BAB IV

PENGUJIAN SISTEM

4.1 Pengujian Terhadap *Software*

Pengujian terhadap sistem dilakukan di laboratorium jaringan komputer, Universitas Islam Indonesia. Agar sistem dapat berjalan dan dapat diakses melalui *browser client* yang berada pada tempat yang jauh, maka diperlukan sebuah *ip public*. *Ip (Internet Protocol)* ini diperoleh dari ISP yang menawarkan akses *internet*. Dalam uji coba ini penulis mendapatkan *ip public* yang masih tersedia di laboratorium tersebut yaitu 202.168.20.13.

4.1.1 *Host Computer*

Sebelum menjalankan pengujian terhadap sistem, *software* yang berada pada *host computer* harus dijalankan terlebih dahulu. Caranya adalah :

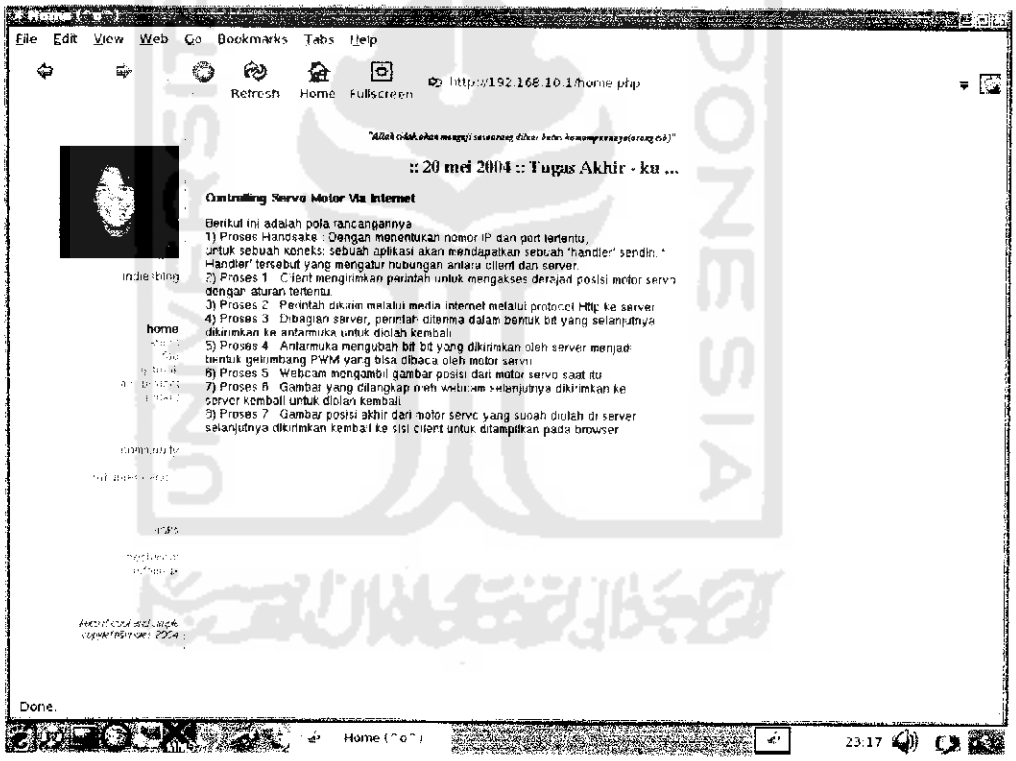
1. Jalankan Apache (pada linux mandrake apache *web server* sudah langsung berjalan saat komputer dihidupkan), tetapi jika belum berjalan, dapat mengetikkan : **shell_prompt$:apachectl start.**
2. Buka akses untuk menggunakan *serial port* yang akan digunakan, dalam percobaan ini adalah **/dev/ttyS0,** dengan mengetikkan: **shell_prompt$:chmod 777 /dev/ttyS0.**
3. Jalankan *camserv* dengan mengetikkan : **shell_prompt$: camserv.** Untuk menggambarkan koneksi yang terjadi setelah *camserv* diaktifkan adalah sebagai berikut :

2004-07-20 16:31:37 [mainloop] Accepted new socket : 192.168.10.2

2004-07-20 16:33:13 [mainloop] Closing socket : 192.168.10.2

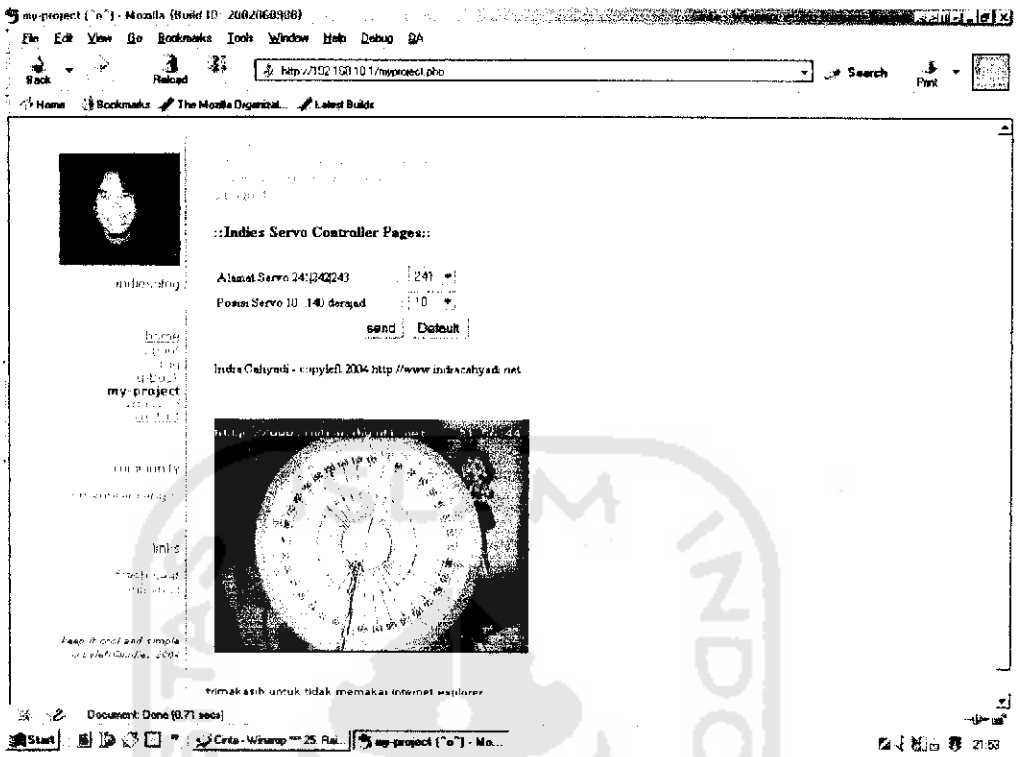
4.1.2 *Client Computer*

Setelah mendapatkan akses melalui internet, maka barulah komputer *client* dapat mengakses alat yang berada *host computer / server*. Cara untuk mengendalikan alat ini dari browser sangat mudah, cukup dengan *click* pada *browser* favorit anda, dan ketikkan : http://remote_host/lokasi index.php. Maka akan mendapatkan tampilan sebagai berikut.



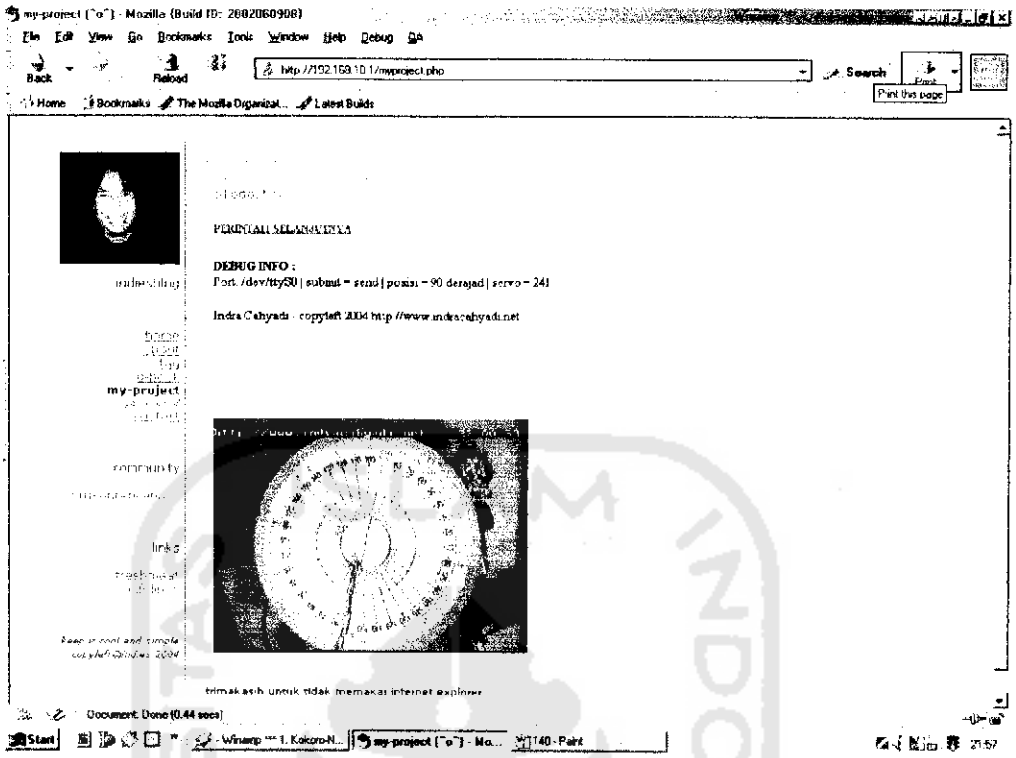
Gambar 4.1 Halaman home.php

Selanjutnya klik pada bagian ***myproject***, dan masukkan nama sandi dan kata kunci (jika belum punya, maka bisa dibuat terlebih dahulu), maka anda akan mendapatkan tampilan sebagai berikut :



Gambar 4.2 Halaman inisialisasi

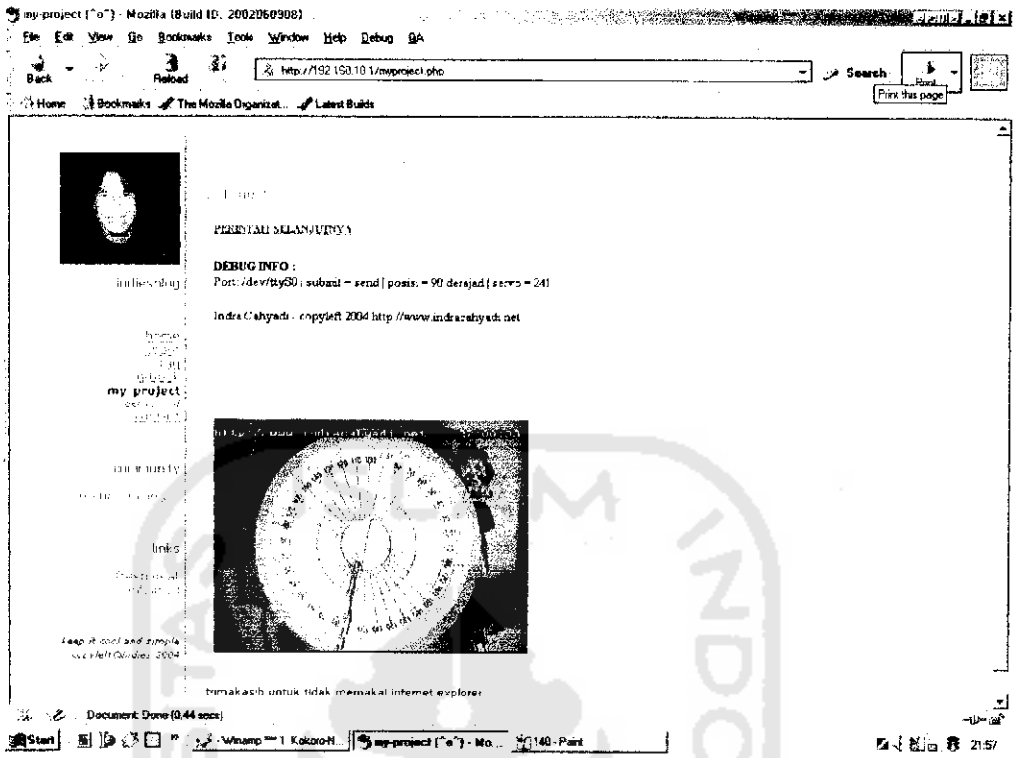
Masukkan alamat dari servo yang akan dikendalikan (mikrokontroler PIC16F84A dapat dihubungkan dengan 12 motor servo sekaligus namun untuk tugas akhir ini penulis hanya menggunakan 1 motor servo saja, sehingga untuk *default* alamatnya adalah 241). Kemudian masukkan posisi/derajat yang dikehendaki,lalu tekan *send*. Selanjutnya data tersebut akan dikirimkan ke motor servo. Data yang anda kirimkan akan ditampilkan pada halaman *debug info* (data dari *port serial* yang dipergunakan serta alamat dan posisi servo yang dikendaki) seperti yang ditampilkan pada gambar 4.3, setelah itu maka dapat dipastikan motor servo akan bergerak sesuai dengan perintah yang diberikan.



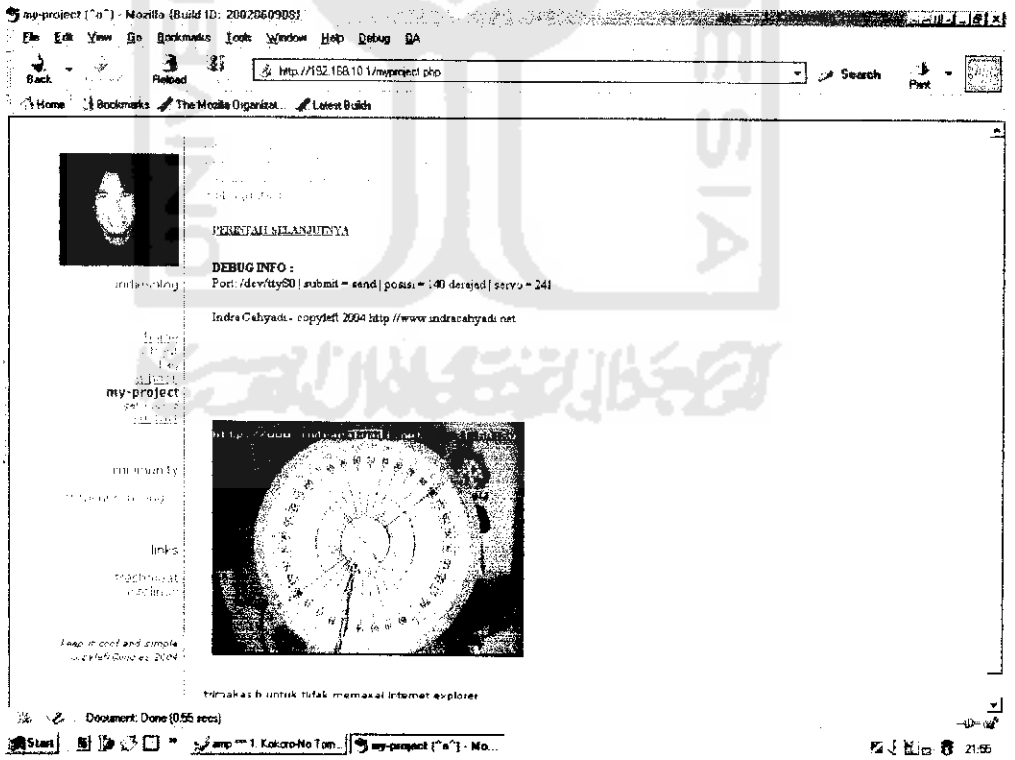
Gambar 4.3 Hasil *debug info*

Diperlukan beberapa detik untuk menampilkan gambar dan hasil dari pengiriman kode perintah, hal ini tergantung dari koneksi internet saat itu. Jika ingin memasukkan perintah yang lain, *click* pada perintah selanjutnya, maka anda akan dibawa kembali ke halaman inisialisasi.

Berikut adalah 2 gambar *screen shoot* dari browser (mozilla *under* windows) yang digunakan untuk melihat pergerakan dari motor servo.



Gambar 4.4 Pergerakan servo sebesar 90 derajat



Gambar 4.5 Pergerakan Servo sebesar 140 derajat.

4.2 Pengujian Terhadap *Webcam*

Untuk melakukan pengujian terhadap gambar yang dikirimkan dari *webcam*, maka *camserv* harus dijalankan terkebih dahulu. Caranya ketikkan perintah berikut pada shell prompt :

shell_prompt$: camserv.

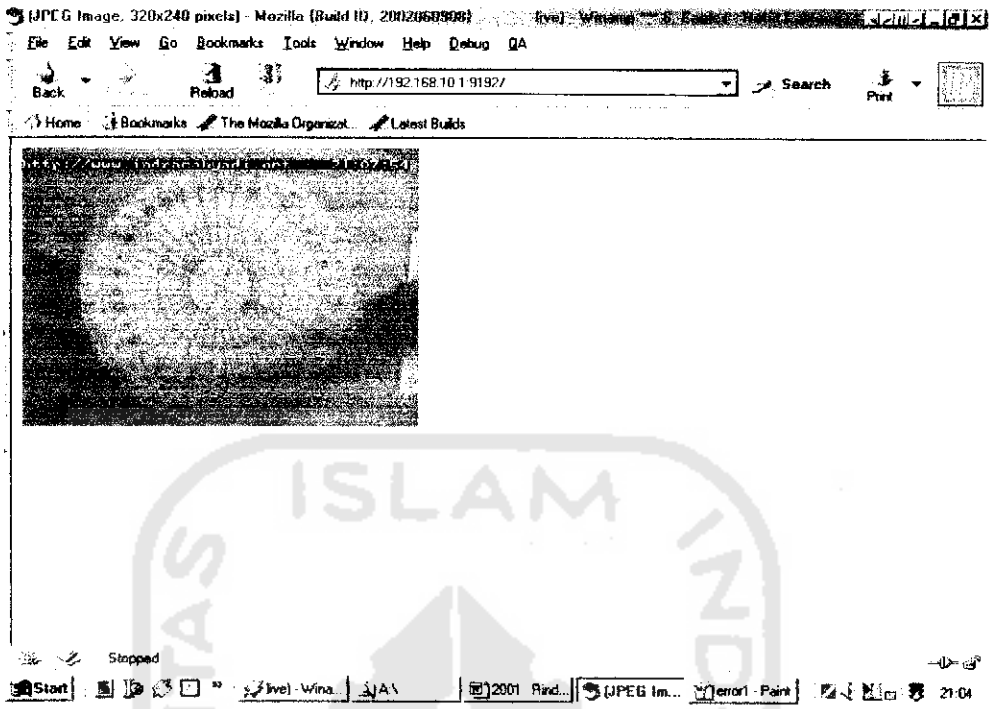
Maka akan diperoleh hasil seperti berikut ini :

```
[indra@indienet indra]$ camserv
camserv v0.5.1 - by Jon Travis (jtravis@p00p.org)
Syntax: camserv <cfg file>
Will try /home/indra/.camserv and
/usr/local/share/camserv/camserv.cfg
2004-07-27 21:45:58 [main] Trying to read config file
"/home/indra/.camserv":
2004-07-27 21:45:58 [main] Error reading config
"/home/indra/.camserv": No such file or directory
2004-07-27 21:45:58 [main] Trying to read config file
"/usr/local/share/camserv/camserv.cfg":
2004-07-27 21:45:58 [main] Success reading config
"/usr/local/share/camserv/camserv.cfg"
2004-07-27 21:45:58 [camconfig] Using default of "2359296" for
[main]:shm_alloc
2004-07-27 21:45:58 [mainloop] Setup signals
2004-07-27 21:45:58 [filter] Loading filter [time_stamp]
2004-07-27 21:45:58 [filter] Loading filter [jpg_filter]
2004-07-27 21:45:58 [video_init] image width: default:320 max:924
min:48 config:320 used:320

2004-07-27 21:45:58 [video_init] image height: default:240 max:576
min:32 config:240 used:240

2004-07-27 21:45:58 [camconfig] Using default of "0" for
[video_v4l_bttv]:frequency
```

Setelah camserv dijalankan barulah gambar *streaming* dari kamera dapat dikirimkan. Untuk dapat mangambil gambar dari *host* komputer, maka ketikkan perintah berikut pada *browser client* :

http://alamat_host_komputer:9192

9192 merupakan *port* yang digunakan *camsev* untuk proses *streaming* gambar. Pengujian ini dilakukan dengan meggunakan LAN, dengan panjang kabel 15 meter. Dan hasil yang diperoleh adalah seperti gambar dibawah ini :



Gambar 4.6 Hasil *streaming* gambar pada Mozilla

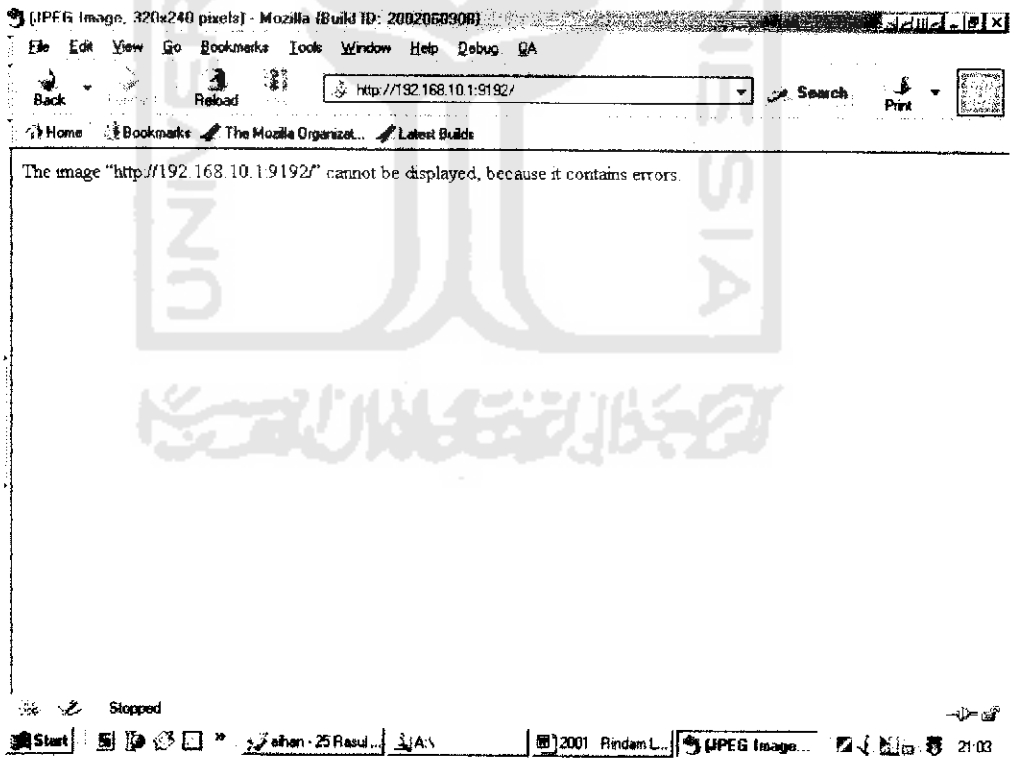
Gambar 4.6 diambil dengan menggunakan *browser mozilla* untuk windows. Untuk browser-browser seperti *mozilla, opera, galeon* tidak terjadi masalah dalam proses *streaming* gambar. Namun untuk *Internet Explorer* dari windows tidak akan dapat dilakukan proses *streaming*, karena browser tersebut memang tidak mendukung untuk *multiple JPEG*. Untuk itu perlu sebuah *script* yang dapat menjalankan proses *streaming* tersebut pada Internet Explorer. Secara garis besar prinsip kerja dari *script* tersebut adalah ia akan melakukan peng-*update*-an gambar setiap 1 detik dari server, sehingga pada *shell prompt* akan terlihat perintah *accept new socket* dari komputer *client* yang dikirimkan secara berurutan setiap 1 detik. Berikut adalah *report* dari camserv untuk *client* yang menggunakan browser Internet Eksplorer :

```
2004-07-27 21:31:48 [mainloop] Accepted new socket: 192.168.10.2
```

```
2004-07-27 21:31:49 [mainloop] Accepted new socket: 192.168.10.2
2004-07-27 21:31:50 [mainloop] Accepted new socket: 192.168.10.2
2004-07-27 21:31:51 [mainloop] Accepted new socket: 192.168.10.2
2004-07-27 21:31:53 [mainloop] Accepted new socket: 192.168.10.2
2004-07-27 21:31:54 [mainloop] Accepted new socket: 192.168.10.2
2004-07-27 21:31:55 [mainloop] Accepted new socket: 192.168.10.2
2004-07-27 21:31:56 [mainloop] Accepted new socket: 192.168.10.2
2004-07-27 21:31:57 [mainloop] Accepted new socket: 192.168.10.2
```

Script dari program ini dapat dilihat pada lampiran.

Namun demikian tidak sepenuhnya gambar dapat dikirimkan secara sempurna dari *host* ke *client* komputer. Terkadang juga terjadi *error* dalam pengiriman gambar tersebut. Hal ini dapat terjadi karena lebar band yang diperlukan untuk proses *streaming* tidak cukup (untuk kecepatan 250 kbps dapat meng-*upload* 3 fps untuk 11 kbyte 320x240 *jpeg image*), kabel yang terlipat atau terlalu panjang, atau dapat juga disebabkan karena *hardware* yang tidak bagus. Berikut adalah pesan *error* yang ditampilkan pada browser client :



Gambar 4.7 Tampilan error pada *streaming* gambar

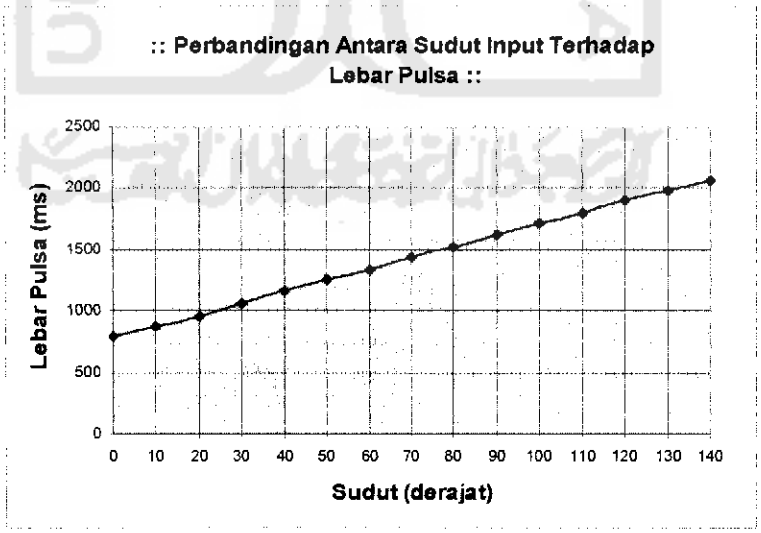
4.3 Pengujian terhadap pergerakan Servo

Berikut hasil yang diperoleh dari pergerakan motor servo terhadap lebar pulsa dan frekuensi :

Tabel 4.1 hasil pengujian pergerakan motor servo

No	Besar Sudut (derajat)	Lebar Pulsa (ms)	Frekuensi (Hz)
1	0	0,800	37,88
2	10	0,880	37,88
3	20	0,960	37,88
4	30	1,060	37,88
5	40	1,160	37,88
6	50	1,260	37,88
7	60	1,340	37,88
8	70	1,440	37,88
9	80	1,520	37,88
10	90	1,620	37,88
11	100	1,720	37,88
12	110	1,800	37,88
13	120	1,900	37,88
14	130	1,980	37,88
15	140	2,060	37,88

Dari tabel diatas maka dapat digambarkan grafik hubungan antara besar sudut dan lebar pulsa seperti berikut ini



Gambar 4.8 Perbandingan antara sudut input terhadap lebar pulsa

Dari table 4.1 dapat dilihat, bahwa servo motor hanya dapat bergerak maksimal sejauh 140 derajat. Hal ini dapat disebabkan oleh beberapa faktor, yaitu dari motor servo itu sendiri dan dari *software.* Seperti yang telah dijelaskan sebelumnya, motor servo memiliki 3 komponen utama, yaitu *control board,* potensiometer dan motor. Besarnya tahanan yang diberikan potensiometer digunakan oleh *control board* untuk menghasilkan *error signal,* ketika posisi yang dikehendaki tidak sama dengan posisi pada saat itu. Jika dikirimkan perintah agar servo bergerak ke posisi 90 derajat, sementara posisi saat itu adalah 45 derajat, maka *error signal* tersebut akan menyebabkan motor menggerakkan potensiometer melalui *gear* hingga *error signal* menjadi nol (posisi mencapai 90 derajat). Artinya jika *control board* tidak dapat bekerja dengan baik, maka posisi yang diinginkan tidak akan tercapai.

Untuk faktor penyebab yang kedua adalah *software. Software* yang digunakan tidak dapat mencapai *pulse range* (selisih pulsa maksimum terhadap pulsa minimum) dimana motor servo dapat bergerak maksimal. *Parallax* tidak memberikan nilai pulsa minimum dan pulsa maksimum terhadap perputaran motor *standard servo* yang dikeluarkannya, sehingga untuk mengetahuinya harus dilakukan dengan cara *try and error.* Sementara dari *software* sendiri hanya mampu memberikan *range* pulsa sebesar 1242 us.

Untuk mendapatkan angka tersebut, maka ada beberapa komponen yang harus diketahui terlebih dahulu, yaitu *clock* dan kecepatan *baudrate.* Clock yang dihasilkan oleh osilator masuk ke mikrokontroler melalui pin OSC1 dimana rangkaian internal dari mikrokontroler akan membagi clock tersebut menjadi empat clock yang tidak saling *overlap* (balapan). Satu siklus instruksi adalah

waktu yang diperlukan untuk menyelesaikan proses eksekusi sebuah instruksi, mulai dari mengambil instruksi, mendekode, mengeksekusi hingga menuliskan kembali pada register instruksi. Pada PICmicro, proses ini memerlukan 4 clock osilator. Hal ini berarti bahwa apabila kita menggunakan kristal osilator 20 MHz, PIC akan bekerja pada kecepatan 5 MHz. Siklus instruksinya adalah 0,2 us.

Sementara itu kecepatan *baudrate* yang digunakan penulis adalah sebesar 38400 bps (hal ini disesuaikan dengan kecapatan osilator yang digunakan, jika menggunakan 20 MHz maka *baudrate* yang digunakan adalah 38400, jika menggunakan 4 MHz maka *baudrate* yang digunakan adalah 9600). Dari nilai *baudrate* yang dipakai inilah dapat ditentukan besarnya nilai *timer period*, yang besarnya adalah 1/*baudrate*. Sehingga didapat 1/38400 = 26 us. Dari nilai-nilai yang ada diatas, maka dapat ditentukan besarnya nilai *pulse range* (untuk lebih jelasnya dapat dilihat pada *servo controller.asm code*) yaitu :

End of period count = (pulse range x timer period scale) / timer period

Pulse range = (end of period count x timer period) / timer period scale

Pulse range = 239 x 26 / 5

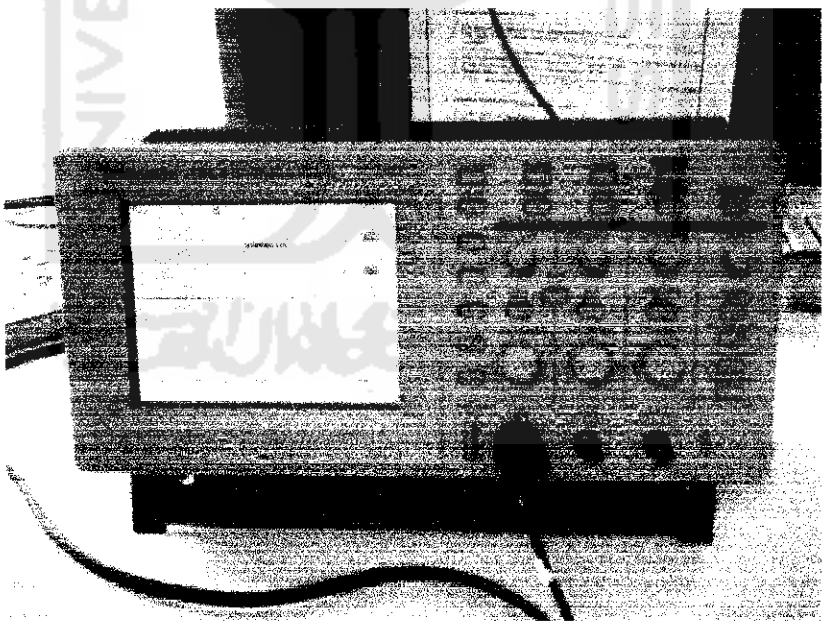
Pulse range = 1242 us

Pulse range = 1,2 ms

Berdasarkan nilai *pulse range* tersebut, kita dapat menentukan besarnya nilai pulsa minimum dan pulsa maksimum dari motor servo yang digunakan. Secara umum motor servo dapat menerima besarnya lebar pulsa positip dari 1 sampai 2 ms (untuk gambaran mengenai perubahan pergerakan motor servo terhadap pulsa yang diberikan dapat dilihat pada gambar 2.3), namun nilai tersebut sangat bervariasi untuk semua motor servo. Dalam percobaan ini penulis

menggunakan lebar pulsa positip sebesar 0,8 sampai 2 ms (selisih antara pulsa maksimum terhadap pulsa minimum ini tidak boleh lebih dari 1,2 ms, meskipun kita memberikan lebar pulsa untuk pergerakan maksimumnya sebesar 3 ms, sistem akan tetap membaca lebar pulsa yang diperbolehkan hanya 1,2 ms, jadi lebar pulsa yang 1 ms-nya lagi tidak akan terbaca sebagai sebuah perintah).

Selanjutnya pencuplikan dilakukan sebanyak 15 kali. Nilai ini mewakili banyaknya variasi derajat yang akan digunakan untuk melihat perputaran motor servo yang diinginkan. Pencuplikan diambil untuk perputaran motor servo dari 0 derajat sampai 140 derajat. Sehingga dapat diperoleh besarnya perubahan setiap derajat yang diinginkan adalah 1,2 ms / 15 = 0,08 ms. Dari besarnya nilai perubahan tersebut maka dapat digambarkan sebuah grafik linear hubungan antara lebar pulsa terhadap derajat yang diinginkan.



Gambar 4.9 Sinyal *output* PWM Motor