Implementasi *Embedded Web Server* Via Modem Berbasiskan Mikrokontroler

Donny K. Sutantyo, Darmawan Utomo
Fakultas Teknik jurusan Teknik Elektro, Universitas Kristen Satya Wacana, Salatiga
Email:

Abstrak

Pada makalah ini telah direalisasikan sebuah *web server* yang menyimpan halaman-halaman *web* pada sebuah mikrokontroler. Untuk merealisasikan sebuah *web server* yang terhubung ke jaringan internet, maka protokol TCP/IP harus diimplementasikan sebagai perangkat lunak terlebih dahulu pada sistem mikrokontroler yang digunakan sebagai *web server*. Protokol-protokol penyusun TCP/IP yang diimplementasikan yaitu *Serial Line Internet Protocol* (SLIP), *Internet Protocol* (IP), *Transmission Control Protocol* (TCP), *Internet Control Message Protocol* (ICMP), dan *Hypertext Transfer Protocol* (HTTP). Sistem mikrokontroler dihubungkan ke jaringan internet dengan menggunakan modem eksternal jenis V90, karena itu dibuat pula perangkat keras antarmuka (*interface*) modem dan perangkat lunak pengendali (*driver*) modem.Kelebihan sistem *Embedded Web Server* pada makalah ini dibandingkan yang sudah ada di pasaran adalah protokol TCP/IP terletak di mikrokontroler sebagai perangkat lunak, sehingga realisasi sistem menjadi jauh lebih efisien, karena tidak memerlukan IC TCP/IP s menjadikan sistem lebih ringkas dan murah. Dari hasil pengujian dapat disimpulkan bahwa sistem dapat bekerja dengan baik.

Kata kunci : sistem embedded, mikrokontroler, protokol komunikasi, TCP/IP.

Abstract

In this paper there has been implemented a web server which store web pages on a microcontroller chip. To create a web server which connect to internet, TCP/IP protocol must be implemented as a software in the microcontroller system which use as web server. Protocols in TCP/IP which implemented are Serial Line Internet Protocol (SLIP), Internet Protocol (IP), Transmission Control Protocol (TCP), Internet Control Message Protocol (ICMP), and Hypertext Transfer Protocol (HTTP). Microcontroller connected physically to internet with use external V90 modem, so it is necessary to develop modem hardware interface and modem driver software. The benefit of this Embedded Web Server system compares with available systems on the market is that the TCP/IP protocol located inside the microcontroller as a software, so its realization can be cheaper and more compact, because it is not necessary to use TCP/IP chip to connect the microcontroller to internet.

***Keywords**: embedded system, microcontroler, communication protocol, TCP/IP.*

Latar Belakang

Dengan berkembangnya teknologi internet yang merupakan suatu sistem komunikasi yang *reliable*, maka internet sering digunakan sebagai media pada pemantauan dan pengendalian jarak jauh. Internet diharapkan dapat menjadi sebuah media pemantau dan pengendali jarak jauh yang cukup baik karena banyaknya protokol-protokol yang digunakan serta kemampuan protokol-protokol komunikasi tersebut untuk mengurangi kesalahan informasi yang dikirimkan seminimal mungkin.

Akan tetapi kendala yang dihadapi adalah masalah ketidakpraktisan (tidak *portable*) dan boros daya karena minimal harus menggunakan sebuah *Personal Computer* (PC) sebagai *web server* untuk menghubungkan alat yang dikendalikan dengan jaringan internet. Karena itu untuk tujuan portabilitas dan efisiensi daya, pada penelitian ini diciptakan suatu perangkat kecil pengganti PC sebagai *web server* untuk dapat menghubungkan alat yang dikendalikan dengan jaringan internet, yang dalam hal ini maka seperti pada judul, alat ini dinamai *Embedded Web Server*. *Embedded Web Server*, sesuai namanya, diharapkan dapat ditanamkan (*embedded*) pada perangkat-perangkat elektronik sehingga perangkat elektronik tersebut dapat terkoneksi ke internet dan dapat dikendalikan maupun dipantau jarak jauh melalui internet. Jenis koneksi dipilih melalui *modem* karena diharapkan alat ini bisa diaplikasikan untuk pengguna rumahan (*home user*). Kelebihan sistem ini dibandingkan *Embedded Web Server* yang sudah ada di pasaran adalah protokol TCP/IP terletak di dalam mikrokontroler sebagai perangkat lunak, sehingga realisasi sistem menjadi jauh lebih efisien, karena tidak memerlukan IC TCP/IP sehingga lebih ringkas dan murah.

Catatan: Diskusi untuk makalah ini diterima sebelum tanggal 1 Juni 2006. Diskusi yang layak muat akan diterbitkan pada Jurnal Teknik Elektro volume 6, nomor 2, September 2006.

Dasar Teori

Arsitektur Jaringan TCP/IP

TCP/IP adalah sebuah arsitektur jaringan yang dipakai dalam jaringan internet. TCP/IP dilahirkan oleh ARPANET, sebuah pusat riset yang disponsori oleh DoD (Departemen Pertahanan AS). Sampai sekarang model ini sudah berkembang sedemikian pesatnya. Semua referensinya di atur di dalam standar *RFC (Request For Comment)*. Standar ini bersifat *open source*, semua referensinya bisa didapat secara gratis dari internet (www.rfc_index.com).

TCP/IP terdiri dari 4 *layer* (lapis) yang berisikan bermacam-macam protokol. Susunan *layer* tersebut adalah sebagai berikut:

Application Layer	→ HTTP/SMTP/FTP/dll
Transport Layer	→ TCP/UDP/ICMP
Internet Layer	→ IP
Network Access Layer	→ Ethernet/SLIP/PPP

Gambar 1. Susunan Layer Pada Protokol TCP/IP[5]

Tiap *layer* dari TCP/IP ini mempunyai fungsi-fungsi yang berbeda dan didalam tiap *layer* ini dapat bervariasi protokol yang bekerja pada *layer* tersebut, karena itu struktur dan ukuran data yang terdapat pada tiap *layer* juga tidak sama. Meskipun demikian format data setiap *layer*-nya dibuat saling kompatibel agar transmisi data menjadi efisien.

Masing-masing layer TCP/IP mempunyai *header* yang berisi informasi kontrol. Proses penambahan dan pengurangan setiap paket data dengan *header* masing-masing layer dilakukan dengan proses enkapsulasi dan dekapsulasi. Untuk mengirimkan sebuah data mesin akan melakukan enkapsulasi *header*, dan sebaliknya untuk menerima data mesin akan melakukan dekapsulasi *header* lalu mengolah informasi-informasi kontrol yang terdapat di dalamnya.

Pada implementasi ini, protokol pada Network Access Layer dipakai SLIP karena merupakan protokol yang dibutuhkan jika koneksinya menggunakan modem, pada Internet Layer dipilih IP yang merupakan protokol yang harus ada untuk bisa koneksi ke internet, pada Transport Layer dipilih TCP untuk mengarahkan port aplikasi ke HTTP, dan pada Application Layer dipilih HTTP[1].
Serial Line Internet Protocol (SLIP)

Protokolnya yang dijelaskan dalam RFC 1055, sangatlah sederhana. *Host* cukup mengirimkan paket-paket IP melalui saluran dengan *byte flag* khusus (0xC0) untuk keperluan pembuatan *frame*. Bila terjadi *byte flag* di dalam paket data IP, maka digunakan pengisian karakter *(byte stuffing)*, dan dua *byte* yang berturutan (0xDB dan 0xDC) disisipkan menggantikan *byte* tersebut. Bila yang terjadi di dalam paket IP adalah 0xDB, maka dalam hal ini juga akan digunakan pengisian karakter, yaitu 0xDB dan 0xDD.

Internet Protokol (IP)

Internet Protocol (IP) yang terdokumentasi dalam standar RFC 791, berada pada *Internet Layer* dari model protokol TCP/IP.

Satuan paket data dari IP disebut *datagram*. Sebuah *datagram* terdiri dari bagian *header* dan data. *Header* mempunyai bagian tetap sebesar 20 *bytes* (5x32bit) dan bagian *optional* yang panjangnya dapat berubah-ubah. *Datagram* ditransmisikan dari kiri ke kanan.

<table>
<tr><td>4</td><td>8</td><td>16</td><td>32 bits</td></tr>
<tr><td>Ver.</td><td>IHL</td><td>Type of service</td><td>Total length</td></tr>
<tr><td colspan="3">Identification</td><td>Flags</td><td>Fragment offset</td></tr>
<tr><td>Time to live</td><td colspan="2">Protocol</td><td>header checksum</td></tr>
<tr><td colspan="4">Source address</td></tr>
<tr><td colspan="4">Destination address</td></tr>
<tr><td colspan="4">Option + Padding</td></tr>
<tr><td colspan="4">Data</td></tr>
</table>

Gambar 2. Format *Datagram* IP

Guna masing-masing *field* pada *datagram* dapat dilihat di referensi RFC 791.

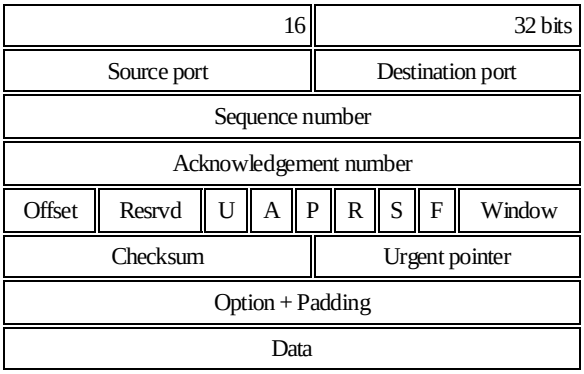
Transmission Control Protocol (TCP)

TCP yang terdokumentasi dalam RFC 793, merupakan salah satu protokol *Transport Layer* pada model TCP/IP. Dengan berjalannya waktu, TCP mengalami banyak perkembangan. Berbagai perbaikan di dokumentasikan dalam RFC 1122 dan RFC 1323.

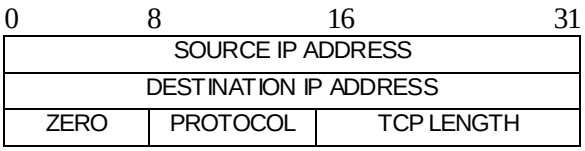
Satuan paket data dari TCP disebut *segment*. Sebuah *segment* terdiri dari *header* dan data. *Header* mempunyai bagian tetap 20 *byte*s dan bagian *optional* yang panjangnya dapat berubah-ubah. Berikut adalah format *header* TCP. *Header* dikirimkan dari kiri ke kanan.

TCP menambahkan *pseudo-header* untuk keperluan perhitungan *checksum* pada *segment*-nya. Sehingga perhitungan *checksum* pada TCP meliputi *header* dan data TCP serta *pseudo-header* TCP. *Pseudo-*

header ini hanya berguna untuk perhitungan *checksum* saja dan tidak ikut dikirimkan bersama dengan *header* dan data. Maksud penggunaan *pseudo-header* ini adalah untuk memastikan bahwa *segment* telah mencapai tujuannya dengan benar.



Gambar 3. Format TCP *Header Segment*



Gambar 4. Format Pseudo-header TCP

Guna masing-masing *field* pada *segment* dan *pseudo-header* dapat dilihat di referensi RFC 793, mengingat keterbatasan panjang penulisan.

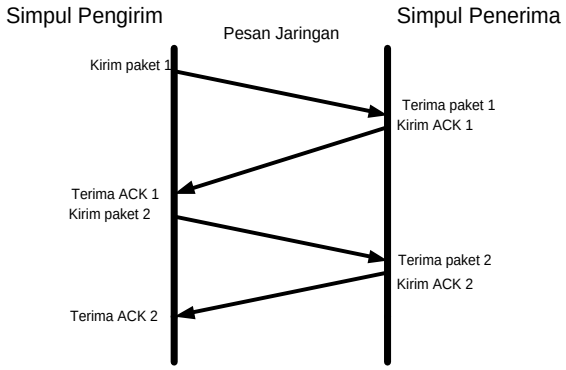
Cara Kerja TCP

Seperti telah disebutkan di atas, TCP melakukan *handshaking* untuk memulai koneksi, mengakhiri koneksi, dan mengirim data. Untuk menjelaskan cara kerja TCP dalam melakukan *handshaking* dapat dimisalkan dengan sebuah jaringan *client-server* sederhana.

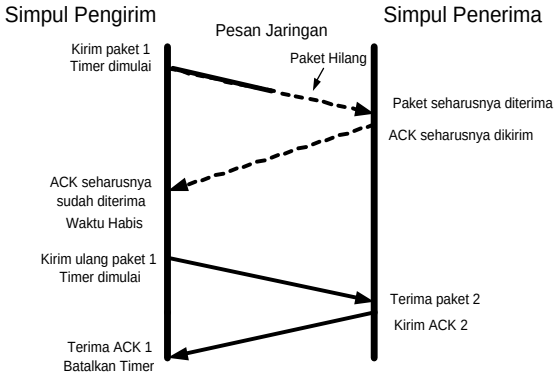
Suatu simpul memulai hubungan *(establish connection)* bila simpul tersebut akan berkomunikasi dengan simpul yang lain dalam jaringan. Proses ini dilakukan dengan membentuk proses *client* dengan menggunakan TCP. Selanjutnya proses *client* akan mengirimkan permintaan ke simpul tujuan, yang cara pengalamatannya diatur oleh IP. Proses *server* pada simpul tujuan akan menanggapi permintaan ini dengan mengirimkan jawaban ke *client*. Dengan demikian terbentuklah hubungan semu antara dua simpul. Aplikasi pada lapis yang lebih tinggi dari *Transport Layer* akan menggunakannya untuk komunikasi. Setelah berkomunikasi, lapis yang lebih tinggi akan memberitahukan *Transport Layer* bahwa jalur telah selesai digunakan dan hubungan dapat diputuskan.

TCP mengirim paket data dalam bentuk urutan yang disebut *stream*. Data dari lapis di atasnya dipecah menjadi beberapa paket dan dikirimkan secara berurutan. Penerima akan menerimanya secara berurutan pula. Pengiriman ini dilakukan melalui untai semu yang dibentuk pada awal hubungan.

Hubungan yang dibentuk oleh TCP bersifat *full duplex*, yaitu pertukaran data dapat berlangsung dua arah pada waktu yang sama. Sifat *full duplex* ini juga dapat dimanfaatkan untuk menyisipkan informasi kontrol.



Gambar 5. Teknik *Positive Acknowledment*



Gambar 6. *Timeout* dan *Retransmission*

Informasi kontrol ini berupa bukti terima (ACK) untuk setiap paket data yang dikirim. Teknik pengiriman data dengan bukti terima ini disebut *positive acknowledment*. Simpul pengirim mengirimkan permintaan data ke simpul penerima. Simpul penerima membalasnya dengan mengirimkan data yang diminta dan disisipi dengan informasi kontrol bukti terima, yaitu bahwa simpul penerima telah menerima data dari simpul pengirim dengan sempurna.

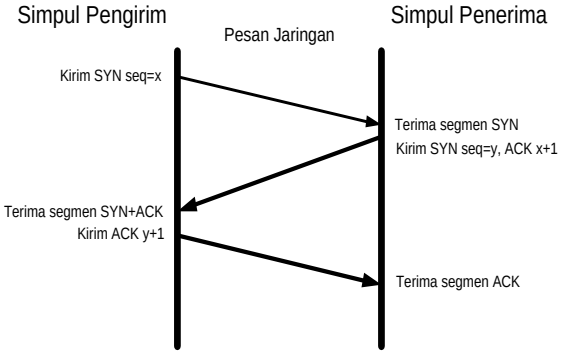
Bila suatu saat terjadi kehilangan atau kerusakan pada paket data, simpul penerima akan menunggu terus sedangkan simpul pengirim tidak akan menerima bukti terima dan kemungkinan akan menunggu

terus datangnya bukti terima tersebut. Akibatnya sistem jaringan akan menggantung *(hang up)*.

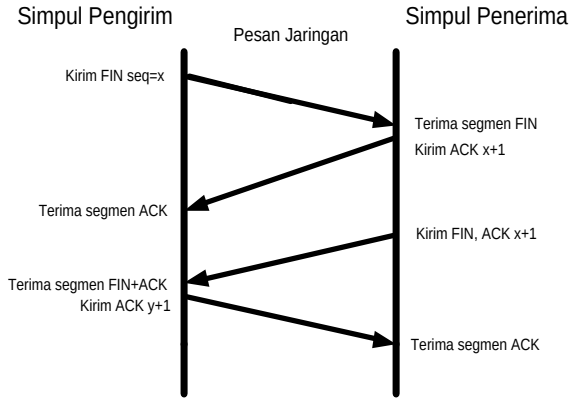
Untuk menghindari hal tersebut di atas digunakan cara *timeout*, yaitu setelah mengirimkan data, simpul pengirim akan menunggu datangnya bukti terima selama selang waktu tertentu. Setelah selang waktu ini habis dan bukti terima belum diterima, simpul pengirim akan *timeout*. Kemudian data yang hilang tersebut akan dikirimkan kembali oleh simpul pengirim *(retransmission)*.

Membuka dan Menutup Koneksi TCP

Seperti telah dijelaskan sebelumnya, agar dua simpul dapat berkomunikasi dengan menggunakan protokol TCP, simpul tersebut harus membentuk hubungan TCP. Di sini kedua simpul akan bertukar informasi kontrol *(handshake)* untuk membentuk suatu untai semu yang menyatakan kedua belah pihak telah siap untuk melakukan transfer data yang sesungguhnya. Untuk membuka koneksi ini TCP menggunakan metode *three-way-handshake*:



Gambar 7. *Three-way-handshake* Pembukaan Koneksi



Gambar 8. *Three-way-handshake* Menutup Koneksi

Secara garis besar proses ini berlangsung untuk tukar menukar nomor urut *(Sequence Number)* antara kedua simpul agar didapat perjanjian mengenai *Initial Sequence Number* (nomor urut pertama) yang akan digunakan oleh masing-masing simpul. *Segment* pertama pada proses *handshake* dapat diidentifikasi dari bit SYN yang diset '1'. Pada *segment* kedua bit SYN dan ACK keduanya diset '1'. *Segment* kedua ini menyatakan bahwa *segment* pertama sudah diterima dan simpul penerima mengirimkan tanda terima (ACK). Pada *segment* terakhir hanya bit ACK saja yang diset '1', yang menandakan bahwa antara kedua simpul telah terbentuk hubungan.

Hal lain yang perlu mendapat perhatian adalah *Sequence Number* dan *Acknowledment Number*. Masing-masing simpul akan memulai koneksi dengan *Sequence Number*-nya sendiri-sendiri yang biasanya dibangkitkan secara acak. Misal simpul pertama memulai dengan *Sequence Number* (seq) = x, simpul kedua dengan seq = y. *Segment* pertama dikirimkan dengan seq = x. *Segment* kedua dari simpul penerima dikirim dengan seq = y dan tanda terima untuk ack = x+1 (*Acknowledment Number* berisi nomor urut dari *byte* data yang diharapkan diterima berikutnya). Pada *segment* terakhir, *segment* dengan seq = y yang telah diterima diberi tanda terima dengan ack = y+1.

Ketika program aplikasi memberitahu TCP bahwa sudah tidak ada lagi data yang dikirim, TCP akan menutup koneksi dalam satu arah dengan *three-way-handshake* yang dimodifikasi. Untuk menutup setengah koneksi yang lain (satu arah yang lain), simpul yang sedang mengirimkan *segment* akan menyelesaikan pengiriman sisa data, menunggu tanda terima dari penerima, dan kemudian mengirimkan *segment* dengan bit FIN diset '1'. TCP penerima akan memberi tanda terima untuk *segment* FIN tersebut dan memberitahu program aplikasi bahwa sudah tidak ada lagi data dari simpul pengirim. Saat koneksi ditutup satu arah, data hanya dapat mengalir pada arah yang lain saja sampai koneksi ditutup seluruhnya oleh simpul yang lain.

Perbedaan *three-way-handshake* untuk menutup dan membuka koneksi ialah bahwa untuk menutup koneksi ditandai dengan *segment* dengan bit FIN diset '1'. *Segment* ini akan dibalas dengan ACK dan diikuti oleh pemberitahuan kepada program aplikasi bahwa koneksi telah ditutup satu arah. Pada akhirnya setelah program aplikasi selesai mengirimkan datanya, TCP diperintahkan untuk menutup koneksi seluruhnya dengan mengirimkan *segment* dengan bit FIN diset '1'. Yang kemudian akan diberikan tanda terima (ACK) terakhir oleh penerimanya.

Sequence Number dan *Acknowledment Number*

Saat proses pembukaan koneksi TCP dikenal *Initial Sequence Number* (ISN), yang merupakan titik awal sistem penomoran *byte* data. Karena alasan keaman-

an, ISN ditentukan secara acak, meski nilai awalnya bisa nol. Setiap *byte* data diberi nomor, diawali dari ISN, sehingga *byte* data pertama yang sesungguhnya akan bernomor urut ISN+1. *Sequence Number* mengidentifikasikan posisi dari *byte* data dalam *stream* data TCP. Sebagai contoh jika *byte* pertama dalam *stream* data bernomor urut 1 (ISN=0) dan sejumlah 4000 byte data sudah dikirim, maka *byte* pertama pada *segment* yang akan dikirim berikutnya akan bernomor urut 4001.

Segment dengan ACK melakukan dua macam fungsi yaitu *positive acknowledment* dan *flow control*. *Acknowledment* ini memberitahukan pengirim banyaknya data yang telah diterima oleh tujuan. *Acknowledment Number* adalah nomor urut dari *byte* berikutnya yang diharapkan diterima oleh simpul penerima. Standar TCP tidak memerlukan penerima mengirimkan ACK untuk setiap *byte* data yang diterimanya, tetapi hanya cukup ACK untuk *byte* terakhir yang diterima dari keseluruhan *segment* data yang dikirimkan. Misalkan jika *byte* pertama bernomor urut 1 dan 2000 *byte* data sudah diterima, maka *Acknowledment Number* akan berisi 2001.

Internet Control Message Protocol (ICMP)

Protokol ICMP didokumentasikan dalam standar RFC 792. Protokol ini dikembangkan karena adanya kebutuhan suatu mekanisme yang dapat digunakan untuk melakukan pelacakan bila terjadi kegagalan jaringan (*network fail*) dalam suatu jaringan komputer.

ICMP Echo

Dalam jaringan TCP/IP, ICMP *echo* digunakan oleh sebuah simpul untuk melakukan diagnostik terhadap keberadaan sebuah simpul lainnya. Selain itu juga digunakan untuk menguji kualitas media transmisi yang digunakan jaringan.
Urut-urutan cara kerjanya adalah sebagai berikut:

1. Simpul asal mengirimkan sebuah paket ke simpul yang diuji keberadaannya. Paket tersebut berisi permintaan yang disebut dengan ICMP *echo request*.
2. Bila simpul tujuan berada dalam jaringan dalam keadaan baik, ia akan menerima paket tersebut dan mengirimkannya kembali ke simpul asal. Paket balasan tersebut disebut dengan ICMP *echo reply*.
3. Langkah 1 dan 2 dilakukan sebanyak empat kali. Selanjutnya simpul asal akan menghitung jumlah paket yang berhasil dikirimkan balik oleh simpul tujuan dalam keadaan utuh. Dari perhitungan tersebut dapat diketahui kualitas media transmisi pada jaringan yang digunakan.

Pada komputer (PC) dengan sistem operasi Windows 98, perintah yang digunakan untuk melakukan diagnostik ini adalah "PING" (pada DOS prompt).
Satuan paket data pada protokol ICMP adalah *datagram*. Berikut adalah format *datagram* ICMP *echo*:

16		32 bits
Type	Code	Checksum
Identifier		Sequence Number
Optional Data		
……..		

Gambar 10. Format *Datagram* ICMP *echo*

HTTP *(Hypertext Transfer Protocol)*

Hypertext Transfer Protocol versi 1.0 (HTTP 1.0) yang dibahas dalam standar RFC 1945, merupakan salah satu protokol utama untuk menyebarkan dan mendistribusikan *web page* di internet. Protokol ini adalah protokol level tinggi dan bekerja pada *Application Layer*. Protokol HTTP ini ditetapkan untuk berkomunikasi dengan menggunakan protokol TCP pada port 80. *Web server* bekerja dengan menggunakan protokol HTTP, sehingga sering pula disebut sebagai *HTTP server*.

Mekanisme protokol HTTP didasarkan pada metode permintaan dan balasan (*request and response*). Sebuah terminal yang meminta layanan ini akan mengirimkan permintaan HTTP (*HTTP request*) ke *server* dengan format tertentu.

request chain → UA –--v---O ← response chain

Untuk melakukan *request* dan *response* pada HTTP dilakukan dalam bentuk teks ASCII. Dalam satu baris HTTP *request* akan terdiri dari metode yang akan diterapkan ke *resource*, identitas *resource* yang dimaksud, dan versi protokol yang digunakan

Beberapa contoh *HTTP request* yang dapat dilakukan seperti :

"GET / CRLF" atau "GET /1.html CRLF"

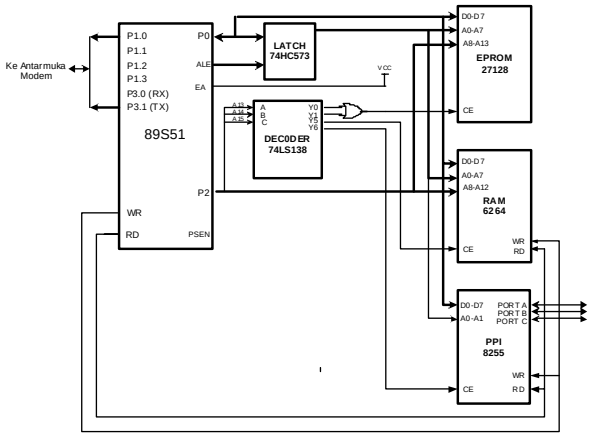
Lebih detil tentang protokol ini dapat dilihat pada referensi RFC 1945.

Perancangan dan Realisasi Sistem

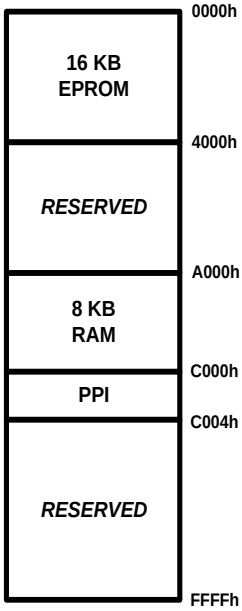
Perancangan Perangkat Keras Sistem Mikrokontroler

Sistem perangkat keras mikrokontroler yang menggunakan mikrokontroler dari keluarga MCS51 ini memiliki fasilitas RAM eksternal, EPROM

eksternal dan *multi I/O*. RAM eksternal berfungsi untuk buffer komunikasi data, EPROM difungsikan untuk menyimpan *file* HTML (halaman *Web*), sedangkan fasilitas multi I/O berfungsi untuk I/O serbaguna. Perangkat lunak TCP/IP diletakkan pada ROM internal mikrokontroler.



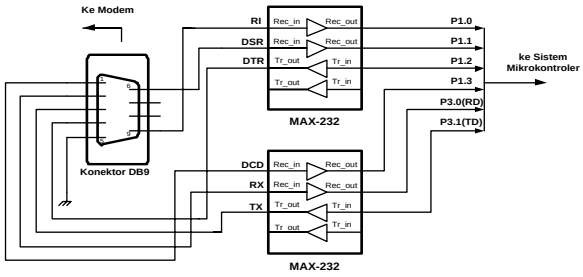
Gambar 11. Desain Sistem Mikrokontroler yang Digunakan



Gambar 12. Peta Pengalamatan Memori

Perancangan Perangkat Keras Antarmuka (*Interface*) Modem

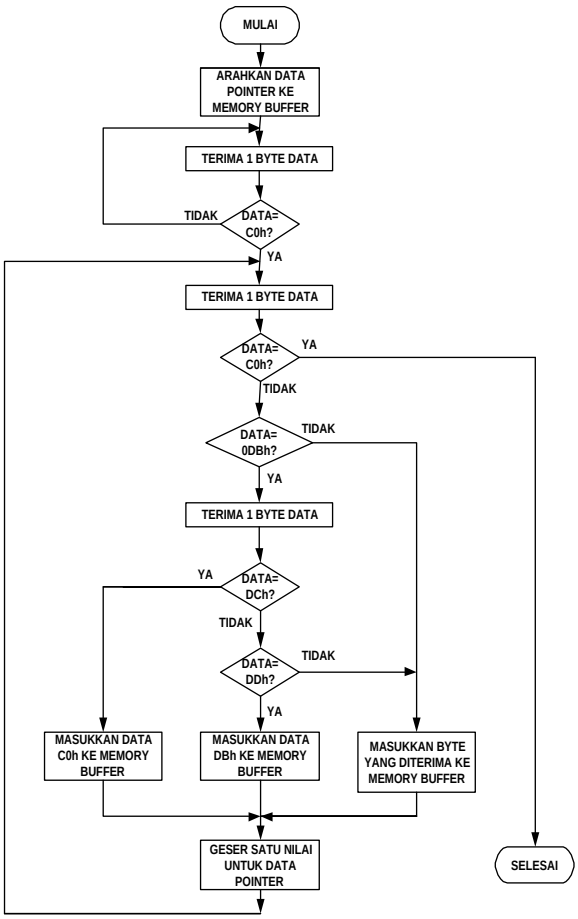
Pada sistem ini digunakan modem eksternal jenis V90 karena penggunaan modem internal akan memerlukan antarmuka yang lebih rumit karena modem internal berbentuk kartu ISA (*ISA card*) atau bahkan kartu PCI (PCI *card*). Antarmuka yang dirancang terhubung dengan *port* mikrokontroler dan konektor DB-9 dari modem. Antarmuka tersebut berisi sebuah untai yang berguna untuk mengkonversi level tegangan TTL yang digunakan mikrokontroler menjadi level tegangan RS-232 yang digunakan modem, dan sebaliknya. Implementasi antarmuka modem beserta perangkat lunak drivernya tidak dibahas secara mendetail, karena sudah pernah ditulis secara mendetail pada penelitian sebelumnya[2].



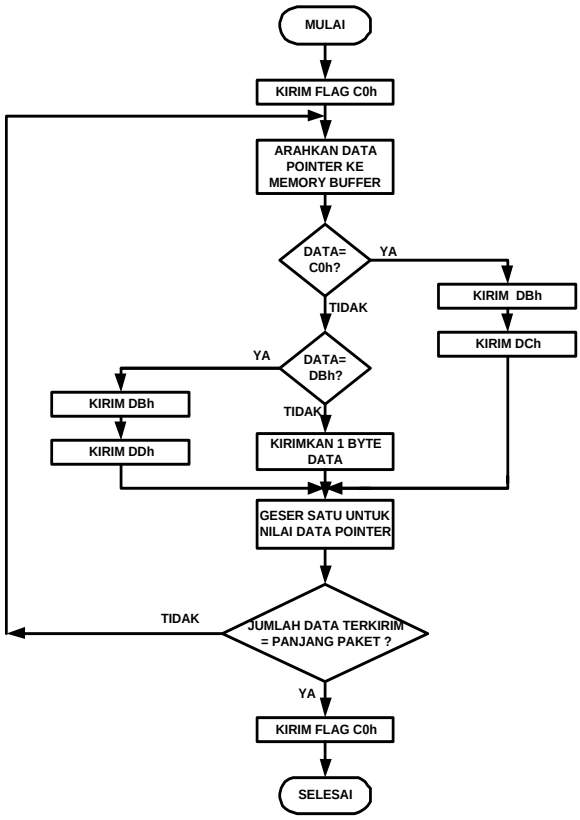
Gambar 13. Desain Antarmuka (*Interface*) Modem

Implementasi Protokol SLIP

Seperti yang sudah dijelaskan, SLIP mengirimkan paket-paket IP dengan *byte flag* khusus dan melakukan proses *byte stuffing* seperti yang telah dijelaskan di bagian dasar teori. Karena itu harus direalisasikan algoritma yang dapat mengimplementasikan aturan tersebut ke dalam mikrokontroler.



Gambar 14 Flowchart Penerima Paket SLIP



Gambar 15. Flowchart Pengirim Paket Data SLIP

Paket data yang akan dikirim dan yang baru diterima akan tersimpan di RAM.

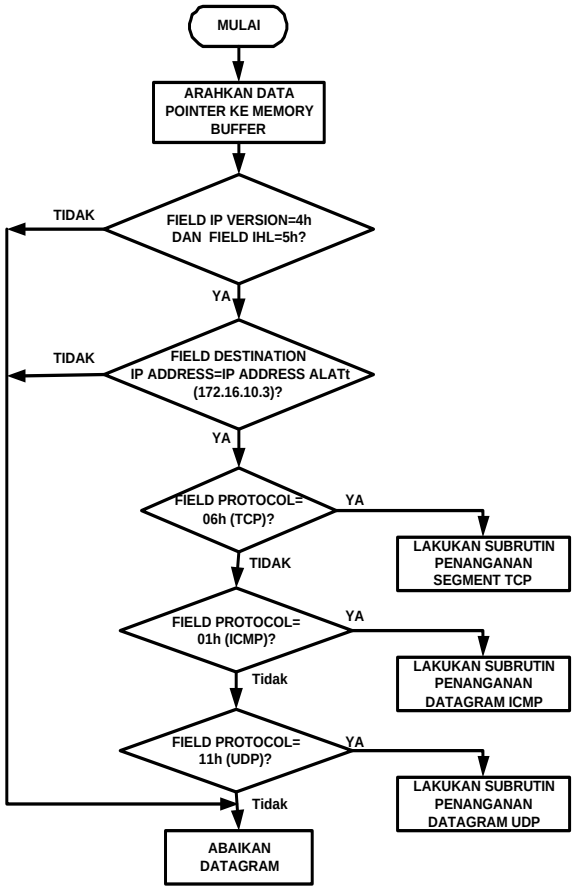
Implementasi Protokol IP

Sebuah *datagram* IP terdiri dari bagian *header* dan data. Untuk mengimplementasikan protokol IP kita harus merancang sebuah perangkat lunak yang bisa membentuk *datagram* IP serta menangani setiap *field* pada *header*.

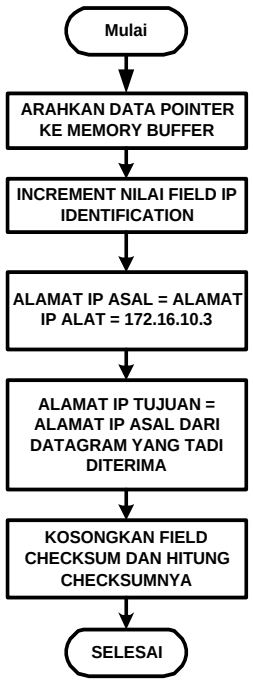
Setelah data berada di *memory buffer* oleh subrutin penerimaan paket SLIP, perangkat lunak sekarang akan menangani protokol pada *layer* di atasnya, yaitu IP. Penanganan *datagram* IP yang diterima meliputi pemeriksaan isi dari beberapa *field* pada *header* IP. Untuk mempermudah perancangan perangkat lunak, pemeriksaan masing-masing *field* dilakukan setiap kelipatan 8 bit (1 *byte*). *Field* diperiksa sesuai aturan RFC.

Untuk mengirimkan sebuah *datagram* IP, harus dilakukan enkapsulasi *header* IP terhadap data yang akan dikirim. Untuk mempermudah pembentukan *header* maka digunakan *header* IP yang sebelumnya telah diterima dengan beberapa perubahan pada *field*-nya. Pemeriksaan field dan pengubahan nilainya didasarkan pada aturan pada RFC yang kemudian diubah ke bentuk algoritma pada gambar . . . . untuk diintegrasikan ke mikrokontroler.

Setelah enkapsulasi ini selesai, *datagram* dikirimkan dengan sebelumnya diproses dengan protokol SLIP.



Gambar 16. Diagram Alir Penanganan *Datagram* IP Yang Diterima

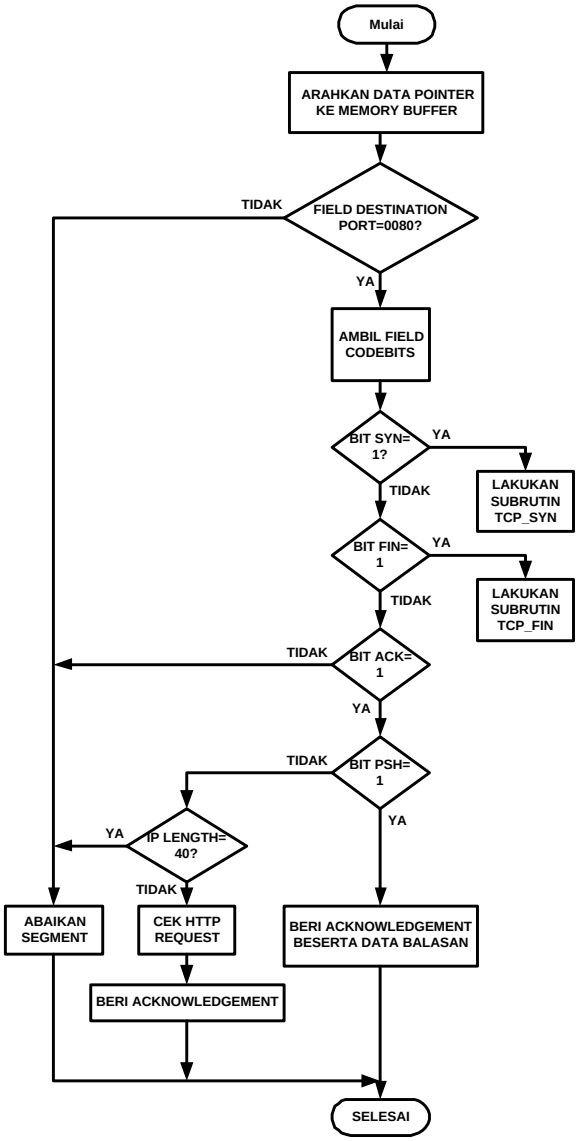


Gambar 17. Diagram Alir Enkapsulasi IP

Implementasi Protokol TCP

TCP melakukan *handshaking* dalam memulai dan mengakhiri koneksi, serta dalam pengiriman data. Setiap *segment* TCP yang diterima atau dikirim akan diikuti oleh sebuah tanda terima yang menyatakan *segment* telah diterima.

Subrutin yang mengolah *segment* TCP akan dipanggil oleh perangkat lunak dan selanjutnya diperiksa isi fieldnya sesuai standar RFC dan diubah ke bentuk algoritma pada gambar … untuk diimplementasikan ke dalam mikrokontroler, begitu pula subrutin untuk pengiriman paket data TCP yang sebelumnya dilakukan enkapsulasi.



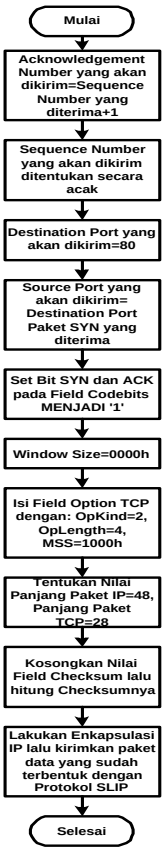
Gambar 18. Diagram Alir Penanganan *Segment* TCP Yang Diterima

Gambar 19. Diagram Alir Pembentukan dan Pengiriman *Segment* TCP

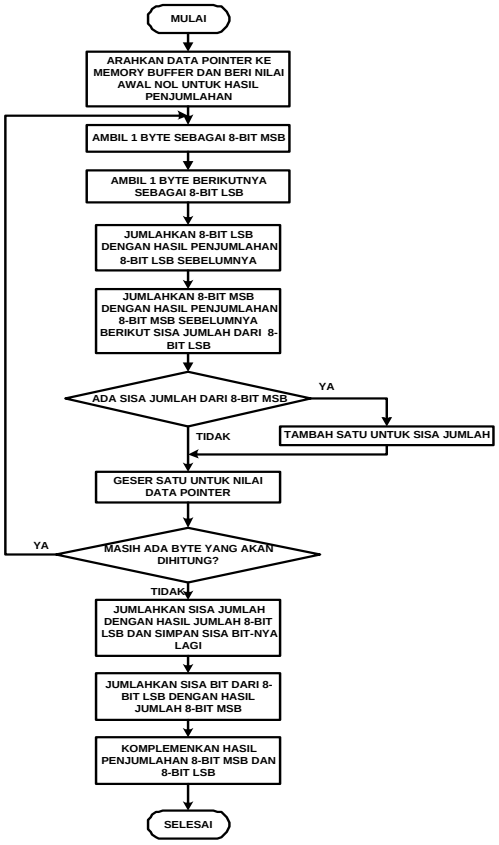
Implementasi *Three-way-handshake* Pembukaan Koneksi TCP

Jika terdeteksi bit SYN diset ‘1’ (pada aturan RFC), perangkat lunak harus dapat mengetahui bahwa *client* meminta untuk membuka koneksi TCP. Untuk itu perangkat lunak akan memanggil subrutin TCP SYN untuk melayani permintaan tersebut.

Implementasi algoritmanya dibuat seperti pada aturan yang telah dijelaskan di bagian 2, dan menghasilkan kedua flowchart berikut:



Gambar 20. Diagram Alir Subrutin TCPSYN Untuk Pembukaan Koneksi TCP



Gambar 21. Diagram Alir Dari Algoritma *Checksum*

Implementasi Algoritma *Checksum*

Seperti yang telah ditulis pada subbab-subbab sebelumnya, hampir setiap protokol pada tiap lapis melakukan perhitungan *checksum* dengan algoritma *1's complement 16 bits*. Pernyataan *16 bits checksum* menunjukkan bahwa penjumlahan dilakukan setiap 16 bit data dan hasilnya juga 16 bit. Pernyataan *1's complement* menunjukkan bahwa hasil penjumlahan dikomplemenkan dengan terlebih dulu dijumlahkan dengan sisa penjumlahan (jika hasil penjumlahan yang didapat lebih besar dari 16 bit).

Kesulitan terjadi karena register-register mikrokontroler adalah 8-bit. Oleh karena itu maka dalam implementasi algoritma ini pemrosesan penjumlahan 16-bit dibagi menjadi 8-bit MSB dan 8-bit LSB.

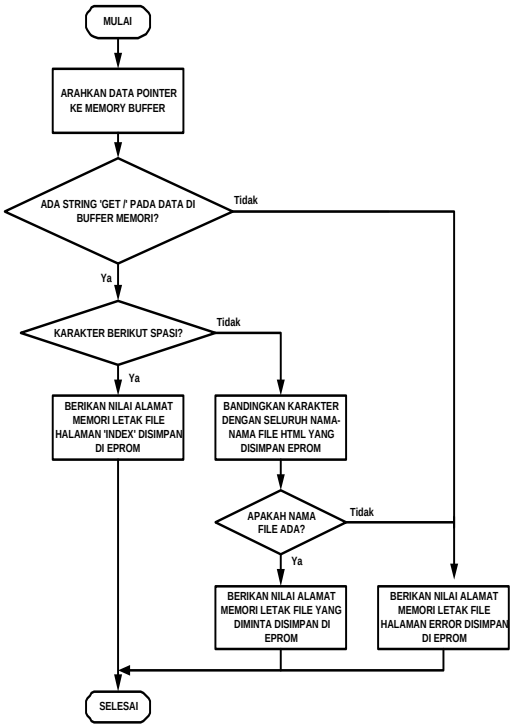
Implementasi Protokol HTTP

Setelah semua implementasi protokol TCP/IP dirancang, akan dibuat sebuah aplikasi *web server* sederhana sebagai aplikasi dari protokol TCP/IP yang diimplementasikan. Aplikasi *web server* tersebut juga dipakai untuk menunjukkan berfungsinya protokol TCP/IP di mikrokontroler. Rancangan *web server* ini cukup sederhana, di mana mikrokontroler akan mampu menjawab permintaan HTTP (*HTTP Request*) dengan mengirimkan halaman-halaman HTML pada *web browser*.

File-file HTML tersebut disimpan dalam EPROM di alamat tertentu beserta kode-kode perintah perangkat lunak mikrokontroler. Kode-kode HTML merupakan *string* ASCII, maka selama disimpan dalam EPROM dia akan diperlakukan seperti sebuah data *string* pada kode perintah perangkat lunak mikrokontroler. Untuk lebih jelasnya, berikut adalah format salah satu file HTML yang disimpan di EPROM pada alamat 1000h:

```
ORG 01000H
htm:   db 'HTTP/1.0 200 ',CR,LF
       db 'Content-Type:
text/html',CR,LF,CR,LF
       db '<html><body
bgcolor=#FFFFCC><center>'
       db '<font face=Arial size=6>'
       db 'Embedded Web server Via Modem<br>'
       db 'Berbasiskan Mikrokontroler MCS-
51<br> <br>'
       db '<font face=Arial size=3>'
       db 'Oleh :<br>Donny Kurnia
Sutantyo<br>6198062<br> <br>'
       db 'Fakultas Teknik Elektro <br>'
       db 'Universitas Kristen Satya Wacana
Salatiga<br> <br>'
        db 'Goto :<br><a
        href=1.html>Halaman1</a> | <a
        href=2.html>Halaman2</a>
        | <a
        href=3.html>Halaman3</a><br> <br
        >'
        db
'</font></center></body></html>',0,'\'
```

Bisa dilihat di atas, penulisan sebuah file HTML didahului dengan *HTTP Response*-nya, diikuti sebuah karakter *Carriage Return* (CR) dan sebuah karakter *Line Feed* (LF) serta diikuti pula dengan parameter *HTTP Response* yang lain. Setelah itu barulah data-data file HTML dituliskan. Untuk menandai akhir sebuah file, dipakai karakter "\".



Gambar 22. Diagram Alir Proses Pencarian File

HTML oleh Mikrokontroler

Metoda yang dipakai oleh *web server* yang dirancang untuk mengolah HTTP *request* ialah mencari *string* ASCII "GET /" dengan menggunakan pencarian pada *segment HTTP request* yang diterima. *String* ASCII ini merupakan format *simple request HTTP* versi 1.0. *String* "GET /" biasanya diikuti dengan nama file yang diminta, sebagai contohnya "GET /skripsi.html". Pada perangkat lunak *web server* yang dibuat, pencarian dilakukan per *byte* pada *field* data TCP di *memory buffer* dan dibandingkan dengan karakter-karakter *HTTP Request* yang dicari.

Prosedur pencarian dilakukan secara bertahap. Yang pertama adalah mencari *string* "GET /", kemudian jika tahap pertama lolos berikutnya dicari nama file. Jika karakter pertama dari nama file yang dicari adalah spasi, berarti file yang diminta adalah file indeks dari *web page*. Jika karakter pertama dari nama file yang dicari bukan spasi, berarti karakter tersebut beserta urutan karakter yang menyertainya adalah nama file yang diminta. Setelah file yang diinginkan *client* ditemukan, file HTML akan dipindahkan ke *memory buffer* untuk diproses oleh subrutin protokol TCP, IP, dan SLIP. Untuk halaman yang dibuka melalui *hyperlink*, metoda yang digunakan sama karena sebenarnya sebuah *hyperlink* yang di-klik oleh user akan menghasilkan HTTP Request yang sama dengan yang dijelaskan di atas.

Pengujian dan Analisa

Untuk menguji bahwa sistem yang dibuat sesuai dengan kriteria keberhasilan yang telah dicantumkan pada usulan tugas akhir, maka dilakukan dua tahap pengujian yang dianggap cukup mewakili, yaitu :

1. Kemampuan sistem dan implementasi yang dibuat untuk menjawab terhadap permintaan/ panggilan program diagnostik "PING" (ICMP Echo).
2. *Web server* yang diimplementasikan harus mampu melayani atau menjawab *HTTP request* dari sebuah *web browser*, serta dapat digunakan untuk keperluan pengendalian dan pemantauan jarak jauh.

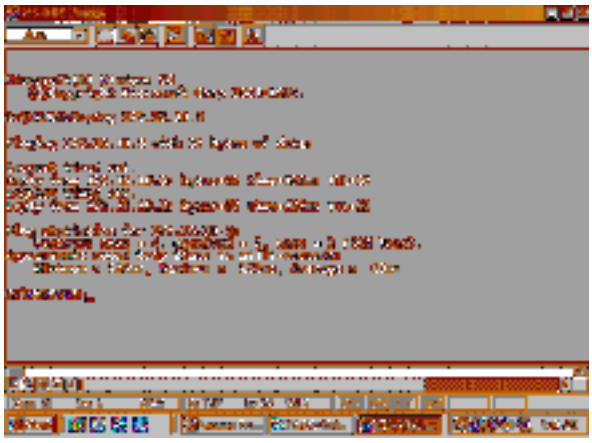
Pengujian Permintaan Diagnostik "PING"

Diagnostik "PING" di sini digunakan untuk menguji apakah alat benar-benar terkoneksi di jaringan dan untuk menguji kualitas transmisi yang digunakan.

Program diagnostik "PING" bekerja dengan menggunakan protokol ICMP. Sedangkan protokol ICMP sendiri membutuhkan protokol IP serta SLIP untuk membawanya sampai ke tempat tujuan. Protokol ICMP di sini diaplikasikan untuk proses ICMP *echo*. Setiap permintaan (ICMP *echo request*) dari komputer (PC) harus dijawab oleh alat yang dibuat dengan menggunakan ICMP *echo reply* beserta muatan data yang diterima jika ada.

Dalam pengujian digunakan penguji berupa sebuah komputer (PC) dengan alamat IP 172.16.10.2 dan alat yang akan diuji dengan alamat IP 172.16.10.3. Pada masing-masing sistem terpasang modem eksternal jenis V90 dan diantara keduanya dihubungkan lewat saluran telepon yang diemulasikan menggunakan sebuah PABX. Komputer (PC) mula-mula akan melakukan *dial-up* ke alat untuk memulai koneksi.

Pengujian dilakukan dengan memberikan perintah "PING" melalui *dos prompt* (di PC) yang ditujukan ke alamat IP alat (172.16.10.3). Perintah "PING" dengan standar muatan data 32 *bytes* dieksekusi dengan "PING 172.16.10.3".



Gambar 23. Hasil Pengujian Alat Dengan Diagnostik "PING"

Dari hasil pengujian terlihat bahwa dua dari empat buah ICMP *Echo Request* yang dihasilkan program "PING" berhasil dijawab oleh alat dengan baik. Hal tersebut menunjukkan bahwa alat sedang berada di jaringan bersama-sama dengan PC serta alat mampu menangani ICMP *Echo Request* yang diberikan kepadanya.

Dari hasil pengujian terlihat bahwa dua buah ICMP *Echo Request* gagal dijawab oleh alat. Hal tersebut disebabkan oleh derau dan distorsi pada saluran telepon yang digunakan. Oleh karena itu dalam perancangan *web server* terdapat protokol yang mempunyai fasilitas koreksi *error*.

Pengujian *Web Server*

Gambar 24. Tampilan Halaman Indeks Dari *Web Server*

Permintaan (HTTP *request*) ke *server* dilakukan dengan menggunakan perangkat lunak *web browser* (digunakan Microsoft Internet Explorer versi 6.0) pada komputer/PC client yang dialamatkan ke http://172.16.10.3 (alamat *embedded web server*/ mikrokontroler). Jika permintaan berhasil akan tampil halaman indeks pada perangkat lunak *web browser*. Untuk meminta halaman yang lain, misalnya halaman 1.html dilakukan dengan dialamatkan ke http://172.16.10.3/1.html. Maka akan ditampilkan halaman 1.html pada layar browser, jika halaman yang diminta ada. Jika tidak ada akan ditampilkan halaman *error*.

Lingkaran 1 menunjukkan bahwa alamat *browser* diarahkan ke alamat IP alat pada http://172.16.10.3. Lingkaran 2 menunjukkan bahwa *web browser* tidak bekerja dalam mode *offline*, yang apabila bekerja pada mode *offline* akan ada *icon* tambahan pada lokasi lingkaran 2. Lingkaran 3 menunjukkan bahwa *web browser* bekerja dalam zona internet. Hal ini berlaku pula untuk halaman-halaman *web* yang akan ditampilkan berikutnya.
Pada pengujian ini dapat dilihat bahwa *web server* yang dirancang dapat bekerja dengan baik.

Kesimpulan

Dari alat yang telah dirancang dan direalisasikan, ada beberapa pokok kesimpulan yang dapat ditampilkan disini bahwa:

- Protokol TCP/IP bisa diimplementasikan pada mikrokontroler, sehingga mikrokontroler dapat dikoneksikan ke jaringan komputer pada umumnya dan jaringan internet pada khususnya. Adapun protokol pembentuk TCP/IP yang telah berhasil diimplementasikan adalah SLIP, IP, TCP, ICMP, dan HTTP.
- Aplikasi-aplikasi yang ada di internet dapat diimplementasikan pada mikrokontroler. Diantaranya adalah aplikasi *web server* dan "PING" .
- Mengimplementasikan protokol jaringan komputer (dalam hal ini adalah protokol TCP/IP beserta protokol pendukungnya) pada sebuah mikrokontroler adalah hal yang sulit, karena protokol tersebut biasanya diimplementasikan pada sistem komputer yang memiliki sistem operasi dan kecepatan proses yang lebih tinggi.

Daftar Pustaka

[1] [1] Sutantyo, Donny K., *Embedded Web Server via Modem Berbasiskan Mikrokontroler MCS51 dan Aplikasinya Untuk Pengendalian Jarak Jauh Melalui Internet*, Skripsi Fakultas Teknik jurusan Teknik Elektro, UKSW, Salatiga, 2002

[2] Sutantyo, Donny K., *Implementasi Perangkat Lunak Driver Modem Pada Mikrokontroler MCS51*, Jurnal Techne edisi Oktober 2003, jurusan Teknik Elektro UKSW, Salatiga, 2003.

[3] Sutantyo, Donny K., *Diktat Kuliah Arsitektur dan Pemrograman Mikrokontroler*, Fakultas Teknik jurusan Teknik Elektro, UKSW, Salatiga, 2003
[4] Mackenzie, I. Scott, *The 8051 Microcontroler, 2nd Edition*, Prentice Hall, 1995.
[5] Tanenbaum, Andrew S., *Computer Networks, 3rd Edition*, Prentice Hall, 1996.
[6] Postel, J., *RFC 791: Internet Protocol*, RFC Standard, 1981.
[7] Postel, J., *RFC 792: Internet Control Message Protocol*, RFC Standard, 1981.
[8] Postel, J., *RFC 793: Transmission Control Protocol*, RFC standard, 1981.
[9] Romkey, J.L., *RFC 1055: Transmission of IP Datagrams Over Serial Lines: SLIP*, RFC Standard, 1988.
[10] Lee, Berners, *RFC 1945: Hypertext Transfer Protocol (HTTP)*, Request For Comment Standard, 1996.