangkat lunak Wireshark 0.99.3a
10. Perangkat lunak yang sudah dirancang pada BAB III

4.5.3 Langkah – langkah Percobaan

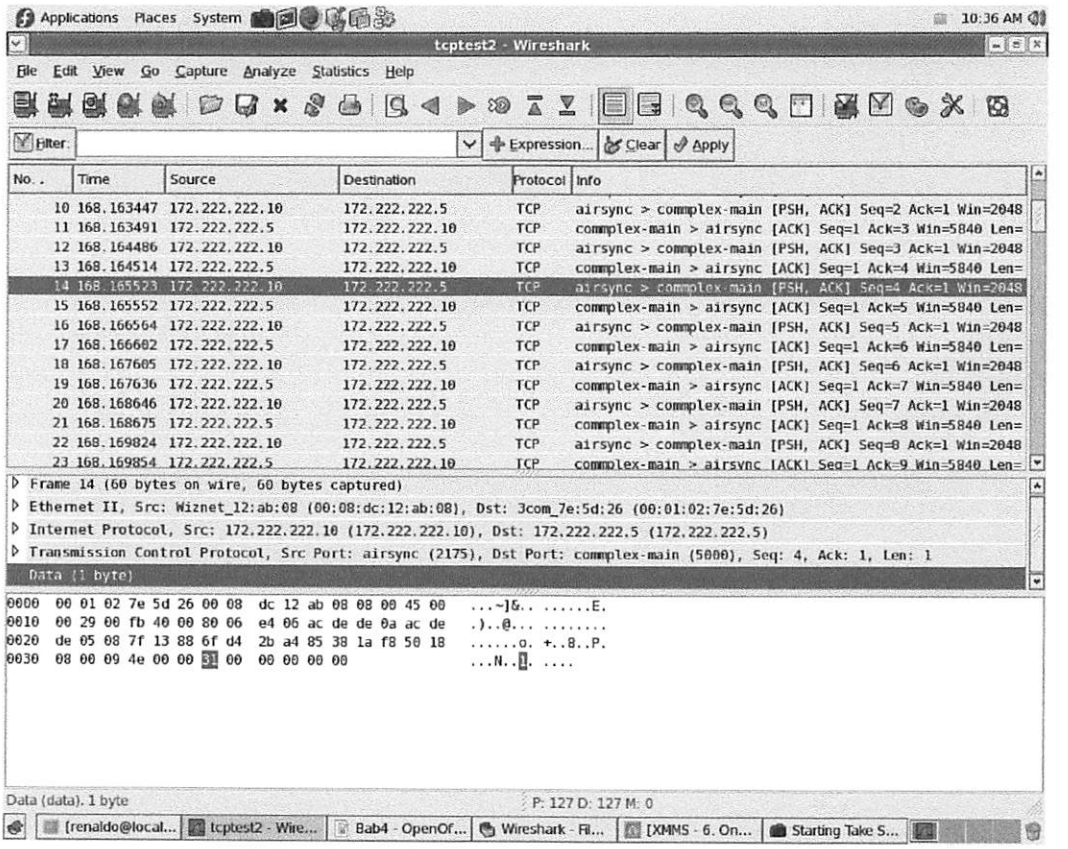
1. Hubungkan rangakaian mikrokontoller pada modul wiznet lalu kemudian hubungkan modul wiznet ke PC dengan menggunakan kabel UTP terakhir pasang power supply
2. Jalankan Wireshark. Perlu diperhatikan bahwa pada waktu menjalankan perangkat lunak ini kita butuh hak akses root, jadi apabila dijalankan sebagai user biasa maka pada saat program berjalan maka kita diminta memasukkan kata kunci root.
3. Setelah Wireshark sudah jalan, pada *Capture Filter* yang terdapat pada menu

Capture Option pilih tcp. Untuk lebih jelas dapat dilihat pada gambar dibawah.



Gambar 4.13 *Pengaturan Pada Capure Option*

4. Setalah dilakukan pengaturan maka klik tombol start untuk memulai perekaman
5. Jalankan perangkat lunak java yang telah kita buat sebelumnya dan melakukan pengambilan data.
6. Setelah pengambilan data dilakukan catat hasil yang diperoleh dan program ditutup. Pada Wireshark hentikan proses perekaman untuk dapat melihat hasilnya. Gambar dibawah merupakan hasil rekaman data pada kartu jaringan.



Gambar 4.14 *Hasil Rekaman Data Pada Kartu Jaringan*

7. Melakukan analisa terhadap setiap paket data yang dikirim.

4.5.4 Analisa

Sebelum data kita analisa terlebih dahulu kita lihat metode penerimaan data pada program di PC. Perhatikan potongan program dibawah

```
String TempCur = FromWiznet.readLine();
```

kode tersebut akan membaca karakter terus-menerus sampai ada baris baru. Data yang ditampilkan adalah “30.14” jadi data yang masuk “30.14\n” tanda “\n” berarti baris baru.

Data yang ditampilkan 30.14 pada waktu 09:29:53

Data hasil rekaman pada kartu jaringan :

1. Paket data pertama

   Waktu : 09:29:53.788353

   Data : 00 01 02 7e 5d 26 00 08 dc 12 ab 08 08 00 45 00

   00 29 00 f8 40 00 80 06 e4 09 ac de de 0a ac de

   de 05 08 7f 13 88 6f d4 2b a1 85 38 1a f8 50 18

   08 00 07 51 00 00 33 00 00 00 00 00

   Paket data diatas terbagi menjadi beberapa bagian :

Network Layer	Destination MAC address[00 01 02 7e 5d 26] Source MAC address[00 08 dc 12 ab 08] Type : IP (0x0800) [08 00]
Internet Layer (IP)	Version dan Header length [45] Differentiated Services Field : 0x00 [00] Total length : 41 [00 29] Identification [00 f8] Flags [40] Fragment offset [00] Time to live [80] Protokol : TCP [06] Header checksum [e4 09] Source : 172.222.222.10 [ac de de 0a] Destination : 172.222.222.5[ac de de 05]
Transport Layer (TCP)	Source port [08 7f] Destination port [13 88] Sequence number [6f d4 2b a1] Ack number [85 38 1a f8] Header length [50] Flags [18] Window size [08 00] Checksum [07 51]
	Data 1 byte [33] 33 hexadesimal merupakan kode ASCII dari karakter "3"

2. Paket data ke dua

   Waktu : 09:29:53.789387

   Data : 00 01 02 7e 5d 26 00 08 dc 12 ab 08 08 00 45 00

   00 29 00 f9 40 00 80 06 e4 08 ac de de 0a ac de

   de 05 08 7f 13 88 6f d4 2b a2 85 38 1a f8 50 18

   08 00 0a 50 00 00 30 00 00 00 00 00

Paket data diatas terbagi menjadi beberapa bagian:

Network Layer	Destination MAC address[00 01 02 7e 5d 26] Source MAC address[00 08 dc 12 ab 08] Type : IP (0x0800) [08 00]
Internet Layer (IP)	Version dan Header length [45] Differentiated Services Field : 0x00 [00] Total length : 41 [00 29] Identification [00 f9] Flags [40] Fragment offset [00] Time to live [80] Protokol : TCP [06] Header checksum [e4 08] Source : 172.222.222.10 [ac de de 0a] Destination : 172.222.222.5[ac de de 05]
Transport Layer (TCP)	Source port [08 7f] Destination port [13 88] Sequence number [6f d4 2b a2] Ack number [85 38 1a f8] Header length [50] Flags [18] Window size [08 00] Checksum [0a 50]
	Data 1 byte [33] 33 hexadesimal merupakan kode ASCII dari karakter "0"

3. Paket data ke tiga

Waktu : 09:29:53.790426

Data : 00 01 02 7e 5d 26 00 08 dc 12 ab 08 08 00 45 00

00 29 00 fa 40 00 80 06 e4 07 ac de de 0a ac de

de 05 08 7f 13 88 6f d4 2b a3 85 38 1a f8 50 18

08 00 0c 4f 00 00 2e 00 00 00 00 00

Paket data diatas terbagi menjadi beberapa bagian :

Network Layer	Destination MAC address[00 01 02 7e 5d 26] Source MAC address[00 08 dc 12 ab 08] Type : IP (0x0800) [08 00]
Internet Layer (IP)	Version dan Header length [45] Differentiated Services Field : 0x00 [00] Total length : 41 [00 29] Identification [00 fa] Flags [40] Fragment offset [00] Time to live [80] Protokol : TCP [06] Header checksum [e4 07] Source : 172.222.222.10 [ac de de 0a]

	Destination : 172.222.222.5[ac de de 05]
Transport Layer (TCP)	Source port [08 7f] Destination port [13 88] Sequence number [6f d4 2b a3] Ack number [85 38 1a f8] Header length [50] Flags [18] Window size [08 00] Checksum [0c 4f]
	Data 1 byte [2e] 2e hexadesimal merupakan kode ASCII dari karakter "."

4. Paket data ke empat

Waktu : 09:29:53.791463

Data : 00 01 02 7e 5d 26 00 08 dc 12 ab 08 08 00 45 00

00 29 00 fb 40 00 80 06 e4 06 ac de de 0a ac de

de 05 08 7f 13 88 6f d4 2b a4 85 38 1a f8 50 18

08 00 09 4e 00 00 31 00 00 00 00 00

Paket data diatas terbagi menjadi beberapa bagian,

Network Layer	Destination MAC address[00 01 02 7e 5d 26] Source MAC address[00 08 dc 12 ab 08] Type : IP (0x0800) [08 00]
Internet Layer (IP)	Version dan Header length [45] Differentiated Services Field : 0x00 [00] Total length : 41 [00 29] Identification [00 fb] Flags [40] Fragment offset [00] Time to live [80] Protokol : TCP [06] Header checksum [e4 06] Source : 172.222.222.10 [ac de de 0a] Destination : 172.222.222.5[ac de de 05]
Transport Layer (TCP)	Source port [08 7f] Destination port [13 88] Sequence number [6f d4 2b a4] Ack number [85 38 1a f8] Header length [50] Flags [18] Window size [08 00] Checksum [09 4e]
	Data 1 byte [31] 31 hexadesimal merupakan kode ASCII dari karakter "1"

5. Paket data ke lima

Waktu : 09:29:53.792504

Data : 00 01 02 7e 5d 26 00 08 dc 12 ab 08 08 00 45 00

00 29 00 fc 40 00 80 06 e4 05 ac de de 0a ac de

de 05 08 7f 13 88 6f d4 2b a5 85 38 1a f8 50 18

08 00 06 4d 00 00 34 00 00 00 00 00

Paket data diatas terbagi menjadi beberapa bagian,

Network Layer	Destination MAC address[00 01 02 7e 5d 26] Source MAC address[00 08 dc 12 ab 08] Type : IP (0x0800) [08 00]
Internet Layer (IP)	Version dan Header length [45] Differentiated Services Field : 0x00 [00] Total length : 41 [00 29] Identification [00 fc] Flags [40] Fragment offset [00] Time to live [80] Protokol : TCP [06] Header checksum [e4 05] Source : 172.222.222.10 [ac de de 0a] Destination : 172.222.222.5[ac de de 05]
Transport Layer (TCP)	Source port [08 7f] Destination port [13 88] Sequence number [6f d4 2b a5] Ack number [85 38 1a f8] Header length [50] Flags [18] Window size [08 00] Checksum [06 4d]
	Data 1 byte [34] 34 hexadesimal merupakan kode ASCII dari karakter "4"

6. Paket data ke enam

Waktu : 09:29:53.793545

Data : 00 01 02 7e 5d 26 00 08 dc 12 ab 08 08 00 45 00

00 29 00 fd 40 00 80 06 e4 04 ac de de 0a ac de

de 05 08 7f 13 88 6f d4 2b a6 85 38 1a f8 50 18

08 00 30 4c 00 00 0a 00 00 00 00 00

Paket data diatas terbagi menjadi beberapa bagian,

Network Layer	Destination MAC address[00 01 02 7e 5d 26] Source MAC address[00 08 dc 12 ab 08] Type : IP (0x0800) [08 00]
Internet Layer (IP)	Version dan Header length [45] Differentiated Services Field : 0x00 [00] Total length : 41 [00 29] Identification [00 fd] Flags [40] Fragment offset [00] Time to live [80] Protokol : TCP [06] Header checksum [e4 04] Source : 172.222.222.10 [ac de de 0a] Destination : 172.222.222.5[ac de de 05]
Transport Layer (TCP)	Source port [08 7f] Destination port [13 88] Sequence number [6f d4 2b a6] Ack number [85 38 1a f8] Header length [50] Flags [18] Window size [08 00] Checksum [30 4c]
	Data 1 byte [0a] 0a hexadesimal merupakan kode ASCII dari karakter "Line feed"

Dari analisa data diatas terlihat bahwa data yang direkam pada kartu jaringan sama dengan data yang ditampilkan pada program yaitu 30.14.

4.6 Pengujian Sistem

4.6.1 Tujuan

Setelah melakukan pengujian pada tiap komponen, maka bagian ini akan melakukan pengujian pada sistem secara keseluruhan untuk mengetahui apakah perangkat yang telah kita rancang dan kita buat berjalan sesuai dengan yang diharapkan.

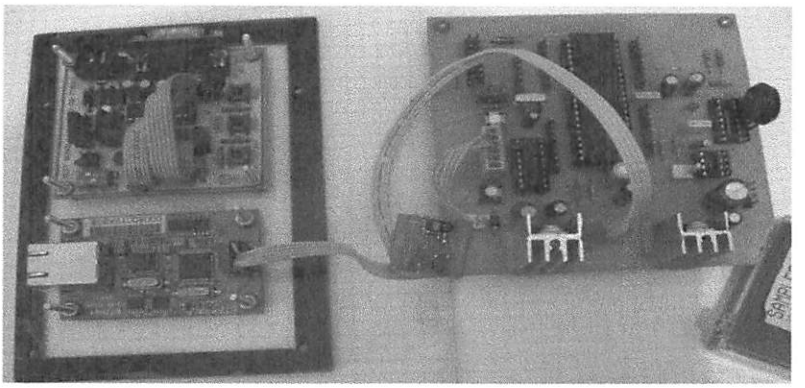
4.6.2 Peralatan Yang Digunakan

1. Rangakaian mikrokontoller ATmega16
2. Rangkaian sensor SHT11 dan MPX4100
3. Modul EG-SR 7150 MJ

4. Power Supply
5. PC

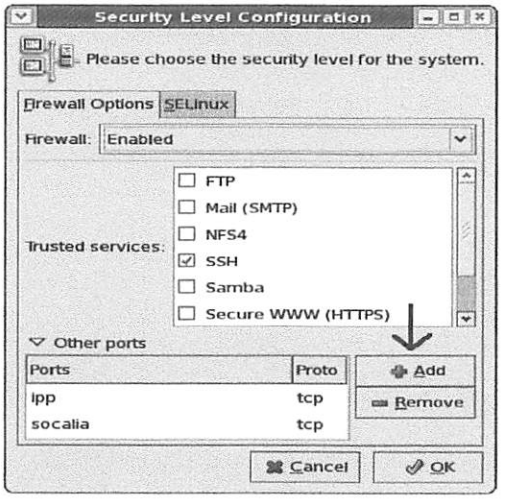
4.6.3 Langkah – langkah Percobaan

1. Hubungkan rangkaian mikrokontoller dengan modul wiznet



Gambar 4.15 *Rangkaian mikrokontoller dan Modul Wiznet*

2. Hubungkan modul wiznet dengan PC menggunakan kabel UTP
3. Pasang power supply pada rangkaian mikrokontoller
4. Pada PC atur *firewall,* untuk mengizinkan port yang kita gunakan pada program Client/Server. Untuk mengizinkan port yang digunakan klik *Add* atau kita *disable* saja. Seperti yang ditunjukan pada gambar 4.11.



Gambar 4.16 *Pengaturan firewall*

5. Jalankan program *Client/Server* yang telah kita buat dan lihat hasilnya.

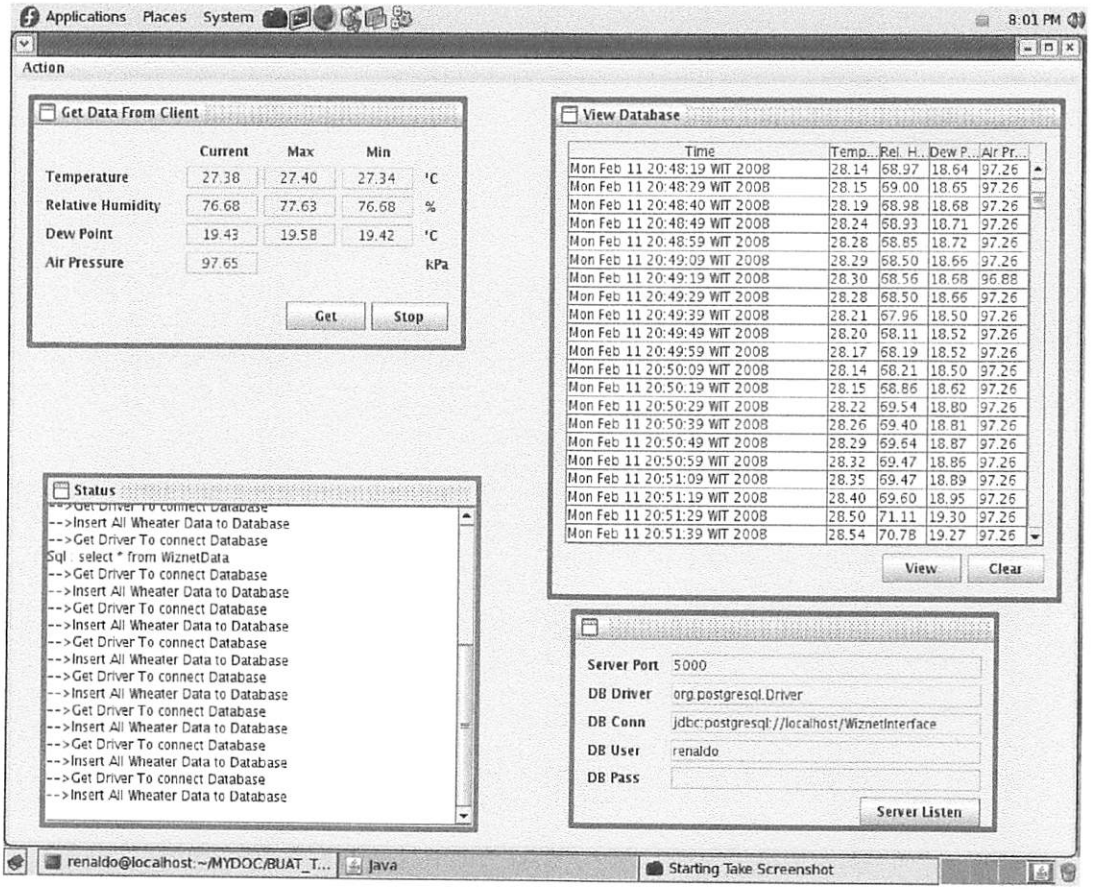
4.6.4 Hasil Pengujian

Pengujian pada sistem berhasil, karena sudah dapat mengukur suhu, kelembapan, dan tekanan udara. Sistem juga sudah dapat mengirimkan data hasil pengukuran untuk ditampilkan pada PC dan disimpan pada *data base*.

***Tabel 4.3** Hasil Pengukuran Suhu, Kelembapan, Titik Embun, dan Tekanan Udara*

Waktu	Suhu (°C)	Kelembapan (%)	Titik Embun (°C)	Tekanan (kPa)
Wed Feb 06 21:40:32 WIT 2008	29.02	73.18	20.05	96.50
Wed Feb 06 21:40:42 WIT 2008	29.02	73.20	20.05	95.74
Wed Feb 06 21:40:52 WIT 2008	29.02	73.26	0	96.88
Thu Feb 07 16:19:30 WIT 2008	30.33	66.97	19.88	96.88
Mon Feb 11 20:45:39 WIT 2008	28.14	68.56	18.56	97.26
Mon Feb 11 20:45:49 WIT 2008	28.13	68.27	18.50	96.88
Sat Feb 16 20:01:09 WIT 2008	27.34	76.97	19.45	97.26
Sat Feb 16 20:01:19 WIT 2008	27.35	76.75	19.42	97.65
Sat Feb 16 20:01:29 WIT 2008	27.36	76.98	19.46	97.26

LAMPIRAN



Gambar 4.17 *Program Client/Server Pada PC*

BAB V

PENUTUP

5.1 Kesimpulan

1. Pada komunikasi serial, penginisialisasian bautrate dilakukan pada modul wiznet dan mikrokontoller. Nilai bautrate harus sama, pada sistem bautrate yang digunakan adalah 9600 bps.
2. Pada ADC mikrokontoller ATmega16 rata-rata error 0% pada tegangan antara 0V-0,99V, 2,75% pada tegangan antara 1V-1,99V, 2,99% pada tegangan antara 1,2V-1,49V, 3,24% pada tegangan antara 1,5V-1,59V, dan 2,98% pada tegangan antara 1,6V-2,5V. Error yang terjadi meruapakan *gain error.*
3. Dengan penggunaan CRC8 pada proses pengambilan data sensor SHT11, maka apabila data mengalami kerusakan pada saat proses transmisi maka data tersebut tidak akan diproses.
4. Pada sensor MPX4100 nilai tegangan supply harus konstan antara 4,85V-5,1V , apabila terjadi perubahan nilai maka akan terjadi kesalahan pada saat proses perhitungan di mikrokontoller.

5.2 Saran

1. Sistem ini hanya bekerja dengan satu klient saja, pada kenyataannya diperlukan beberapa client untuk mendapatkan informasi cuaca pada suatu daerah. Diharapkan untuk pengembangan selanjutnya sistem dapat bekerja dengan beberapa client dan mengamati unsur iklim yang lain sehingga dapat

digunakan untuk melakukan peramalan cuaca.

2. Untuk fleksibilitas media transmisi yang digunakan adalah gelombang radio. Dengan penggunaan gelombang radio penempatan perangkat bisa lebih jauh dari kantor pengamatan asal segala unsur teknis dalam pembangunan sistem transmisi gelombang radio terpenuhi.

DAFTAR PUSTAKA

Bayong, T.H.K.,2004. **Klimatologi,** Penerbit ITB, Bandung,2004

Hayt, William dan Jack Kemmerly,1996, **Rangkaian Listrik,** Penerbit Erlangga,jakarta,1996

Wardhana, Lingga, 2006, **Belajar Sendiri Mikrokontroller AVR Seri ATMEGA8535,** Penerbit Andi, Yogyakarta, 2006

Cokorda dkk, 2000, **Trik Pemograman Java untuk Jaringan dan Internet,** Elex Media Komputindo, jakarta, 2000

Susanto, Budi,2003, **Pemograman Client/Server dengan Java 2,** Elex Media Komputindo,jakarta,2003

John Bibin,2006, **AVR Book,** www.geocities.com/njbibin

www.tuxgraphics.org/electronics

www.user.on.net/~symes/CwithAVR/IntroCwithAVR.htm

INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
JURUSAN TEKNIK ELEKTRO

FORMULIR BIMBINGAN SKRIPSI

Nama : Renaldo Ariai Panauhan
NIM : 03.17.084
Masa Bimbingan : 15 Agustus 2007 s/d 15 Januari 2008
Judul Skripsi : Perancangan Dan Pembuatan Alat Pengukur Suhu, Kelembapan dan Tekanan Udara Berbasis Protokol TCP/IP

NO	Tanggal	Uraian	Paraf Pembimbing
1	7/8 07.	Konsultasi judul, Gambar Blok diagram rangkaian.	
2	11/9 07.	Untuk mengguakan sensor suhu, amati Characteristiknya	
3	19/9 07.	Sebelum digunakan, Cekkan karakterisasi sensor tek & kelembaban.	
4	14/10 07.	Dari karakterisasi sensor buat grafik, bandingkan dg datasheet	
5	20/11 07.	Lakukan pengujian data, tulis di Bab IV, Hasilnya ditulis kimp	
6	14/12 07.	Pada kesimpulan, Cantumkan nilai kesalahan/ketelitian sensor.	
7	17/2 08.	Seminar hasil (tertutup).	
8	22/2 08.	Pada Bab II, Uraikan juga tg kondisi kelembaban udara	
9	3/3 08.	Tgaskan kembali bahan masalah di Bab I.	
10	22/3 08.	Konsultasi semua Bab, lanjut ujian	

Malang,
Dosen Pembimbing

20/3 08.

Ir. Sidik Noertjahjono, MT.
NIP. 1028700167

Form. S – 4b

INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
JURUSAN TEKNIK ELEKTRO

FORMULIR BIMBINGAN SKRIPSI

Nama : Renaldo Ariai Panauhan
NIM : 03.17.084
Masa Bimbingan : 15 Agustus 2007 s/d 15 Januari 2008
Judul Skripsi : Perancangan Dan Pembuatan Alat Pengukur Suhu, Kelembapan dan Tekanan Udara Berbasis Protokol TCP/IP

NO	Tanggal	Uraian	Paraf Pembimbing
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			

Malang,
Dosen Pembimbing

DR. Cahyo Crysdian, Msc
NIP. 1030400412

Form. S – 4b

INSTITUT TEKNOLOGI NASIONAL
FAKULTAS TEKNOLOGI INDUSTRI
JURUSAN TEKNIK ELEKTRO

Formulir Perbaikan Ujian Skripsi

Dalam pelaksanaan Ujian Skripsi Janjang Strata 1 Jurusan Teknik Elektro Konsentasi T. Energi Listrik / T Elektronika, maka perlu adanya perbaikan skripsi untuk mahasiswa :

NAMA : Renaldo Ariqi P.
N I M : 03170084
Perbaikan meliputi :

Malang,

( )

INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
JURUSAN TEKNIK ELEKTRO

FORMULIR BIMBINGAN SKRIPSI

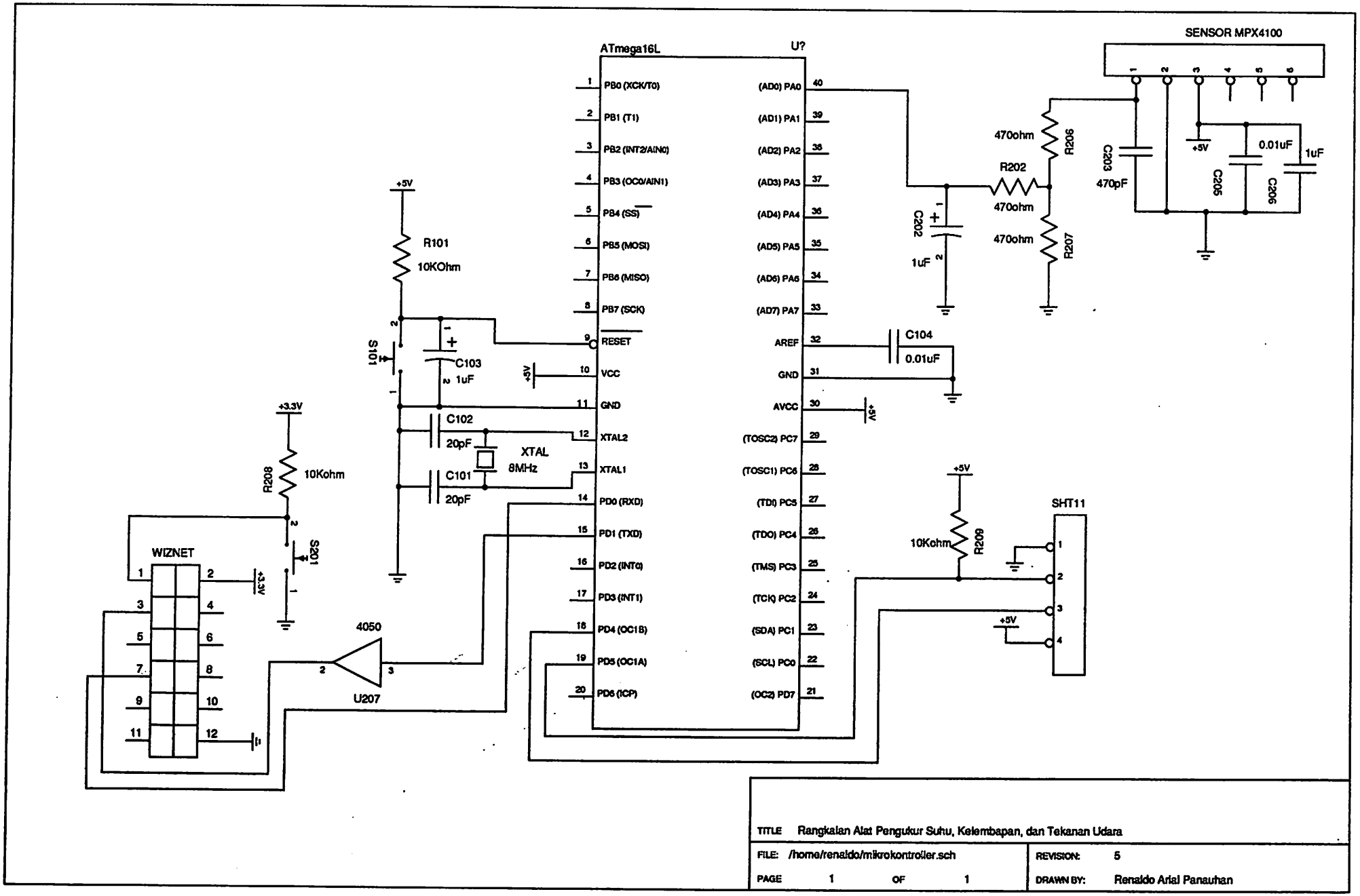
Nama : Renaldo Ariai Panauhan
NIM : 03.17.084
Masa Bimbingan : 15 Agustus 2007 s/d 15 Januari 2008
Judul Skripsi : Perancangan Dan Pembuatan Alat Pengukur Suhu, Kelembapan dan Tekanan Udara Berbasis Protokol TCP/IP

NO	Tanggal	Uraian	Paraf Pembimbing
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			

Malang,
Dosen Pembimbing

DR. Cahyo Crysdian, Msc
NIP. 1030400412

Form. S – 4b


ATmega16L
U?
SENSOR MPX4100
PB0 (XCK/T0)
PB1 (T1)
PB2 (INT2/AIN0)
PB3 (OC0/AIN1)
PB4 (SS)
PB5 (MOSI)
PB6 (MISO)
PB7 (SCK)
RESET
VCC
GND
XTAL2
XTAL1
PD0 (RXD)
PD1 (TXD)
PD2 (INT0)
PD3 (INT1)
PD4 (OC1B)
PD5 (OC1A)
PD6 (ICP)
(AD0) PA0
(AD1) PA1
(AD2) PA2
(AD3) PA3
(AD4) PA4
(AD5) PA5
(AD6) PA6
(AD7) PA7
AREF
GND
AVCC
(TOSC2) PC7
(TOSC1) PC6
(TDI) PC5
(TDO) PC4
(TMS) PC3
(TCK) PC2
(SDA) PC1
(SCL) PC0
(OC2) PD7
R101
10KOhm
C103
1uF
S101
C102
20pF
XTAL
8MHz
C101
20pF
R208
10Kohm
S201
+3.3V
WIZNET
4050
U207
C202
1uF
R202
470ohm
R206
470ohm
R207
470ohm
C203
470pF
C205
0.01uF
C206
1uF
C104
0.01uF
R209
10Kohm
SHT11
+5V
TITLE Rangkaian Alat Pengukur Suhu, Kelembapan, dan Tekanan Udara
FILE: /home/renaldo/mikrokontroller.sch
REVISION: 5
PAGE 1 OF 1
DRAWN BY: Renaldo Arial Panauhan

```
#include <stdlib.h>
#include <math.h>
#include "sensirion_protocol.c"
void Init_ADC(void);
int Read_ADC(void);
void InitUART( unsigned char baudrate );
void KirimSerial(char *Teks);
char ReceiveByte( void );
void TransmitByte( char dataByte );
void GetData();
void ErroR(char Eror);
char n=1;

main(){
        //Inisialisasi
        Init_ADC();
        InitUART(51);
        char tombol;
        while(1){
                tombol=ReceiveByte();
                if(tombol=='G'){
                        GetData();
                }
        }
        return 0;
}

void Init_ADC(void){
        ADMUX|=(_BV(ADLAR) | _BV(REFS1)| _BV(REFS0)) ;
        ADMUX&=~(_BV(MUX0)|_BV(MUX1)|_BV(MUX2)|_BV(MUX3)|_BV(MUX4));
        ADCSRA|=(_BV(ADEN) | _BV(ADPS2) | _BV(ADPS0)) ;
        ADCSRA&=~(_BV(ADSC)|_BV(ADATE)|_BV(ADIF)|_BV(ADIE)|_BV(ADPS1));
}
int Read_ADC(void){
        ADCSRA|=_BV(ADSC);
        while(!((ADCSRA&_BV(ADIF))>>ADIF));
        return ADCH;
}



void InitUART( unsigned char baudrate ){
        UBRRL =  baudrate;
        UCSRB = (UCSRB | _BV(RXEN) | _BV(TXEN) );
}
char ReceiveByte( void ){
        while ( !(UCSRA & (_BV(RXC))) );
        return UDR;
}
void TransmitByte( char dataByte ){
        while ( !(UCSRA & (_BV(UDRE))) );
        UDR =  dataByte;
}
void KirimSerial(char *Teks){
        char i=0;
        while(Teks[i]!=0){
        TransmitByte(Teks[i]);
        i++;
  }
}
```

```
void GetData(){
        float Tekanan,Suhu,Kelembapan,DP,H,Suhu_Max,Suhu_Min;
        float DP_Max,DP_Min,Kelembapan_Max,Kelembapan_Min;
        char Press[6],Temp[6],RH[6],Dp[6],Temp_Max[6],Temp_Min[6],RH_Max[6],RH_Min[6];
        char Dp_Max[6],Dp_Min[6];
        unsigned int humival_raw,tempval_raw,inADC;
        unsigned char error;
        //Tekanan Udara
        inADC=Read_ADC();
        Tekanan=inADC*0.01;
        if(Tekanan<1){
                Tekanan=Tekanan;
        }
        else if(Tekanan<1.2){
                Tekanan=Tekanan-0.03;
        }
        else if(Tekanan<1.5){
                Tekanan=Tekanan-0.04;
        }
        else if(Tekanan<1.6){
                Tekanan=Tekanan-0.05;
        }
        else if(Tekanan<2.5){
                Tekanan=Tekanan-0.06;
        }
        Tekanan=((((Tekanan*2)+0.0784719)/4.94)+0.1518)/0.01059;
        dtostrf(Tekanan,5,2,Press);

        //proses pengukuran pada sensor sht11
        error=s_measure(&tempval_raw,0);
        if (error==0){
                error=s_measure(&humival_raw,1);
                if(error>0){
                        ErroR(error);
                }
        }
        else{
                ErroR(error);
        }

        //proses untuk suhu dan kelembapan
        Suhu=calc_sth11_temp(tempval_raw);
        dtostrf(Suhu,4,2,Temp);

        Kelembapan=rhcalc_int(humival_raw,Suhu);
        dtostrf(Kelembapan,4,2,RH);

        H=(log10(Kelembapan)-2)/0.4343+(17.62*Suhu)/(243.12+Suhu);
        DP=(243.12*H)/17.62-H;
        dtostrf(DP,4,2,Dp);
        //---memproses Max dan Min----
        if(n==1){
                Suhu_Max=Suhu;
                Suhu_Min=Suhu;
                Kelembapan_Max=Kelembapan;
                Kelembapan_Min=Kelembapan;
                DP_Max=DP;
                DP_Min=DP;
                n=2;
        }
```

```
        if(Suhu > Suhu_Max) Suhu_Max=Suhu;
        if(Suhu < Suhu_Min) Suhu_Min=Suhu;
        if(Kelembapan > Kelembapan_Max) Kelembapan_Max=Kelembapan;
        if(Kelembapan < Kelembapan_Min) Kelembapan_Min=Kelembapan;
        if(DP > DP_Max) DP_Max=DP;
        if(DP < DP_Min) DP_Min=DP;

        //---convert all to string---- ~_^  ----<<@
        dtostrf(Suhu_Max,4,2,Temp_Max);
        dtostrf(Suhu_Min,4,2,Temp_Min);
        dtostrf(Kelembapan_Max,4,2,RH_Max);
        dtostrf(Kelembapan_Min,4,2,RH_Min);
        dtostrf(DP_Max,4,2,Dp_Max);
        dtostrf(DP_Min,4,2,Dp_Min);
        //----write to serial----
        KirimSerial(Temp);
        KirimSerial("\n");
        KirimSerial(Temp_Max);
        KirimSerial("\n");
        KirimSerial(Temp_Min);
        KirimSerial("\n");
        KirimSerial(RH);
        KirimSerial("\n");
        KirimSerial(RH_Max);
        KirimSerial("\n");
        KirimSerial(RH_Min);
        KirimSerial("\n");

        KirimSerial(Dp);
        KirimSerial("\n");
        KirimSerial(Dp_Max);
        KirimSerial("\n");
        KirimSerial(Dp_Min);
        KirimSerial("\n");

        KirimSerial(Press);
        KirimSerial("\n");

}

void ErroR(char Eror){
        char i;
        for(i=10;i>0;i++){
                KirimSerial(Eror);
                KirimSerial("\n");
        }
}
```

```
#include <avr/io.h>
#define F_CPU 8000000UL
#include <util/delay.h>

#define STATUS_REG_W 0x06 //000 0011 0
#define STATUS_REG_R 0x07 //000 0011 1
#define MEASURE_TEMP 0x03 //000 0001 1
#define MEASURE_HUMI 0x05 //000 0010 1

#define SETSCK1 PORTD|=(1<<PD4)
#define SETSCK0 PORTD&=~(1<<PD4)
#define SCKOUTP DDRD|=(1<<DDD4)

#define SETDAT1 PORTD|=(1<<PD5)
#define SETDAT0 PORTD&=~(1<<PD5)
#define GETDATA (PIND&(1<<PIND5))

#define DMODEIN DDRD&=~(1<<DDD5)
#define PULLUP1 PORTD|=(1<<PIND5)
#define DMODEOU DDRD|=(1<<DDD5)


#define S_PULSLONG _delay_us(3.0)
#define S_PULSSHORT _delay_us(1.0)


unsigned char computeCRC8(unsigned char inData, unsigned char seed)
{
    unsigned char bitsLeft;
    unsigned char tmp;

    for (bitsLeft = 8; bitsLeft > 0; bitsLeft--)
    {
        tmp = ((seed ^ inData) & 0x01);
        if (tmp == 0)
        {
            seed >>= 1;
        }
        else
        {
            seed ^= 0x18;
            seed >>= 1;
            seed |= 0x80;
        }
        inData >>= 1;
    }
    return seed;
}


unsigned char bitswapbyte(unsigned char byte)
{
        unsigned char i=8;
        unsigned char result=0;
        while(i){
                result=(result<<1);
                if (1 & byte) {
                        result=result | 1;
                }
                i--;
                byte=(byte>>1);
```

```
        }
        return(result);
}


char s_write_byte(unsigned char value)
{
        unsigned char i=0x80;
        unsigned char error=0;
        DMODEOU;
        while(i){
                if (i & value) {
                        SETDAT1;
                }else{
                        SETDAT0;
                }
                SETSCK1;
                S_PULSLONG;
                SETSCK0;
                S_PULSSHORT;
                i=(i>>1);
        }
        DMODEIN;
        PULLUP1;
        SETSCK1;
        S_PULSLONG;
        if (GETDATA){
                error=1;
        }
        S_PULSSHORT;
        SETSCK0;
        return(error);
}


unsigned char s_read_byte(unsigned char ack)
{
        unsigned char i=0x80;
        unsigned char val=0;
        DMODEIN;
        PULLUP1;
        while(i){
                SETSCK1;
                S_PULSSHORT;
                if (GETDATA){
                        val=(val | i);
                }
                SETSCK0;
                S_PULSSHORT;
                i=(i>>1);
        }
        DMODEOU;
        if (ack){

                SETDAT0;
        }else{
                SETDAT1;
        }
        SETSCK1;
        S_PULSLONG;
        SETSCK0;
```

```
        S_PULSSHORT;
        DMODEIN;
        PULLUP1;
        return (val);
}


void s_transstart(void)
{

        SCKOUTP;
        SETSCK0;
        DMODEOU;
        SETDAT1;
        //
        S_PULSSHORT;
        SETSCK1;
        S_PULSSHORT;
        SETDAT0;
        S_PULSSHORT;
        SETSCK0;
        S_PULSLONG;
        SETSCK1;
        S_PULSSHORT;
        SETDAT1;
        S_PULSSHORT;
        SETSCK0;
        S_PULSSHORT;
        //
        DMODEIN;
        PULLUP1;
}



char s_measure(unsigned int *p_value, unsigned char mode)
{
        unsigned char i=0;
        unsigned char msb,lsb;
        unsigned char checksum;

        unsigned char crc_state=0;
        *p_value=0;
        s_transstart();
        if(mode){
                mode=MEASURE_HUMI;
        }else{
                mode=MEASURE_TEMP;
        }
        if (s_write_byte(mode)){
                return(1);
        }
        crc_state=computeCRC8(bitswapbyte(mode),crc_state);

        while(i<240){
                _delay_ms(3.0);
                if (GETDATA==0){
                        i=0;
                        break;
                }
```

```
                i++;
        }
        if(i){

                return(3);
        }
        msb=s_read_byte(1);
        crc_state=computeCRC8(bitswapbyte(msb),crc_state);
        lsb=s_read_byte(1);
        *p_value=(msb<<8)|(lsb);
        crc_state=computeCRC8(bitswapbyte(lsb),crc_state);
        checksum =s_read_byte(0);
        if (crc_state != checksum ) {
                return(2);
        }
        return(0);
}


float calc_sth11_temp(unsigned int t)
{
        float Suhu;
        Suhu=(t*0.01)-40;
        return Suhu;
}

float rhcalc_int(unsigned int s,float T)
{
        float Kelembapan,Kelembapan2;
        Kelembapan=(-0.0000028*s*s) +(0.0405*s) - 4.0;
        Kelembapan2=(T-25)*(0.01+(0.00008*s))+Kelembapan;
        return Kelembapan2;

}
```

```
/*
 * DataAccuisition.java
 *
 * Created on December 28, 2007, 7:29 PM
 */

package MyJavaSoft;
import java.net.*;
import javax.swing.Timer;
import java.io.*;
import java.sql.*;
import javax.swing.table.DefaultTableModel;

/**
 *
 * @author  Renaldo Ariai Panauhan
 */
public class DataAccuisition extends javax.swing.JFrame {
    char menu1=1;
    char menu2=1;
    char menu3=1;
    char menu4=1;
    String DBDriver,DBConn,DBUser,DBPass;

    //soket dan timer
    Socket WiznetInterface;
    BufferedReader FromWiznet;
    DataOutputStream ToWiznet;
    Timer time;

    final String[]header={"Time","Temperature","Rel. Humidity","Dew Point","Air
Pressure"};
    DefaultTableModel tabMode;

    /** Creates new form DataAccuisition */
    public DataAccuisition() {
        initComponents();
    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-
BEGIN:initComponents
    private void initComponents() {
          //this code generate by netbean
    }// </editor-fold>//GEN-END:initComponents

    private void MenuServerListenActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_MenuServerListenActionPerformed
          if(menu4==1){
             IFServerListen.setVisible(false);
             menu4=2;
        }
        else {
             IFServerListen.setVisible(true);
             menu4=1;
        }
    }//GEN-LAST:event_MenuServerListenActionPerformed
```

```
    private void ButtonServerListenActionPerformed(java.awt.event.ActionEvent evt) {//
GEN-FIRST:event_ButtonServerListenActionPerformed
        try{
                String PortServer=TFServerPort.getText();
                int ServerPort =Integer.valueOf(PortServer).intValue();
                ServerSocket WiznetServer=new ServerSocket(ServerPort);
                //--get value from tF
                DBDriver=TFDBDriver.getText();
                DBConn=TFDBConn.getText();
                DBUser=TFDBUser.getText();
                DBPass=TFDBPass.getText();
                TFServerPort.setEditable(false);
                TFDBDriver.setEditable(false);
                TFDBConn.setEditable(false);
                TFDBUser.setEditable(false);
                TFDBPass.setEditable(false);
                ButtonGet.setEnabled(true);
                ButtonStop.setEnabled(true);
                WiznetInterface=WiznetServer.accept();
                TAStatus.append("-->Get Connection From Client\n");

        }
        catch(Exception e){
            TAStatus.append("@Error Running Server\n");
            TAStatus.append("@Chek your Sistem Configuration");
        }
    }//GEN-LAST:event_ButtonServerListenActionPerformed


    private void ButtonClearActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_ButtonClearActionPerformed
         tabMode = new DefaultTableModel(null,header);
         TableDatabase.setModel(tabMode);
         TAStatus.append("-->All Table has been deleted\n");
    }//GEN-LAST:event_ButtonClearActionPerformed

    private void ButtonViewActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_ButtonViewActionPerformed
        Object[][]data=new Object[0][0];
        int n;
        try{
                Class.forName(DBDriver);
                String koneksi=DBConn;
                String user=DBUser;
                String pass=DBPass;
                TAStatus.append("-->Get Driver To connect Database\n");
                Connection connection = DriverManager.getConnection(koneksi,user,pass);
                Statement statement = connection.createStatement();
                TAStatus.append("Sql : select * from WiznetData\n");
                String DataCuaca="select * from WiznetData;";
                ResultSet rs=statement.executeQuery(DataCuaca);
               //       rs.last();
                //n=rs.getRow();
                data=new Object[10000][5];
                int p=0;
                //rs.beforeFirst();
                while(rs.next()){
                    data[p][0]=rs.getString("waktu");
                    data[p][1]=rs.getString("Temp");
                    data[p][2]=rs.getString("Humidity");
```

```
                    data[p][3]=rs.getString("DewPoint");
                    data[p][4]=rs.getString("AirPressure");
                    p++;
                }
        }
        catch(Exception e){
            TAStatus.append("@Data Base Error\n");
        }
        tabMode = new DefaultTableModel(data,header);
        TableDatabase.setModel(tabMode);

    }//GEN-LAST:event_ButtonViewActionPerformed

    private void ButtonStopActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_ButtonStopActionPerformed
        try{
                WiznetInterface.close();
                FromWiznet.close();
                ToWiznet.close();
                time.stop();
                ButtonGet.setEnabled(false);
                ButtonStop.setEnabled(false);
                TFServerPort.setEditable(true);
                TFDBDriver.setEditable(true);
                TFDBConn.setEditable(true);
                TFDBUser.setEditable(true);
                TFDBPass.setEditable(true);
                TFTempCur.setText("");
                TFTempMax.setText("");
                TFTempMin.setText("");
                TFHumCur.setText("");
                TFHumMax.setText("");
                TFHumMin.setText("");
                TFDewCur.setText("");
                TFDewMax.setText("");
                TFDewMin.setText("");
                TFPressure.setText("");
        }
        catch(Exception e){
        }
    }//GEN-LAST:event_ButtonStopActionPerformed

    private void ButtonGetActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_ButtonGetActionPerformed
        getData();
        RegetData();
    }//GEN-LAST:event_ButtonGetActionPerformed

    public void getData(){
        try{
                FromWiznet = new BufferedReader(new InputStreamReader
(WiznetInterface.getInputStream()));
                ToWiznet = new DataOutputStream(WiznetInterface.getOutputStream());
                ToWiznet.writeBytes("G");
                //-------get Temperature-----------
                String TempCur = FromWiznet.readLine();
                String TempMax = FromWiznet.readLine();
                String TempMin = FromWiznet.readLine();
                //------get humidity---------
                String HumCur = FromWiznet.readLine();
                String HumMax = FromWiznet.readLine();
```

```
                String HumMin = FromWiznet.readLine();
                //-----get dew point----------
                String DewCur = FromWiznet.readLine();
                String DewMax = FromWiznet.readLine();
                String DewMin = FromWiznet.readLine();
                //-----get Air Pressure-----
                String AirPressure = FromWiznet.readLine();
                //----Set Time for data arrival---
                String Waktu = new java.util.Date().toString();
                //----Write all to TextField----
                //----add temperature value to TextField
                TFTempCur.setText(TempCur);
                TFTempMax.setText(TempMax);
                TFTempMin.setText(TempMin);
                //---add humidity value to TextField
                TFHumCur.setText(HumCur);
                TFHumMax.setText(HumMax);
                TFHumMin.setText(HumMin);
                //---add Dew Point Value to TextField
                TFDewCur.setText(DewCur);
                TFDewMax.setText(DewMax);
                TFDewMin.setText(DewMin);
                //---add Air value to TextField
                TFPressure.setText(AirPressure);
                //---add all to database---
                //
                Class.forName(DBDriver);
                String koneksi=DBConn;
                String user=DBUser;
                String pass=DBPass;
                TAStatus.append("-->Get Driver To connect Database\n");
                Connection connection = DriverManager.getConnection(koneksi,user,pass);
                Statement statement = connection.createStatement();
                String DataCuaca="insert into WiznetData values('"+Waktu+"','"+TempCur
+"','"+HumCur+"','"+DewCur+"','"+AirPressure+"');";
                statement.executeUpdate(DataCuaca);
                statement.close();
                connection.close();
                TAStatus.append("-->Insert All Wheater Data to Database\n");
        }
        catch(Exception e){
                TAStatus.append("@Data Base Error\n");
        }
    }

    public void RegetData(){
        // Action Listener untuk keperluan Timer
        java.awt.event.ActionListener Tm = new java.awt.event.ActionListener()  {
            @Override
                public void actionPerformed(java.awt.event.ActionEvent evt) {
                    getData();
                }
        };
        // Membuat Timer dengan waktu delay = 1 milisecond
        time = new Timer(1000,Tm);
        // Menjalankan Timer
        time.start();
    }
    private void MenuExitActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_MenuExitActionPerformed
        System.exit(0);
```

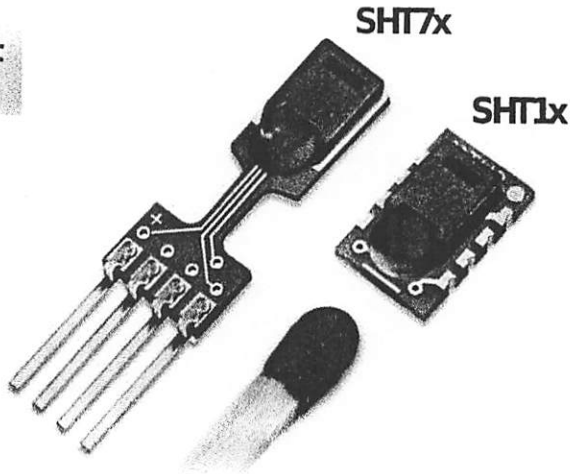
```
    }//GEN-LAST:event_MenuExitActionPerformed

    private void MenuStatusActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_MenuStatusActionPerformed
        if(menu3==1){
             IFStatus.setVisible(false);
             menu3=2;
        }
        else {
             IFStatus.setVisible(true);
             menu3=1;
        }
    }//GEN-LAST:event_MenuStatusActionPerformed

    private void MenuViewDataActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_MenuViewDataActionPerformed
        if(menu2==1){
             IFViewDataBase.setVisible(false);
             menu2=2;
        }
        else {
             IFViewDataBase.setVisible(true);
             menu2=1;
        }
    }//GEN-LAST:event_MenuViewDataActionPerformed

    private void MenuGetDataActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_MenuGetDataActionPerformed
        if(menu1==1){
             IFGetData.setVisible(false);
             menu1=2;
        }
        else {
             IFGetData.setVisible(true);
             menu1=1;
        }
    }//GEN-LAST:event_MenuGetDataActionPerformed


    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            @Override
            public void run() {
                new DataAccuisition().setVisible(true);
            }
        });
    }


}
```

SENSIRION
THE SENSOR COMPANY

SHT1x / SHT7x

Humidity & Temperature Sensor

Evaluation Kit Available



- Relative humidity and temperature sensors
- Dew point
- Fully calibrated, digital output
- Excellent long-term stability
- No external components required
- Ultra low power consumption
- Surface mountable or 4-pin fully interchangeable
- Small size
- Automatic power down

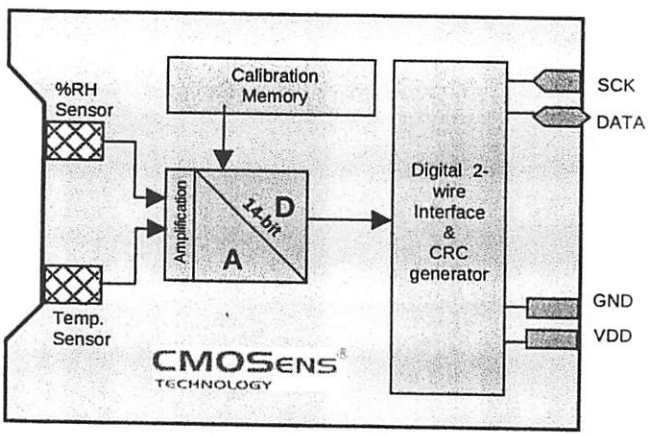
SHT1x / SHT7x Product Summary

The SHTxx is a single chip relative humidity and temperature multi sensor module comprising a calibrated digital output. Application of industrial CMOS processes with patented micro-machining (CMOSens® technology) ensures highest reliability and excellent long term stability. The device includes a capacitive polymer sensing element for relative humidity and a bandgap temperature sensor. Both are seamlessly coupled to a 14bit analog to digital converter and a serial interface circuit on the same chip. This results in superior signal quality, a fast response time and insensitivity to external disturbances (EMC) at a very competitive price.
Each SHTxx is individually calibrated in a precision humidity chamber. The calibration coefficients are programmed into the OTP memory. These coefficients are used internally during measurements to calibrate the signals from the sensors.
The 2-wire serial interface and internal voltage regulation allows easy and fast system integration. Its tiny size and low power consumption makes it the ultimate choice for even the most demanding applications.
The device is supplied in either a surface-mountable LCC (Leadless Chip Carrier) or as a pluggable 4-pin single-in-line type package. Customer specific packaging options may be available on request.

Applications

_ HVAC
_ Automotive
_ Consumer Goods
_ Weather Stations
_ Humidifiers
_ Dehumidifiers
_ Test & Measurement
_ Data Logging
_ Automation
_ White Goods
_ Medical

Block Diagram



Ordering Information

Part Number	Humidity accuracy [%RH]	Temperature accuracy [K] @ 25 °C	Package
SHT10	±4.5	±0.5	SMD (LCC)
SHT11	±3.0	±0.4	SMD (LCC)
SHT15	±2.0	±0.3	SMD (LCC)
SHT71	±3.0	±0.4	4-pin single-in-line
SHT75	±1.8	±0.3	4-pin single-in-line

1 Sensor Performance Specifications

Parameter	Conditions	Min.	Typ.	Max.	Units
Humidity					
Resolution (1)		0.5	0.03	0.03	%RH
		8	12	12(2)	bit
Repeatability			±0.1		%RH
Accuracy (3) Uncertainty	linearized	see figure 1			
Interchangeability		Fully interchangeable			
Nonlinearity	raw data		±3		%RH
	linearized		<<1		%RH
Range		0		100	%RH
Response time	1/e (63%) at 25°C, 1m/s air	6	8	10	s
Hysteresis			±1		%RH
Long term stability	typical		< 0.5		%RH/yr
Temperature					
Resolution (1)		0.04	0.01	0.01	°C
		0.07	0.02	0.02	°F
		12	14	14	bit
Repeatability			±0.1		°C
			±0.2		°F
Accuracy (3)		see figure 1			
Range		-40		123.8	°C
		-40		254.9	°F
Response Time	1/e (63%)	5		30	s

Table 1 Sensor Performance Specifications

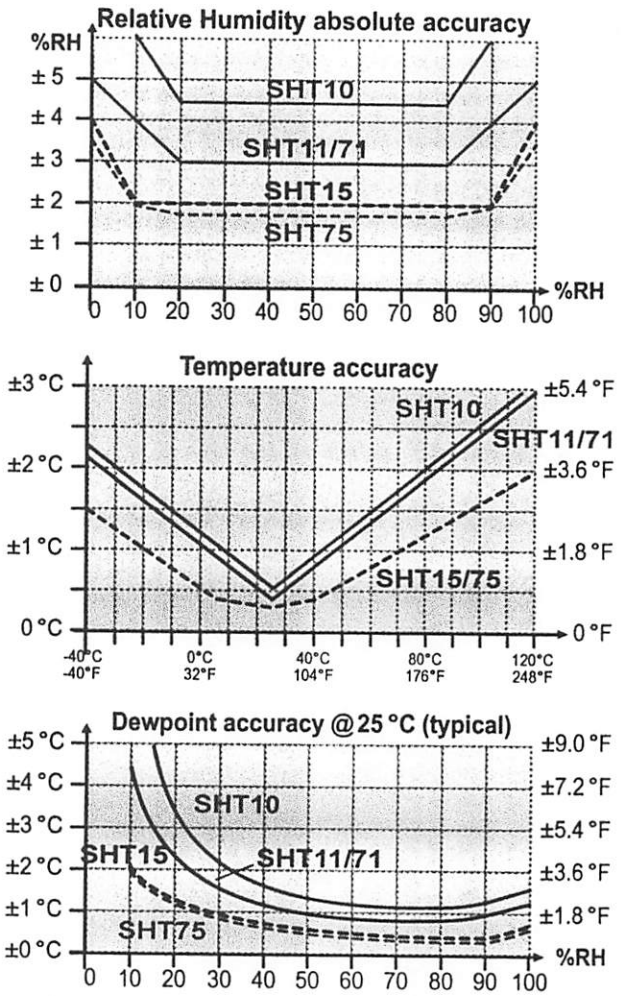


Figure 1 Rel. Humidity, Temperature and Dewpoint accuracies

2 Interface Specifications

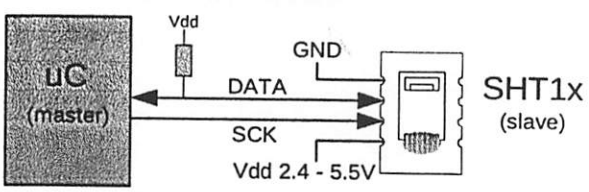


Figure 2 Typical application circuit

2.1 Power Pins

The SHTxx requires a voltage supply between 2.4 and 5.5 V. **After powerup the device needs 11ms to reach its "sleep" state. No commands should be sent before that time.**
Power supply pins (VDD, GND) may be decoupled with a 100 nF capacitor.

2.2 Serial Interface (Bidirectional 2-wire)

The serial interface of the SHTxx is optimized for sensor readout and power consumption and is not compatible with I²C interfaces, see FAQ for details.

2.2.1 Serial clock input (SCK)

The SCK is used to synchronize the communication between a microcontroller and the SHTxx. Since the interface consists of fully static logic there is no minimum SCK frequency.

2.2.2 Serial data (DATA)

The DATA tristate pin is used to transfer data in and out of the device. DATA **changes after the falling edge** and is **valid on the rising edge** of the serial clock SCK. During transmission the DATA line must remain stable while SCK is high. To avoid signal contention the microcontroller should only drive DATA low. An external pull-up resistor (e.g. 10 kΩ) is required to pull the signal high. (See Figure 2) Pull-up resistors are often included in I/O circuits of microcontrollers.
See Table 5 for detailed IO characteristics

(1) The default measurement resolution of 14bit (temp.) and 12bit (humidity) can be reduced to 12 and 8 bit through the status register. (2) Effective number of bits is 11bit.
(3) Each SHTxx is tested to be fully within accuracy specifications at 25°C (77°F) and 3.3V.

2.2.3 Sending a command

To initiate a transmission, a "Transmission Start" sequence has to be issued. It consists of a lowering of the DATA line while SCK is high, followed by a low pulse on SCK and raising DATA again while SCK is still high.

Figure 3 "Transmission Start" sequence

The subsequent command consists of three address bits (only "000" is currently supported) and five command bits. The SHTxx indicates the proper reception of a command by pulling the DATA pin low (ACK bit) after the falling edge of the 8th SCK clock. The DATA line is released (and goes high) after the falling edge of the 9th SCK clock.

Command	Code
Reserved	0000x
Measure Temperature	**00011**
Measure Humidity	**00101**
Read Status Register	00111
Write Status Register	00110
Reserved	0101x-1110x
Soft reset, resets the interface, clears the status register to default values wait minimum 11 ms before next command	**11110**

Table 2 SHTxx list of commands

2.2.4 Measurement sequence (RH and T)

After issuing a measurement command ('00000101' for RH, '00000011' for Temperature) the controller has to wait for the measurement to complete. This takes approximately 11/55/210 ms for a 8/12/14bit measurement. The exact time varies by up to ±15% with the speed of the internal oscillator. To signal the completion of a measurement, the SHTxx pulls down the data line and enters idle mode. The controller **must** wait for this "data ready" signal before restarting SCK to readout the data. Measurement data is stored until readout, therefore the controller can continue with other tasks and readout as convenient.

Two bytes of measurement data and one byte of CRC checksum will then be transmitted. The uC must acknowledge each byte by pulling the DATA line low. All values are MSB first, right justified. (e.g. the 5th SCK is MSB for a 12bit value, for a 8bit result the first byte is not used). Communication terminates after the acknowledge bit of the CRC data. If CRC-8 checksum is not used the controller may terminate the communication after the measurement data LSB by keeping ack high.
The device automatically returns to sleep mode after the measurement and communication have ended.

Warning: To keep self heating below 0.1 °C the SHTxx should not be active for more than 10% of the time (e.g. max. 2 measurements / second for 12bit accuracy).

2.2.5 Connection reset sequence

If communication with the device is lost the following signal sequence will reset its serial interface:
While leaving DATA high, toggle SCK 9 or more times. This must be followed by a "Transmission Start" sequence preceding the next command. This sequence resets the interface only. The status register preserves its content.

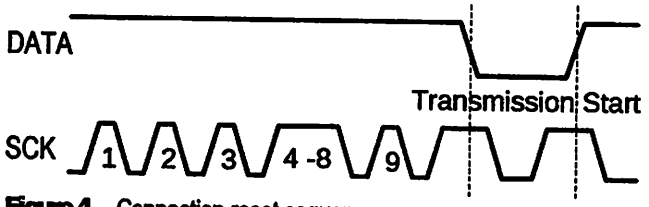


Figure 4 Connection reset sequence

2.2.6 CRC-8 Checksum calculation

The whole digital transmission is secured by a 8 bit checksum. It ensures that any wrong data can be detected and eliminated.
Please consult application note "CRC-8 Checksum Calculation" for information on how to calculate the CRC.

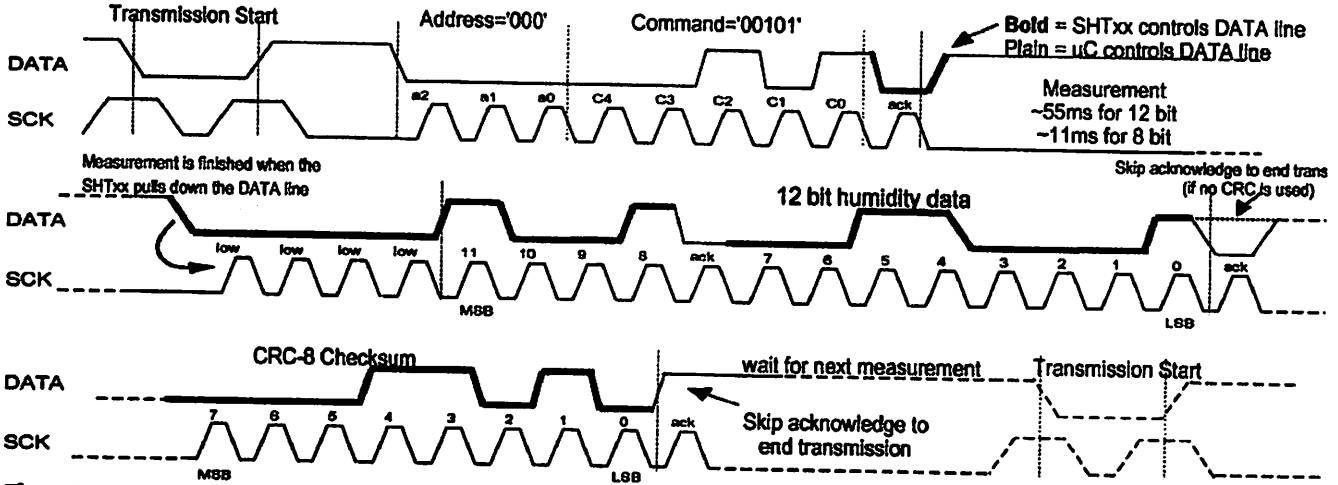


Figure 5 Example RH measurement sequence for value "0000'1001 ' 0011'0001"= 2353 = 75.79 %RH (without temperature compensation)

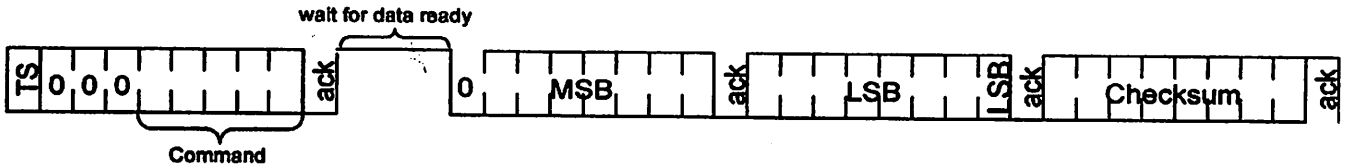


Figure 6 Overview of Measurement Sequence (TS = Transmission Start)

2.3 Status Register

Some of the advanced functions of the SHTxx are available through the status register. The following section gives a brief overview of these features. A more detailed description is available in the application note "Status Register"

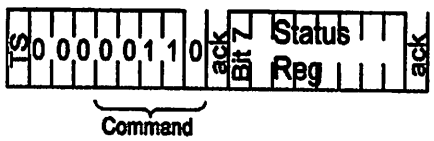


Figure 7 Status Register Write

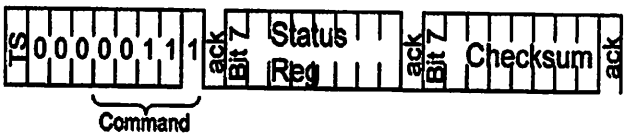


Figure 8 Status Register Read

Bit	Type	Description	Default	
7		reserved	0	
6	R	End of Battery (low voltage detection) '0' for Vdd > 2.47 '1' for Vdd < 2.47	X	No default value, bit is only updated after a measurement
5		reserved	0	
4		reserved	0	
3		For Testing only, do not use	0	
2	R/W	Heater	0	off
1	R/W	no reload from OTP	0	reload
0	R/W	'1' = 8bit RH / 12bit Temperature resolution '0' = 12bit RH / 14bit Temperature resolution	0	12bit RH 14bit Temp.

Table 3 Status Register Bits

2.3.1 *Measurement resolution*

The default measurement resolution of 14bit (temperature) and 12bit (humidity) can be reduced to 12 and 8bit. This is especially useful in high speed or extreme low power applications.

2.3.2 *End of Battery*

The "End of Battery" function detects VDD voltages below 2.47 V. Accuracy is ±0.05 V

2.3.3 *Heater*

An on chip heating element can be switched on. It will increase the temperature of the sensor by 5-15 °C (9-27 °F). Power consumption will increase by ~8 mA @ 5 V.
Applications:
By comparing temperature and humidity values before and after switching on the heater, proper functionality of both sensors can be verified.

- In high (>95 %RH) RH environments heating the sensor element will prevent condensation, improve response time and accuracy

Warning: While heated the SHTxx will show higher temperatures and a lower relative humidity than with no heating.

2.4 Electrical Characteristics[1]

VDD=5V, Temperature = 25 °C unless otherwise noted

Parameter	Conditions	Min.	Typ.	Max.	Units
Power supply DC		2.4	5	5.5[2]	V
Supply current	measuring		550		µA
	average	2[3]	28[4]		µA
	sleep		0.3	1	µA
Low level output voltage		0		20%	Vdd
High level output voltage		75%		100%	Vdd
Low level input voltage	Negative going	0		20%	Vdd
High level input voltage	Positive going	80%		100%	Vdd
Input current on pads				1	µA
Output peak current	on			4	mA
	Tristated (off)		10		µA

Table 4 SHTxx DC Characteristics

	Parameter	Conditions	Min	Typ.	Max.	Unit
F_{SCK}	SCK frequency	VDD > 4.5 V			10	MHz
		VDD < 4.5 V			1	MHz
T_{RFO}	DATA fall time	Output load 5 pF	3.5	10	20	ns
		Output load 100 pF	30	40	200	ns
T_{CLK}	SCK hi/low time		100			ns
T_V	DATA valid time			250		ns
T_{SU}	DATA set up time		100			ns
T_{HO}	DATA hold time		0	10		ns
T_R/T_F	SCK rise/fall time			200		ns

Table 5 SHTxx I/O Signals Characteristics

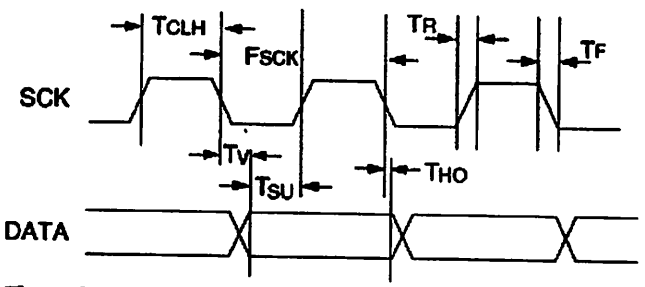


Figure 9 Timing Diagram

(1) Parameters are periodically sampled and not 100% tested
(2) Recommended voltage supply for highest accuracy is between 2.4...3.6V, due to sensor calibration at 3.3V.
(3) With one measurement of 8 bit accuracy without OTP reload per second
(4) With one measurement of 12bit accuracy per second

3 Converting Output to Physical Values

3.1 Relative Humidity

To compensate for the non-linearity of the humidity sensor and to obtain the full accuracy it is recommended to convert the readout with the following formula[1]:

$$RH_{linear} = c_1 + c_2 \bullet SO_{RH} + c_3 \bullet SO_{RH}^2$$

SO_{RH}	c_1	c_2	c_3
12 bit	-4	0.0405	$-2.8 * 10^{-6}$
8 bit	-4	0.648	$-7.2 * 10^{-4}$

Table 6 Humidity conversion coefficients

For simplified, less computation intense conversion formulas see application note "RH and Temperature Non-Linearity Compensation".

Values higher than 99% RH indicate fully saturated air and must be processed and displayed as 100%[2] RH.

The humidity sensor has no significant voltage dependency.

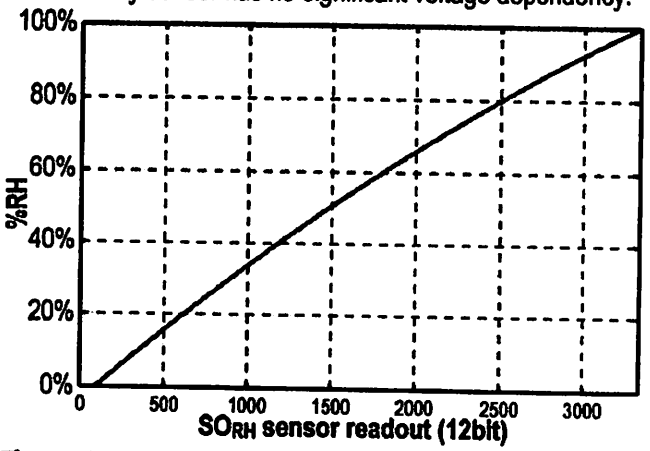


Figure 10 Conversion from SO_{RH} to relative humidity

3.1.1 *Humidity Sensor RH/Temperature compensation*

For temperatures significantly different from 25 °C (~77 °F) the temperature coefficient of the RH sensor should be considered:

$$RH_{true} = (T_{°C} - 25) \bullet (t_1 + t_2 \bullet SO_{RH}) + RH_{linear}$$

SO_{RH}	t_1	t_2
12 bit	0.01	0.00008
8 bit	0.01	0.00128

Table 7 Temperature compensation coefficients

This equals ~0.12 %RH / °C @ 50 %RH

3.2 Temperature

The bandgap PTAT (Proportional To Absolute Temperature) temperature sensor is very linear by design. Use the following formula to convert from digital readout to temperature:

$$Temperature = d_1 + d_2 \bullet SO_T$$

VDD	d_1 [°C]	d_1 [°F]
5V	-40.00	-40.00
4V	-39.75	-39.55
3.5V[3]	-39.66	-39.39
3V[3]	-39.60	-39.28
2.5V[3]	-39.55	-39.19

SO_T	d_2 [°C]	d_2 [°F]
14bit	0.01	0.018
12bit	0.04	0.072

Table 8 Temperature conversion coefficients

For improved accuracies in extreme temperatures with more computation intense conversion formulas see application note "RH and Temperature Non-Linearity Compensation".

3.3 Dewpoint

Since humidity and temperature are both measured on the same monolithic chip, the SHTxx allows superb dewpoint measurements. See application note "Dewpoint calculation" for more.

[1] Where SO_{RH} is the sensor output for relative humidity

[2] If wetted excessively (strong condensation of water on sensor surface), sensor output signal can drop below 100%RH (even below 0%RH) in some cases, but sensor will recover completely when water droplets evaporate. Sensor is not damaged by water immersion or condensation.

[3] For improved temperature accuracy with SHTxx-V4 set d1
for 3.5V: d1[°C]_3.5V=-39.60 / d1[°F]_3.5V=-39.28;
for 3V: d1[°C]_3V=-39.50 / d1[°F]_3V=-39.10;
for 2.5V: d1[°C]_2.5V=-39.45 / d1[°F]_2.5V=-39.01

Integrated Silicon Pressure Sensor Manifold Absolute Pressure Sensor On-Chip Signal Conditioned, Temperature Compensated and Calibrated

MPX4100 SERIES

INTEGRATED PRESSURE SENSOR 20 TO 105 kPA (2.9 TO 15.2 psi) 0.3 TO 4.9 V OUTPUT

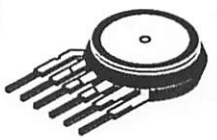
The MPX4100 series Manifold Absolute Pressure (MAP) sensor for engine control is designed to sense absolute air pressure within the intake manifold. This measurement can be used to compute the amount of fuel required for each cylinder. The small form factor and high reliability of on-chip integration makes the MAP sensor a logical and economical choice for automotive system designers.

Features

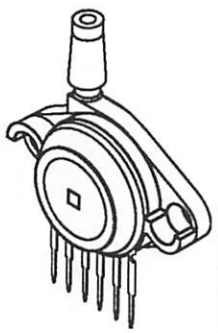
- 1.8% Maximum Error Over 0° to 85°C
- Specifically Designed for Intake Manifold Absolute Pressure Sensing in Engine Control Systems
- Ideally Suited for Microprocessor Interfacing
- Temperature Compensated Over -40°C to +125°C
- Durable Epoxy Unibody Element
- Ideal for Non-Automotive Applications

Typical Applications

- Manifold Sensing for Automotive Systems

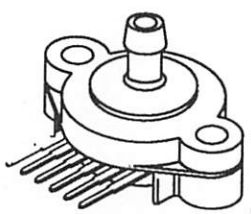


MPX4100A
CASE 867-08



MPX4100AP
CASE 867B-04

MPX4100AS
CASE 867E-03



MPX4100ASX
CASE 867F-03

ORDERING INFORMATION[1]				
Device Type	**Options**	**Case No.**	**MPX Series Order Number**	**Device Marking**
Basic Element	Absolute, Element Only	867-08	MPX4100A	MPX4100A
Ported Elements	Absolute, Ported	867B-04	MPX4100AP	MPX4100AP
	Absolute, Stove Pipe Port	867E-03	MPX4100AS	MPX4100A
	Absolute, Axial Port	867F-03	MPX4100ASX	MPX4100A

1. The MPX4100A series MAP silicon pressure sensors are available in the Basic Element, or with pressure port fittings that provide mounting ease and barbed hose connections.

PIN NUMBERS			
1	V_{OUT}	4	NC
2	GND	5	NC
3	V_S	6	NC

freescale semiconductor

The MPX4100 series piezoresistive transducer is a state-of-the-art, monolithic, signal conditioned, silicon pressure sensor. This sensor combines advanced micromachining techniques, thin film metallization, and bipolar semiconductor processing to provide an accurate, high level analog output signal that is proportional to applied pressure.

Figure 1 shows a block diagram of the internal circuitry integrated on a pressure sensor chip.

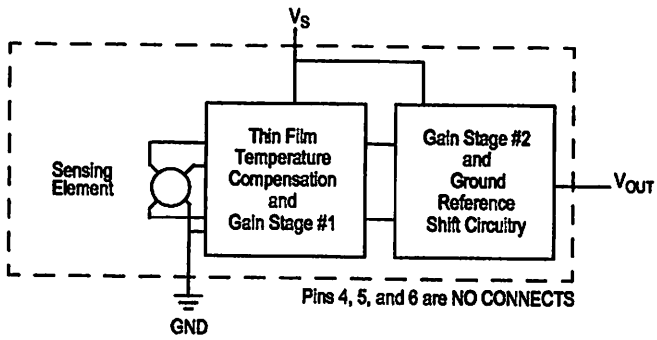


Figure 1. Fully Integrated Pressure Sensor Schematic

Table 1. MAXIMUM RATINGS[1]

Rating	Symbol	Value	Unit
Overpressure[2] (P1 > P2)	P_{max}	400	kPa
Burst Pressure[2] (P1 > P2)	P_{burst}	1000	kPa
Storage Temperature	T_{stg}	-40 to +125	°C
Operating Temperature	T_A	-40 to +125	°C

1. T_C = 25°C unless otherwise noted.
2. Exposure beyond the specified limits may cause permanent damage or degradation to the device.

Table 2. OPERATING CHARACTERISTICS (VS = 5.1 Vdc, TA = 25×C unless otherwise noted, P1 > P2)

Characteristic		Symbol	Min	Typ	Max	Unit
Pressure Range[2]		P_{OP}	20	—	105	kPa
Supply Voltage[2]		V_S	4.85	5.1	5.35	Vdc
Supply Current		I_O	—	7.0	10	mAdc
Minimum Pressure Offset[3] @ V_S = 5.1 V	(0 to 85°C)	V_{off}	0.225	0.306	0.388	Vdc
Full Scale Output[4] @ V_S = 5.1 V	(0 to 85°C)	V_{FSO}	4.815	4.897	4.978	Vdc
Full Scale Span[5] @ V_S = 5.1 V	(0 to 85°C)	V_{FSS}	—	4.59	—	Vdc
Accuracy[6]	(0 to 85°C)	—	—	—	±1.8	$\%V_{FSS}$
Sensitivity		V/P	—	54	—	mV/kPa
Response Time[7]		t_R	—	1.0	—	ms
Output Source Current at Full Scale Output		I_{O+}	—	0.1	—	mAdc
Warm-Up Time[8]		—	—	20	—	ms
Offset Stability[9]		—	—	±0.5	—	$\%V_{FSS}$

1. Decoupling circuit shown in Figure 3 required to meet electrical specifications.
2. 1.0 kPa (kiloPascal) equals 0.145 psi.
3. Offset (V_{off}) is defined as the output voltage at the minimum rated pressure.
4. Full Scale Output (V_{FSO}) is defined as the output voltage at the maximum or full rated pressure.
5. Full Scale Span (V_{FSS}) is defined as the algebraic difference between the output voltage at full rated pressure and the output voltage at the minimum rated pressure.
6. Accuracy (error budget) consists of the following:
   - Linearity: Output deviation from a straight line relationship with pressure over the specified pressure range.
   - Temperature Hysteresis: Output deviation at any temperature within the operating temperature range, after the temperature is cycled to and from the minimum or maximum operating temperature points, with zero differential pressure applied.
   - Pressure Hysteresis: Output deviation at any pressure within the specified range, when this pressure is cycled to and from the minimum or maximum rated pressure, at 25°C.
   - TcSpan: Output deviation over the temperature range of 0 to 85°C, relative to 25°C.
   - TcOffset: Output deviation with minimum rated pressure applied, over the temperature range of 0 to 85°C, relative to 25°C.
   - Variation from Nominal: The variation from nominal values, for Offset or Full Scale Span, as a percent of V_{FSS}, at 25°C.
7. Response Time is defined as the time for the incremental change in the output to go from 10% to 90% of its final value when subjected to a specified step change in pressure.
8. Warm-up is defined as the time required for the product to meet the specified output voltage after the Pressure has been stabilized.
9. Offset stability is the product's output deviation when subjected to 1000 hours of Pulsed Pressure, Temperature Cycling with Bias Test.
10. Device is ratiometric within this specified excitation range.

Table 3. MECHANICAL CHARACTERISTICS

Characteristic	Symbol	Min	Typ	Max	Unit
Weight, Basic Element (Case 867)	—	—	4.0	—	Grams
Common Mode Line Pressure[1]	—	—	—	690	kPa

1. Common mode pressures beyond specified may result in leakage at the case-to-lead interface.

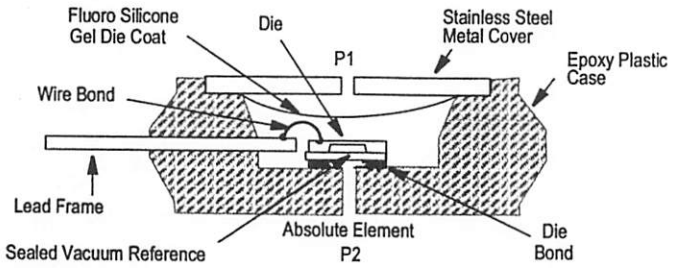


Figure 2. Cross-Sectional Diagram (Not to Scale)

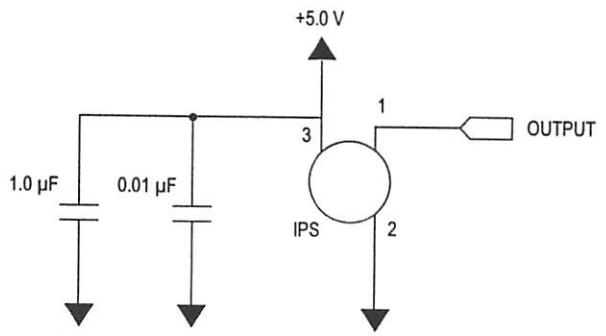


Figure 3. Recommended Power Supply Decoupling
(For output filtering recommendations, refer to Application Note AN1646.)

Figure 2 illustrates an absolute sensing chip in the basic chip carrier (Case 867). A fluorosilicone gel isolates the die surface and wire bonds from the environment, while allowing the pressure signal to be transmitted to the sensor diaphragm. The MPX4100A series pressure sensor operating characteristics, and internal reliability and qualification tests are based on use of dry air as the pressure media. Media, other than dry air, may have adverse effects on sensor performance and long-term reliability. Contact the factory for information regarding media compatibility in your application.

Figure 4 shows the sensor output signal relative to pressure input. Typical, minimum, and maximum output curves are shown for operation over a temperature range of 0° to 85°C. (The output will saturate outside of the specified pressure range.)

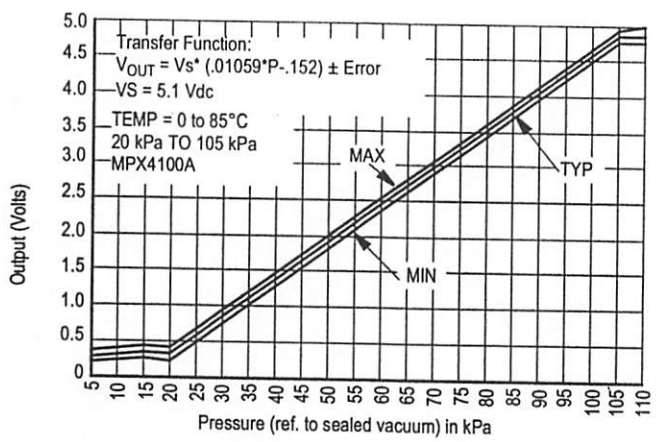


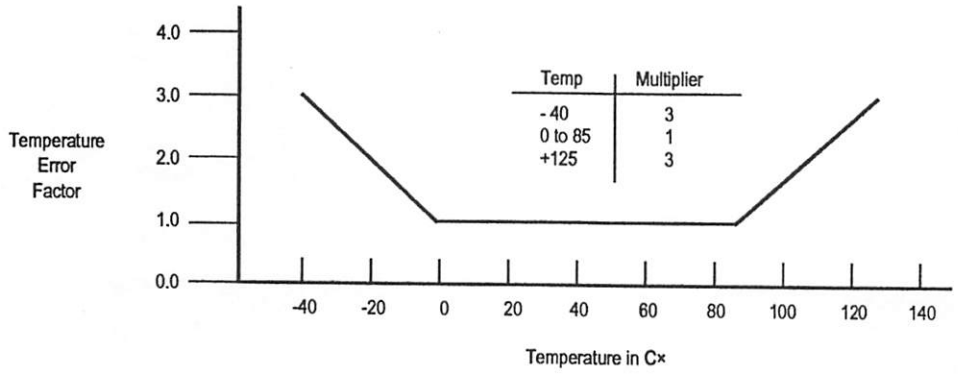
Figure 4. Output versus Absolute Pressure

Transfer Function (MPX4100A)

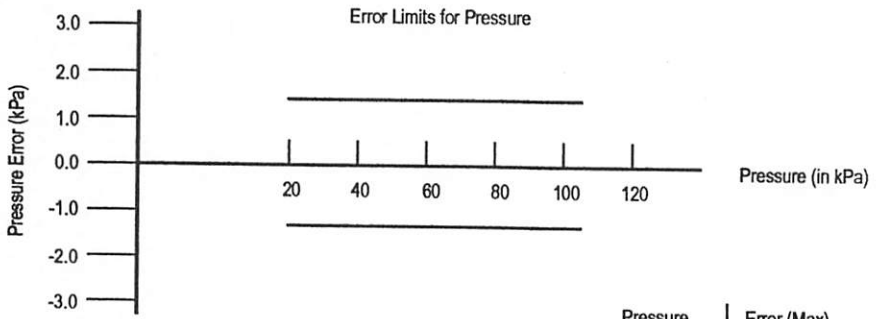
Nominal Transfer Value: $V_{out} = V_S (P \times 0.01059 - 0.1518)$
$\pm$ (Pressure Error x Temp. Factor x 0.01059 x V_S)
$V_S = 5.1\ V \pm 0.25\ Vdc$

Temperature Error Band

MPX4100A Series



Pressure Error Band



Pressure	Error (Max)
20 to 105 (kPa)	±1.5 (kPa)

PRESSURE (P1)/VACUUM (P2) SIDE IDENTIFICATION TABLE

The two sides of the pressure sensor are designated as the Pressure (P1) side and the Vacuum (P2) side. The Pressure (P1) side is the side containing fluorosilicone gel, which protects the die from harsh media. The MPX pressure sensor is designed to operate with positive differential pressure applied, P1 > P2.

The Pressure (P1) side may be identified by using the table below:

Part Number	Case Type	Pressure (P1) Side Identifier
MPX4100A	867	Stainless Steel Cap
MPX4100AP	867B	Side with Port Marking
MPX4100AS	867E	Side with Port Attached
MPX4100ASX	867F	Side with Port Attached

1. Introduction

The EG-SR-7150MJ is a gateway module that converts serial data into TCP/IP data type. It transmits the data sent by a serial equipment to the Internet and TCP/IP data to the equipment.

With the EG-SR-7150MJ mounted with an RJ-45 connector, users can have an easier and quicker interface with the Ethernet.

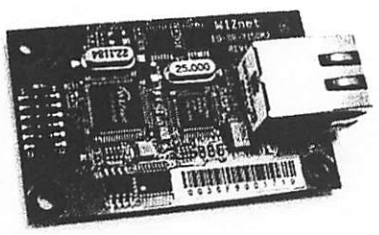
The EG-SR-7150MJ provides serial commands, with which the developers of any serial device can add local configuration capability to their products. For example, a card reader developer can program the keypad on a card reader to configure serial or network on-site without the use of a laptop or PC.

1.1. Key Features

- Ready-to-go serial to Ethernet gateway module mounted with an RJ-45 connector
- Serial Command Support
  - Simple command frame format
  - Comprehensive & readable command set for network and serial settings
  - On-site configuration without PC
- High stability & reliability by using a W3150A WIZnet Chip, a fully-hardwired TCP/IP stack
- Easy and powerful configuration program
- 10/100Mbps Ethernet interface, Max. 230Kbps Serial interface
- RoHS compliant

1.2. Products Contents (EG-SR-7150MJ-EVB)

The EG-SR-7150MJ-EVB, the evaluation kit for the EG-SR-7150MJ contains the following items;

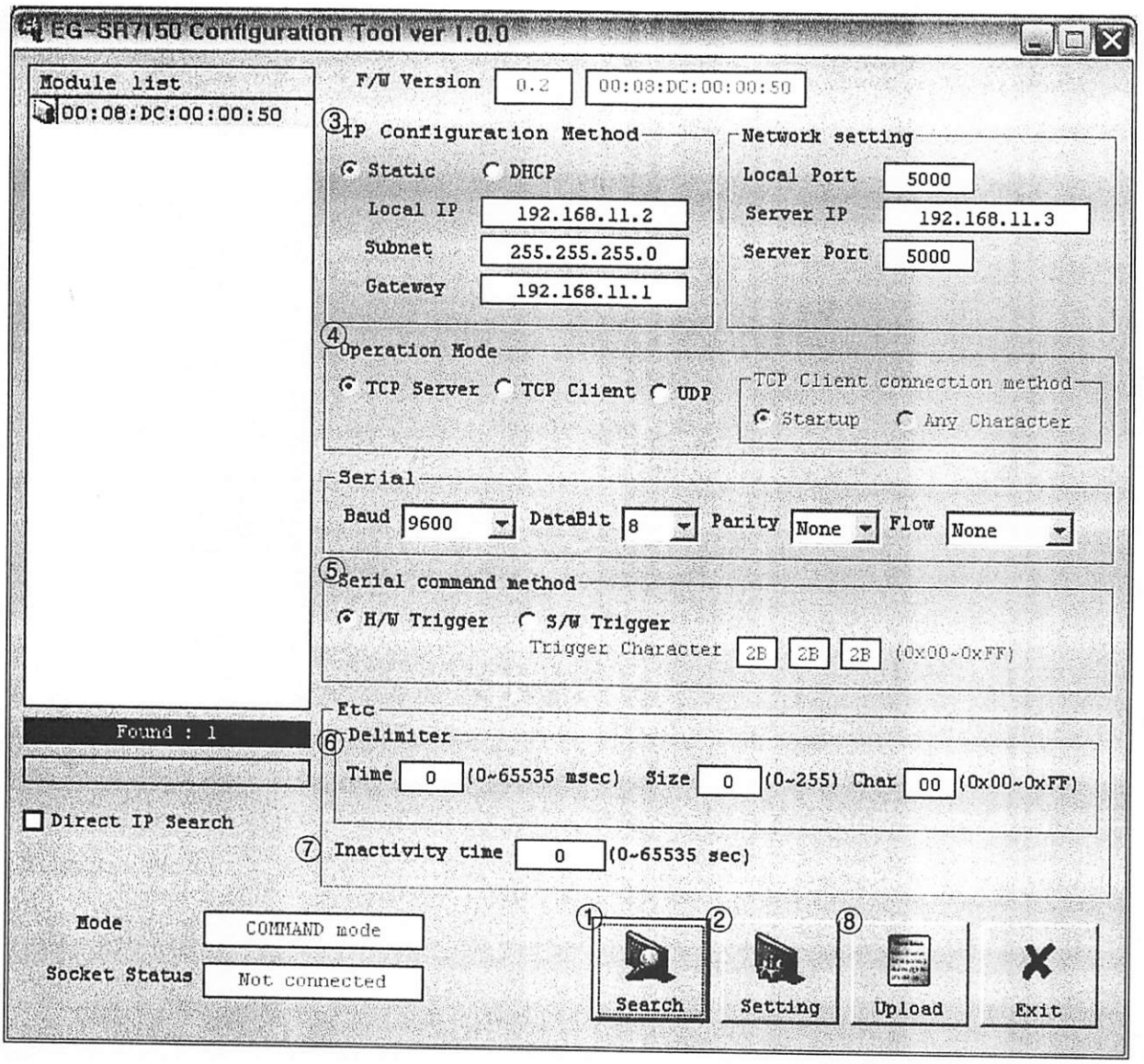
	EG-SR-7150MJ Module

1.3. Specifications of the EG-SR-7150MJ

Category	Specifications
Form Factor	2mm Pitch 2x6 pins, 62x40 mm
LAN Interface	10/100 Mbps auto-sensing, RJ-45 connector
Protocol	TCP, UDP, IP, ARP, ICMP, MAC, (IGMP, PPPoE)
CPU	AT89C51RC2 (8bit MCU and 32K Flash)
Serial Interface	RS 232 (LVTTL)
Serial Signals	TXD, RXD, RTS, CTS, GND
Serial Parameters	Parity : None, Even, Odd Data bits : 7, 8 Flow control : RTS/CTS, XON/XOFF Speed : up to 230Kbps
Management	Configuration utility based on Windows
Temperature	0℃~70℃ (Operating), -40℃~85℃ (Storage)
Humidity	10~90%
Power	150mA @ 3.3V (max)
Size	40mm x 62mm x 17mm

2.2. Configuration Tool

2.2.1. Configuration tool features



① Search

The Search function is used to search all modules existing on the same Subnet. The UDP broadcast is used for searching modules on a LAN.

The MAC address for a searched module will be listed in the "**Module list**".

If **Direct IP Search** is checked, TCP will be used for searching instead of UDP. This mode is used more for searching the EG-SR-7150MJ modules on remote networks than local networks with the same subnet. An IP address assigned to the module will be required.

② Setting

If you select one of the MAC addresses listed in the "**Module list**", the configuration value of the selected module will be displayed. After changing each value in the configuration program, click the "**Setting**" button to complete the configuration.
The module will be initialized with the new configurations.

③ IP Configuration method: Static, DHCP

Static: The IP address can be manually assigned by users.

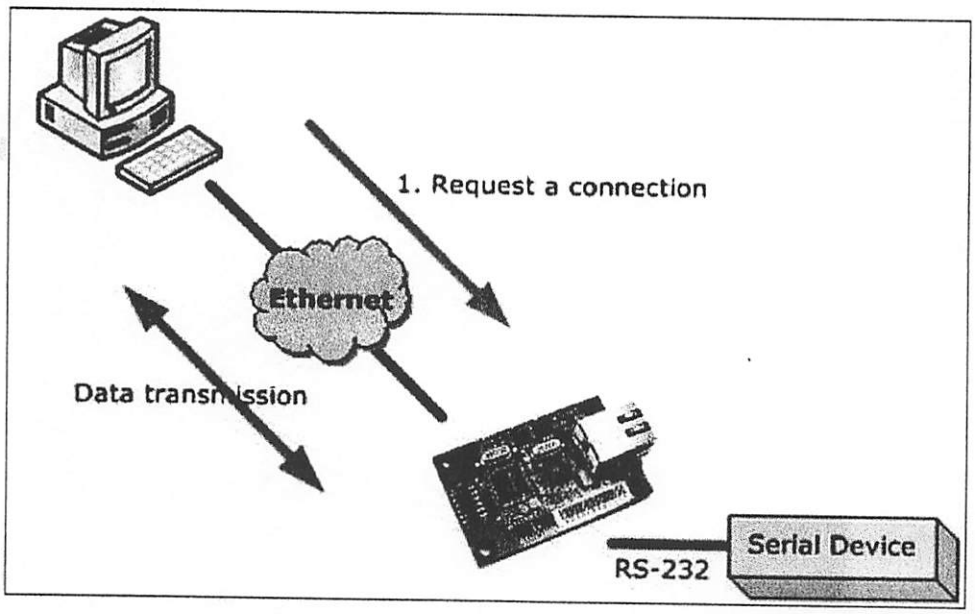
DHCP: The module assigns IP, subnet and gateway addresses by acquiring them from the DHCP server

☞ Other configurations should be set manually except for the IP configuration of DHCP.

④ Operation mode: TCP server, TCP client, UDP

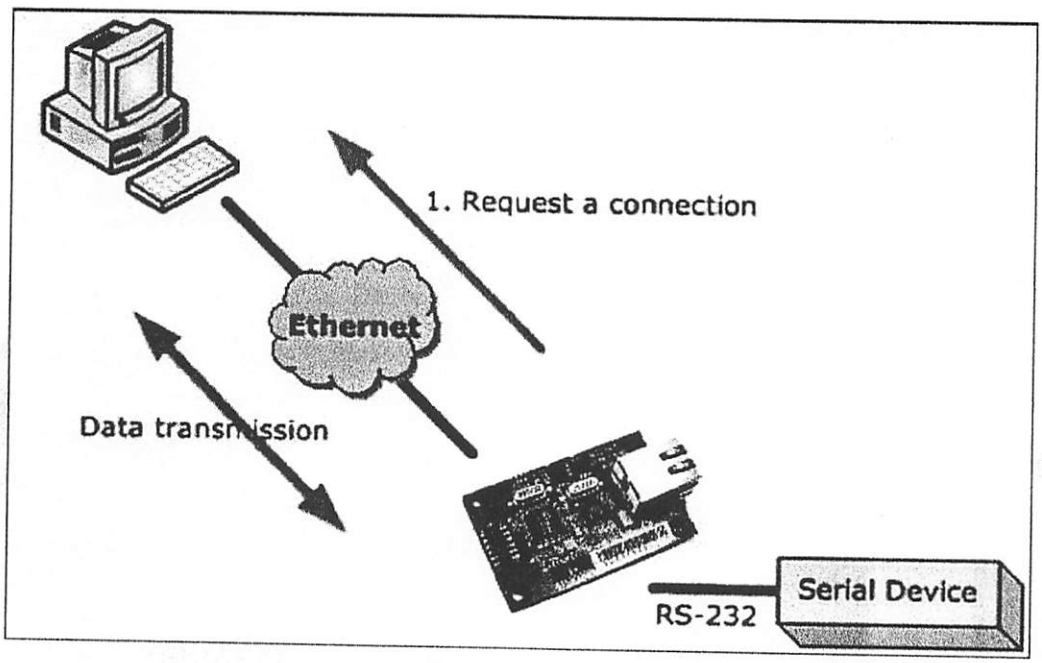
Three different operation modes are supported — TCP Server, TCP Client, and UDP.
The main difference between the TCP and UDP protocols is that TCP guarantees the delivery of data by requesting the recipient to send an acknowledgement to the sender. On the other hand, UDP does not require this type of verification, so data can be delivered quicker, but its delivery can not be guaranteed.
The TCP Server and TCP Client mode are related to the first step of connection establishment. Once the connection is established, data will be transparently transmitted in both directions (from Server to Client or from Client to Server).

TCP server mode



To operate this mode, the **Local IP, Subnet, gateway address and local port number** should be configured. The EG-SR-7150MJ waits to be connected by the host computer, allowing the host computer to establish a connection and get data from the serial device.
As illustrated in the figure above, the data transmission is as follows:
1. The host connects to the EG-SR-7150MJ which is configured as TCP Server Mode.
2. Once the connection is established, data can be transmitted in both directions - from the host to the EG-SR-7150MJ, and from the EG-SR-7150MJ to the host.

TCP client mode

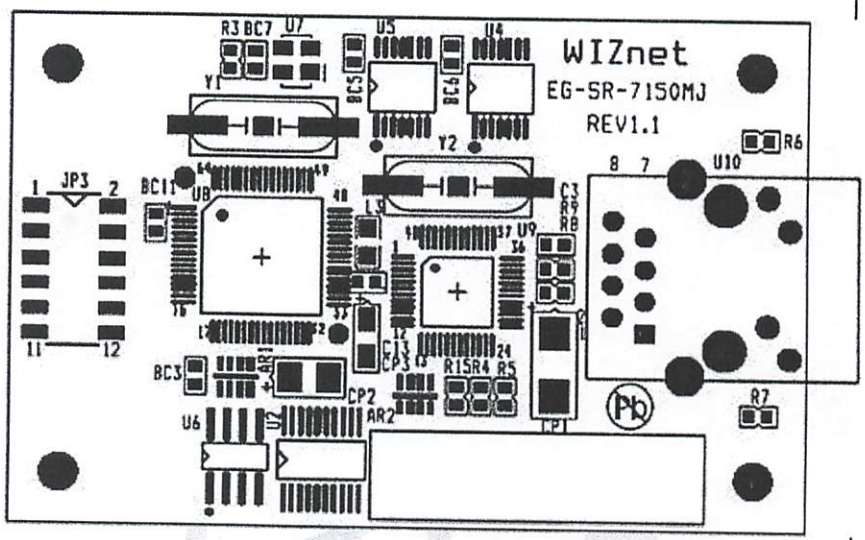


To operate this mode, the **Local IP, Subnet, gateway address, server IP, server port number** should be set. In the **TCP Client mode**, the EG-SR-7150MJ proceeds active open for establishing a TCP connection to a host computer.
As illustrated in the figure above, data transmission is as follows:
1. The EG-SR-7150MJ operating as TCP Client Mode establishes a connection based on the condition set in the **TCP client connection method** (**Startup, Any character**). i.e. the EG-SR-7150MJ can try to connect as soon as one starts up(**Startup**), or later when data from serial device arrives. (**Any character**).
2. After the connection is established, data can be transmitted in both directions - from the host to the EG-SR-7150-MJ, and from the EG-SR-7150-MJ to the host.

UDP mode

4. PIN Assignment and Dimensions

```
-----------------------------------
              JP3
-----------------------------------
/RESET - 1   2 -  3.3V
   RXD - 3   4 - 3.3V
   CTS - 5   6 - /FACTORY_RESET
   TXD - 7   8 - /HW_TRIGGER
   RTS - 9   10 - /PSEN
 GND - 11  12 - GND
```



Name	Functions	I/O	
3.3V	Power		
/RESET	Low active reset Minimum 1.2 usec is required.	Input	
RXD	RS-232 Data Input	Input	
CTS	RS-232 Clear To Send	Input	Optional
TXD	RS-232 Data Output	Output	
RTS	RS-232 Request To Send	Output	Optional
Factory Reset	Pull Factory Reset to low and if /RESET is activated, the configuration is changed to factory default.	Input	
H/W Trigger	Pull H/W Trigger to low, enter the serial command mode	Input	
/PSEN	Pull /PSEN to low and if /RESET is activated, the module enter the bootloader for FLIP connection	Input	

☞ All signal levels are 3.3V LVTTL.