SISTEM KENDALI LAMPU VIA *WIRELESS* 2,4 GHz BERBASIS MIKROKONTROLER ATMEGA 16

PROYEK AKHIR

Diajukan Kepada Fakultas Teknik Universitas Negeri Yogyakarta
Untuk Memenuhi Sebagian Persyaratan
Guna Memperoleh Gelar Ahli Madya Teknik

Oleh :
FAJAR ARI IRAWAN
NIM. 13507134013

PROGRAM STUDI TEKNIK ELEKTRONIKA-D3
JURUSAN PENDIDIKAN TEKNIK ELEKTRONIKA
FAKULTAS TEKNIK
UNIVERSITAS NEGERI YOGYAKARTA
2016

PERSETUJUAN

PROYEK AKHIR

Dengan Judul

Sistem Kendali Lampu Via *Wireless* 2,4 GHz

Berbasis Mikrokontroler ATmega 16

Dipersiapkan dan Disusun Oleh :

FAJAR ARI IRAWAN
NIM. 13507134013

Telah diperiksa dan disetujui oleh Dosen pembimbing untuk diajukan

Di depan Dewan Penguji Tugas Akhir

Jurusan Teknik Elektronika Fakultas Teknik Universitas Negeri Yogyakarta

Guna memperoleh gelar Ahli Madya Teknik

Yogyakarta, 2 Agustus 2016

Mengetahui,
Kaprodi Teknik Elektronika

Dr. Sri Waluyanti
NIP. 19581218 198603 2 001

Menyetujui,
Pembimbing Proyek Akhir

Nurkhamid, Ph.D.
NIP. 19680707 199702 1 001

PENGESAHAN

Proyek akhir yang berjudul " **SISTEM PENGENDALI LAMPU VIA *WIRELESS* 2,4 GHz BERBASIS MIKROKONTROLER ATMEGA 16** " ini telah dipertahankan di depan Dewan Penguji pada tanggal 22 agustus 2016 dan dinyatakan lulus.

DEWAN PENGUJI

Nama	Jabatan	Tanda Tangan	Tanggal
Nurkhamid, Ph.D.	Ketua Penguji	.....	21/9/2016
Satriyo Agung D, M.Pd.	Sekertaris Penguji	.....	21/9/2016
Adi Dewanto, M.Kom.	Penguji	.....	21/9/2016

Yogyakarta, 22 Agustus 2016
Fakultas Teknik
Universitas Negeri Yogyakarta
Dekan,

Dr. Widarto, M.Pd.
NIP. 19631230 198812 1 001

SURAT PERNYATAAN

Yang bertanda tangan dibawah ini :

Nama	: Fajar Ari Irawan
NIM	: 13507134013
Program Studi	: Teknik Elektronika-D3
Fakultas	: Teknik
Judul Proyek Akhir	: **Sistem Kendali Lampu Via *Wireless* 2,4 GHz Berbasis Mikrokontroler ATmega 16**

Dengan ini saya menyatakan bahwa dalam Proyek Akhir ini tidak terdapat karya yang pernah diajukan untuk memperoleh gelar Ahli Madya Teknik atau gelar lainnya di suatu Perguruan Tinggi, dan sepanjang pengetahuan saya juga tidak terdapat karya atau pendapat yang pernah ditulis oleh orang lain, kecuali secara tertulis diacu dalam naskah ini dan disebutkan dalam daftar pustaka.

Yogyakarta, 2 Agustus 2016

Yang Menyatakan,

Fajar Ari Irawan

NIM. 13507134013

PERSEMBAHAN

Dengan penuh rasa syukur karya ini **ku** persembahkan untuk :

Yang **pertama** untuk **Bapak** dan **Ibu** saya tercinta yang telah melahirkan dan membesarkanku dengan penuh kasih sayang serta senantiasa berdoa untuk keselamatan dan kebahagianku.

Yang **kedua** untuk **Kakak** dan **Adikku** tercinta yang selalu memberikan dukungan baik moril maupun material. Terima kasih atas motivasinya.

Yang **ketiga** buat teman-teman Teknik Elektronika UNY **kelas B angkatan 2013** serta teman-teman yang lainnya terutama untuk **(Maul, Bangkit, Fahmi, Hari, Haris, Arifin, Syahrul, Bakhtiar, Tholib)** yang memberikan semangat, motivasi serta masukannya, **semangat terus brother, raihlah mimpi menjadi kenyataan...!!!**

Yang **keempat** Buat **Bapak** dan **Ibu Dosen Teknik Elektronika UNY.** Terima kasih atas bimbinganya selama ini tanpa kalian saya tidak bisa menjadi dewasa. Terima kasih Buat seluruh **Karyawan dan staff Elektronika UNY.**

And yang **terakhir** Buat **SomeOne ku N** tercinta terima kasih atas kasih sayang, dukungan, semangat dan motivasinya.

MOTTO

Berangkat dengan penuh keyakinan

Berjalan dengan penuh keikhlasan

Istiqomah dalam menghadapi cobaan

Jadilah seperti karang di lautan yang kuat di hantam ombak dan kerjakanlah hal yang bermanfaat untuk diri sendiri dan orang lain, karena hidup hanyalah sekali. Ingat hanya pada Allah apapun dan di manapun kita berada kepada Dia-lah tempat meminta dan memohon.

Nilai prestasi adalah keseluruhan pribadi yang cerdas dan beretika.

Kesuksesan itu bukan ditunggu, tetapi diwujudkan lewat usaha dan kegigihan.

"Maka sesungguhnya bersama kesulitan ada kemudahan. Sesungguhnya bersama kesulitan ada kemudahan. Maka apabila engkau telah selesai (dari sesuatu urusan), tetaplah bekerja keras (untuk urusan yang lain). Dan hanya kepada Tuhanmulah engkau berharap." (QS. Al-Insyirah,6-8)

SISTEM KENDALI LAMPU VIA *WIRELESS* 2,4 GHz BERBASIS MIKROKONTROLLER ATMEGA 16

Oleh :

Fajar Ari Irawan
NIM. 13507134013

ABSTRAK

Tujuan utama dari proyek akhir ini untuk merancang dan membuat suatu teknologi yang bermanfaat bagi perkembangan ilmu pengetahuan dan teknologi, khususnya kebutuhan sistem kendali lampu. Teknologi ini berfungsi sebagai sistem kendali lampu jarak jauh yang akan digunakan untuk kebutuhan sehari-hari yang diaplikasikan untuk sistem kendali lampu yang terbaharukan.

Rancang bangun alat sistem kendali lampu via *wireless* 2,4 GHz berbasis mikrokontroler ATmega 16 yang digunakan diwujudkan dari beberapa sistem yaitu rangkaian catu daya, rangkaian logic converter dan ethernet, rangkaian driver relay serta input data ascii yang dikirim melalui telnet melalui *smartphone* atau komputer. Perancangan perangkat lunak sebagai pengolahan dan pengendali program pada mikrokontroler ATmega 16 serta *wiznet serial to ethernet* menggunakan bahasa C dan *sofware* CVAVR sebagai compiler-nya.

Berdasarkan hasil pengujian dari alat sistem kendali lampu via *wireless* 2,4 GHz berbasis mikrokontroler ATmega 16 yang telah dilakukan menunjukan hasil yang sesuai dengan perancangan. Alat ini sudah dapat bekerja sesuai perintah, hal ini ditunjukan pada lampu dapat dikendalikan dari jarak tertentu. Lampu dapat menyala dan mati sesuai dengan intruksi yang diberikan, dengan mengukur kemampuan pengiriman data berbanding dengan jarak antara penerima dan pemancar menggunakan *smartphone* yang dikoneksikan ke router tp-link menggunakan media *wireless*.

Kata kunci : ATmega 16, *Smartphone, Wireless* 2,4 *GHz, Serial to Ethernet*, Lampu

KATA PENGANTAR

Puji syukur penulis panjatkan kepada Allah SWT atas limpahan rahmat, karunia dan nikmat yang telah di berikan-Nya, sehingga penulis dapat menyelesaikan proyek akhir dan penyusunan laporan ini.

Penulis sadar tanpa bantuan berbagai pihak Proyek Akhir ini tidak akan terlaksana dengan baik. Oleh karena itu pada kesempatan ini penulis dengan ketulusan hati mengucapkan terima kasih atas dukungan, bimbingan dan bantuanya baik secara moril maupun materil kepada :

1. Allah SWT yang telah melimpahkan Rahmat dan Nikmat Islam.
2. Bapak Prof. Dr. Rochmat Wahab, M.Pd, M.A, selaku Rektor Universitas Negeri Yogyakarta.
3. Bapak Dr. Widarto, M.Pd, selaku Dekan Fakultas Teknik Universitas Negeri Yogyakarta.
4. Bapak Dr. Fatchul Arifin, S.T., M.T, selaku ketua jurusan Pendidikan Teknik Elektronika Universitas Negeri Yogyakarta.
5. Bapak Nurkhamid, M.Pd, selaku Dosen Pembimbing Proyek Akhir.
6. Bapak Dr. Fatchul Arifin, S.T., M.T, selaku Dosen Penasehat Akademik.
7. Bapak dan Ibu dosen serta, teknisi di jurusan Pendidikan Teknik Elektronika.
8. Bapak, ibu, kakak dan adikku tercinta yang telah memberikan dukungan dan doa.

9. Teman – teman kelas B angkatan 2013 yang telah memberikan banyak masukan, bantuan dan motivasi.
10. Semua pihak yang tidak dapat penulis sebutkan satu persatu atas bantuannya selama penelitian ini.

Penulis menyadari bahwa penelitian ini masih banyak kekurangannya, oleh karena itu saran dan kritik yang sifatnya membangun sangat penulis harapkan demi kesempurnaan penelitian ini. Akhirnya peneliti berharap semoga Tugas Akhir ini bermanfaat bagi diri peneliti dan pembaca semuanya.

Yogyakarta, 2 Agustus 2016

Penulis

Fajar Ari Irawan

DAFTAR ISI

Halaman

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR LAMPIRAN

BAB I

PENDAHULUAN

A. Latar Belakang Masalah

Kebutuhan akan sistem pengendalian jarak jauh semakin meningkat dimana perpindahan dan pergerakan manusia semakin luas dan cepat terutama di kota besar, aktifitas setiap individu masyarakat sangatlah padat dengan berbagai macam pekerjaannya, tentunya memakan waktu dari pagi hingga malam hari. Akibatnya banyak kegiatan dirumah tangga yang tertunda, seperti menghidupkan atau mematikan lampu disetiap ruang saat malam dan pagi hari (Galih Rakasiwi, 2014).

Selama ini masyarakat dapat mengendalikan sesuatu dari jarak jauh dengan menggunakan *remote control* yang berbasis *Bluetooth*, kemudian dengan saklar yang melalui kabel, akan tetapi pengendalian tersebut dibatasi oleh jarak jangkauan sehingga masih kurang efektif mengingat sebagian orang sering berpergian jauh bahkan bisa saja pergi ke luar kota dengan waktu yang cukup lama. Maka dari itu agar lampu bisa di kendalikan dengan cakupan jarak yang semakin luas dan mudah salah satu solusinya dengan menggunakan *wireless* 2,4 GHz serta *smartphone android* sebagai *remote controlnya*. Android merupakan sebuah sistem operasi pada ponsel, android merupakan sebuah sistem operasi pada ponsel berbasis linux yang mencakup sistem operasi dan *middleware*. Fasilitas *opensource* atau sistem operasi yang dapat dikembangkan dengan

bebas bagi penggunanya membuat banyak orang untuk mengembangkannya dengan inovasi– inovasi yang semakin berkembang terhadap sistem operasinya maupun pada pembangunan aplikasi *mobile*-nya tersebut. Tak heran saat ini banyak pengembang yang membangun aplikasi *mobile* pada *platform* android.

Oleh karena itu, penulis mencoba merancang sebuah alat yang dapat mengendalikan lampu dari jarak jauh, yaitu sistem kendali lampu via *wireless* 2,4 GHz berbasis mikrokontroler ATmega 16 yang dibangun di-*platform* android dengan menggunakan *wireless* 2,4 GHz sebagai solusi alternatif baru untuk pengendalian jarak jauh. Aplikasi yang dibangun pada *platform* android ini memiliki tampilan antarmuka (*user interface*) yang menarik dan mudah dipahami, aplikasi yang digunakan dalam pembuatan alat ini yaitu *Mocca Telnet*. Sistem pengendalian yang dibangun memanfaatkan jaringan internet atau *wireless* 2,4 GHz untuk pengiriman instruksi pengendaliannya ke mikrokontroler ATmega 16.

B. Identifikasi Masalah

Berdasarkan latar belakang masalah yang ada, maka dapat diidentifikasikan hal sebagai berikut :

1. Padatnya aktifitas masyarakat dengan berbagai macam pekerjaanya sehingga masyarakat sering lupa untuk menghidupkan dan mematikan lampu disaat malam dan pagi hari.
2. Kurangnya pemanfaatan *smartphone* untuk mengendalikan lampu.

C. Batasan Masalah

Berdasarkan pada pokok permasalahan yang diuraikan pada latar belakang dan identifikasi masalah diatas, sehingga ruang lingkup masalah menjadi lebih jelas maka batasan masalah pada proyek akhir sistem kendali lampu via *wireless* 2,4 GHz berbasis mikrokontroler ATmega 16 ini adalah untuk membantu masyarakat menghidupkan dan mematikan lampu disaat malam dan pagi hari.

D. Rumusan Masalah

Dari identifikasi yang ada, maka dapat ditarik beberapa rumusan masalah, yaitu :

1. Bagaimanakah Desain Sistem Kendali Lampu Via *Wireless* 2,4 GHz Berbasis Mikrokontroler *ATmega 16* ?
2. Bagaimanakah unjuk kerja Sistem Kendali Lampu Via *Wireless* 2,4 GHz Berbasis Mikrokontroler *ATmega 16* ?

E. Tujuan

Adapun tujuan rancang bangun alat sistem kendali lampu via *wireless* 2,4 GHz berbasis mikrokontroler ATmega 16 yaitu :

1. Mampu membuat perancangan Sitem Kendali Lampu Via *Wireless* 2,4 GHz Berbasis Mikrokontroler *ATmega 16.*
2. Dapat mengetahui unjuk kerja Sitem Kendali Lampu Via *Wireless* 2,4 GHz Berbasis Mikrokontroler *ATmega 16.*

F. Manfaat

Pembuatan proyek akhir ini diharapkan dapat bermanfaat bagi mahasiswa, lembaga pendidikan, dan industri. Adapun manfaat yang diharapkan dari pembuatan tugas akhir ini antara lain :

1. Bagi mahasiswa
    a. Sebagai tolak ukur individual setelah mendapatkan ilmu dari bangku kuliah dan kehidupan sehari-hari untuk diimplementasikan dalam bentuk suatu alat yang mampu bermanfaat bagi masyarakat.
    b. Untuk mengaplikasikan ilmu yang didapat selama di bangku kuliah dan menerapkan ilmunya secara nyata.
    c. Dapat digunakan sebagai bahan referensi atau pembelajaran dan penambah wawasan tentang sistem kendali lampu menggunakan komunikasi via *wireless* 2,4 GHz berbasis mikrokontroler ATmega 16 serta sebagai kajian untuk pengembangan selanjutnya.
    d. Sebagai bentuk kontribusi terhadap Universitas dan pengabdian kepada masyarakat dalam bentuk karya alat yang bermanfaat.
2. Bagi program studi Teknik Elektronika
    a. Sebagai wujud dari perkembangan Ilmu Pengetahuan dan Teknologi (IPTEK).

b. Sebagai parameter kulitas dan kuantitas lulusan mahasiswa Fakultas Teknik Universitas Negeri Yogyakarta.

3. Bagi Dunia Usaha dan Dunia Industri

a. Dapat digunakan sebagai pengembangan produk elektronika yang dapat diaplikasikan pada berbagai bidang khususnya pada bidang komunikasi menggunakan *wireless.*

b. Dapat dimanfaatkan sebagai nilai tambah suatu jasa usaha yang menggunakan teknologi mikrokontroler yang mampu digunakan untuk membantu mengendalikan suatu alat.

G. Keaslian Gagasan

Dengan ini saya menyatakan bahwa dalam proyek akhir ini tidak terdapat karya yang pernah diajukan di Fakultas Teknik Elektronka Universitas Negeri Yogyakarta guna untuk memperoleh gelar Ahli Madya Teknik atau gelar yang lainnya di suatu perguruan tinggi, dan sepanjang pengetahuan saya juga tidak terdapat karya atau pendapat yang pernah ditulis oleh orang lain, kecuali secara tertulis diacu dalam naskah ini dan disebutkan dalam daftar pustaka.

BAB II

PENDEKATAN PEMECAHAN MASALAH

A. Sistem Kendali

Sistem kendali adalah suatu susunan komponen fisik yang terhubung atau terkait sedemikian rupa sehinga dapat memerintah, mengarahkan, atau mengatur diri sendiri atau sistem lain.

Dalam kehidupan sehari-hari sadar atau tanpa kita sadari kita terus bertemu dengan suatu perangkat atau peralatan yang kerjanya terkendali secara otomatis baik terkendali sebagian maupun seluruhnya, seperti saat mengendarai mobil, saat menggunakan mesin cuci, menggunakan handphone, dan banyak lagi yang lainnya, singkatnya sistem yang digunakan untuk membuat suatu perangkat menjadi terkendali sesuai dengan keinginan manusia ini biasanya disebut sebagai sistem kendali.

Didalam sistem kendali bisa di aplikasikan beberapa sistem yang mempunyai fungsi dan tujuanya masing-masing. Sebut saja sistem kendali lampu jarak jauh, mengingat lampu pada rumah menjadi salah satu hal yang penting demi sebuah kenyamanan pemiliknya. Pada sistem kendali lampu jarak jauh, pemilik rumah akan dimudahkan dalam menghidupkan serta mematikan lampu hanya dengan perangkat remote menggunakan *smartphone* yang telah dikoneksikan.

B. *Wireless*

Wireless jika dari arti katanya adalah dapat diartikan “tanpa kabel”, yaitu melakukan suatu hubungan telekomunikasi menggunakan gelombang elektromagnetik sebagai pengganti media kabel. Saat ini teknologi *wireless* sudah berkembang pesat, buktinya dapat dilihat dengan semakin banyaknya yang menggunakan telepon sellular, selain itu berkembang juga teknologi *wireless* yang dipakai untuk mengakses internet.

“Wireless merupakan Koneksi antar suatu perangkat dengan perangkat lainnya tanpa menggunakan kabel atau Metode untuk mengirimkan sinyal melalui suatu ruangan bukannya menggunakan kabel” (W Purbo, Onno, 2005).

Wireless yang digunakan berupa *wireless* 2,4 GHz, karena *wireless* 2,4 GHz memiliki jangkauan jaringan yang lebih luas selain itu tingkat toleransi lebih baik untuk pohon dan hambatan kecil dibandingkan 5.8 GHz serta paling kompatibel dengan standar Wi-Fi perangkat, seperti Wi-Fi ponsel, laptop dan Wi-Fi kamera IP, dan bebas lisensi di sebagian besar negara.

C. Mikrokontroler

Mikrokontroler atau pengendali mikro adalah komponen yang sangat umum dalam sistem elektronik modern. Penggunaanya sangat luas, dalam kehidupan sehari-hari baik di rumah, kantor, rumah sakit, bank, sekolah, industri, dan lain-lain.

Mikrokontroler berdasarkan jurnal Anna Nur Nazilah Chamim "Mikrokontroler adalah sebuah system komputer yang seluruh atau bagian besar elemenya dikemas dalam satu chip IC, sehingga sering disebut single chip microcomputer. Mikrokontroler merupakan system computer yang mempunyai satu atau beberapa tugas yang sangat spesifik" (Anna Nur Nazilah Chamim, 2010:4).

Mikrokontroler dapat kita gunakan untuk berbagai aplikasi misalnya untuk pengendalian, otomasi industri, akuisisi data, telekumunikasi dan lain-lain. Keuntungan menggunakan mikrokontroler yaitu harganya murah, dapat diprogram berulang kali, dan dapat kita program sesuai dengan keinginan kita.

D. Mikrokontroler ATmega 16

Mikrokontroler ATmega 16 adalah mikrokontroler jenis AVR yang diproduksi oleh Atmel, keunggulan mikrokontrler AVR ini memiliki 1 siklus clock dengan kecepatan kerjanya yang sangat tinggi sampai 16 MHz, selain kecepatan kerjanya yang tinggi mikroontroler ATmega 16 juga memiliki kapasitas flash memori mencapai 16 Kbyte, SRAM 2 KByte dan EEPROM 1024 byte meliputi 32 buah saluran port I/O, yaitu Port A, Port B, Port C dan Port D dalam mikrokontroler ATmega 16 juga terdapat unit interupsi internal dan external serta port USART untuk komunikasi serial.

Berikut ini adalah susunan pin/kaki dari ATmega 16.

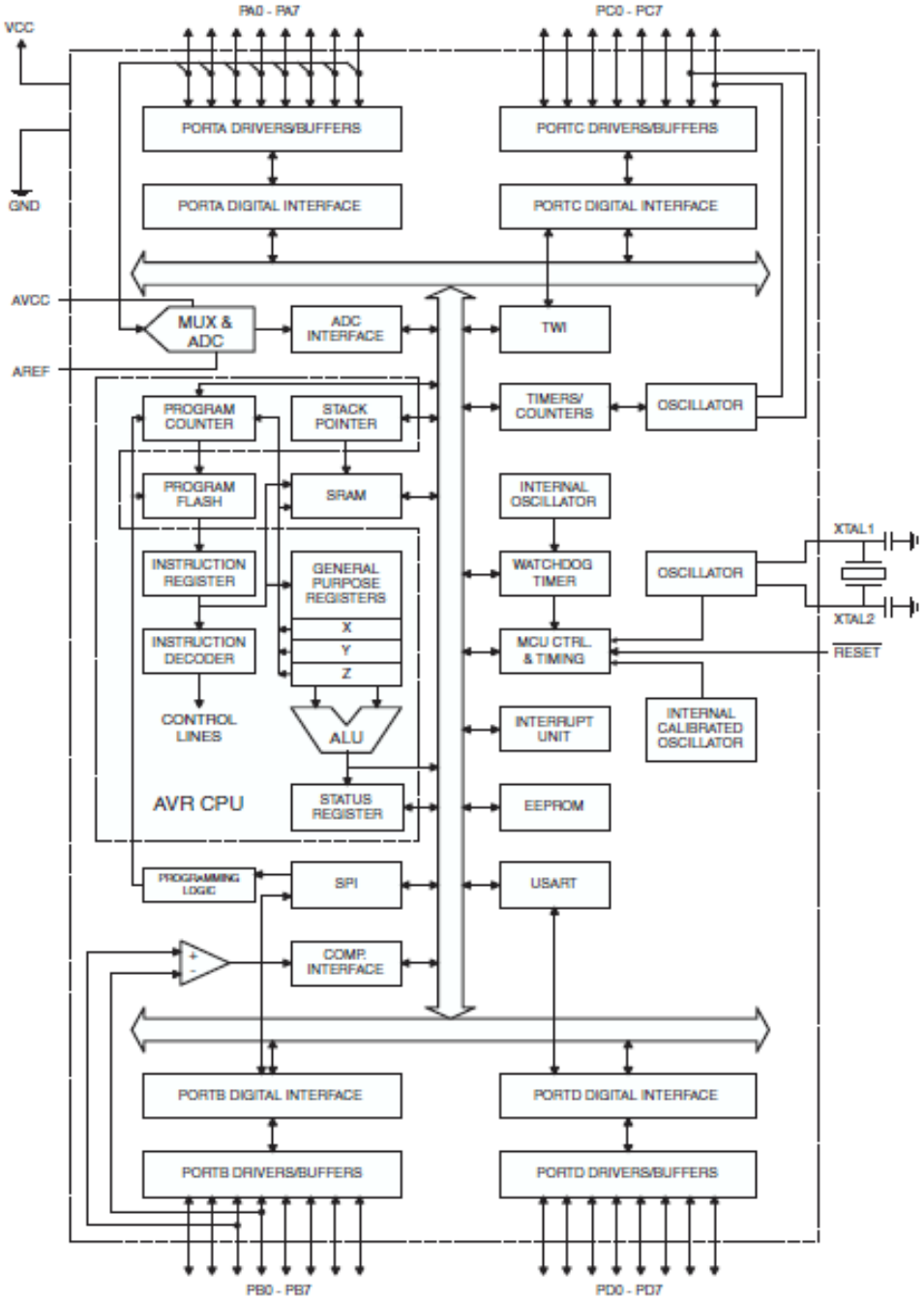
1. VCC sebagai masukan catu daya.
2. GND
3. Port A (PA.0-7) sebagai *input/output* dua arah dan masukan ADC.

4. Port B (PB.0-7) *input/output* dua arah dan pin fungsi khusus seperti SCK, MISO, MOSI, SS, OC0/AIN1, INT2/AIN0, T1, XCK/T0.
5. Port C (PC.0-7) *input/output* dua arah dan pin fungsi khusus seperti TOSC2, TOSC1, TDI,TD0, TMS, TCK, SDA, SCL
6. Port D (PD.0-7) *input/output* dua arah dan pin fungsi khusus seperti RXD, TXD, INT0, INT1, OC1B, OC1A, ICP1,OC2
7. *Reset* untuk mereset mikrokontroler.
8. 8. XTAL 1 dan 2 merupakan pin *clockeksternal*.
9. AVCC untuk tegangan masukan fungsi *ADC.*
10. AREF untuk tegangan referensi *eksternal ADC.*

Berdasarkan jurnal Grendy Christopher dkk, (2002), "Mikrokontroler AVR merupakan pengontrol utama standar industri dan riset saat ini. Hal ini dikarenakan berbagai kelebihan yang dimilikinya dibandingkan mikroprosesor, yaitu murah, dukungan software dan dokumentasi yang memadai dan memerlukan komponen pendukung yang sangat sedikit" (Grendy Christopher dkk, 2002:2).

Digunakan mikrokontroler ATmega 16, karena mikrokontroler AVR ATmega 16 memiliki kecepatan eksekusi yang lebih cepat dan sebagian besar instruksinya dieksekusi dalam satu siklus clock, lebih cepat dibandingkan dengan mikrokontroler MCS 51 yang memiliki arsitektur CISC (Complex Instruction Set Compute) dimana mikrokontroler MCS 51 membutuhkan 12 siklus Clock untuk mengeksekusi 1 instruksi. Selain itu, mikrokontroler ATmega 16 AVR memiliki fitur yang lengkap (ADC Internal, EEPROM Internal, Timer/counter, Watchdog Timer, PWM, PORT I/O, komunikasi serial, komparator, I2C, dll. Sehingga dengan fasilitas yang lengkap ini, programmer dan desainer dapat

menggunakannya untuk berbagai aplikasi sistem elektronika seperti robot, otomasi industri, peralatan telekomunikasi, dan berbagai keperluan lain. Blok diagram ATmega 16 dapat dilihat pada Gambar 1.



Gambar 1. Blok Diagram ATmega 16
(*Datasheet* ATmega 16)

E. Bahasa Pemrograman C

Bahasa pemrograma C merupakan pemrograman yang berfungsi untuk mengolah code program dalam mengolah kode-kode program, compiler C melaksanakan beberapa tahapan yaitu melakukan prapengolahan untuk melakukan persiapan yang diperlukan sebuah berkas program kompilasi.

Berdasarkan jurnal Dian Wirdasari "Akar dari bahasa C adalah bahasa BCPL yang dikembangkan oleh Martin Richard pada tahun 1967. Bahasa ini kemudian dikembangkan oleh Ken Thompson menjadi bahasa B pada tahun 1970. Perkembangan selanjutnya memjadi bahasa C oleh Dennis Richie sekitar 1970-an di Bell Telephone Laboratories (sekarang adalah AT&T Bell Laboratories)"(Dian Wirdasari, 2010:8).

Bahasa C terdiri dari fungsi-fungsi program serta tidak ada perbedaan prosedur, dan fungsi setiap program c mempunyai satu fungsi utama "main" yang bersifat case sensitive (huruf besar dan kecil berbeda setiap statement di akhiri tanda semicolom) (;).

F. Code Vision AVR (CVAVR)

Code Vision AVR merupakan software yang bisa digunakan untuk membuat code program mikrokontroler AVR. Kebanyakan orang yang akan memprogram suatu IC mikrokontroler akan memakai software Code Vision ini.

Berdasarkan naskah Fandhy Bangun Pambajeng, (2015:14), "Code Vision AVR adalah sebuah compiler yang memiliki fasilitas Integrated Development Environment (IDE) dan didesain agar dapat menghasilkan program C secara otomatis untuk mikrokontroler Atmel AVR, program C yang diimplementasikan sesuai dengan arsitektur AVR" (Fandhy Bangun Pambajeng, 2015:14).

Didalam pemrograman Code Vision AVR terdapat fitur-fitur untuk memudahkan orang pada saat akan membuat program selain itu sofware ini juga suport terhadap semua mikrokontroler diantaranya mikrokontroler ATmega 16.

G. Downloader AVR

USBasp adalah USB downloader programer yang diperuntukan untuk mikrokontroler Atmel dari keluarga AVR. Sangat simpel dan mudah dalam membuatnya, hanya membutuhkan sebuah mikrokontroler jenis ATMega48 atau ATMega8 dan beberapa komponen pasif. Programer ini hanya menggunakan firmware USB driver dan tidak diperlukan USB controler khusus.

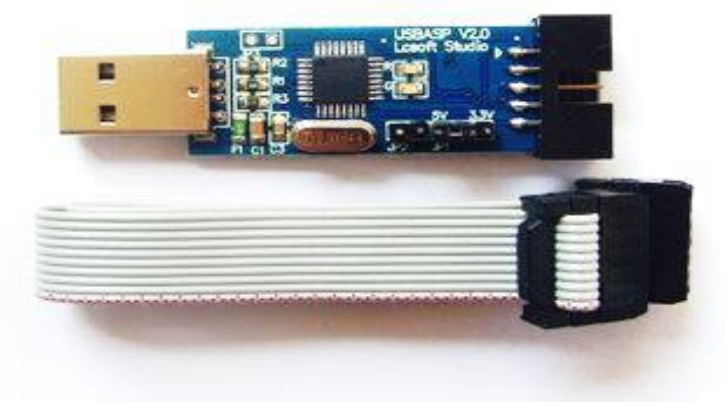
Berdasarkan jurnal Dwi Pipit Haryanto, Anto Cuswanto, (2010). “Downloader berfungsi untuk memasukkan bahasa pemrograman yang telah dibuat kedalam mikrokontroler. Downloader mempunyai beberapa macam merk, namun spesifikasi kegunaannya secara umum adalah sama” (Dwi Pipit Haryanto, Anto Cuswanto, 2010).

Downloader USB berfungsi untuk mendownload atau memindahkan program dari Code Vision AVR ke mikrokontroler yang digunakan yaitu mikrokontroler ATmega 16

Downloader dengan port USBasp ini memiliki kelebihan antara lain :

1. Kompatible dengan OS windows (2k/XP/vista/seven)
2. Tidak memerlukan pengontrol atau komponen smd khusus

3. Kecepatan pemrograman bisa mencapai 5kByte/detik
4. Terdapat jumper untuk opsi slow SCK untuk mendukung mikrokontroler target yang berkecepatan rendah (< 1.5 MHz)
5. Tidak memerlukan tegangan external karena sudah mengambil tegangan dari komputer melalui port usb
6. Terdapat jumper tegangan untuk mikrokontroler target bila ingin mengambil tegangan dari port usb, bila mikrokontroler target ingin menggunakan tegangan external lepas jumpernya. Downloader dapat dilihat pada Gambar 2.



Gambar 2. Bentuk Fisik Downloader AVR
(http://www.priceza.co.id/s/harga/USBasp-Programmer-Downloader)

H. Router

Router adalah perangkat keras jaringan komputer yang dapat digunakan untuk menghubungkan beberapa jaringan yang sama atau berbeda. Router biasa digunakan untuk pemasangan internet oleh sebagian perusahaan (Network) atau intansi tertetu bahkan terkadang oleh masyarakat umum.

Berdasarkan jurnal (Much Aziz Muslim, 2007:12) "Routers adalah peralatan yang bekerja pada layer 3 OSI dan sering digunakan menyambungkan jaringan luas (Wide Area Networking – WAN) atau untuk melakukan segmentasi layer 3 di LAN. WAN seperti halnya dengan LAN juga beroperasi di layer 1, 2 dan 3 OSI sehingga router yang digunakan untuk menyambungkan LAN dan WAN harus mampu saling mendukung" (Much Aziz Muslim, 2007:12).

Router memiliki daya cangkupan yang sangat luas dan memiliki tingkat kecepatan yang sangat tinggi tergantung dari jenis router itu sendiri, selain itu dalam pengoperasian router dapat kita setting sendiri IP addresnya sehingga kita dapat dengan mudah dalam penggunaanya. Biasanya IP yang di setting pada router adalah IP LAN dan IP WAN, IP LAN digunakan hanya untuk penggunaan yang dekat (Lokal) sedangkan IP yang di WAN digunakan untuk cangkupan yang sangat luas bahkan biasa kita akses dimana saja.

Pada proyek akhir ini menggunakan router TP-LINK TL-WR720N karena ada beberapa kelebihan antara lain :

1. Kecepatan *Wireless* sampai dengan 150 Mbps
2. Mendukung sampai dengan 4 SSID jaringan *wireless* dengan SSID dan password yang berbeda
3. Keamanan enkripsi wireless yang mudah dengan menekan Tombol WPS
4. Kontrol bandwidth berbasis IP memungkinkan administrator untuk menentukan berapa banyak bandwidth yang dialokasikan ke setiap PC. Tampilan router dapat dilihat pada Gambar 3.

Gambar 3. Router TP-Link (TL-WR720N)
(http://antarlangit.com/products/Wireless-N-Router-TPLink-TL-WR720N.html)

I. EGSR-7150MJ (Ethernet to Serial Gateway)

EGSR-7150MJ/WIZNET merupakan hardware yang berfungsi untuk melakukan komunikasi serial dengan membuat antarmuka melalui Ethernet dan menggunakan settingan ip untuk dapat terhubung atau berkomunikasi serta menggunakan kabel UTP dan konektor RJ-45 sebagai penghubungnya.

"EGSR-7150MJ adalah suatu modul gateway yang mengubah tipe data serial ke tipe data TCP/IP dan sebaliknya atau lebih dikenal sebagai ethernet to serial gateway" (Stallings William, 1996).

Dengan EGSR-7150MJ yang tersambung dengan konektor RJ-45, user bisa lebih mudah dan lebih cepat untuk membuat antarmuka melalui Ethernet. EGSR- 7150MJ memiliki program Configuration Tool yang

digunakan untuk melakukan konfigurasi TCP/ IP dan serial. Bentuk EGSR-7150MJ dapat dilihat pada Gambar 4.

Gambar 4. Modul EGSR-7150MJ
(http://old.wiznet.co.kr/sub_modules/en/product/Product_Detail.asp?cate1=&cate2=&cate3=&pid=1046)

J. Relay

Relay pada dasarnya adalah sebuah saklar (switch) yang menyambungkan atau memutus kontak tegangan sambung secara mekanik jika diberi tegangan listrik maka relay akan bekerja dan relay akan langsung menutup (terhubung), jika relay tidak mendapatkan tegangan maka relay tidak dapat beroperasi (terputus). Karena relay bersifat normali close (NC) dan normali open (NO).

Menurut Cak Dayat yang tertulis pada websitenya http://www.ulaslistrik.com/ bahwa :

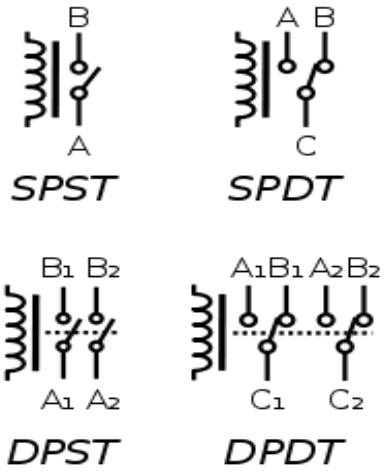
> ”Secara umum, keutamaan penggunaan relay adalah kemampunya bekerja pada rangkaian berdaya rendah dan mampu menangani system pensaklaran dengan daya besar.hal inilah yang membedakan sistem pensaklaran menggunakan transistor.Sistem pada transitor menggunakan

daya kecil penggunaanya pun untuk daya kecil.Relay dengan arus AC tidak lah berbeda kerjanya,hanya coil yang didesign mampu bekerja pada arus jenis AC".

"Dikarenakan relay termasuk dalam golongan saklar,maka relay memiliki istlah pole (Terminal) dan Throw (kondisi),oleh kerena itu dibagi menjadi beberapa kelompok diantranya".

- "Single Pole Single Throw (SPST) : Relay kelompok ini memiliki 1 jenis kontak,dengan 2 terminal 1 terminal input dan satu terminal output".
- "Single Pole Double Throw (SPDT) : Relay kelompok ini memiliki 2 jenis kontak,terdiri dari 3 terminal,satu terminal input dan 2 terminal output dengan 2 kondisi normally NO/NC".
- "Double Pole Single Throw (DPST) : Relay kelompok ini sama seperti dua relay SPST dalam satu rumah namun dikendalikan oleh hanya satu koil,satu kondisi kontak pada masing masing terminal".
- "Double Pole Double Throw (DPDT) : Relay kelompok ini sama dengan relay jenis SPDT dalam satu rumah dan dikendalikan leh hanya satu koil".

"Untuk mendapatkan gambaran lebih jelasnya silahkan perhatikan symbol relay pada Gambar 5".



Gambar 5. Symbol Relay
(http://www.ulaslistrik.com/)

Relay berfungsi sebagai swich/saklar masukan kepada mikrokontroler berdasar kondisi sensor. Dalam penelitian digunakan 4

buah rangkaian driver yang dirangkai secara paralel. Dengan demikian rangkaian driver relay terdiri 4 terminal masukan dan keluaran. Relay dapat dilihat pada Gambar 6.

Gambar 6. Tampilan Relay
(http://teknikelektronika.com/pengertian-relay-fungsi-relay/)

K. Lampu

Lampu dalam kamus besar bahasa indonesia berarti alat untuk menerangi. Dan lampu sudah sangat vital di kehidupan manusia, dan merupakan benda yang paling sering dicari saat matahari mulai tenggelam.

Dioda merupakan komponen elektronik yang terbuat dari semikonduktor. Dioda terdiri atas sambungan P (Positif disebut Anoda) dan N (Negatif disebut Katoda). Lampu dapat dilihat pada Gambar 7.

Gambar 7. Lampu
http://teknikelektronika.com/jenis-jenis-lampu-listrik-simbol-lampu/

L. Android

Pengertian Android adalah sistem operasi berbasis Linux yang dipergunakan sebagai pengelola sumber daya perangkat keras, baik untuk ponsel, smartphone dan juga PC tablet.

"Android adalah sebuah sistem operasi untuk perangkat mobile berbasis linux yang mencakup sistem operasi, middleware, dan aplikasi. Android menyediakan platform terbuka bagi para pengembang untuk membuat aplikasi mereka sendiri" (M. Ichwan, Fifin Hakiky. 2011:2).

Peminat ponsel android saat ini sudah sangat banyak, mengalahkan pasar ponsel lainya seperti iOS, windows phone, symbian, bahkan blackberry sekalipun. Beberapa tahun yang lalu android hanya dipakai oleh para pembisnis dari kalangan menengah ke atas. Alasan mereka menggunakan android adalah untuk memudahkan bisnis mereka. Namun pada zaman sekarang, ponsel android tidak hanya dipakai oleh para

pebisnis saja, banyak para remaja bahkan anak-anak pun telah banyak menggunakan ponsel android.

M. Smartphone

Smartphone merupakan telephone pintar yang saat ini sedang berkembang dan banyak digunakan sebagian besar orang untuk kebutuhan bekomunikasi dengan teman, kerabat, ataupun keluarga.

Berdasarkan jurnal Chuzaimah, Mabruroh, Fereshti Nurdiana Dihan, (2010), "Smartphone atau ponsel cerdas merupakan kombinasi dari PDA dan ponsel, namun lebih berfokus pada bagian ponselnya. Smartphone ini mengintegrasikan kemampuan ponsel dengan fitur komputer - PDA. Smartphone mampu menyimpan informasi, e-mail, dan instalasi program, seperti menggunakan mobile phone dalam satu device" (Chuzaimah, Mabruroh, Fereshti Nurdiana Dihan, 2010),.

Smartphone merupakan telephone yang sangat canggih dengan berbagai macam kelebihan, seiring dengan berkembangnya teknologi saat ini, selain untuk berkomunikasi masih banyak kegunaan lain dari smartphone diantaranya yaitu untuk sistem pengendalian (remot control), seperti halnya dalam proyek ini smartphone digunakan sebagai remot pengendali untuk menghidupkan dan mematikan lampu sehingga lampu dapat dikendalikan dari jarak yang jauh.

N. Telnet

Pada umumnya telnet digunakan untuk mengakses dan mengendalikan komputer yang menggunakan sistem Unix. Selain itu telnet biasa digunakan untuk masuk pada sistem perangkat network seperti router dan switches berbentuk command prompt.

“Telnet adalah salah satu protokol network yang menyediakan remote command-line shell yang berjalan melalui TCP/IP. Telnet pertama kali diciptakan yaitu pada tahun 1970-an dan masih tetap digunakan hingga saat ini” (Mandalamaya, 2014).

Untuk bisa menggunakan telnet terlebih dahulu kita setting ip yang ada pada telnet dan mengikuti ip yang sudah ada pada alat maupun komputer, yang akan kita remote harus memiliki port yang terbuka untuk diakses melalui protokol telnet port pada telnet biasanya masih default yaitu port 23. sedangkan tampilan telnet sendiri dapat dilihat pada Gambar 8.

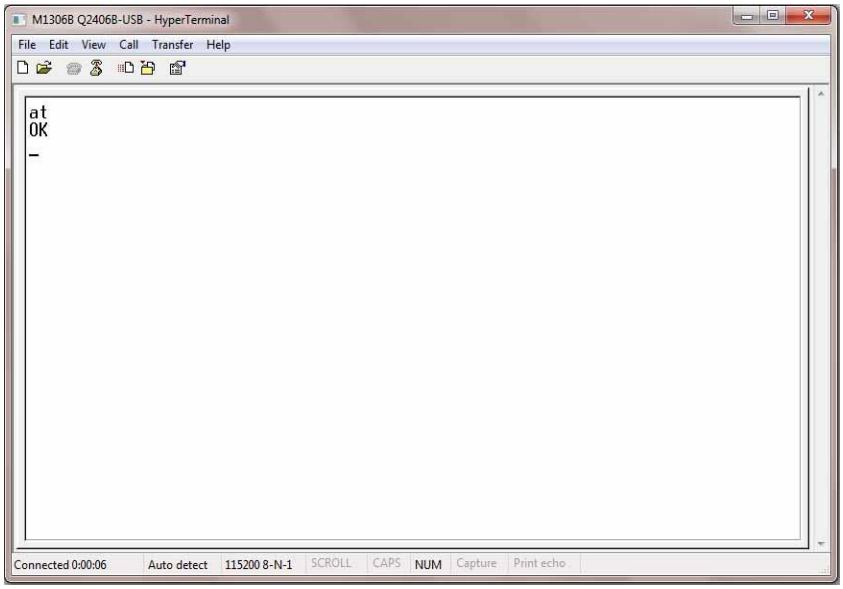
Gambar 8. Tampilan Telnet
(http://www.mochasoft.dk/android_telnet.htm)

O. HyperTerminal

HyperTerminal adalah sebuah program yang dirancang untuk melaksanakan fungsi komunikasi dan emulasi terminal. Program ini akan memanfaatkan modem internal dari host atau komputer utama dan menggunakan layanan seperti Telnet untuk membuat sambungan ke komputer sekunder.

Menurut William, Stalling, (1997) "HyperTerminal adalah sebuah program yang dirancang untuk melaksanakan fungsi komunikasi dan emulasi terminal. Juga dikenal sebagai HyperTerm, tampilan HyperTerminal dapat dilihat pada Gambar 23. Program ini telah ditawarkan sebagai bagian dari sistem operasi Microsoft sejak peluncuran Windows 98. Pada dasarnya, HyperTerminal memungkinkan pengguna komputer memanfaatkan komputer lainnya untuk berhubungan antara dua system" (William, Stalling. 1997).

HyperTerminal dapat dimanfaatkan untuk mentransfer data dan file dari satu sistem ke yang lainnya, tanpa perlu menyimpan data untuk beberapa jenis perangkat luar dan kemudian memuat data secara manual ke sistem lainnya. HyperTerminal memanfaatkan port serial dan kontrol yang terkait dengan perangkat eksternal. Perangkat ini dapat bervariasi dan meliputi opsi sebagai peralatan komunikasi radio, robot, dll. HyperTerminal digunakan sebagai sistem kendali untuk menghidupkan dan mematikan lampu. Tampilan HyperTerminal dapat dilihat pada Gambar 9.



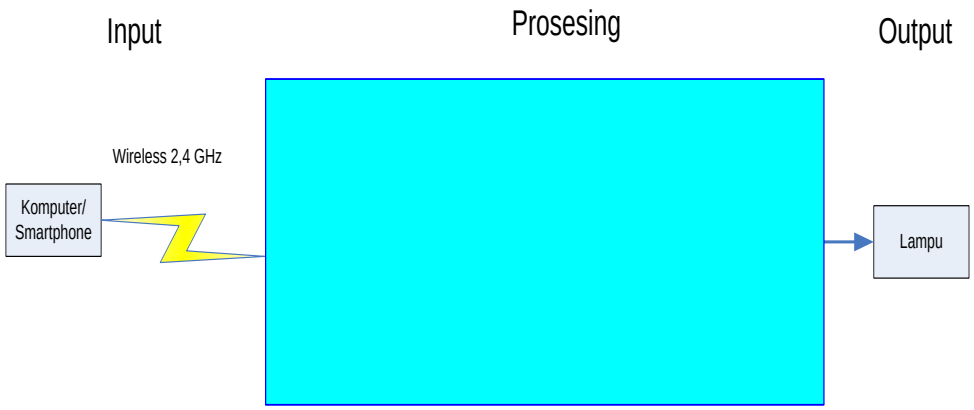
Gambar 9. Tampilan HyperTerminal
(http://toekangmodem.blogspot.co.id/2011/10/test-modem-dengan-at-command-untuk.html)

BAB III

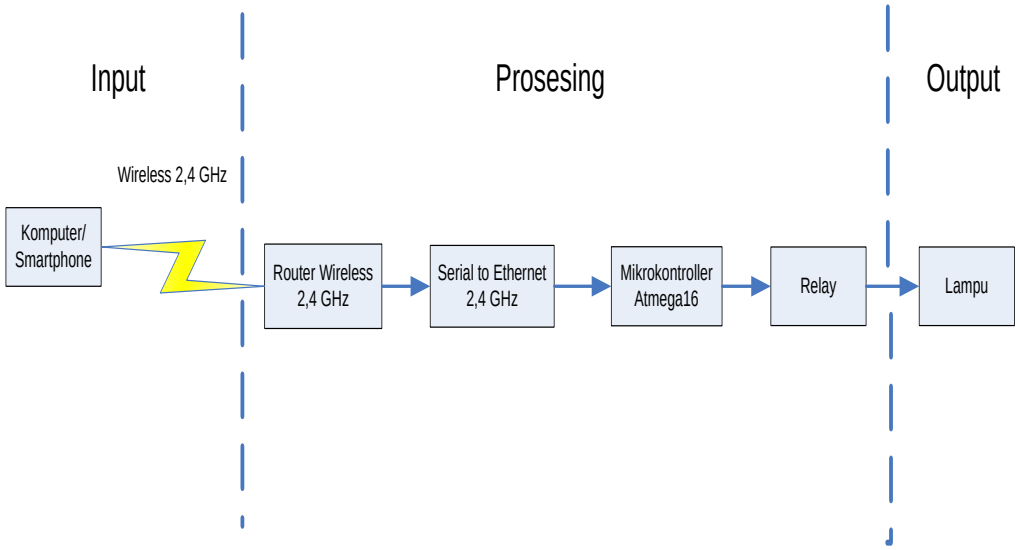
PERANCANGAN ALAT

A. Gambaran Umum Sistem

Aplikasi yang akan dibangun adalah sistem kendali lampu via *wireless* 2,4 GHz berbasis mikrokontroler ATmega 16. Input untuk sistem yang akan dibuat ini berupa data ascii yang dikirim melalui aplikasi telnet. Untuk pengolahnnya digunakan mikrokontroler ATmega 16 sedangkan untuk outputnya berupa 4 lampu dengan tegangan AC 220 V. Diagram blok sistem kendali lampu via *wireless* 2,4 GHz berbasis mikrokontroler ATmega 16 dapat dilihat pada Gambar 10 dan Gambar 11.



Gambar 10. Blok Diagram 1 Sistem Kendali Lampu Via *Wireless* 2,4 GHz Berbasis Mikrokontroler ATmega 16



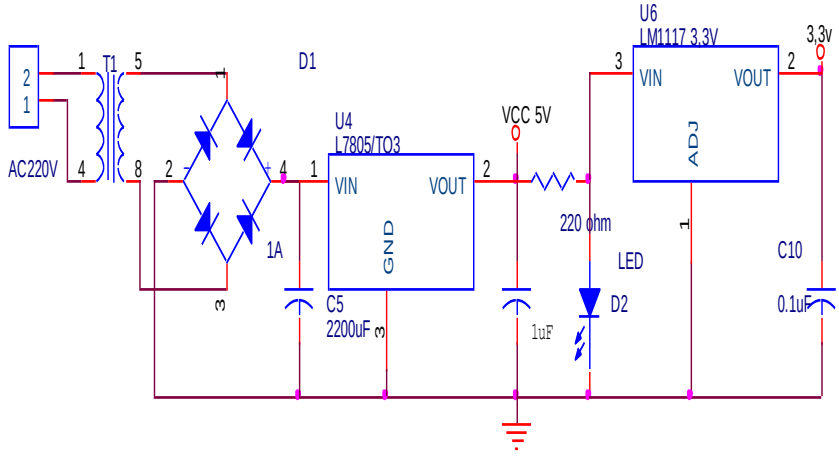
Gambar 11. Blok Diagram 2 Sistem Kendali Lampu Via *Wireless* 2,4 GHz Berbasis Mikrokontroler ATmega 16

Pengguna utama dibuatnya alat ini yaitu masyarakat umum yang biasanya mempunyai aktifitas dan kesibukan yang padat dalam bekerja, terutama di kota besar aktifitas setiap individu masyarakat sangatlah padat dengan berbagai macam pekerjaannya, tentunya memakan waktu dari pagi hingga malam hari. Akibatnya banyak kegiatan rumah tangga yang tertunda, seperti menghidupkan atau mematikan lampu disetiap ruang saat malam dan pagi hari.

Fungsi dari alat sendiri cukup sederhana hanya untuk menghidupkan atau mematikan lampu dengan menggunakan *wireless* 2,4 GHz serta *smartphone* sebagai remot pengendalinya.

B. Rangkaian *Power Supply*

Power Supply yang digunakan dalam pembuatan sistem kendali lampu via *wireless* 2,4 GHz berbasis mikrokontroler ATmega 16, berupa catu daya 5 Volt dan 3,3 Volt. Travo yang digunakan adalah travo step-down (Adaptor) 1 A. Fungsi trafo dalam rangkaian ini untuk menurunkan tegangan dari 220 V AC ke 12 Volt AC. Untuk merubah tegangan AC ke DC menggunakan dioda bridge 1 A, dan untuk menurunkan tegangan 12 V DC ke 5 Volt DC dan 12 Volt ke 3,3 Volt menggunakan regulator LM1117. LM1117 merupakan regulator untuk menurunkan tegangan ke 3,3 Volt. Mikrokontroler membutuhkan tegangan/catu daya 5 Volt ± 10%. Rangkaian *Power Supply* ditunjukkan pada Gambar 12.

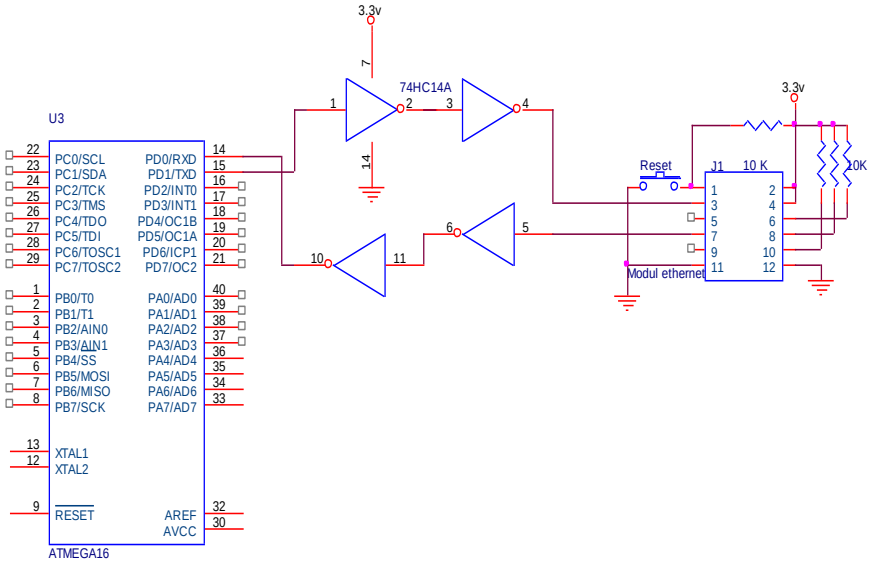


Gambar 12. Rangkaian *Power Supply*

C. Rangkaian Logic Converter dan Ethernet (TCP/IP to serial)

Pada rangkaian modul ethernet ditunjukan pada Gambar 13 terhubung dengan mikrokontroler secara serial, komunikasi serial hanya membutuhkan 2 penghubung yaitu TX dan RX, keluaran dari modul ethernet adalah data serial dengan tegangan LVTTL. Untuk LVTTL

tegangan yang digunakan adalah 3,3 Volt sehingga dalam hubungannya dengan mikrokontroler dibutuhkan suatu rangkaian *buffer*, yaitu menggunakan IC 74HC14 yang dicatu dengan tegangan 3,3 Volt.



Gambar 13. Rangkaian *Interface* Mikrokontroler dengan Modul TCP ke Serial

D. Mikrokontroler ATmega 16

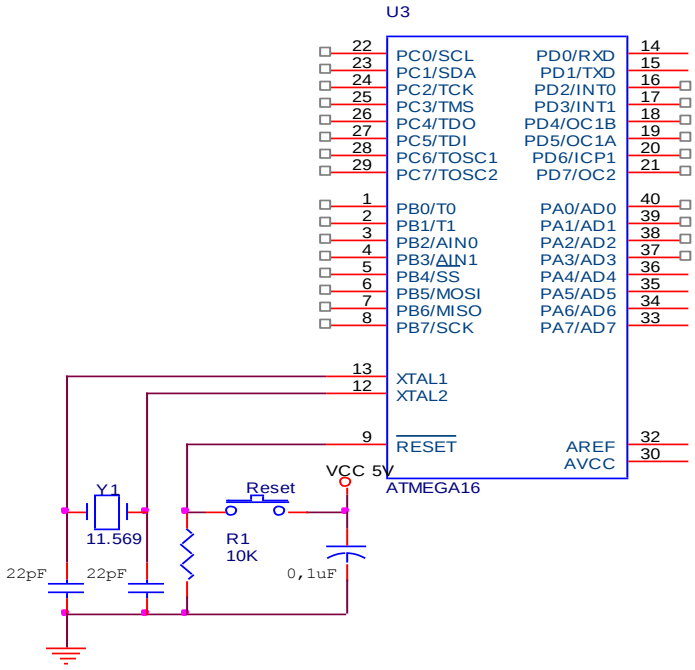
1. Osilator

Pada rangkaian osilator ini digunakan kristal 11,0592 MHz seperti ditunjukkan Gambar 14. Menurut *datasheet* crystal yang bisa digunakan untuk mikrokontroler ATmega 16 adalah 0 – 16 MHz dan dua kapasitor 22 pF. Waktu yang dibutuhkan untuk melakukan eksekusi sebuah instruksi dinamakan waktu siklus instruksi (instruction cycles time) yang nilainya tergantung pada kristal yang digunakan.

2. Reset

Rangkaian *reset* digunakan untuk menghentikan kerja mikrokontroler dengan kembali ke alamat 0000/reset. Rangkaian *reset* dapat dilihat pada Gambar 14. Untuk mereset mikrokontroler ATmega 16 yaitu dengan memberikan logika Low pada pin reset (pin 9) mikrokontroler ATmega 16, logika low ini dibuat minimal 50 ns. Keadaan yang dapat membuat mikrokontroler masuk kedalam kondisi reset adalah:

1) Pada saat Power On
2) Saat reset eksternal terjadi, yaitu ketika pin reset diaktifkan
3) Pada saat watchdog timer mencapai nilai maksimum (overflow)



Gambar 14. Rangkaian Sistem Minimum ATmega 16

E. Rangkaian *Driver Relay*

Untuk mengaktifkan relay, dibutuhkan arus yang cukup besar. Karena arus yang disediakan oleh port mikrokontroler tidak cukup untuk mengaktifkan relay, diperlukan *driver*, yaitu menggunakan IC ULN 2803. IC ULN 2803 ini merupakan susunan tujuh buah transistor darlington yang dirancang khusus untuk keperluan antarmuka dari sistem digital dengan arus lemah ke sistem yang membutuhkan tegangan dan arus yang lebih tinggi.

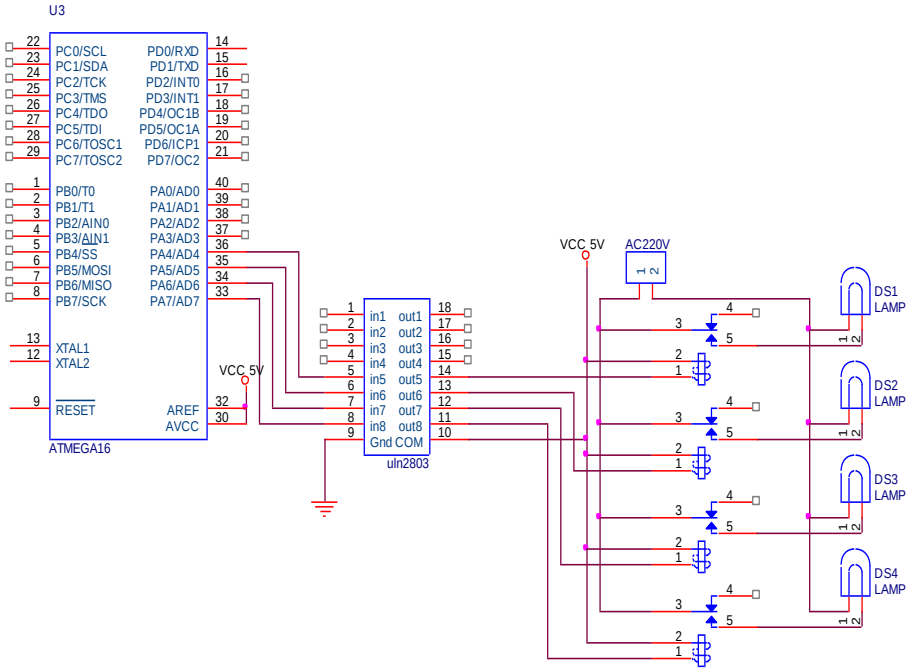
Masukan IC ULN 2803 dihubungkan dengan port mikrokontroler, keluaran IC ULN 2803 dihubungkan ke relay, 'COM' (kaki 10 ULN 2803) dihubungkan ke sumber tegangan 5 Vdc, dan 'GND' dihubungkan ke *ground*.

Bila mendapat masukan logika '1', maka transistor darlington 'on', sehingga menghubungkan salah satu kutub lilitan relay ke *ground*, dan relay akan aktif. dan bila mendapat masukan logika '0', maka transistor darlington 'off' sehingga hubungan kutub lilitan relay ke *ground* terbuka, dan relay 'off'.

IC ULN 2803 telah dilengkapi dengan dioda yang dipasang dengan bias negatif (*reverse bias*) pada keluaran dan com. Dioda ini berfungsi melindungi transistor dari tegangan induksi balik yang timbul dari lilitan relay saat transistor dimatikan.

Relay yang digunakan adalah relay dengan satu buah kontak NO (*normaly open*). Kontak NO digunakan untuk menyaklar peralatan listrik

ke tegangan 220 volt, rangkaian *driver* relay dan objek yang dikontrol dapat dilihat pada Gambar 15.



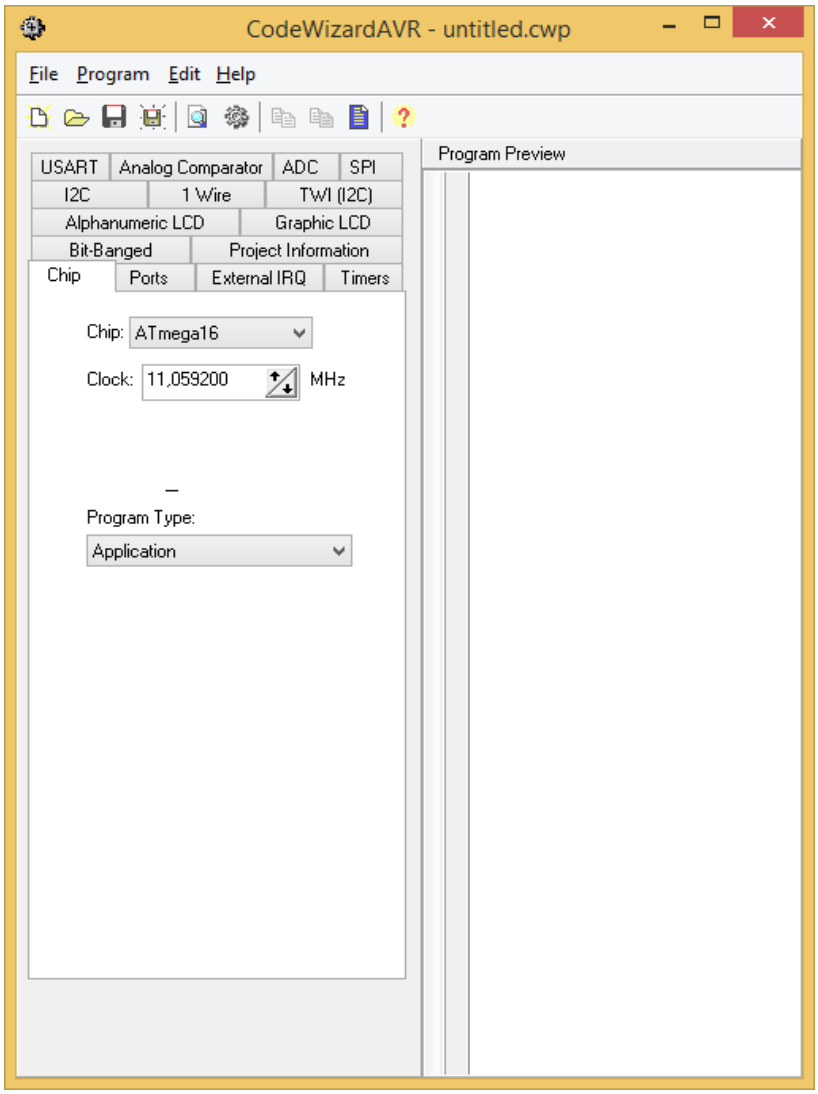
Gambar 15. Rangkaian *Driver Relay*

F. Perancangan Perangkat Lunak

Agar system pada mikrokontroler bekerja dengan sebagaimana mestinya, maka diperlukan perangkat lunak yang mengatur kerja dari keseluruhan rangkaian. Pertama-tama yang dibuat adalah diagram alir (*Flowchart*) dan kemudian dilakukan pembuatan program. Pembuatan program ditulis dengan bahasa C menggunakan tool codevision AVR, dan program tersebut disimpan dalam memori flash mikrokontroler ATmega 16. Pada mikrokontroler ATmega 16 terdapat memori program sebesar 16 kbyte flash, EEPROM 512 byte dan memori data 512 byte RAM.

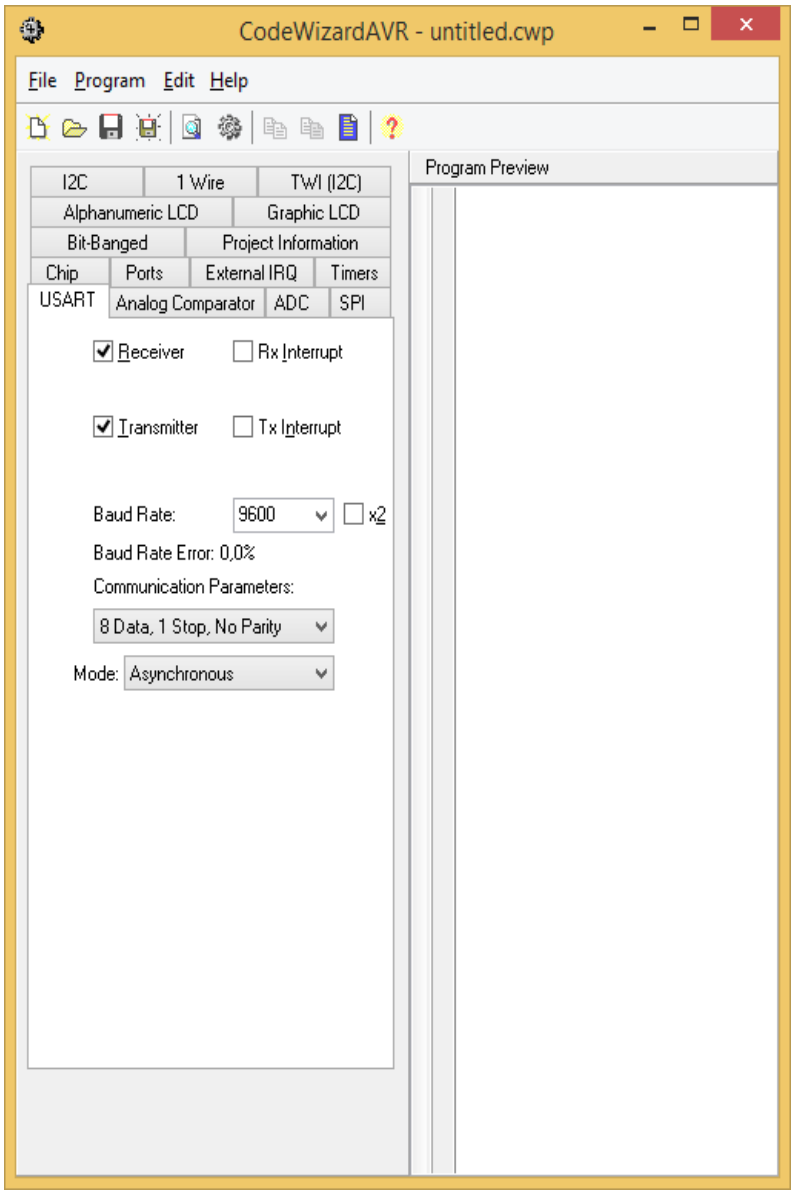
1. **Setting Code Vision**

Setting awal code vision dengan menentukan jenis IC Mikrokontroler yang digunakan. Mikrokontroler yang digunakan adalah ATmega 16, crystal yang digunakan 11,0592 MHz. Pengaturan Code Vision untuk pemilihan jenis mikrokontroler dapat dilihat pada Gambar 16.



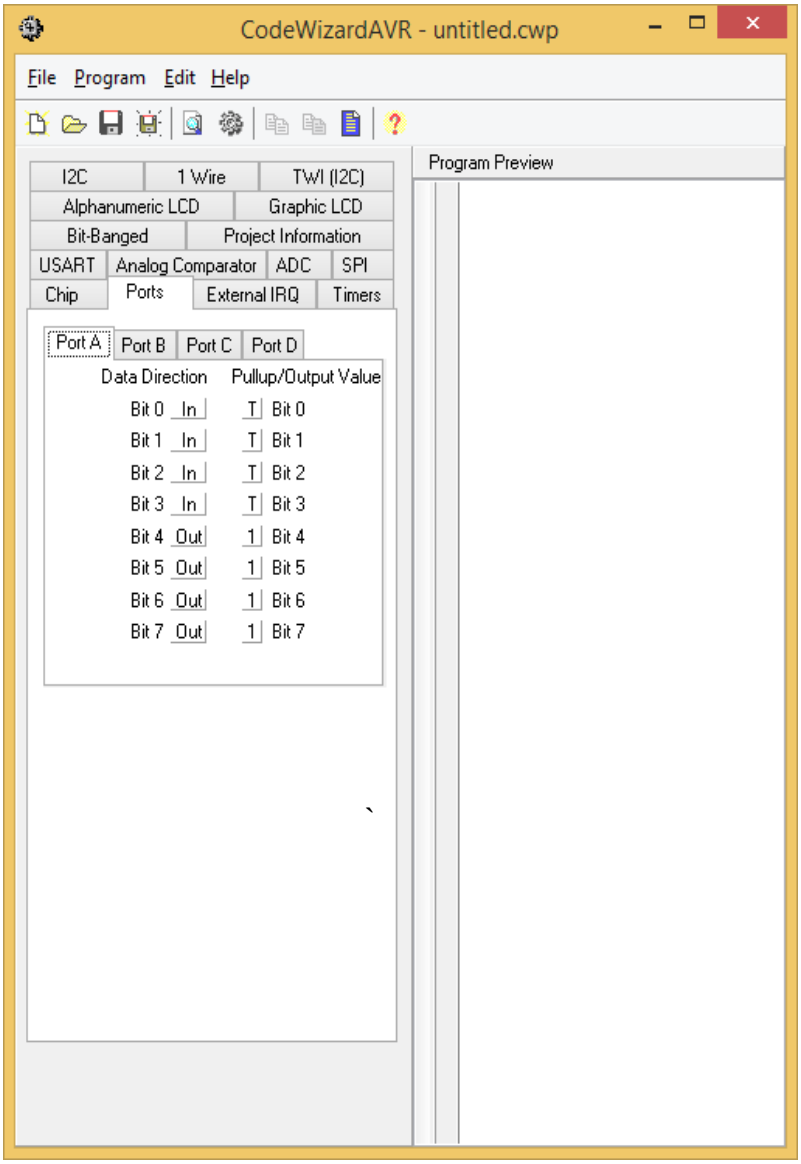
Gambar 16. Setting Code Vision untuk Pemilihan Jenis Mikrokontroler

ATmega 16 mempunyai 1 usart. Komunikasi serial yang digunakan menggunakan kecepatan transfer 9600 bps, dengan menggunakan interupsi ketika ada data serial masuk tetapi tidak memakai interupsi ketika berhasil mengirim data. Pengaturan Code Vision untuk pengaturan komunikasi serial dapat dilihat pada Gambar 17.



Gambar 17. Pengaturan Komunikasi Serial Mikrokontroler pada Code Vision

Untuk pengendalian 4 relay menggunakan PORTA.4 sampai PORTA.7 dengan memilih masing-masing port dijadikan keluaran. Pengaturan Code Vision untuk pengaturan PORTA dapat dilihat pada Gambar 18.



Gambar 18. Setting Keluaran PORTA pada Code Vision

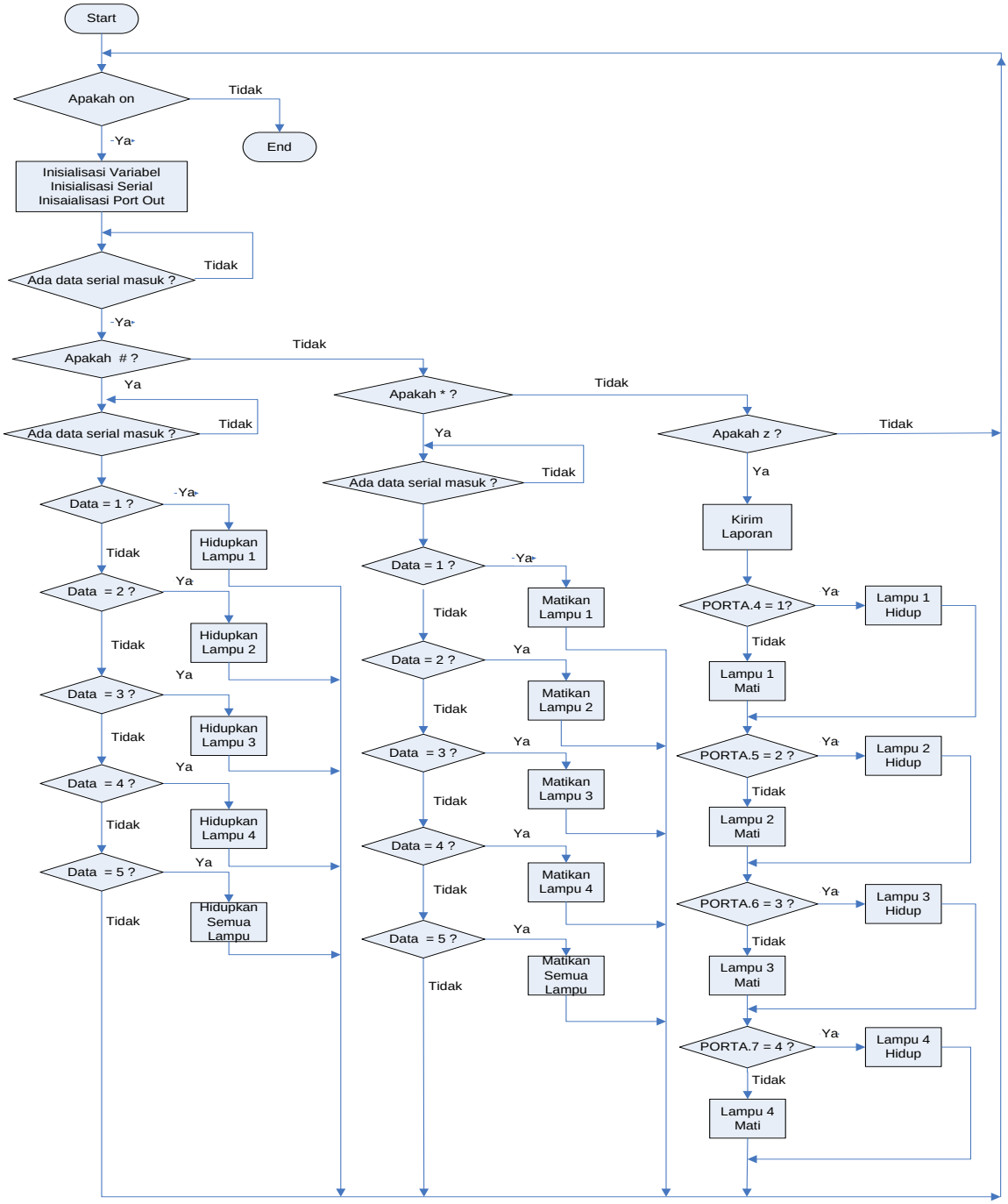
2. Program Utama

Program utama berisi program untuk mendeteksi perintah yang diterima dari serial. Yaitu perintah menghidupkan atau mematikan lampu. Untuk menghidupkan lampu diawali tombol Pagar (#) diikuti kode lampu. Sedangkan untuk mematikan lampu dengan kode Bintang (*) diikuti kode lampu, dan untuk melihat laporan pada lampu menggunakan kode Zet (z). Format mematikan dan mengidupkan serta melihat laporan lampu dapat dilihat pada Tabel 1. sedangkan program utama Flowchart dapat dilihat pada Gambar 19.

Tabel 1. Kode Program

NO	Kode	Fungsi
1	Pagar (#)	Menghidupkan
2	Bintang (*)	Mematikan
3	Zet (z)	Melihat Laporan
4	1	Lampu 1
5	2	Lampu 2
6	3	Lampu 3
7	4	Lampu 4

3. Flowchart



Gambar 19. Flowchart Program Utama

BAB IV

PENGUJIAN DAN PEMBAHASAN

A. Metode Pengujian

Berdasarkan spesifikasi sistem yang telah dijelaskan di bab III sebelumnya, selanjutnya dilakukan pengujian terhadap sistem menggunakan beberapa metode pengujian. Tujuan pengujian ini untuk membuktikan apakah sistem yang diimplementasikan telah memenuhi spesifikasi yang telah direncanakan sebelumnya. Hasil pengujian akan dimanfaatkan untuk menyempurnakan kinerja sistem dan sekaligus digunakan dalam pengembangan lebih lanjut.

Metode pengujian dipilih berdasarkan fungsi operasional dan beberapa parameter yang ingin diketahui dari sistem tersebut. Data yang diperoleh dari metode pengujian yang dipilih tersebut dapat memberikan informasi yang cukup untuk keperluan penyempurnaan sistem.

Dalam penelitian ini dipilih dua macam metode pengujian, yaitu pengujian fungsional dan pengujian kinerja sistem.

Pengujian fungsional digunakan untuk membuktikan apakah sistem yang diimplementasikan dapat memenuhi persyaratan fungsi operasional seperti yang direncanakan.

Pengujian kinerja sistem dimaksudkan untuk memperoleh beberapa parameter yang dapat menunjukkan kemampuan dan kehandalan sistem dalam menjalankan fungsi operasionalnya.

B. Pengujian Fungsional

Sebagaimana dijelaskan sebelumnya, pengujian fungsional bertujuan untuk memeriksa fungsi operasional sistem yang diimplementasikan apakah telah sesuai dengan spesifikasi yang direncanakan dan sistem menjalankan fungsinya sesuai dengan tujuan pengembangannya.

1. Pengujian Rangkaian *Power Supply*

Pengujian rangkaian *power supply* dilakukan dengan mengukur tegangan keluaran dari rangkaian *power supply*.

Hasil pengukuran dapat dilihat pada Tabel 2.

Tabel 2. Hasil Pengukuran Tegangan *Power Supply*

No	Titik Pengukuran	Pengukuran	
		I	II
1	Masukan Trafo	216 VAC	215 VAC
2	Masukan Dioda Bridge	10,5 VAC	10,6 VAC
3	Keluaran Dioda Bridge	13,8 Volt	13,18 Volt
4	Keluaran LM1117 3,3 Volt	3,32 Volt	3,31 Volt
5	Keluaran LM 7805 5 Volt	4,97 Volt	5,03 Volt

Dari hasil pengukuran dapat dilihat keluaran tegangan LM 1117 3,3 Volt maupun 5 Volt memenuhi syarat yaitu tidak lebih atau kurang dari 5 % tegangan yang dikehendaki.

Alat yang digunakan untuk mengukur tegangan power supply diatas menggunakan multimeter digital model DT-830D yang memiliki spesifikasi dan tingkat akurasi sebagai berikut :

❖ Spesifikasi :

- Akurasi dijamin selama 1 tahun, 23 ° C ± 5 ° C, kurang dari 75 % RH

Tabel 3. Tegangan DC

RANGE	RESOLUTION	ACCURACY
200Mv	100Mv	±0.5% 5D
2000mV	1mV	
20V	10mV	
200V	100mV	
1000V	1V	±1.0% 5D

Impedansi masukan : 1MΩ

Max. tegangan input : 1000V DC atau 750V AC rms (untuk rentang 200mV : 500V DC atau rms 350VAC).

Tabel 4. Tegangan AC

RANGE	RESOLUTION	ACCURACY
200V	100mV	±1.2% 10D
750V	1V	

Frekuensi respon : 45-400Hz

Tegangan Max.input : rms 750V AC

Tampilan : gelombang sinus rms. respon rata-rata

Tabel 5. DC Lancar

RANGE	RESOLUTION	ACCURACY
200 μA	100nA	±1.0% 5D
2000 μA	1μA	
20mA	10μA	
200mA	100μA	±1.2% 5D
10A	10mA	±2.0% 5D

Perlindungan yang berlebihan : F 250mA / 250V Fused

(10A disatukan)

Tabel 6. Resistansi

RANGE	RESOLUTION	ACCURACY
200 Ω	0.1 Ω	±1.2% 5D
2000 Ω	1 Ω	±1.0% 5D
20K Ω	10 Ω	
200K Ω	100 Ω	
2000K Ω	1K Ω	±1.2% 5D

Max. tegangan rangkaian terbuka : sekitar 3V.

Tabel 7. Temperature

RANGE	RESOLUTION	ACCURACY
0°C~+1000°C	1ºC	±3°C ±2D<150°C
		±3% >150°C

1. hFE

   Vcc tentang 3V, Ib tentang 10μA, menampilkan hFE 1-1000

2. Dioda dan Buzzer

   Diode : Pengujian Tegangan tentang 2.8V, saat ini sekitar 1mA. Itu perkiraan drop tegangan maju dalam mV akan ditampilkan.

   Buzzer : buzzer akan berbunyi bila kurang dari 50.

3. Signal Output

   Sinyal output : gelombang persegi (50Hz) atau gelombang sinus (1000Hz).

   Tingkat output : 5Vp – p

   Tabel 8. Uji Baterai

Range	Description	Test Condition
1.5V	Tegangan kerja baterai akan ditampilkan pada LCD, sehingga bahwa kualitas baterai bisa dinilai.	Arus kerja sekitar 20mA
9V		Arus kerja sekitar 5mA .

❖ Spesifikasi Umum

- Display : ---------- 3 1/2 LCD digit dengan maksimal bacaan Tahun 1999.
- Polaritas : ---------- Auto polaritas indikasi.

- Indikasi Overrange : - Satunya tokoh " 1 " pada layar.
- Lingkungan operasi : temp. 0 ~ 40 * ; kelembaban relatif < 75 %.
- Suhu penyimpanan: - 15 * ~ 50 * ; kelembaban relatif < 90 %.
- Baterai : ---------- 9V 6F22 / 1.5V X 2 AA (untuk DT830BL saja).
- indikasi baterai rendah : " " atau " BAT " muncul di layar.
- Dimensi : ---------- 139 mm 78mm 40mm.
- Berat: ---------- 170g.
- Konsumsi daya: ---------- 20mW.

Gambaran fisik multimeter digital model DT-830D dapat dilihat pada Gambar 20.

Gambar 20. Multimeter Digital DT-830D

2. Pengujian Rangkaian Reset Mikrokontroler

Rangkaian reset berfungsi menghentikan kerja CPU dan kemudian mengulang dari awal (program counter ke alamat 0000). Saat catu daya dihidupkan rangkaian reset menunda kerja dari CPU hingga tegangan stabil (power on reset). Reset pada mikrokontroler ATmega 16 adalah aktif rendah.

Pengujian dilakukan dengan menghidupkan mikrokontroler mengamati kerja mikrokontroler. Dari hasil pengamatan dapat disimpulkan reset mikrokontroler dapat berfungsi dengan baik, power on reset dapat berfungsi dengan baik, dengan tanda mikrokontroler dapat langsung bekerja ketika power dihidupkan. Begitu juga ketika dilakukan reset pada mikrokontroler melalui tombol maka mikrokontroler dapat melakukan reset.

3. Pengujian Router Tp-Link *Wireless* 2,4 GHz

Pengujian router *wireless* 2,4 GHz dilakukan dengan mengukur kemampuan pengiriman data berbanding dengan jarak antara penerima dan pemancar. Menggunakan *smartphone* yang dikoneksikan ke router tp-link menggunakan media *wireless*. Hasil dari pengujian dapat dilihat pada Tabel 9.

Tabel 9. Hasil Uji Daya Jangkau Router TP_Link TL720N

No	Jarak Router dan Penerima (meter)	Keterangan (signal streng)
1	3 m Tanpa penghalang	Penuh 4 strip
2	4 m Terhalang 1 dinding	3 strip
3	8 m Terhalang 2 dinding	2 strip

4	15 m Terhalang 3 dinding	1 strip
5	15 m Tanpa halangan	Penuh 4 strip

Hasil pengujian pada Tabel 3 dapat dilihat kemampuan pemancar dan penerima untuk dapat berkomunikasi jarak maksimal adalah meter, dalam kondisi tanpa halangan.

4. Pengujian Rangkaian *Driver Relay*

Relay digunakan sebagai piranti antarmuka antara tegangan rendah dengan tegangan tinggi. Dalam rangkaian ini, digunakan relay dengan satu kutub. Relay digunakan untuk mengendalikan empat lampu dangan teganan 220 Volt. Rangkaian *driver* ini akan bekerja ketika ada masukan tinggi (5 Vdc) pada bagian masukan IC ULN 2803 sehingga mengaktifkan pasangan transistor darlington yang bersesuaian dalam IC tersebut sebagai penggerak (*driver*) relay.

Pengujian dilakukan dengan cara memberikan kondisi berbeda pada port-port I/O mikrokontroler ATmega 16 yang mengatur masukan *driver* dan mengukur besar tegangan port-port I/O yang berfungsi mendeteksi keadaan relay. Tabel 10 berikut adalah hasil pengujian rangkaian *driver relay*.

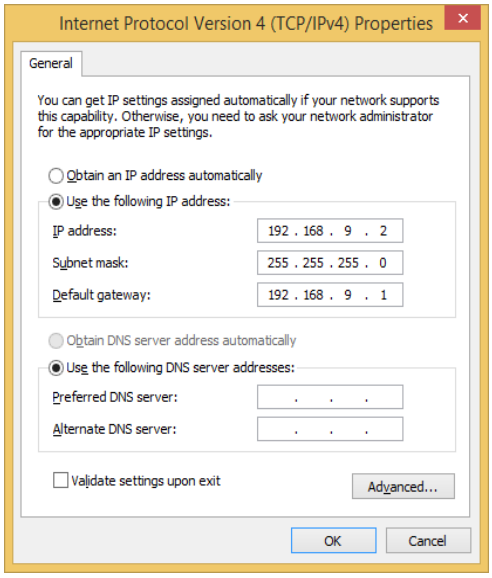
Tabel 10. Pengujian Relay

Relay ke	Port ATMega 16	Kondisi	Keterangan
1	PORTA.4	1	Relay aktif
2	PORTA.5	1	Relay aktif

3	PORTA.6	1	Relay aktif
4	PORTA.7	1	Relay aktif
5	PORTA.4	0	Relay non aktif
6	PORTA.5	0	Relay non aktif
7	PORTA.6	0	Relay non aktif
8	PORTA.7	0	Relay non aktif

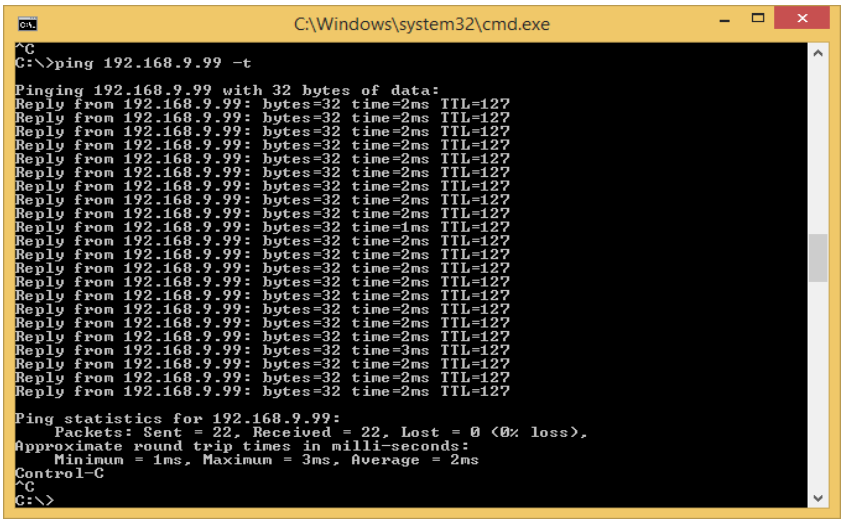
5. Pengujian Rangkaian Modul Ethernet

Pengujian rangkaian ini dengan menghubungkan alat ke komputer. Hubungan modul dengan komputer adalah jenis *peer to peer* sehingga sambungan yang digunakan adalah *cross*. Kabel yang digunakan jenis utp dengan konektor RJ45. Setelah tersambung dengan komputer langkah selanjutnya adalah dengan mengatur IP komputer disesuaikan dengan IP modul ethernet. IP modul ethernet menggunakan IP 192.168.9.99 sehingga komputer kita set dalam range IP modul ethernet yaitu 192.168.9.2 sampai dengan 192.168.9.254, kecuali ip 192.168.9.99 karena sudah dipakai oleh modul ethernet. Tampilan pengaturan IP komputer dapat dilihat pada Gambar 21.



Gambar 21. Pengaturan IP pada Komputer

Pengecekan koneksi antara modul dan komputer dengan menggunakan program DOS yaitu dengan instruksi PING, hasil pengecekan komunikasi antara modul IP dan alarm dapat dilihat pada Gambar 22.



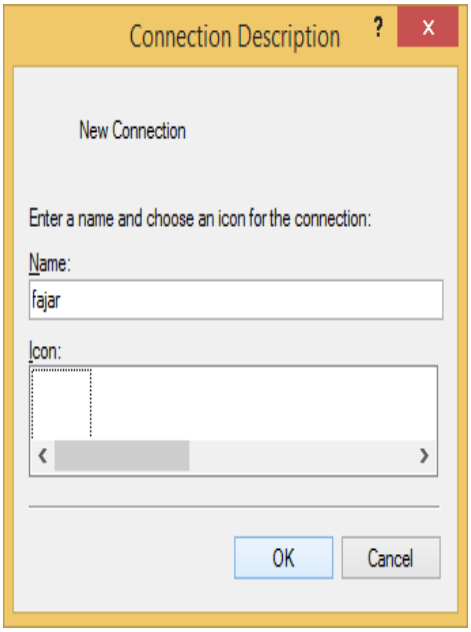
Gambar 22. Pengecekan Komunikasi dengan PING

Dari hasil pengujian dapat dilihat komunikasi antara modul ethernet dan komputer bagus, dengan ping time < 10 ms.

C. Pengujian dan Pembahasan Sistem Keseluruhan

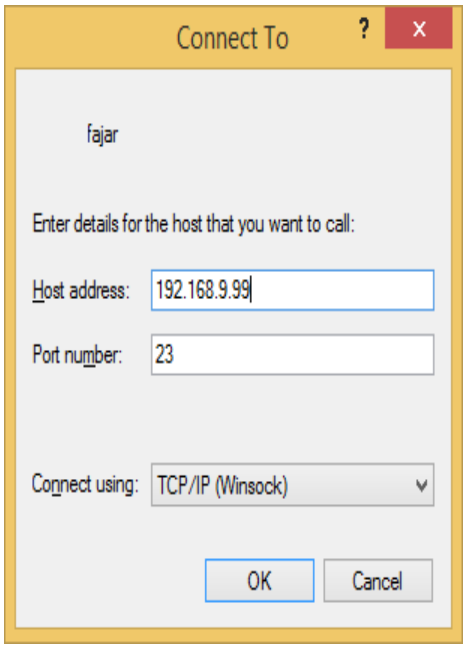
Untuk memulai menjalankan alat ini yaitu dengan menghubungkan alat sistem kendali lampu via *wireless* 2,4 GHz berbasis mikrokontroler ATmega 16 dengan *power supply* PLN 220 VAC, sedangkan *power supply* radio router tp-link diambilkan dari adaptor 12 V. Kemudian dilanjutkan dengan menjalankan aplikasi HyperTerminal komputer dan telnet di perangkat *smartphone*.

Pengujian keseluruhan dengan komputer windows. Untuk memulai menjalankan alat ini yaitu dengan membuka aplikasi bawaan dari windows. Untuk penampilan hasil bisa menggunakan DOS bisa juga menggunakan HyperTerminal. Untuk pengujian kali ini dengan menggunakan HyperTerminal. Step-step penggunaan HyperTerminal dapat dilihat pada Gambar 23.



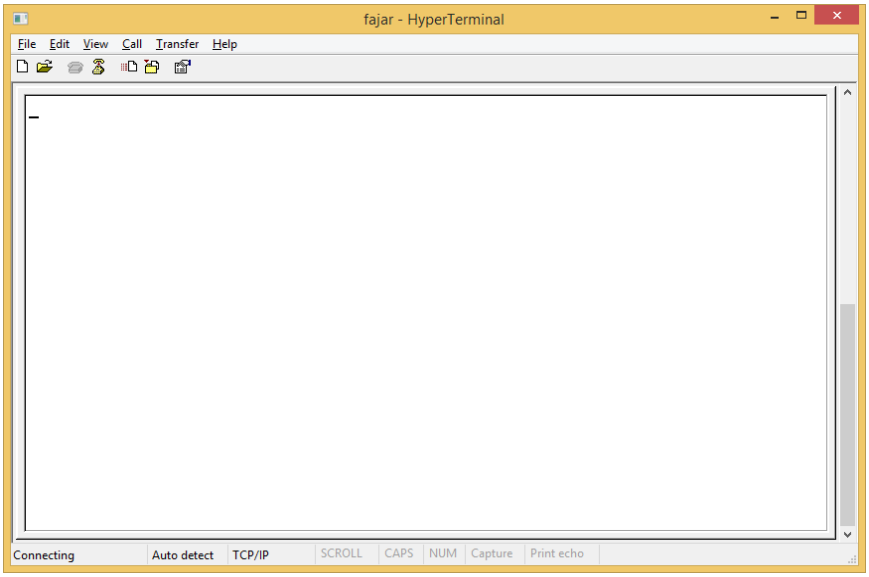
Gambar 23. Step 1 Pengaturan HyperTerminal

Setelah step 1 dikerjakan kemudian dengan menekan OK, tampilan selanjutnya dapat dilihat pada Gambar 24.



Gambar 24. Step 2 Pengaturan HyperTerminal

Setelah step 2 akan tampil jendela HyperTerminal. Tampilan program HyperTerminal dapat dilihat pada Gambar 25.



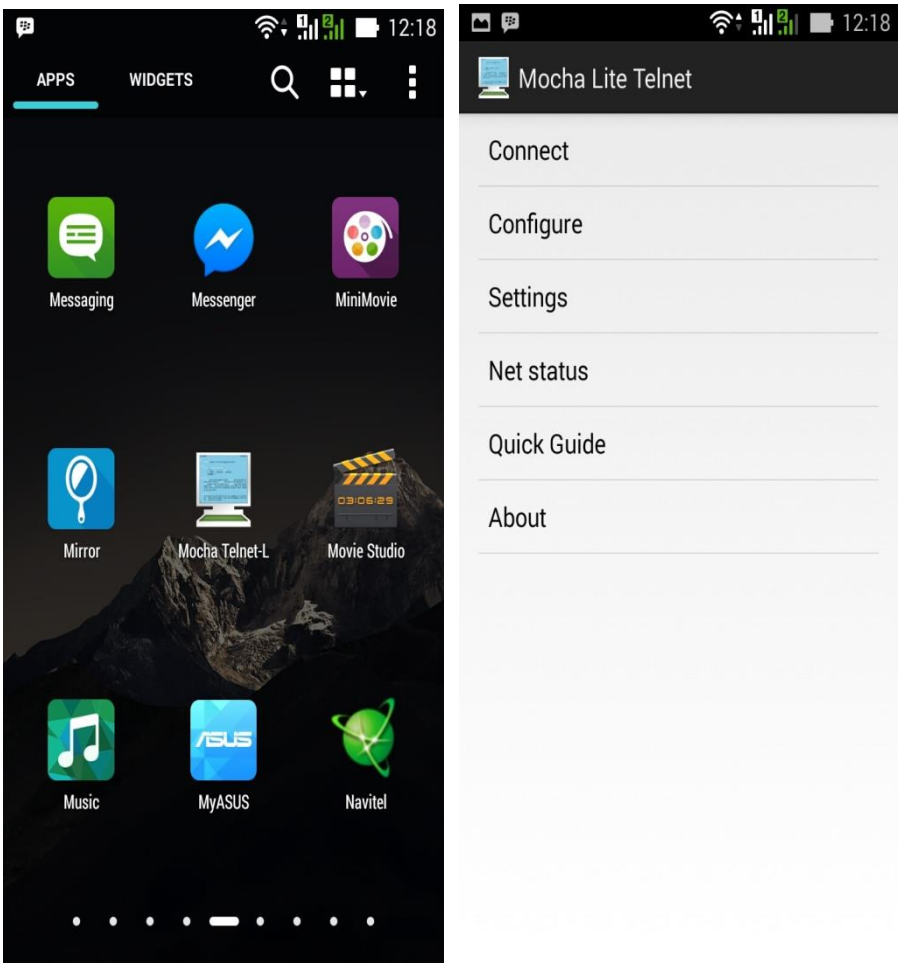
Gambar 25. Tampilan HyperTerminal

Setelah terkoneksi dilakukan pengujian untuk mematikan dan menghidupkan lampu. Hasil pengujian menggunakan HyperTerminal windows dapat dilihat pada Tabel 11.

Tabel 11. Hasil Pengujian Menggunakan HyperTerminal Windows

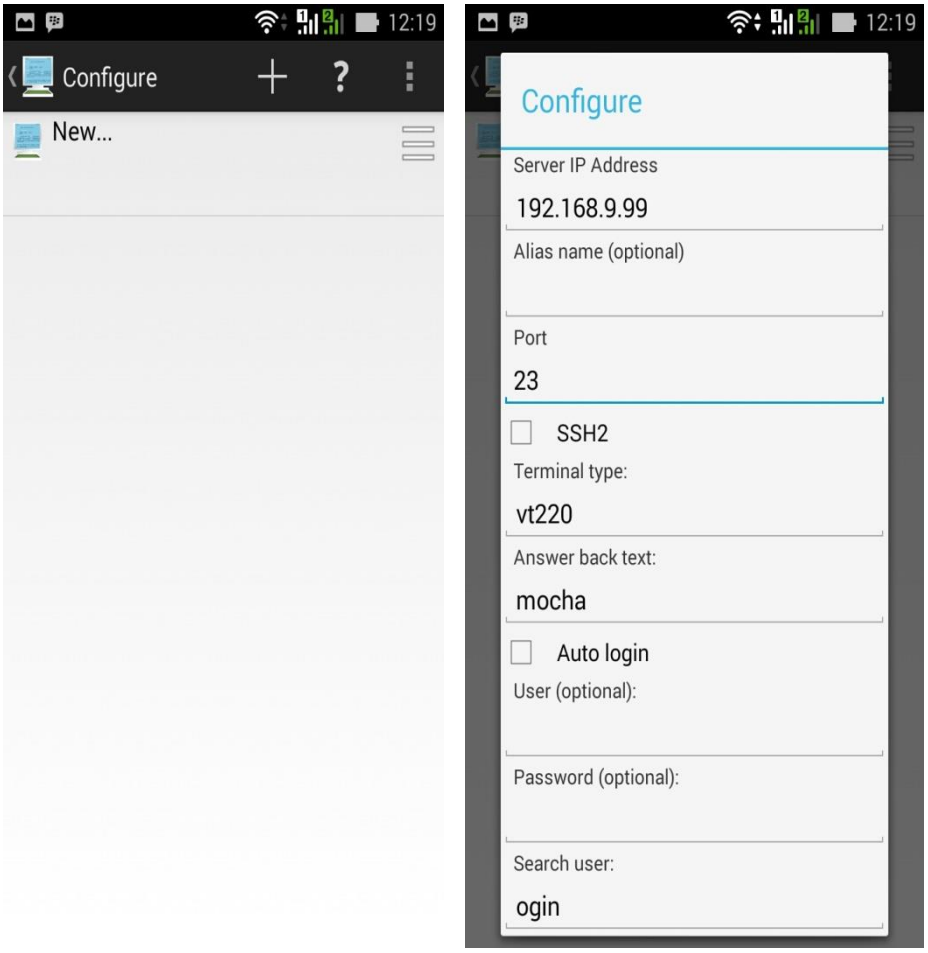
NO	Perintah	Kondisi Lampu
1	#1	Lampu 1 hidup
2	*1	Lampu 1 mati
3	#2	Lampu 2 hidup
4	*2	Lampu 2 mati
5	#3	Lampu 3 hidup
6	*3	Lampu 3 mati
7	#4	Lampu 4 hidup
8	*4	Lampu 4 mati
9	#5	Lampu semua hidup
10	*5	Lampu semua mati

Pengujian keseluruhan dengan *smartphone android*, aplikasi yang dipakai untuk pengujian sistem kendali lampu via *wireless* 2,4 GHz berbasis mikrokontroler ATmega 16 adalah mocca telnet, aplikasi dapat diunduh lewat playstore. Step-step penggunaan mocca telnet yang pertama buka aplikasi mocca telnet, tampilan mocca telnet diandroid dapat dilihat pada Gambar 26.



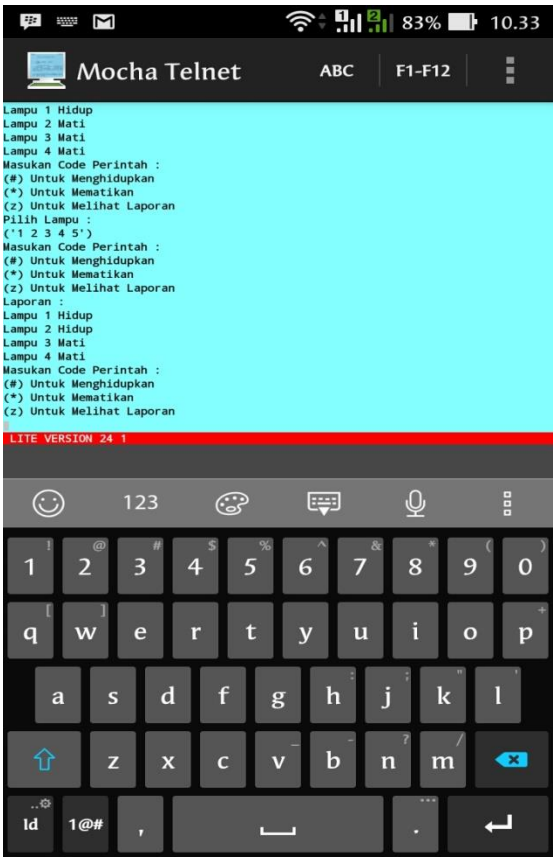
Gambar 26. Tampilan Aplikasi Mocca Telnet di Android

Setelah aplikasi mocca telnet dibuka, dilakukan configurasi awal untuk setting ip modul ethernet sistem kendali lampu via *wireless* 2,4 GHz berbasis mikrokontroler ATmega 16. Pilih menu configure, pilih new, Isi ip dengan 192.168.9.99 dan Port 23 (port default untuk telnet). Tampilan menu configure mocca telnet dapat dilihat pada Gambar 27.



Gambar 27. Tampilan Menu Configure

Setelah dilakukan configure, step selanjutnya adalah dengan memilih menu connect. Tampilan menu connect dapat dilihat pada Gambar 28.



Gambar 28. Tampilan Aplikasi Mocca Telnet Kondisi Connected dengan Alat Sistem Kendali Lampu Via *Wireless* 2,4 GHz Berbasis Mikrokontroler ATmega 16

Setelah terkoneksi dilakukan pengujian untuk mematikan dan menghidupkan lampu. Hasil pengujian menggunakan mocca telnet dapat dilihat pada Tabel 12.

Tabel 12. Hasil Pengujian Menggunakan Mocca Telnet

NO	Perintah	Kondisi Lampu
1	#1	Lampu 1 hidup
2	*1	Lampu 1 mati
3	#2	Lampu 2 hidup
4	*2	Lampu 2 mati
5	#3	Lampu 3 hidup
6	*3	Lampu 3 mati
7	#4	Lampu 4 hidup
8	*4	Lampu 4 mati
9	#5	Lampu semua hidup
10	*5	Lampu semua mati

BAB V

PENUTUP

A. Kesimpulan

Berdasarkan perancangan, pembuatan, pengujian alat, dan pembahasan. Dapat diambil beberapa kesimpulan sebagai berikut :

1. Rangkaian Sistem Kendali Lampu Via *Wireless* 2.4 GHz berbasis mikrokontroler ATmega 16 terdiri dari bagian *input* data ascii yang dikirim melalui telnet melalui *smartphone* atau komputer. Untuk pengolahnnya digunakan mikrokontroler ATmega 16 dan *Wiznet serial to ethernet* sedangkan untuk outputnya berupa empat buah relay untuk menggerakkan lampu 220 V AC.
2. Unjuk kerja dari Sistem Kendali Lampu Via *Wireless* 2.4 GHz berbasis mikrokontroler ATmega 16 yaitu dapat mengendalikan empat buah lampu 220 VAC dengan menggunakan media *wireless* 2,4 GHz.

B. Saran

Penelitian ini dapat dikembangkan lagi untuk mencapai hasil yang lebih baik. Di antaranya yaitu, dibagian input dengan berbasis aplikasi dengan tampilan yang menggunakan tombol seperti *remot control* untuk mempermudah pengguna bukan dengan tampilan text.

DAFTAR PUSTAKA

W Purbo, Onno, Internet *Wireless* dan Hotspot, PT Elex Media Komputindo, Jakarta, 2005.

Anna Nur Nazilah Chamim. (2010:4). Penggunaan Microcontroller Sebagai Pendeteksi Posisi dengan Menggunakan Sinyal GSM. Politeknik PPKP, Yogyakarta.

Grendy Christopher, dkk. (2002). Robot *RGB Color Sorter* Berbasis Mikrokontroler ATmega 16. STMIK GI MDP.

Dian Wirdasari. (2010:8). Membuat Program dengan Menggunakan Bahasa C. SAINTIKOM

Fandhy Bangun Pambajeng. (2015:14). Rancang Bangun Alarm Kendali Parkir Mobil Berbasis Mikrokontroler ATmega 16 dan Sensor Ultrasonik SRF 04. AMIKOM Yogyakarta.

ATMEL. (2013). *ATmega16 Datasheet*. Atmel corporation

Dwi Pipit Haryanto, Anto Cuswanto. (2010). Otomatisasi Pengisian Penampung Air Berbasis Mikrokontroller AT8535. AMIKOM Yogyakarta.

Much Aziz Muslim. (2007:12). Analisa Teknis Perbandingan Router Linux dengan Router Mikrotik pada Jaringan Wireless. Universitas Sitikubang Semarang.

Stallings William, Dasar – Dasar Komunikasi Data, Prentice Hall.Inc, New Jersey, 1996.

M. Ichwan, Fifin Hakiky. (2011:2). Pengukuran Kinerja Goodreads Application Programming Interface (API) Pada Aplikasi Mobile Android. Institut Teknologi Nasional Bandung.

Chuzaimah, Mabruroh, Fereshti Nurdiana Dihan. (2010). Smartphone : Antara Kebutuhan dan E-Lifestyle. Seminar Nasional Informatika (semnasIF 2010) UPN ”Veteran” Yogyakarta.

http://www.mandalamaya.com/pengertian-telnet-dan-kegunaan-telnet/ Diakses pada tanggal 27 Agustus 2016

William, Stalling. 1997. Komunikasi & jaringan Nirkabel Jilid 1 Edisi Kedua. Jakarta: Erlangga

Galih Rakasiwi. (2014). *Prototype Pengontrolan Lampu Dengan Arduindo Berbasis Arduino Via Wifi*. Surakarta : Tugas Akhir, Universitas Muhammadiyah Surakarta.

LAMPIRAN

1. Data Sheet ATmega 16

Features

- High-performance, Low-power AVR® 8-bit Microcontroller
- Advanced RISC Architecture
  - 131 Powerful Instructions – Most Single-clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 16 MIPS Throughput at 16 MHz
  - On-chip 2-cycle Multiplier
- Nonvolatile Program and Data Memories
  - 16K Bytes of In-System Self-Programmable Flash
    Endurance: 10,000 Write/Erase Cycles
  - Optional Boot Code Section with Independent Lock Bits
    In-System Programming by On-chip Boot Program
    True Read-While-Write Operation
  - 512 Bytes EEPROM
    Endurance: 100,000 Write/Erase Cycles
  - 1K Byte Internal SRAM
  - Programming Lock for Software Security
- JTAG (IEEE std. 1149.1 Compliant) Interface
  - Boundary-scan Capabilities According to the JTAG Standard
  - Extensive On-chip Debug Support
  - Programming of Flash, EEPROM, Fuses, and Lock Bits through the JTAG Interface
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescalers and Compare Modes
  - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Four PWM Channels
  - 8-channel, 10-bit ADC
    8 Single-ended Channels
    7 Differential Channels in TQFP Package Only
    2 Differential Channels with Programmable Gain at 1x, 10x, or 200x
  - Byte-oriented Two-wire Serial Interface
  - Programmable Serial USART
  - Master/Slave SPI Serial Interface
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analog Comparator
- Special Microcontroller Features
  - Power-on Reset and Programmable Brown-out Detection
  - Internal Calibrated RC Oscillator
  - External and Internal Interrupt Sources
  - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby and Extended Standby
- I/O and Packages
  - 32 Programmable I/O Lines
  - 40-pin PDIP, 44-lead TQFP, and 44-pad MLF
- Operating Voltages
  - 2.7 - 5.5V for ATmega16L
  - 4.5 - 5.5V for ATmega16
- Speed Grades
  - 0 - 8 MHz for ATmega16L
  - 0 - 16 MHz for ATmega16

ATMEL

8-bit AVR® Microcontroller with 16K Bytes In-System Programmable Flash

ATmega16
ATmega16L

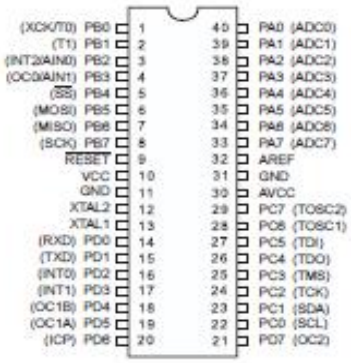
Preliminary

Rev. 2466E–AVR–10/02

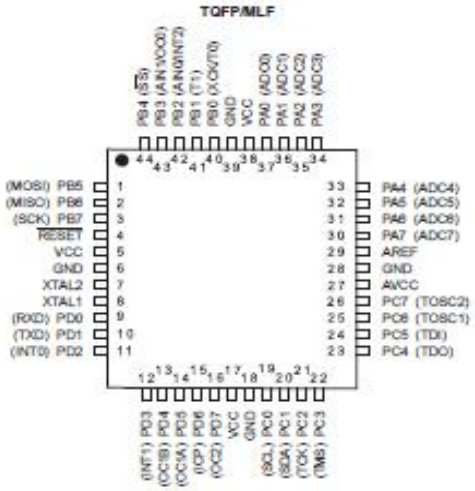
Pin Configurations

Figure 1. Pinouts ATmega16

PDIP



TQFP/MLF



Disclaimer

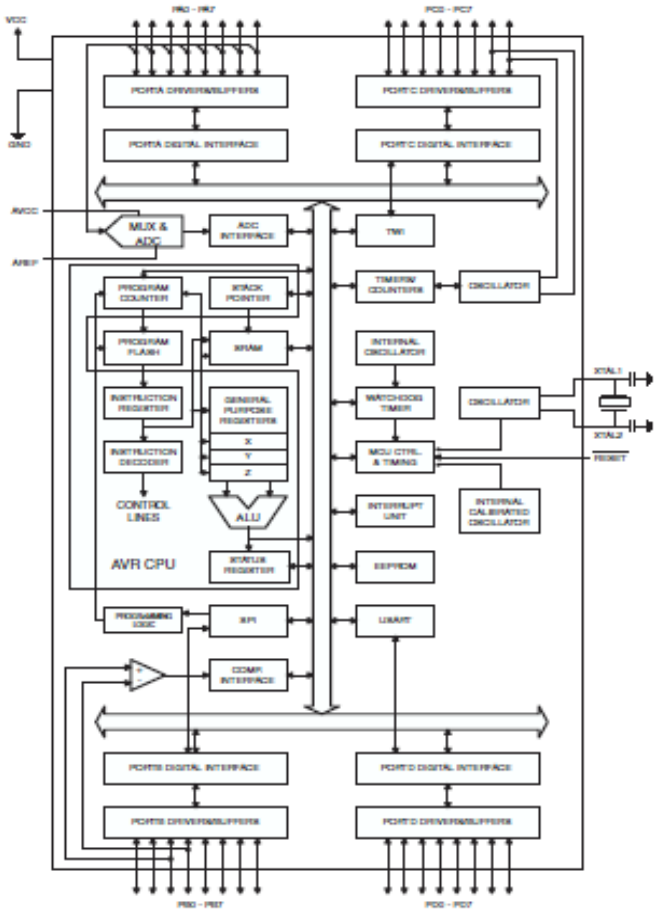
Typical values contained in this data sheet are based on simulations and characterization of other AVR microcontrollers manufactured on the same process technology. Min and Max values will be available after the device is characterized.

Overview

The ATmega16 is a low-power CMOS 8-bit microcontroller based on the AVR enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the ATmega16 achieves throughputs approaching 1 MIPS per MHz allowing the system designer to optimize power consumption versus processing speed.

Block Diagram

Figure 2. Block Diagram



The AVR core combines a rich instruction set with 32 general purpose working registers. All the 32 registers are directly connected to the Arithmetic Logic Unit (ALU), allowing two independent registers to be accessed in one single instruction executed in one clock cycle. The resulting architecture is more code efficient while achieving throughputs up to ten times faster than conventional CISC microcontrollers.

The ATmega16 provides the following features: 16K bytes of In-System Programmable Flash Program memory with Read-While-Write capabilities, 512 bytes EEPROM, 1K byte SRAM, 32 general purpose I/O lines, 32 general purpose working registers, a JTAG interface for Boundary-scan, On-chip Debugging support and programming, three flexible Timer/Counters with compare modes, Internal and External Interrupts, a serial programmable USART, a byte oriented Two-wire Serial Interface, an 8-channel, 10-bit ADC with optional differential input stage with programmable gain (TQFP package only), a programmable Watchdog Timer with Internal Oscillator, an SPI serial port, and six software selectable power saving modes. The Idle mode stops the CPU while allowing the USART, Two-wire interface, A/D Converter, SRAM, Timer/Counters, SPI port, and interrupt system to continue functioning. The Power-down mode saves the register contents but freezes the Oscillator, disabling all other chip functions until the next External Interrupt or Hardware Reset. In Power-save mode, the Asynchronous Timer continues to run, allowing the user to maintain a timer base while the rest of the device is sleeping. The ADC Noise Reduction mode stops the CPU and all I/O modules except Asynchronous Timer and ADC, to minimize switching noise during ADC conversions. In Standby mode, the crystal/resonator Oscillator is running while the rest of the device is sleeping. This allows very fast start-up combined with low-power consumption. In Extended Standby mode, both the main Oscillator and the Asynchronous Timer continue to run.

The device is manufactured using Atmel's high density nonvolatile memory technology. The On-chip ISP Flash allows the program memory to be reprogrammed In-system through an SPI serial interface, by a conventional nonvolatile memory programmer, or by an On-chip Boot program running on the AVR core. The boot program can use any interface to download the application program in the Application Flash memory. Software in the Boot Flash section will continue to run while the Application Flash section is updated, providing true Read-While-Write operation. By combining an 8-bit RISC CPU with In-System Self-Programmable Flash on a monolithic chip, the Atmel ATmega16 is a powerful microcontroller that provides a highly-flexible and cost-effective solution to many embedded control applications.

The ATmega16 AVR is supported with a full suite of program and system development tools including: C compilers, macro assemblers, program debugger/simulators, In-circuit emulators, and evaluation kits.

Pin Descriptions

VCC

Digital supply voltage.

GND

Ground.

Port A (PA7..PA0)

Port A serves as the analog inputs to the A/D Converter.

Port A also serves as an 8-bit bi-directional I/O port, if the A/D Converter is not used. Port pins can provide internal pull-up resistors (selected for each bit). The Port A output buffers have symmetrical drive characteristics with both high sink and source capability. When pins PA0 to PA7 are used as inputs and are externally pulled low, they will source current if the internal pull-up resistors are activated. The Port A pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port B (PB7..PB0)

Port B is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port B output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port B pins that are externally pulled low will source current if the pull-up resistors are activated. The Port B pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port B also serves the functions of various special features of the ATmega16 as listed on page 55.

Port C (PC7..PC0)

Port C is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port C output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port C pins that are externally pulled low will source current if the pull-up resistors are activated. The Port C pins are tri-stated when a reset condition becomes active, even if the clock is not running. If the JTAG interface is enabled, the pull-up resistors on pins PC5(TDI), PC3(TMS) and PC2(TCK) will be activated even if a reset occurs.

Port C also serves the functions of the JTAG interface and other special features of the ATmega16 as listed on page 58.

Port D (PD7..PD0)

Port D is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port D output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port D pins that are externally pulled low will source current if the pull-up resistors are activated. The Port D pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port D also serves the functions of various special features of the ATmega16 as listed on page 60.

$\overline{\text{RESET}}$

Reset input. A low level on this pin for longer than the minimum pulse length will generate a reset, even if the clock is not running. The minimum pulse length is given in Table 15 on page 35. Shorter pulses are not guaranteed to generate a reset.

XTAL1

Input to the inverting Oscillator amplifier and input to the internal clock operating circuit.

XTAL2

Output from the inverting Oscillator amplifier.

AVCC

AVCC is the supply voltage pin for Port A and the A/D Converter. It should be externally connected to V_{CC}, even if the ADC is not used. If the ADC is used, it should be connected to V_{CC} through a low-pass filter.

AREF

AREF is the analog reference pin for the A/D Converter.

About Code Examples

This documentation contains simple code examples that briefly show how to use various parts of the device. These code examples assume that the part specific header file is included before compilation. Be aware that not all C Compiler vendors include bit definitions in the header files and interrupt handling in C is compiler dependent. Please confirm with the C Compiler documentation for more details.

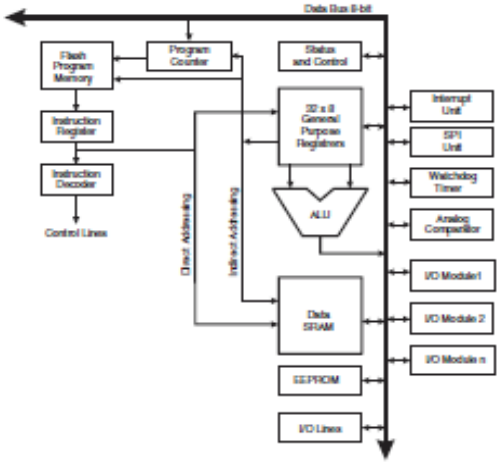
AVR CPU Core

Introduction

This section discusses the AVR core architecture in general. The main function of the CPU core is to ensure correct program execution. The CPU must therefore be able to access memories, perform calculations, control peripherals, and handle interrupts.

Architectural Overview

Figure 3. Block Diagram of the AVR MCU Architecture



In order to maximize performance and parallelism, the AVR uses a Harvard architecture – with separate memories and buses for program and data. Instructions in the program memory are executed with a single level pipelining. While one instruction is being executed, the next instruction is pre-fetched from the program memory. This concept enables instructions to be executed in every clock cycle. The program memory is In-System Reprogrammable Flash memory.

The fast-access Register file contains 32 x 8-bit general purpose working registers with a single clock cycle access time. This allows single-cycle Arithmetic Logic Unit (ALU) operation. In a typical ALU operation, two operands are output from the Register file, the operation is executed, and the result is stored back in the Register file – in one clock cycle.

Six of the 32 registers can be used as three 16-bit indirect address register pointers for Data Space addressing – enabling efficient address calculations. One of the these address pointers can also be used as an address pointer for look up tables in Flash Program memory. These added function registers are the 16-bit X-, Y-, and Z-register, described later in this section.

The ALU supports arithmetic and logic operations between registers or between a constant and a register. Single register operations can also be executed in the ALU. After

an arithmetic operation, the Status Register is updated to reflect information about the result of the operation.

Program flow is provided by conditional and unconditional jump and call instructions, able to directly address the whole address space. Most AVR instructions have a single 16-bit word format. Every program memory address contains a 16- or 32-bit instruction.

Program Flash memory space is divided in two sections, the Boot program section and the Application Program section. Both sections have dedicated Lock bits for write and read/write protection. The SPM instruction that writes into the Application Flash memory section must reside in the Boot Program section.

During interrupts and subroutine calls, the return address program counter (PC) is stored on the Stack. The Stack is effectively allocated in the general data SRAM, and consequently the stack size is only limited by the total SRAM size and the usage of the SRAM. All user programs must initialize the SP in the reset routine (before subroutines or interrupts are executed). The Stack Pointer SP is read/write accessible in the I/O space. The data SRAM can easily be accessed through the five different addressing modes supported in the AVR architecture.

The memory spaces in the AVR architecture are all linear and regular memory maps.

A flexible interrupt module has its control registers in the I/O space with an additional global interrupt enable bit in the Status Register. All interrupts have a separate interrupt vector in the interrupt vector table. The interrupts have priority in accordance with their interrupt vector position. The lower the interrupt vector address, the higher the priority.

The I/O memory space contains 64 addresses for CPU peripheral functions as Control Registers, SPI, and other I/O functions. The I/O Memory can be accessed directly, or as the Data Space locations following those of the Register file, $20 - $5F.

ALU – Arithmetic Logic Unit

The high-performance AVR ALU operates in direct connection with all the 32 general purpose working registers. Within a single clock cycle, arithmetic operations between general purpose registers or between a register and an immediate are executed. The ALU operations are divided into three main categories – arithmetic, logical, and bit-functions. Some implementations of the architecture also provide a powerful multiplier supporting both signed/unsigned multiplication and fractional format. See the "Instruction Set" section for a detailed description.

Status Register

The Status Register contains information about the result of the most recently executed arithmetic instruction. This information can be used for altering program flow in order to perform conditional operations. Note that the Status Register is updated after all ALU operations, as specified in the Instruction Set Reference. This will in many cases remove the need for using the dedicated compare instructions, resulting in faster and more compact code.

The Status Register is not automatically stored when entering an interrupt routine and restored when returning from an interrupt. This must be handled by software.

The AVR Status Register – SREG – is defined as:

Bit	7	6	5	4	3	2	1	0	
	I	T	H	S	V	N	Z	C	SREG
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – I: Global Interrupt Enable**

The Global Interrupt Enable bit must be set for the interrupts to be enabled. The individual interrupt enable control is then performed in separate control registers. If the Global Interrupt Enable Register is cleared, none of the interrupts are enabled independent of the individual interrupt enable settings. The I-bit is cleared by hardware after an interrupt has occurred, and is set by the RETI instruction to enable subsequent interrupts. The I-bit can also be set and cleared by the application with the SEI and CLI instructions, as described in the instruction set reference.

- **Bit 6 – T: Bit Copy Storage**

The Bit Copy Instructions BLD (Bit LoaD) and BST (Bit STore) use the T-bit as source or destination for the operated bit. A bit from a register in the Register file can be copied into T by the BST instruction, and a bit in T can be copied into a bit in a register in the Register file by the BLD instruction.

- **Bit 5 – H: Half Carry Flag**

The Half Carry Flag H indicates a half carry in some arithmetic operations. Half Carry is useful in BCD arithmetic. See the "Instruction Set Description" for detailed information.

- **Bit 4 – S: Sign Bit, S = N ⊕ V**

The S-bit is always an exclusive or between the negative flag N and the two's complement overflow flag V. See the "Instruction Set Description" for detailed information.

- **Bit 3 – V: Two's Complement Overflow Flag**

The Two's Complement Overflow Flag V supports two's complement arithmetics. See the "Instruction Set Description" for detailed information.

- **Bit 2 – N: Negative Flag**

The Negative Flag N indicates a negative result in an arithmetic or logic operation. See the "Instruction Set Description" for detailed information.

- **Bit 1 – Z: Zero Flag**

The Zero Flag Z indicates a zero result in an arithmetic or logic operation. See the "Instruction Set Description" for detailed information.

- **Bit 0 – C: Carry Flag**

The Carry Flag C indicates a carry in an arithmetic or logic operation. See the "Instruction Set Description" for detailed information.

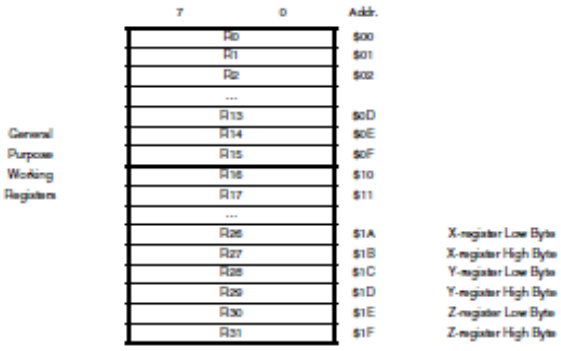
General Purpose Register File

The Register File is optimized for the AVR Enhanced RISC instruction set. In order to achieve the required performance and flexibility, the following input/output schemes are supported by the Register file:

- One 8-bit output operand and one 8-bit result input
- Two 8-bit output operands and one 8-bit result input
- Two 8-bit output operands and one 16-bit result input
- One 16-bit output operand and one 16-bit result input

Figure 4 shows the structure of the 32 general purpose working registers in the CPU.

Figure 4. AVR CPU General Purpose Working Registers



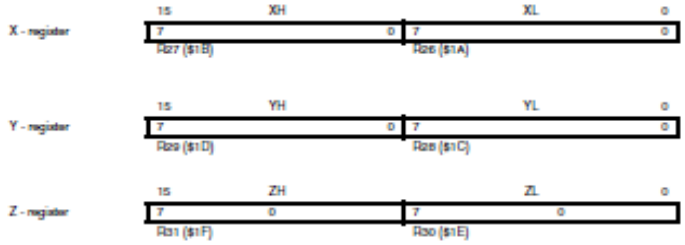
Most of the instructions operating on the Register File have direct access to all registers, and most of them are single cycle instructions.

As shown in Figure 4, each register is also assigned a data memory address, mapping them directly into the first 32 locations of the user Data Space. Although not being physically implemented as SRAM locations, this memory organization provides great flexibility in access of the registers, as the X-, Y-, and Z-pointer Registers can be set to index any register in the file.

The X-register, Y-register and Z-register

The registers R26..R31 have some added functions to their general purpose usage. These registers are 16-bit address pointers for indirect addressing of the Data Space. The three indirect address registers X, Y, and Z are defined as described in Figure 5.

Figure 5. The X-, Y-, and Z-registers



In the different addressing modes these address registers have functions as fixed displacement, automatic increment, and automatic decrement (see the Instruction Set Reference for details).

Stack Pointer

The Stack is mainly used for storing temporary data, for storing local variables and for storing return addresses after interrupts and subroutine calls. The Stack Pointer Register always points to the top of the stack. Note that the stack is implemented as growing from higher memory locations to lower memory locations. This implies that a stack PUSH command decreases the Stack Pointer.

The Stack Pointer points to the data SRAM stack area where the Subroutine and Interrupt Stacks are located. This Stack space in the data SRAM must be defined by the program before any subroutine calls are executed or interrupts are enabled. The Stack Pointer must be set to point above $60. The Stack Pointer is decremented by one when data is pushed onto the Stack with the PUSH instruction, and it is decremented by two when the return address is pushed onto the Stack with subroutine call or interrupt. The Stack Pointer is incremented by one when data is popped from the Stack with the POP instruction, and it is incremented by two when data is popped from the Stack with return from subroutine RET or return from interrupt RETI.

The AVR Stack Pointer is implemented as two 8-bit registers in the I/O space. The number of bits actually used is implementation dependent. Note that the data space in some implementations of the AVR architecture is so small that only SPL is needed. In this case, the SPH Register will not be present.

Bit	15	14	13	12	11	10	9	8	
	SP15	SP14	SP13	SP12	SP11	SP10	SP9	SP8	SPH
	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	SPL
	7	6	5	4	3	2	1	0	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

Instruction Execution Timing

This section describes the general access timing concepts for instruction execution. The AVR CPU is driven by the CPU clock clk_{CPU}, directly generated from the selected clock source for the chip. No internal clock division is used.

Figure 6 shows the parallel instruction fetches and instruction executions enabled by the Harvard architecture and the fast-access Register file concept. This is the basic pipelining concept to obtain up to 1 MIPS per MHz with the corresponding unique results for functions per cost, functions per clocks, and functions per power-unit.

Figure 6. The Parallel Instruction Fetches and Instruction Executions

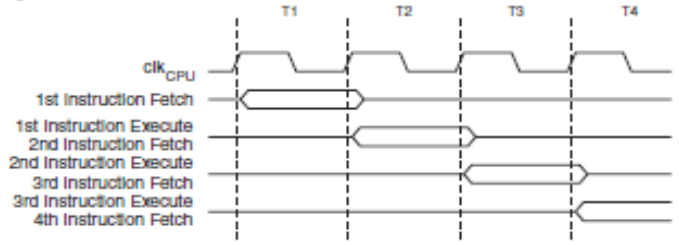
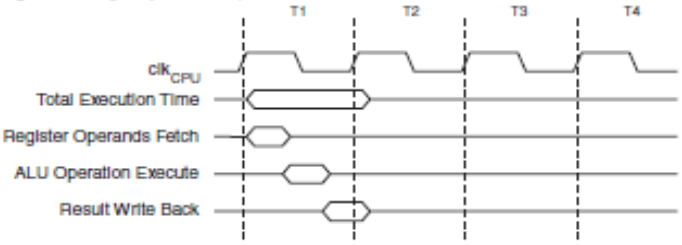


Figure 7 shows the internal timing concept for the Register file. In a single clock cycle an ALU operation using two register operands is executed, and the result is stored back to the destination register.

Figure 7. Single Cycle ALU Operation



Reset and Interrupt Handling

The AVR provides several different interrupt sources. These interrupts and the separate reset vector each have a separate program vector in the program memory space. All interrupts are assigned individual enable bits which must be written logic one together with the Global Interrupt Enable bit in the Status Register in order to enable the interrupt. Depending on the program counter value, interrupts may be automatically disabled when Boot Lock bits BLB02 or BLB12 are programmed. This feature improves software security. See the section "Memory Programming" on page 254 for details.

The lowest addresses in the program memory space are by default defined as the Reset and Interrupt Vectors. The complete list of vectors is shown in "Interrupts" on page 42. The list also determines the priority levels of the different interrupts. The lower the address the higher is the priority level. RESET has the highest priority, and next is INT0

– the External Interrupt Request 0. The Interrupt Vectors can be moved to the start of the Boot Flash section by setting the IVSEL bit in the General Interrupt Control Register (GICR). Refer to "Interrupts" on page 42 for more information. The Reset Vector can also be moved to the start of the boot Flash section by programming the BOOTRST fuse, see "Boot Loader Support – Read-While-Write Self-Programming" on page 241.

When an interrupt occurs, the Global Interrupt Enable I-bit is cleared and all interrupts are disabled. The user software can write logic one to the I-bit to enable nested interrupts. All enabled interrupts can then interrupt the current interrupt routine. The I-bit is automatically set when a Return from Interrupt instruction – RETI – is executed.

There are basically two types of interrupts. The first type is triggered by an event that sets the interrupt flag. For these interrupts, the Program Counter is vectored to the actual Interrupt Vector in order to execute the interrupt handling routine, and hardware clears the corresponding interrupt flag. Interrupt flags can also be cleared by writing a logic one to the flag bit position(s) to be cleared. If an interrupt condition occurs while the corresponding interrupt enable bit is cleared, the interrupt flag will be set and remembered until the interrupt is enabled, or the flag is cleared by software. Similarly, if one or more interrupt conditions occur while the Global Interrupt Enable bit is cleared, the corresponding interrupt flag(s) will be set and remembered until the global interrupt enable bit is set, and will then be executed by order of priority.

The second type of interrupts will trigger as long as the interrupt condition is present. These interrupts do not necessarily have interrupt flags. If the interrupt condition disappears before the interrupt is enabled, the interrupt will not be triggered.

When the AVR exits from an interrupt, it will always return to the main program and execute one more instruction before any pending interrupt is served.

Note that the Status Register is not automatically stored when entering an interrupt routine, nor restored when returning from an interrupt routine. This must be handled by software.

When using the CLI instruction to disable interrupts, the interrupts will be immediately disabled. No interrupt will be executed after the CLI instruction, even if it occurs simultaneously with the CLI instruction. The following example shows how this can be used to avoid interrupts during the timed EEPROM write sequence.

Assembly Code Example

```
in  r16, SREG       ; store SREG value
cli     ; disable interrupts during timed sequence
sbi EECR, EEMWE     ; start EEPROM write
sbi EECR, EEWE
out SREG, r16       ; restore SREG value (I-bit)
```

C Code Example

```
char cSREG;
cSREG = SREG;  /* store SREG value */
/* disable interrupts during timed sequence */
_CLI();
EECR |= (1<<EEMWE); /* start EEPROM write */
EECR |= (1<<EEWE);
SREG = cSREG; /* restore SREG value (I-bit) */
```

When using the SEI instruction to enable interrupts, the instruction following SEI will be executed before any pending interrupts, as shown in this example.

Assembly Code Example

```
sei   ; set global interrupt enable
sleep ; enter sleep, waiting for interrupt
; note: will enter sleep before any pending
; interrupt(s)
```

C Code Example

```
_SEI(); /* set global interrupt enable */
_SLEEP(); /* enter sleep, waiting for interrupt */
/* note: will enter sleep before any pending interrupt(s) */
```

Interrupt Response Time

The interrupt execution response for all the enabled AVR interrupts is four clock cycles minimum. After four clock cycles the program vector address for the actual interrupt handling routine is executed. During this four clock cycle period, the Program Counter is pushed onto the Stack. The vector is normally a jump to the interrupt routine, and this jump takes three clock cycles. If an interrupt occurs during execution of a multi-cycle instruction, this instruction is completed before the interrupt is served. If an interrupt occurs when the MCU is in sleep mode, the interrupt execution response time is increased by four clock cycles. This increase comes in addition to the start-up time from the selected sleep mode.

A return from an interrupt handling routine takes four clock cycles. During these four clock cycles, the Program Counter (two bytes) is popped back from the Stack, the Stack Pointer is incremented by two, and the I-bit in SREG is set.

AVR ATmega16 Memories

This section describes the different memories in the ATmega16. The AVR architecture has two main memory spaces, the Data Memory and the Program Memory space. In addition, the ATmega16 features an EEPROM Memory for data storage. All three memory spaces are linear and regular.

In-System Reprogrammable Flash Program Memory

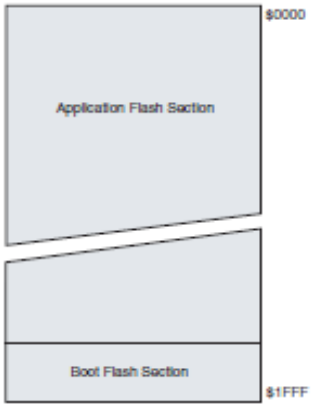
The ATmega16 contains 16K bytes On-chip In-System Reprogrammable Flash memory for program storage. Since all AVR instructions are 16 or 32 bits wide, the Flash is organized as 8K x 16. For software security, the Flash Program memory space is divided into two sections, Boot Program section and Application Program section.

The Flash memory has an endurance of at least 10,000 write/erase cycles. The ATmega16 Program Counter (PC) is 13 bits wide, thus addressing the 8K program memory locations. The operation of Boot Program section and associated Boot Lock bits for software protection are described in detail in "Boot Loader Support – Read-While-Write Self-Programming" on page 241. "Memory Programming" on page 254 contains a detailed description on Flash data serial downloading using the SPI pins or the JTAG interface.

Constant tables can be allocated within the entire program memory address space (see the LPM – Load Program Memory instruction Description).

Timing diagrams for instruction fetch and execution are presented in "Instruction Execution Timing" on page 11.

Figure 8. Program Memory Map



SRAM Data Memory

Figure 9 shows how the ATmega16 SRAM Memory is organized.

The lower 1120 Data Memory locations address the Register file, the I/O Memory, and the internal data SRAM. The first 96 locations address the Register file and I/O Memory, and the next 1024 locations address the internal data SRAM.

The five different addressing modes for the data memory cover: Direct, Indirect with Displacement, Indirect, Indirect with Pre-decrement, and Indirect with Post-increment. In the Register file, registers R26 to R31 feature the indirect addressing pointer registers.

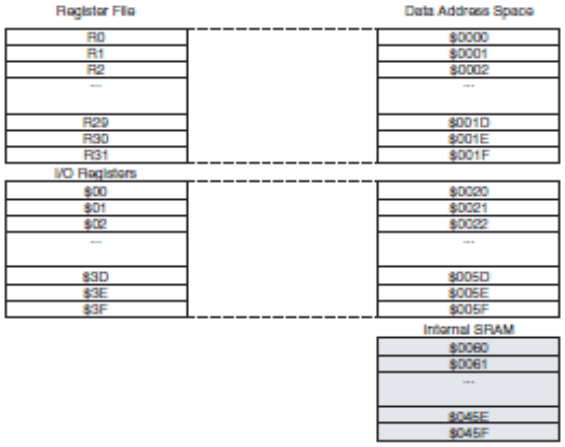
The direct addressing reaches the entire data space.

The Indirect with Displacement mode reaches 63 address locations from the base address given by the Y- or Z-register.

When using register indirect addressing modes with automatic pre-decrement and post-increment, the address registers X, Y, and Z are decremented or incremented.

The 32 general purpose working registers, 64 I/O Registers, and the 1024 bytes of internal data SRAM in the ATmega16 are all accessible through all these addressing modes. The Register file is described in "General Purpose Register File" on page 8.

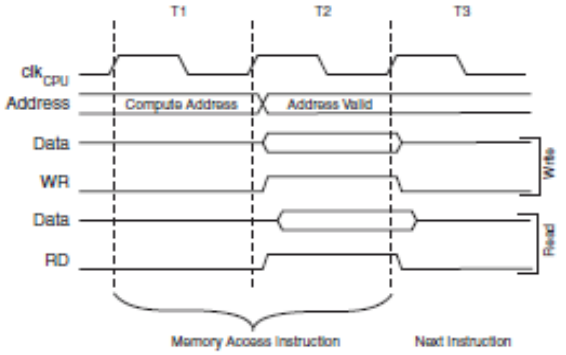
Figure 9. Data Memory Map



Data Memory Access Times

This section describes the general access timing concepts for internal memory access. The internal data SRAM access is performed in two clk_{CPU} cycles as described in Figure 10.

Figure 10. On-chip Data SRAM Access Cycles



EEPROM Data Memory

The ATmega16 contains 512 bytes of data EEPROM memory. It is organized as a separate data space, in which single bytes can be read and written. The EEPROM has an endurance of at least 100,000 write/erase cycles. The access between the EEPROM and the CPU is described in the following, specifying the EEPROM Address Registers, the EEPROM Data Register, and the EEPROM Control Register.

For a detailed description of SPI and JTAG data downloading to the EEPROM, see page 268 and page 272, respectively.

EEPROM Read/Write Access

The EEPROM Access Registers are accessible in the I/O space.

The write access time for the EEPROM is given in Table 1. A self-timing function, however, lets the user software detect when the next byte can be written. If the user code contains instructions that write the EEPROM, some precautions must be taken. In heavily filtered power supplies, V_{CC} is likely to rise or fall slowly on Power-up/down. This causes the device for some period of time to run at a voltage lower than specified as minimum for the clock frequency used. See "Preventing EEPROM Corruption" on page 20 for details on how to avoid problems in these situations.

In order to prevent unintentional EEPROM writes, a specific write procedure must be followed. Refer to the description of the EEPROM Control Register for details on this.

When the EEPROM is read, the CPU is halted for four clock cycles before the next instruction is executed. When the EEPROM is written, the CPU is halted for two clock cycles before the next instruction is executed.

The EEPROM Address Register – EEARH and EEARL

Bit	15	14	13	12	11	10	9	8	
	–	–	–	–	–	–	–	EEAR8	EEARH
	EEAR7	EEAR6	EEAR5	EEAR4	EEAR3	EEAR2	EEAR1	EEAR0	EEARL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	X	
	X	X	X	X	X	X	X	X	

- **Bits 15..9 – Res: Reserved Bits**

These bits are reserved bits in the ATmega16 and will always read as zero.

- **Bits 8..0 – EEAR8..0: EEPROM Address**

The EEPROM Address Registers – EEARH and EEARL – specify the EEPROM address in the 512 bytes EEPROM space. The EEPROM data bytes are addressed linearly between 0 and 511. The initial value of EEAR is undefined. A proper value must be written before the EEPROM may be accessed.

The EEPROM Data Register – EEDR

Bit	7	6	5	4	3	2	1	0	
	MSB							LSB	EEDR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7..0 – EEDR7.0: EEPROM Data**

For the EEPROM write operation, the EEDR Register contains the data to be written to the EEPROM in the address given by the EEAR Register. For the EEPROM read operation, the EEDR contains the data read out from the EEPROM at the address given by EEAR.

The EEPROM Control Register – EECR

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	EERIE	EEMWE	EEWE	EERE	EECR
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	X	0	

- **Bits 7..4 – Res: Reserved Bits**

These bits are reserved bits in the ATmega16 and will always read as zero.

- **Bit 3 – EERIE: EEPROM Ready Interrupt Enable**

Writing EERIE to one enables the EEPROM Ready Interrupt if the I bit in SREG is set. Writing EERIE to zero disables the interrupt. The EEPROM Ready Interrupt generates a constant interrupt when EEWE is cleared.

- **Bit 2 – EEMWE: EEPROM Master Write Enable**

The EEMWE bit determines whether setting EEWE to one causes the EEPROM to be written. When EEMWE is set, setting EEWE within four clock cycles will write data to the EEPROM at the selected address If EEMWE is zero, setting EEWE will have no effect.

When EEMWE has been written to one by software, hardware clears the bit to zero after four clock cycles. See the description of the EEWE bit for an EEPROM write procedure.

- **Bit 1 – EEWE: EEPROM Write Enable**

The EEPROM Write Enable Signal EEWE is the write strobe to the EEPROM. When address and data are correctly set up, the EEWE bit must be written to one to write the value into the EEPROM. The EEMWE bit must be written to one before a logical one is written to EEWE, otherwise no EEPROM write takes place. The following procedure should be followed when writing the EEPROM (the order of steps 3 and 4 is not essential):

1. Wait until EEWE becomes zero.
2. Wait until SPMEN in SPMCR becomes zero.
3. Write new EEPROM address to EEAR (optional).
4. Write new EEPROM data to EEDR (optional).
5. Write a logical one to the EEMWE bit while writing a zero to EEWE in EECR.
6. Within four clock cycles after setting EEMWE, write a logical one to EEWE.

The EEPROM can not be programmed during a CPU write to the Flash memory. The software must check that the Flash programming is completed before initiating a new EEPROM write. Step 2 is only relevant if the software contains a Boot Loader allowing the CPU to program the Flash. If the Flash is never being updated by the CPU, step 2 can be omitted. See "Boot Loader Support – Read-While-Write Self-Programming" on page 241 for details about boot programming.

Caution: An interrupt between step 5 and step 6 will make the write cycle fail, since the EEPROM Master Write Enable will time-out. If an interrupt routine accessing the EEPROM is interrupting another EEPROM Access, the EEAR or EEDR register will be modified, causing the interrupted EEPROM Access to fail. It is recommended to have the global interrupt flag cleared during all the steps to avoid these problems.

When the write access time has elapsed, the EEWE bit is cleared by hardware. The user software can poll this bit and wait for a zero before writing the next byte. When EEWE has been set, the CPU is halted for two cycles before the next instruction is executed.

- **Bit 0 – EERE: EEPROM Read Enable**

The EEPROM Read Enable Signal – EERE – is the read strobe to the EEPROM. When the correct address is set up in the EEAR register, the EERE bit must be written to a logic one to trigger the EEPROM read. The EEPROM read access takes one instruction, and the requested data is available immediately. When the EEPROM is read, the CPU is halted for four cycles before the next instruction is executed.

The user should poll the EEWE bit before starting the read operation. If a write operation is in progress, it is neither possible to read the EEPROM, nor to change the EEAR register.

The calibrated Oscillator is used to time the EEPROM accesses. Table 1 lists the typical programming time for EEPROM access from the CPU.

Table 1. EEPROM Programming Time

Symbol	Number of Calibrated RC Oscillator Cycles[1]	Typ Programming Time
EEPROM write (from CPU)	8448	8.5 ms

Note: 1. Uses 1 MHz clock, independent of CKSEL Fuse setting.

The following code examples show one assembly and one C function for writing to the EEPROM. The examples assume that interrupts are controlled (for example by disabling interrupts globally) so that no interrupts will occur during execution of these functions. The examples also assume that no Flash Boot Loader is present in the software. If such code is present, the EEPROM write function must also wait for any ongoing SPM command to finish.

Assembly Code Example

```
EEPROM_write:
  ; Wait for completion of previous write
  sbic EECR,EEWE
  rjmp EEPROM_write
  ; Set up address (r18:r17) in address register
  out  EEARH, r18
  out  EEARL, r17
  ; Write data (r16) to data register
  out  EEDR,r16
  ; Write logical one to EEMWE
  sbi  EECR,EEMWE
  ; Start eeprom write by setting EEWE
  sbi  EECR,EEWE
  ret
```

C Code Example

```
void EEPROM_write(unsigned int uiAddress, unsigned char ucData)
{
  /* Wait for completion of previous write */
  while(EECR & (1<<EEWE))
    ;
  /* Set up address and data registers */
  EEAR = uiAddress;
  EEDR = ucData;
  /* Write logical one to EEMWE */
  EECR |= (1<<EEMWE);
  /* Start eeprom write by setting EEWE */
  EECR |= (1<<EEWE);
}
```

The next code examples show assembly and C functions for reading the EEPROM. The examples assume that interrupts are controlled so that no interrupts will occur during execution of these functions.

Assembly Code Example

```
EEPROM_read:
  ; Wait for completion of previous write
  sbic EECR,EEWE
  rjmp EEPROM_read
  ; Set up address (r18:r17) in address register
  out  EEARH, r18
  out  EEARL, r17
  ; Start eeprom read by writing EERE
  sbi  EECR,EERE
  ; Read data from data register
  in   r16,EEDR
  ret
```

C Code Example

```
unsigned char EEPROM_read(unsigned int uiAddress)
{
  /* Wait for completion of previous write */
  while(EECR & (1<<EEWE))
    ;
  /* Set up address register */
  EEAR = uiAddress;
  /* Start eeprom read by writing EERE */
  EECR |= (1<<EERE);
  /* Return data from data register */
  return EEDR;
}
```

Preventing EEPROM Corruption

During periods of low V_{CC}, the EEPROM data can be corrupted because the supply voltage is too low for the CPU and the EEPROM to operate properly. These issues are the same as for board level systems using EEPROM, and the same design solutions should be applied.

An EEPROM data corruption can be caused by two situations when the voltage is too low. First, a regular write sequence to the EEPROM requires a minimum voltage to operate correctly. Secondly, the CPU itself can execute instructions incorrectly, if the supply voltage is too low.

EEPROM data corruption can easily be avoided by following this design recommendation:

Keep the AVR RESET active (low) during periods of insufficient power supply voltage. This can be done by enabling the internal Brown-out Detector (BOD). If the detection level of the internal BOD does not match the needed detection level, an external low V_{CC} Reset Protection circuit can be used. If a reset occurs while a write operation is in progress, the write operation will be completed provided that the power supply voltage is sufficient.

I/O Memory

The I/O space definition of the ATmega16 is shown in "Register Summary" on page 298.

All ATmega16 I/Os and peripherals are placed in the I/O space. The I/O locations are accessed by the IN and OUT instructions, transferring data between the 32 general purpose working registers and the I/O space. I/O Registers within the address range $00 - $1F are directly bit-accessible using the SBI and CBI instructions. In these registers, the value of single bits can be checked by using the SBIS and SBIC instructions. Refer to the Instruction Set section for more details. When using the I/O specific commands IN and OUT, the I/O addresses $00 - $3F must be used. When addressing I/O Registers as data space using LD and ST instructions, $20 must be added to these addresses.

For compatibility with future devices, reserved bits should be written to zero if accessed. Reserved I/O memory addresses should never be written.

Some of the status flags are cleared by writing a logical one to them. Note that the CBI and SBI instructions will operate on all bits in the I/O Register, writing a one back into any flag read as set, thus clearing the flag. The CBI and SBI instructions work with registers $00 to $1F only.

The I/O and peripherals control registers are explained in later sections.

Minimizing Power Consumption

There are several issues to consider when trying to minimize the power consumption in an AVR controlled system. In general, sleep modes should be used as much as possible, and the sleep mode should be selected so that as few as possible of the device's functions are operating. All functions not needed should be disabled. In particular, the following modules may need special consideration when trying to achieve the lowest possible power consumption.

Analog to Digital Converter

If enabled, the ADC will be enabled in all sleep modes. To save power, the ADC should be disabled before entering any sleep mode. When the ADC is turned off and on again, the next conversion will be an extended conversion. Refer to "Analog to Digital Converter" on page 198 for details on ADC operation.

Analog Comparator

When entering Idle mode, the Analog Comparator should be disabled if not used. When entering ADC Noise Reduction mode, the Analog Comparator should be disabled. In the other sleep modes, the Analog Comparator is automatically disabled. However, if the Analog Comparator is set up to use the Internal Voltage Reference as input, the Analog Comparator should be disabled in all sleep modes. Otherwise, the Internal Voltage Reference will be enabled, independent of sleep mode. Refer to "Analog Comparator" on page 195 for details on how to configure the Analog Comparator.

Brown-out Detector

If the Brown-out Detector is not needed in the application, this module should be turned off. If the Brown-out Detector is enabled by the BODEN Fuse, it will be enabled in all sleep modes, and hence, always consume power. In the deeper sleep modes, this will contribute significantly to the total current consumption. Refer to "Brown-out Detection" on page 37 for details on how to configure the Brown-out Detector.

Internal Voltage Reference

The Internal Voltage Reference will be enabled when needed by the Brown-out Detector, the Analog Comparator or the ADC. If these modules are disabled as described in the sections above, the internal voltage reference will be disabled and it will not be consuming power. When turned on again, the user must allow the reference to start up before the output is used. If the reference is kept on in sleep mode, the output can be used immediately. Refer to "Internal Voltage Reference" on page 39 for details on the start-up time.

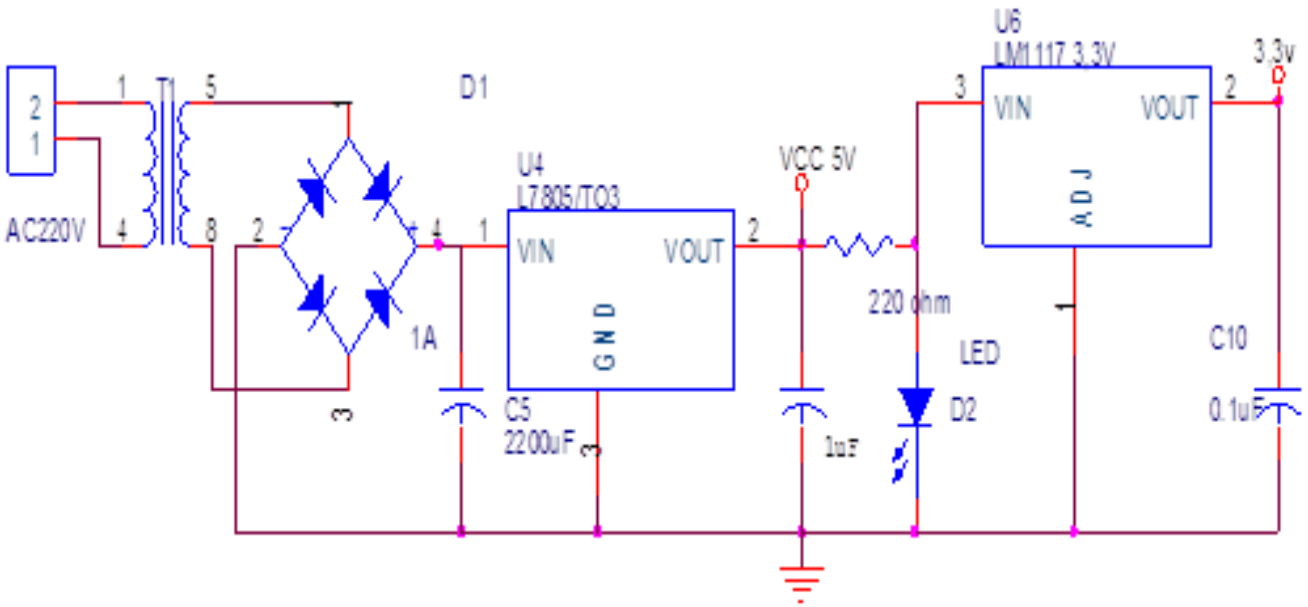
Watchdog Timer

If the Watchdog Timer is not needed in the application, this module should be turned off. If the Watchdog Timer is enabled, it will be enabled in all sleep modes, and hence, always consume power. In the deeper sleep modes, this will contribute significantly to the total current consumption. Refer to "Watchdog Timer" on page 39 for details on how to configure the Watchdog Timer.

Port Pins

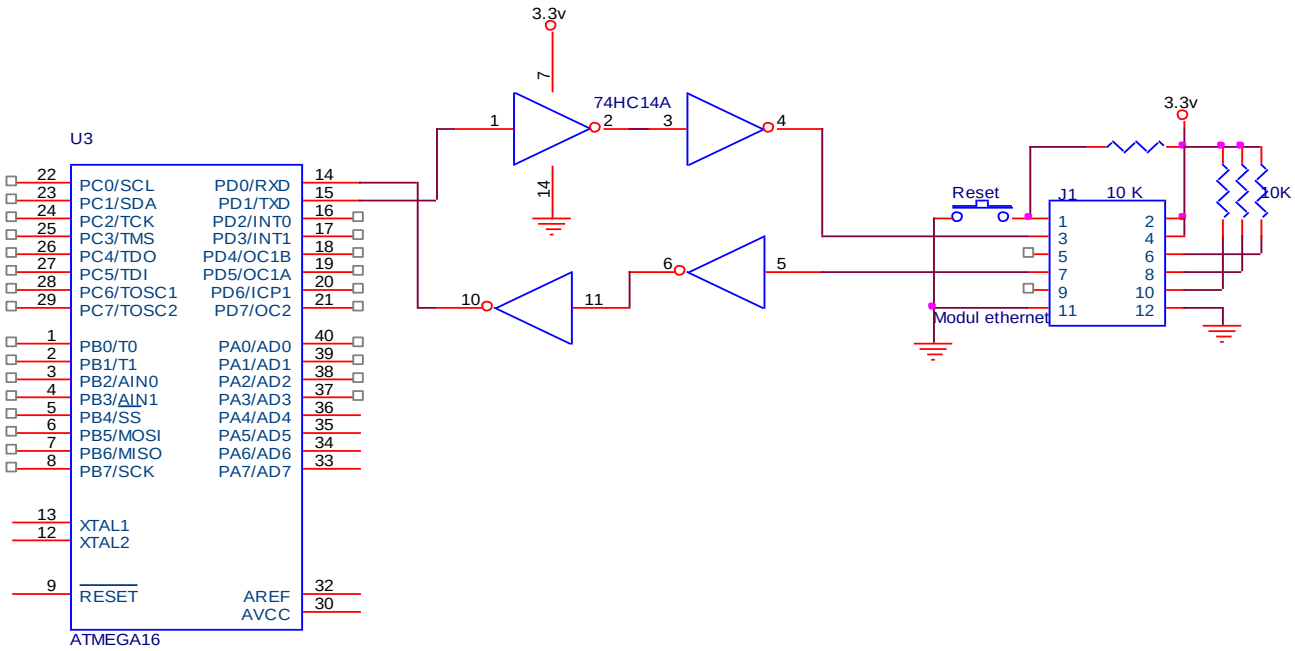
When entering a sleep mode, all port pins should be configured to use minimum power. The most important thing is then to ensure that no pins drive resistive loads. In sleep modes where the both the I/O clock ($clk_{I/O}$) and the ADC clock (clk_{ADC}) are stopped, the input buffers of the device will be disabled. This ensures that no power is consumed by the input logic when not needed. In some cases, the input logic is needed for detecting wake-up conditions, and it will then be enabled. Refer to the section "Digital Input Enable and Sleep Modes" on page 51 for details on which pins are enabled. If the input buffer is enabled and the input signal is left floating or have an analog signal level close to $V_{CC}/2$, the input buffer will use excessive power.

2. Rangkaian *Power Supply*



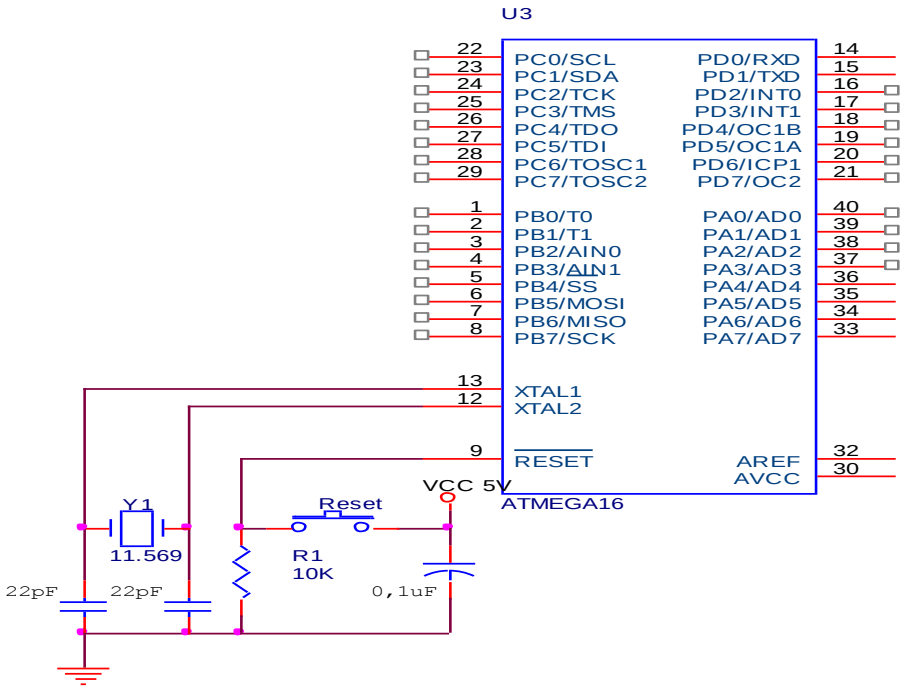
RANGKAIAN POWER SUPPLY			KETERANGAN	
FT UNY	SKALA: -	DIG. FAJAR	A4	No.1
	DIP. NURKHAMI	DIST. NURKHAMI	NIM. 13507134013	

3. Rangkaian Logic Converter dan Ethernet (TCP/IP to serial)



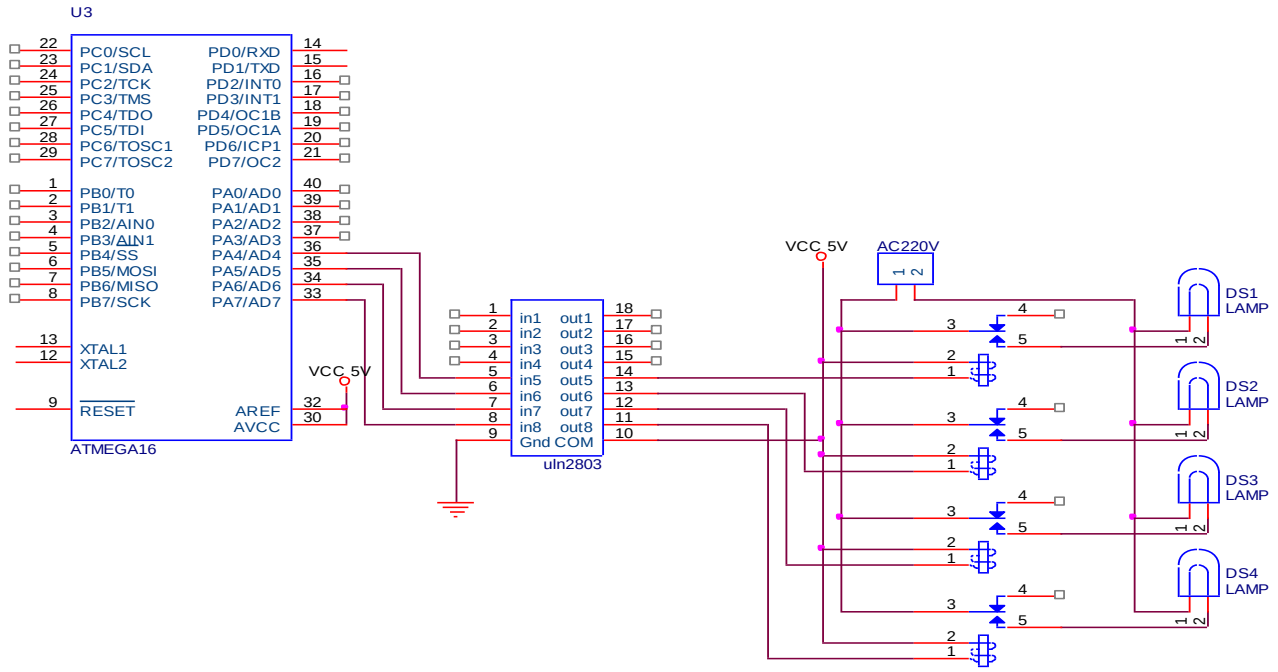
RANGKAIAN LOGIC CONVERTER			KETERANGAN
FT UNY	SKALA: -	DIG. FAJAR	A4 / No.2
	DIP. NURKHAMI	DIST. NURKHAMI	NIM. 13507134013

4. Rangkaian Sistem Minimum ATmega 16



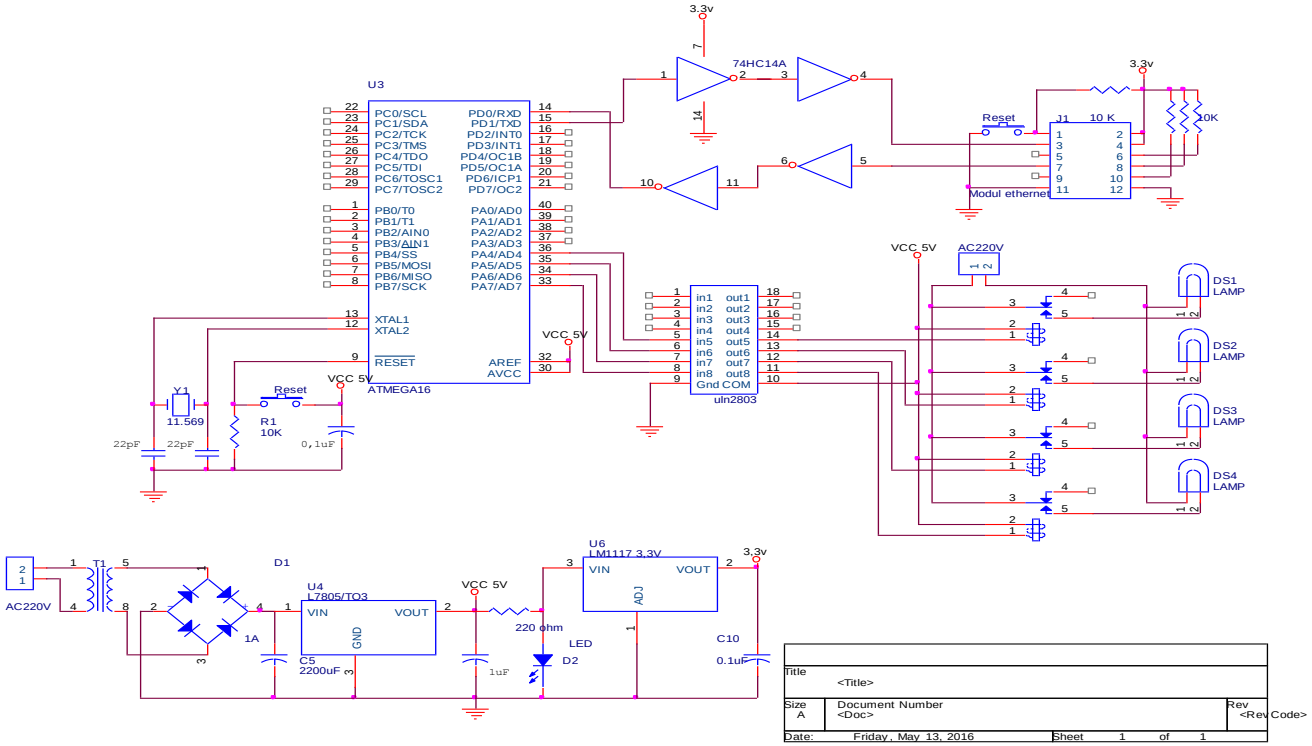
RANGKAIAN SISTEM MINIMUM ATMEGA 16			KETERANGAN
FT UNY	SKALA: -	DIG. FAJAR	A4 / No.3
	DIP. NURKHAMI	DIST. NURKHAMI	NIM. 13507134013

5. Rangkaian *Driver Relay*



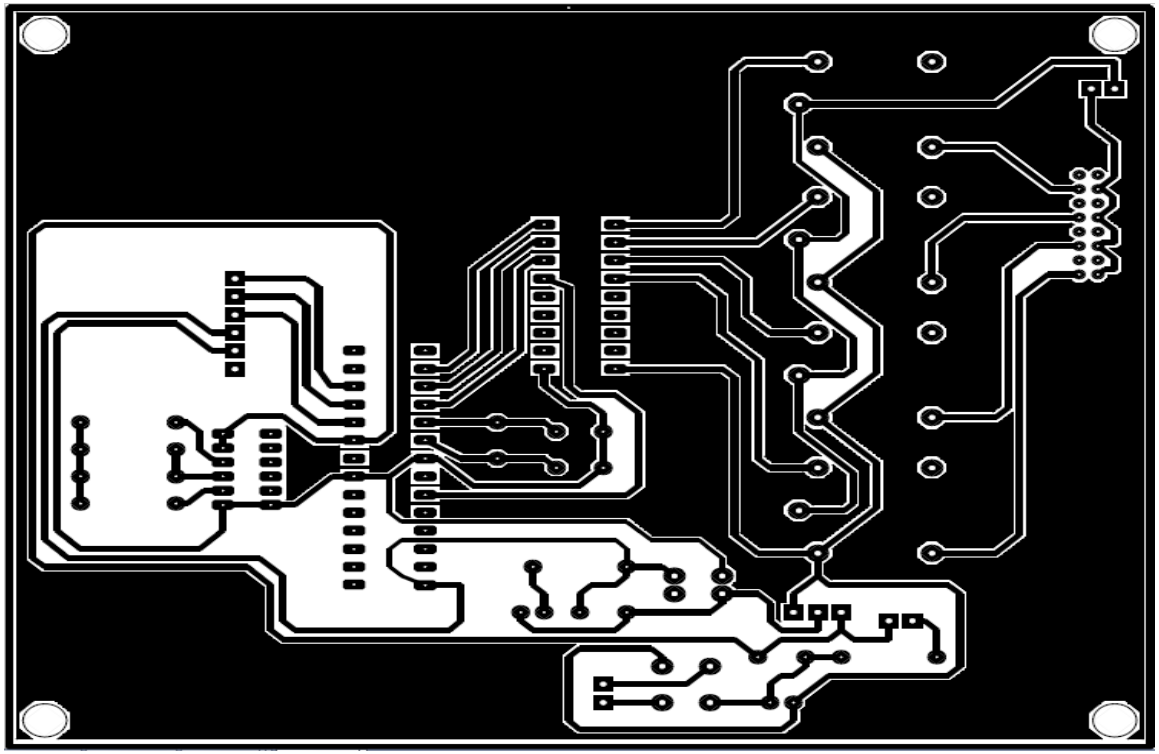
RANGKAIAN DRIVER RELAY			KETERANGAN
FT UNY	SKALA: -	DIG. FAJAR	A4 / No.4
	DIP. NURKHAMI	DIST. NURKHAMI	NIM. 13507134013

6. Gambar Rangkaian Keseluruhan



GAMBAR RANGKAIAN KESELURUHAN			KETERANGAN
FT UNY	SKALA: -	DIG. FAJAR	A4 / No.5
	DIP. NURKHAMI	DIST. NURKHAMI	NIM. 13507134013

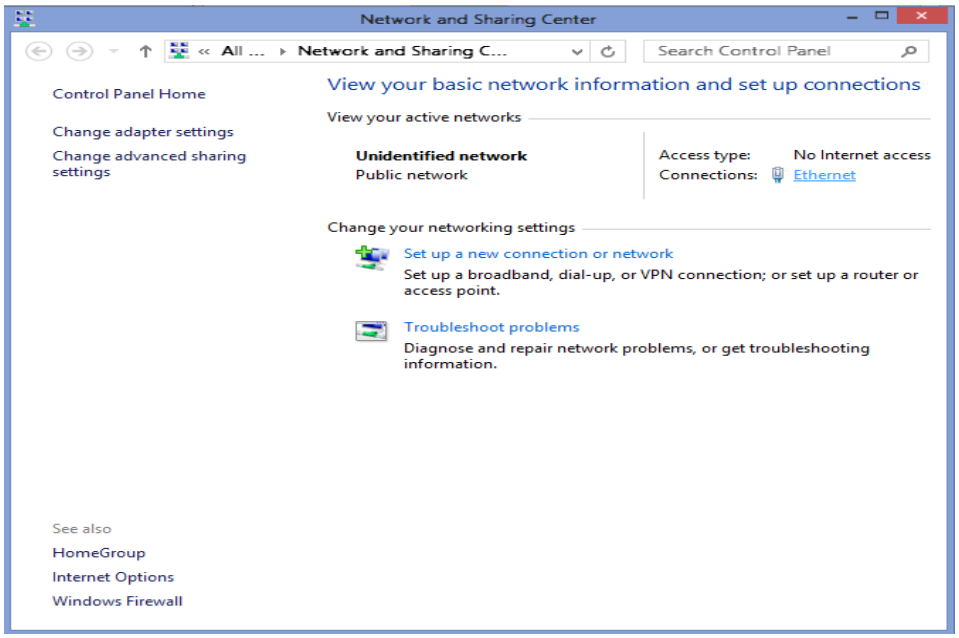
7. Layout Rangkaian PCB



LAYOUT RANGKAIAN PCB			KETERANGAN	
FT UNY	SKALA: -	DIG. FAJAR	A4	No.6
	DIP. NURKHAMI	DIST. NURKHAMI	NIM. 13507134013	

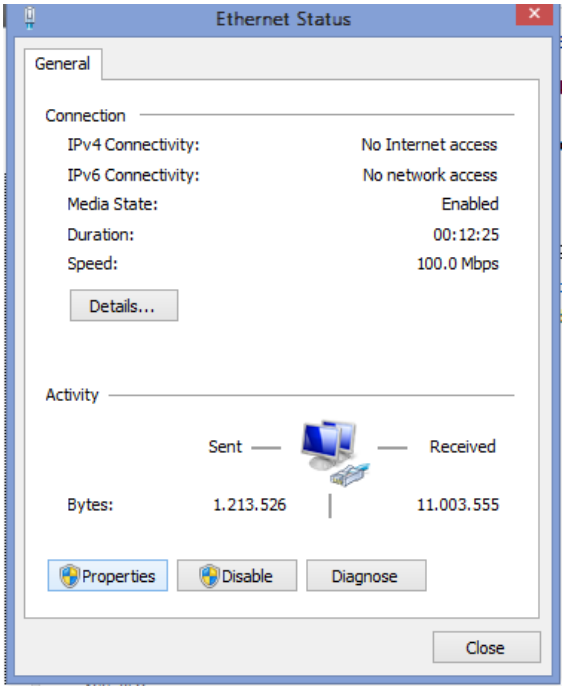
8. Setting Router TP-Link *Wireless* 2,4 GHz

Setting router *Wireless* 2,4 GHz berfungsi untuk mengkoneksikan jaringan *wireless* pada alat yang digunakan, sebelum setting router terlebih dahulu setting ip pada komputer/laptop dengan cara tancapkan kabel LAN ke laptop yang sudah tersedia pada router lalu arahkan kursor pada bagian kanan bawah laptop yang ada gambar komputernya setelah itu, klik kanan pada bagian gambar komputer tersebuat lalu akan ada tampilan (Open Network and Sharing Center) kemudian akan ada tampilannya seperti pada Gambar 1 setelah itu klik ethernet.



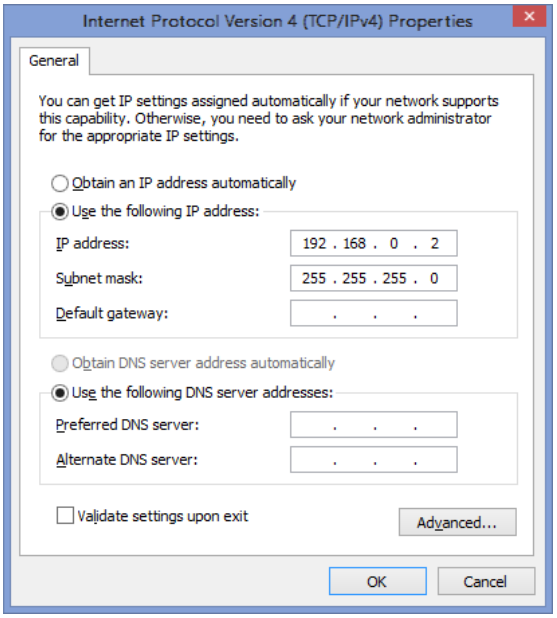
Gambar 1. Tampilan Open Network and Sharing Center

Langkah kedua, klik tulisan ethernet setelah diklik pada tulisan ethernet maka akan keluar tampilan (Ethernet Status), pada tampilan ini terdapat tiga nama diantaranya ada propertis, disable dan diagnose. Karena akan di-setting ip pada laptop maka pilih propertis tampilanya bisa dilihat pada Gambar 2.



Gambar 2. Tampilan Ethernet Status

Langkah ketiga, setelah diklik pada tulisan propertis maka akan keluar tampilan (Internet Protokol Version 4), setelah kelur tampilanya langsung saja setting ip yang akan dipakai. Ip yang di-setting pada laptop harus sama dengan ip default dari router, biasanya ip default pada router yaitu 192.168.0.1 maka settingan ip pada router harus sama hanya yang membedakan angka bagian belakang asalkan lebih dari angka 1 agar nantinya ip yang di-settings pada laptop tidak akan bentrok dengan ip router, langsung saja setting ip 192.168.0.2 pada laptop dan subnet mask 255.255.255.0 subnet mask-nya sendiri diisikan default bahkan biasanya akan keluar dengan sendirinya setelah di-setting ip localnya. Untuk tampilan ip-nya dapat dilihat pada Gambar 3.

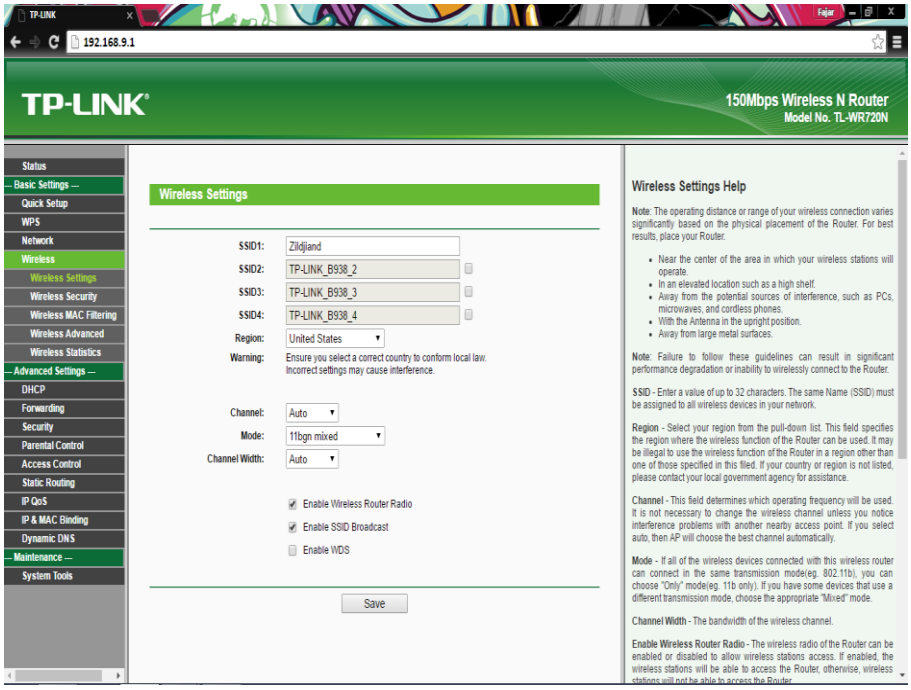


Gambar 3. Tampilan Internet Protokol Version 4

Langkah ke empat, karena pada laptop sudah di-setting semua maka selanjutnya setting ip pada router. Untuk masuk ke router terlebih dahulu buka browser yang biasa digunakan baik itu google chrome ataupun mozila firefox, setelah browser dibuka lalu tuliskan ip pada router biasnya ip default pada router yaitu 192.168.0.1 lalu klik enter maka akan ada tampilan password pada tampilan browser setelah itu isikan username dan password-nya, password-nya sendiri biasanya masih default tinggal diisi kan dengan username : admin dan password : admin lalu klik enter maka akan langsung keluar tampilan pada router.

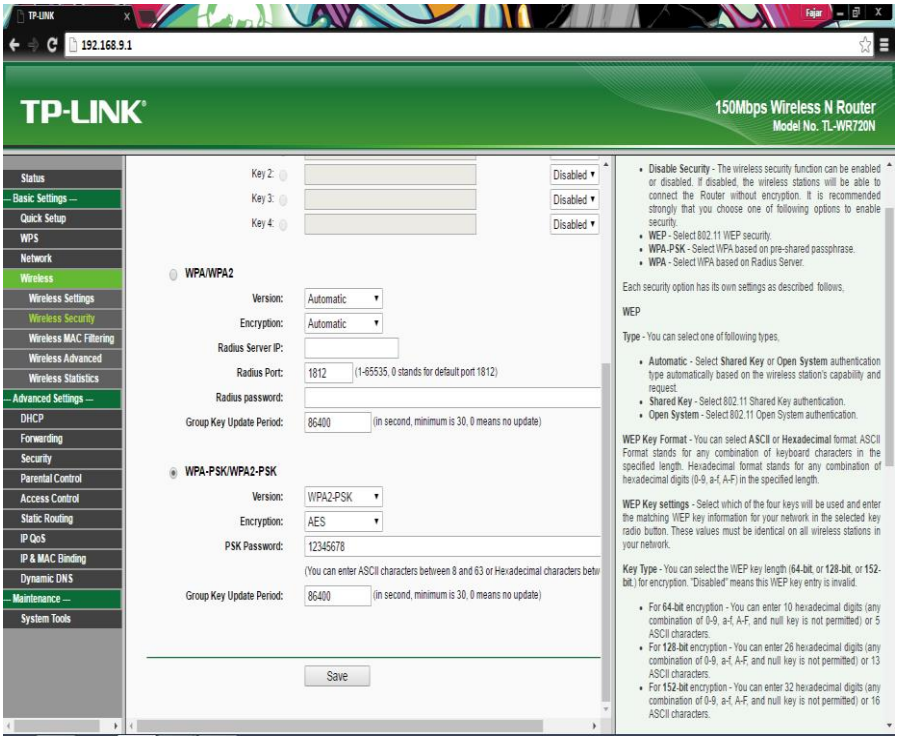
Langkah ke lima, sesudah itu pilih *wireless* lalu pilih *wireless* setings kemudian klik maka akan keluar tampilan *wireless* settings. Tampilanya dapat dilihat pada Gambar 4, di bagian ini merupakan nama (SSID) yang di miliki oleh router disini nama (SSID) sebelumnya diganti dengan nama (Zildjian) karena SSID ada 4 pilihan maka cukup dengan

mengisikan pada kolom SSID pertama. Nama yang diisikan bebas karena nama pada SSID ini yang nantinya akan di tampilkan pada wifi yang akan di gunakan dan agar mudah di kenal.



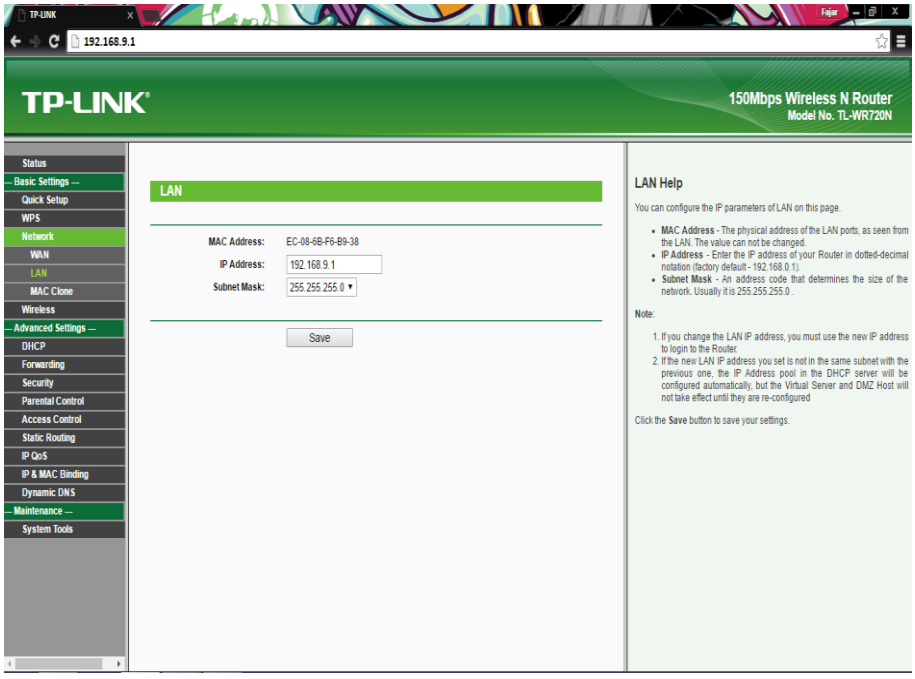
Gambar 4. Tampilan *Wireless* Setting

Langkah ke enam, setelah diganti nama pada SSID kemudian pilih *wireless* security dan klik maka akan keluar tampilanya. Bagian ini merupakan pengaman atau password pada *wireless* yang akan dipakai, guna untuk menjaga keamanan maka disini diisikan password 1 sampai 8 agar mudah diingat. Tampilanya dapat dilihat pada Gambar 5.



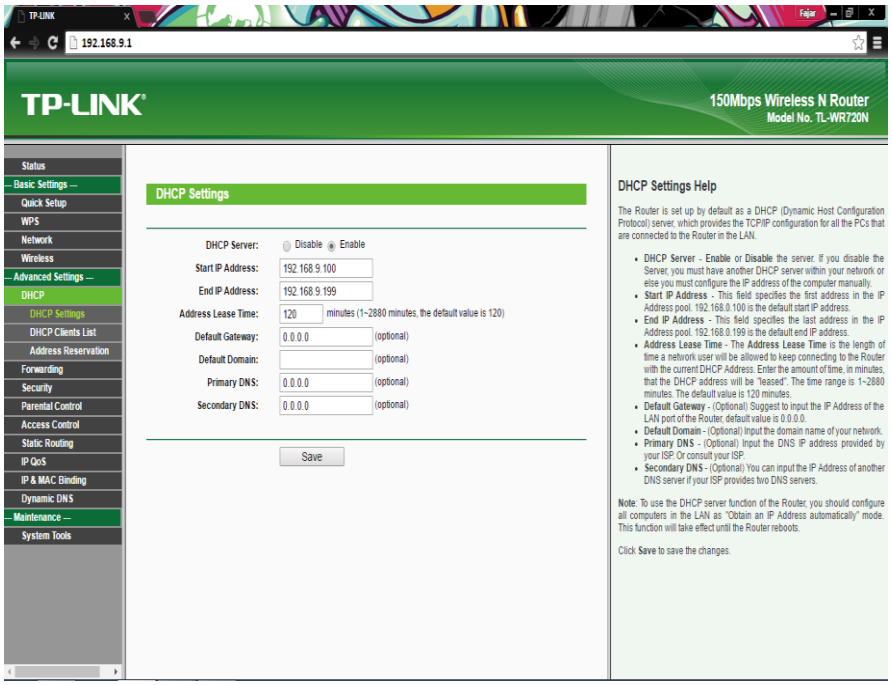
Gambar 5. Tampilan *Wireless* Security

Langkah ke tujuh, setelah setting password kemudian setting ip pada LAN dengan klik pada bagian network karena yang di-setting hanya ip localnya maka pilih LAN kemudian klik maka akan langsung keluar tampilan LAN setelah itu setting ip yang ada pada kolom (IP Address) ip local yang akan digunakan pada router yaitu 192.168.9.1 dengan subnet mask 255.255.255.0 untuk tampilan setting-nya bisa dilihat di Gambar 6.



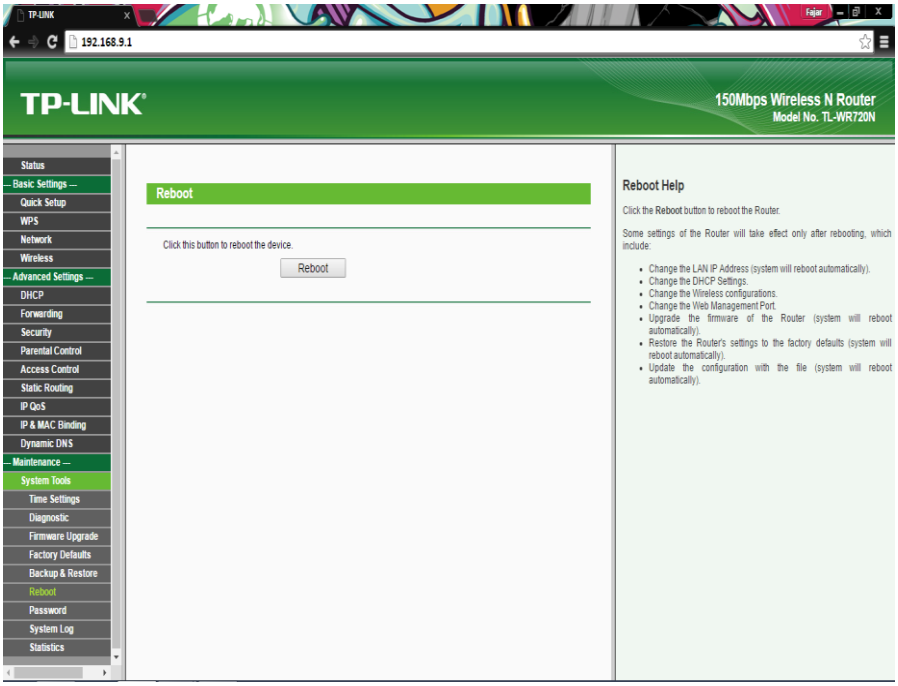
Gambar 6. Tampilan DHCP Setting

Langkah ke delapan, setelah ganti ip LAN, buka kembali browser dengan ip baru yaitu 192.168.9.1 pilih menu DHCP lalu klik maka akan keluar tampilan DHCP setting. Bagian ini merupakan bagian untuk membatasi client yang menggunakan *wireless* untuk membatasinya dengan memilih enabel lalu isikan ip 192.168.9.100 dan 192.168.9.199 yang artinya client yang dapat menggunakan *wireless* ini bisa sampai 199-100=99 client, tampilanya sendiri dapat dilihat pada Gambar 7.



Gambar 7. Tampilan LAN Setting

Langkah ke sembilan atau langkah terakhir, setelah semuanya di-setting kemudian pilih maintenance lalu pilih sistem tools setelah itu pilih Reboot dan akan terlihat tampilanya lalu klik nama Reboot tampilanya dapat dilihat pada Gambar 8. Reboot sendiri berfungsi untuk menyimpan (Save) semua setting yang sudah dilakukan dari awal agar settingannya tidak hilang.



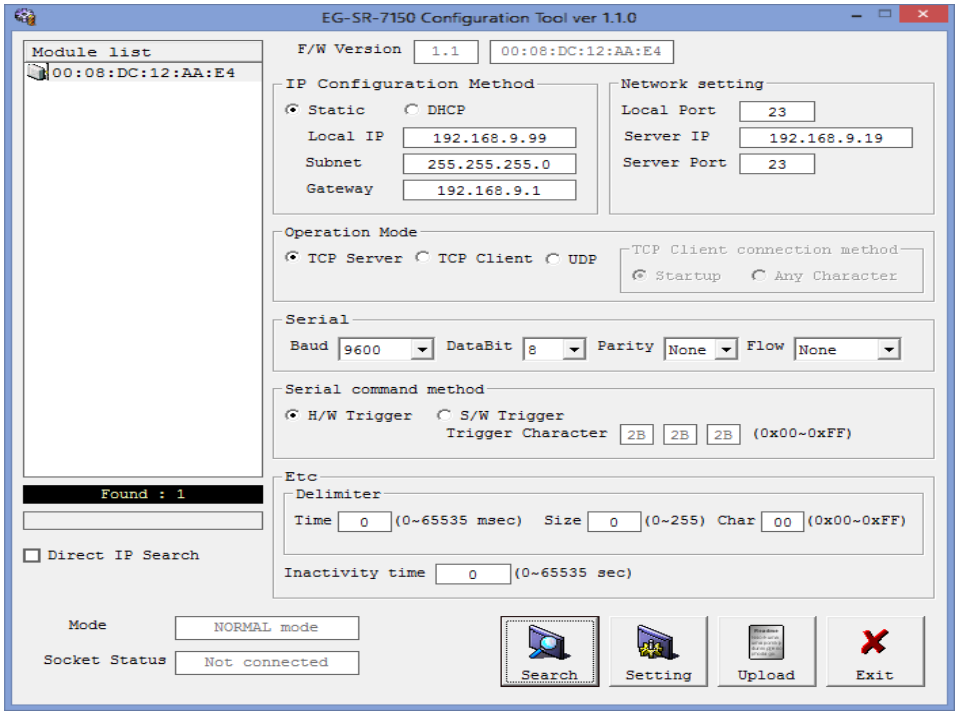
Gambar 8. Tampilan Reboot

9. Setting *Wiznet Egsr* 7150 MJ

Setting pada *wiznet* berfungsi untuk menerima data yang akan dikirim melalui *wireless* 2,4 GHz yang terdapat pada router dan juga melalui *smartphone,* setelah itu *wiznet* akan mengolah data yang telah diterima menjadi data serial karena rangkaian pada alat tidak dapat mendeteksi data yang di kirim melalui *wireless*. Maka dari itu, digunakanlah *wiznet* untuk menerima data sekaligus mengirim perintah pada mikrokontroler ATmega 16 yang nantinya akan menjalankan alat terutama ke empat buah relay untuk menghidupkan lampu. Sedangkan untuk setting *wiznet* sendiri, dengan mengunakan aplikasi bawaan dari *wiznet*. Cara setting diaplikasi *wizne*t yaitu pilih nama (Static) kemudian isikan ip yang sudah ada, maksud dari ip yang sudah ada yaitu ip yang diisikan pada kolom local ip harus sama dengan ip yang sudah di-setting pada router sebelumnya karena *wiznet* hanya akan bekerja dengan ip yang sama dengan pengirimnya (Satu kelas). Karena ip pada router yaitu 192.168.9.1 dengan subnet mask 255.255.255.0 maka ip yang harus diisikan pada *wiznet* bisa dipilih mulai dari 192.168.9.2-192.168.9.254 dengan subnet mask sama 255.255.255.0 selain subnet mask isikan juga gateway sesuai ip di atas hanya saja bagian belakangnya 1 di samakan dengan ip pada router yaitu 192.168.9.1. Ip yang dimaksud dengan ip satu kelas yang berada pada bagian titik ke tiga yaitu yang berisi angka 9, selain itu isikan juga pada bagian local port, server ip dan server port. Pada bagian port isikan 23 karena port yang di gunakan untuk mengendalikan

lampu pada mocca telnet biasanya port 23 lalu pada server ip isikan ip 192.168.9.19 sedangkan pada bagian server port mengikuti port yang sudah diisi tadi yaitu 23.

Setelah setting ip *wiznet* dilanjutkan setting baud rate / kecepatan transfer data serial antara *wiznet* dan mikrokontroler. Baud rate yang akan digunakan adalah 9600 sesuai dengan program yang sudah dibuat, maka setting boudrate diaplikasi *wiznet* juga diisikan 9600 guna agar dapat terhubung dengan mikrokontroler ATmega 16 pada bagian databit isikan 8 karena yang digunakan hanya 8 bit untuk bagian parity dan flow diisi none. Setelah semua bagian pada kolom-kolom tertentu sudah diisi, maka klik setting untuk menyimpan semua settingan pada *wiznet*. Sedangkan tampilanya dapat dilihat pada Gambar 9.



Gambar 9. Tampilan Setting *Wiznet Egsr* 7150 Mj

10. Listing Program pada ATmega 16

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

Project :
Version :
Date    : 19/07/2016
Author  : Fajar Ari
Company :
Comments:


Chip type               : ATmega16
Program type            : Application
AVR Core Clock frequency: 11,059200 MHz
Memory model            : Small
External RAM size       : 0
Data Stack size         : 256
*****************************************************/

#include <mega16.h>

// Standard Input/Output functions
#include <stdio.h>
#include <stdio.h>
#include <delay.h>
unsigned char zildjian;


// Declare your global variables here

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

// Input/Output Ports initialization
// Port A initialization
// Func7=Out Func6=Out Func5=Out Func4=Out Func3=Out Func2=Out Func1=Out
Func0=Out
// State7=1 State6=1 State5=1 State4=1 State3=1 State2=1 State1=1 State0=1
PORTA=0xFF;
DDRA=0xFF;
```

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

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

// Port D initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTD=0x00;
DDRD=0x00;

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

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: Timer1 Stopped
// Mode: Normal top=0xFFFF
// OC1A output: Discon.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=0x00;
TCCR1B=0x00;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
```

```
OCR1BL=0x00;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer2 Stopped
// Mode: Normal top=0xFF
// OC2 output: Disconnected
ASSR=0x00;
TCCR2=0x00;
TCNT2=0x00;
OCR2=0x00;

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
// INT2: Off
MCUCR=0x00;
MCUCSR=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x00;

// USART initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART Receiver: On
// USART Transmitter: On
// USART Mode: Asynchronous
// USART Baud Rate: 9600
UCSRA=0x00;
UCSRB=0x18;
UCSRC=0x86;
UBRRH=0x00;
UBRRL=0x47;

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

// ADC initialization
// ADC disabled
ADCSRA=0x00;

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

```
// TWI initialization
// TWI disabled
TWCR=0x00;
putchar(0x0a);putchar(0x0d);
PORTA=0x00;
while (1)
     {
     printf("Masukan Code Perintah :"); putchar(0x0a);putchar(0x0d);
     printf("(#) Untuk Menghidupkan"); putchar(0x0a);putchar(0x0d);
     printf("(*) Untuk Mematikan"); putchar(0x0a);putchar(0x0d);
     printf("(z) Untuk Melihat Laporan"); putchar(0x0a);putchar(0x0d);
     zildjian=getchar();
     if (zildjian=='*')
       {printf("Pilih Lampu :");putchar(0x0a);putchar(0x0d);
        printf("('1 2 3 4 5')"); putchar(0x0a);putchar(0x0d);
           zildjian=getchar();
          // putchar(zildjian);
           if (zildjian=='1'){PORTA.4=0;zildjian=0x00;}
           if (zildjian=='2'){PORTA.5=0;zildjian=0x00;}
           if (zildjian=='3'){PORTA.6=0;zildjian=0x00;}
           if (zildjian=='4'){PORTA.7=0;zildjian=0x00;}
           if (zildjian=='5'){PORTA=0b00000000;zildjian=0x00;}
       }

     if (zildjian=='#')
       {printf("Pilih Lampu :");putchar(0x0a);putchar(0x0d);
        printf("('1 2 3 4 5')"); putchar(0x0a);putchar(0x0d);
           zildjian=getchar();
           //putchar(zildjian);
           if (zildjian=='1'){PORTA.4=1;zildjian=0x00;}
           if (zildjian=='2'){PORTA.5=1;zildjian=0x00;}
           if (zildjian=='3'){PORTA.6=1;zildjian=0x00;}
           if (zildjian=='4'){PORTA.7=1;zildjian=0x00;}
           if (zildjian=='5'){PORTA=0b11111111;zildjian=0x00;}
       }
     if (zildjian=='z')
       {printf("Laporan :");putchar(0x0a);putchar(0x0d);
 if (PORTA.4==1){printf("Lampu 1 Hidup");putchar(0x0a);putchar(0x0d);}
              else {printf("Lampu 1 Mati");putchar(0x0a);putchar(0x0d);}
           if (PORTA.5==1){printf("Lampu 2 Hidup");putchar(0x0a);putchar(0x0d);}
              else {printf("Lampu 2 Mati");putchar(0x0a);putchar(0x0d);}
           if (PORTA.6==1){printf("Lampu 3 Hidup");putchar(0x0a);putchar(0x0d);}
              else {printf("Lampu 3 Mati");putchar(0x0a);putchar(0x0d);}
           if (PORTA.7==1){printf("Lampu 4 Hidup");putchar(0x0a);putchar(0x0d);}
              else {printf("Lampu 4 Mati");putchar(0x0a);putchar(0x0d);}
       }
       }
}
```

11. Bentuk Alat Sistem Kendali Lampu Via *Wireless* 2,4 GHz Berbasis Mikrokontroler ATmega 16

12. Cara Pengoperasian Alat

1. Menghubungkan *power supply* pada stop kontak.
2. Hubungkan *Smartphone* pada jaringan *wereless/*wifi.
3. Buka aplikasi *Mocca Telnet* pada *Smartphone android*.
4. Klik tampilan Connect pada Mocha Lite Telnet, android siap untuk memberikan perintah kepada lampu.
5. Masukan code perintah “Pagar ” (#) untuk menghidupkan lampu lalu masukan nomer lampu (1 2 3 4) yang akan di hidupkan {#1, #2, #3, #4}. Code Bintang (*) untuk mematikan lampu lalu masukan nomer lampu (1 2 3 4) yang akan di matikan {*1, *2, *3, *4} dan Zet (z) untuk melihat laporan dari lampu.