Jurnal Qua Teknika, Vol.6 No.1 Maret 2016
p-ISSN: 2088-2424; e-ISSN: 2527-3892
UNISBA Blitar, Http://qua.unisbablitar.ejournal.web.id
Ahmad Yufron, 2016. Mikrokontroler At Mega 8535 Sebagai Pengendali Intensitas Lampu Pijar. *Jurnal Qua Teknika*, (2016), 6(1): 69-87

MIKROKONTROLER AT MEGA 8535 SEBAGAI PENGENDALI INTENSITAS LAMPU PIJAR

Ahmad Yufron
Jurusan Teknik Sipil, Fakultas Teknik
Universitas Islam Balitar Blitar
Jl. Majapahit no 4 Blitar

Ringkasan

Perancangan sistem pengendalian intensitas lampu Pijar berbasis mikrokontroler ATmega8535. Jenis lampu yang dikendalikan intensitasnya adalah lampu pijar yang cenderung bersifat resistif. Pemrograman mikrokontroler untuk mengendalikan intensitas lampu Pijar menggunakan bahasa Basic dengan compiler dari Bascom AVR dan PonyProg2000 sebagai downloader ke mikrokontroler.

Kata kunci*:* mikrokontroler, lampu pijar, pengendali.

1. PENDAHULUAN

Sumber daya energi Listrik yang menjadi kebutuhan manusia juga banyak yang disediakan dari pembangkit yang menggunakan minyak bumi misalnya melalui Pembangkit Listrik Tenaga Disel (PLTD), Pembengkit listrik Tenaga Nuklir (PLTN), Pembengkit listrik Tenaga Air (PLTA) Pembangkit listrik tenaga uap (PLTU) dll. Usaha untuk mengatur daya yang masuk lampu ini salah satunya adalah dengan mengatur intensitasnya. Perubahan intensitas lampu ini dikendalikan dengan banyak cara. Salah satu cara untuk mengatur intensitas lampu adalah dengan mikrokontroler yang banyak dijumpai di pasaran. Mikrokontroler sebagai piranti yang semakin berkembang memiliki banyak manfaat, dapat diprogram sesuai dengan kebutuhan. Mikrokontroler merupakan mikroprosesor kecil yang di dalamnya memiliki fungsi khusus. Mikrokontroler ini umumnya dapat diprogram melalui komputer dengan *interface* seperti COM atau LPT bahkan yang terbaru melalui USB (*Universal Serial Bus*). Mikrokontroler ini dapat digunakan sesuai dengan kebutuhan karena dapat diprogram dengan banyak bahasa. ATmega8535 merupakan salah satu mikrokontroler buatan Atmel yang memiliki banyak kegunaan. Harga mikrokontroler ini tergolong murah jika dilihat dari fasilitas yang dibawanya. Fasiltasnya cukup banyak misalnya portnya ada 4 yang dapat digunakan untuk banyak masukan atau keluaran, adanya ADC, Timer dan fasilitas lain. Keuntungan lain mikrokontroler ini adalah cara memrogramnya juga mudah karena tidak memerlukan *downloader* yang sangat merepotkan seperti mikrokontroler generasi sebelumnya karena dapat diprogram menggunakan sistem minimalnya. Dengan fasiltas-faslitas tersebut ATmega8535 cocok digunakan sebagai pengendali intensitas lampu karena fasilitas pendukungnya cukup memadai.

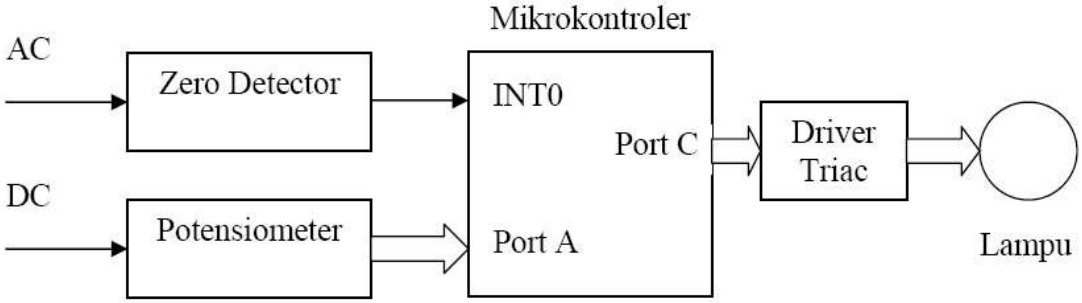
Adapun tujuan penelitian ini adalah : 1) Menghasilkan sistem kendali intensitas lampu berbasis mikrokontroler ATmega8535, 2) Menghasilkan rancang dan membuat program

pengendali intensitas lampu dengan menggunakan bahasa Basic dengan *compiler* dari Bascom AVR dan PonyProg2000 sebagai *downloader* ke mikrokontroler, dan 3) Memahami prinsip kerja pengendalian instensitas lampu dengan menggunakan mikrokontroler.

2. PERANCANGAN SISTEM PENGENDALI

2.1 Perancangan Perangkat Keras

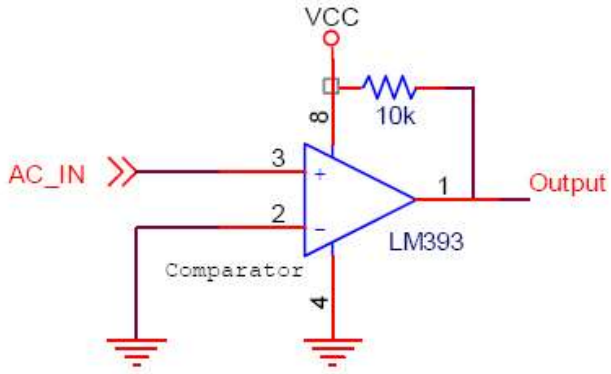
Perancangan perangkat keras yang dibutuhkan meliputi *zero detector*, potensiometer pembagi tegangan sebagai input masukan, sistem minimal mikrokontroler ATmega8535, *driver* Triac beserta Triacnya dan output berupa lampu pijar. Masukan mikrokontroler dari *zero detector* dihubungkan ke kaki INT0 sedangkan masukan tegangan dari potensiometer dihubungkan ke Port A. Output berupa pulsa *low* dikeluarkan melalui Port C. Secara diagram, pengendali intensitas lampu terlihat seperti gambar di bawah ini.



Gambar 3.1. Blok Diagram sistem kendali intensitas lampu

2.1.1 *Zero Detector*

Bagian ini berfungsi untuk mendeteksi sinyal AC saat mengalami tegangan nol volt (saat *zero*). Proses pendeteksiannya dengan menggunakan komparator. Komparator yang digunakan dalam penelitian ini adalah LM393. Komparator ini termasuk jenis *Low Power Low Offset Voltage Dual Comparator*. Outputnya menggunakan *open collector*. Rangkaian komparator ini dirangkai berdasarkan rangkaian *datasheet*. Rangkaiannya terlihat di Gambar 3.2.



Gambar 3.2. Rangkaian *Zero Detector*

Prinsip kerjanya dengan membandingkan tegangan AC terhadap tegangan referensi yang dihubungkan ke *ground* (0 volt). Masukan AC berasal dari trafo *step down* yang sudah diturunkan lagi dengan resistor pembagi tegangan. Saat fase positif komparator akan

menghasilkan output *high* (Vcc) dan saat fase negatif komparator akan menghasilkan output *low* (0 volt). Jadi outputnya adalah gelombang kotak dengan frekuensi sesuai dengan frekuensi AC-nya yaitu 50 Hz. Pada saat *Positive Going Transition* (PGT) atau *Negative Going Transition* (NGT) inilah saat terjadi *zero*. PGT atau NGT inilah yang dibaca oleh mikrokontroler sebagai *zero*. Dalam penelitian ini penulis menggunakan PGT dan dibaca sebagai interupsi 0 (INT0) pada ATmega8535.

2.1.2 Potesiometer

Potensiometer ini berfungsi untuk membagi tegangan. Input tegangan potensiometer ini dari 0 sampai 5 volt. Output tegangan potensiometer ini nantinnya akan dibaca oleh ADC yang sudah terintegrasi dalam ATmega8535 melalui port A. Masukan ADC ada 8 channel tetapi dalam penelitian ini yang digunakan hanya 4 sehingga potensiometernya juga hanya butuh 4. Rangkaian keempat potensiometer sebagai input ADC terlihat seperti di bawah ini.

Gambar 3.3. Potensiometer Pembagi Tegangan

Perlu diingat bahwa tegangan output potensiometer ini tidak berfungsi untuk menyediakan daya, namun hanya dibaca tegangannya. Tegangan output ini hanya sebagai data untuk mengatur sudut picu pada pentrigeran Triac. Data inilah yang nantinya diolah oleh mikrokontroler.

2.1.3 Mikrokontroler ATmega8535

ATmega8535 berfungsi sebagai pemroses data masukan potensiometer dan *zero detector*. Data berupa tegangan diubah menjadi data digital oleh ADC dalam ATmega8535. Operasi ADC menggunakan *single ended conversion* yang berarti tegangan masukan merupakan tegangan terhadap 0 volt. Resolusi ADC ini adalah 10 bit sehingga total level yang dibangkitkan adalah:

Total level = 2^{10} = 1024 level

Data 10 bit hasil konversi ADC yang selanjutnya disebut NKADC yaitu:

$$NKADC = \frac{Vin.1024}{Vref} \quad \text{.....(3.1)}$$

NKADC disimpan dalam register 8 bit ADCL dan ADCH. Register ADCH hanya dipakai 2 bit pada LSB-nya pada kondisi *default*.

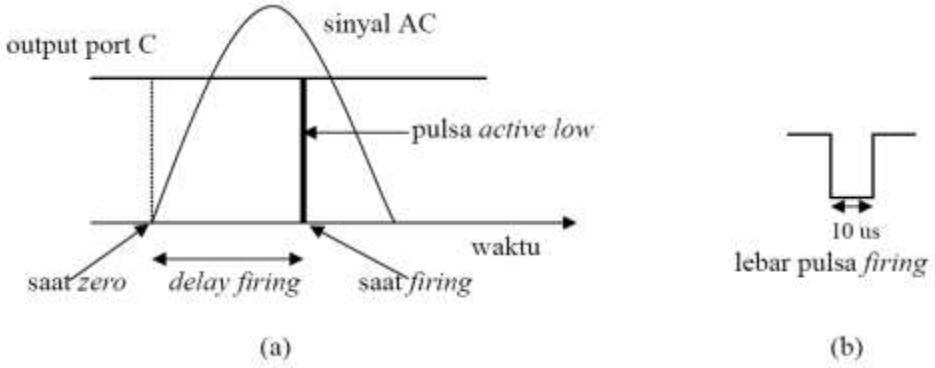
Tegangan referensi sama dengan Vcc yaitu 5 volt. Nilai tegangan masukan 0 sampai 5 volt, maka ADC ini memilki resolusi sebagai berikut.

Resolusi ADC = 5/1024 volt/level = 4,8828125 mV/level

Besarnya NKADC ini nantinya digunakan sebagai acuan *delay* pentrigeran. Pada penelitian ini basis waktu terkecil yang digunakan adalah 10 mikrodetik. Basis waktu terkecil dipilih 10 mikrodetik untuk memudahkan perancangan. Selain itu kemampuan membuat *delay* minimum dari program Basic adalah 5 mikrodetik itupun tidak dapat variabel. Sehingga untuk membuat *delay* berupa variabel dibuat subrutin tersendiri untuk menghasilkan *delay* yang dapat diubah-ubah. Subrutin untuk membuat *delay* dengan basis 10 mikrodetik dapat dilihat di lampiran A. Besarnya *delay* pemicuan triac ditentukan dengan persamaan berikut.

Delay = NKADC x 10 us .....(3.2)

Berdasar Persamaan 3.2, *delay* terkecil adalah 0 mikrodetik dan terbesar 10240 mikrodetik. *Delay* terbesar yaitu 10240 us = 10,24 ms telah melebihi setengah periode gelombang AC PLN yaitu ½ x 20 = 10 ms. Sistem ini dirancang hanya mengendalikan *firing* untuk setengah gelombang AC, maka dalam implementasinya perlu adanya pembatasan NKADC-nya melalui program. Besarnya *delay firing* untuk setengah gelombang AC ditunjukkan Gambar 3.4a.

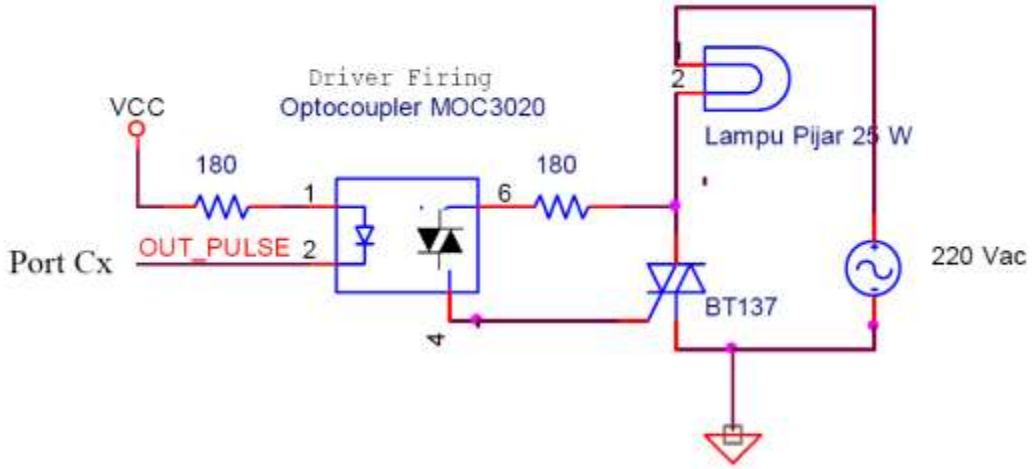


Gambar 3.4. *Delay firing* dan lebar pulsa *firing*

Pada penelitian ini lebar pulsa *firing* yang digunakan adalah 10 mikrodetik. Lebar pulsa minimum *firing* berdasar *datasheet* adalah 2 mikrodetik. Lebar pulsa dipilih 10 mikrodetik untuk memudahkan perancangan walaupun minimumnya dapat mencapai 5 mikrodetik. Gambar pulsa *low* untuk *firing* ditunjukkan Gambar 3.4b

2.1.4 *Driver* Triac

Driver triac yang digunakan adalah tipe MOC3020. *Driver* ini termasuk jenis optocoupler sehingga relatif aman jika terjadi ketidaknormalan pada bagian beban. Rangkaian *driver* ini ditunjukkan Gambar 3.5.



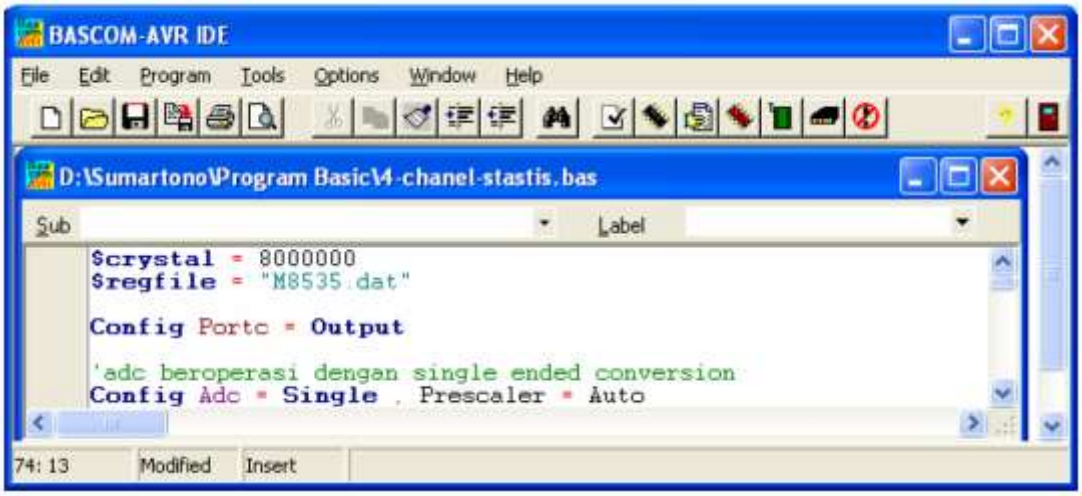
Gambar 3.5. *Driver Firing* Triac MOC3020

Pengaktifan Triac melalui *driver* ini adalah dengan pulsa *low* yang berasal dari port C (C0 – C3). Pada saat ada pulsa *low* (0 volt) pada kaki 2 maka akan terjadi beda potensial antara kaki 1 dan 2 sehingga arus mengalir dan dioda dalam MOC3020 memancarkan cahaya sehingga *bilateral switch* ON, arus mengalir dari kaki 6 ke 4 akan mengaktifkan Triac menjadi ON sehingga Triac dapat mengalirkan arus. Fungsi resistor 180 ohm adalah untuk membatasi arus yang mengalir melalui dioda. Untuk bagian tegangan rendah (sebelah kiri) arus yang mengalir maksimum 30 mA. Apabila hambatan dioda dalam MOC3020 diabaikan maka arus yang mengalir dapat dihitung sebagai berikut.
I = 5/180 = 0,02778 A = 27,78 mA (masih di bawah nilai yang diizinkan)
Arus 27,78 mA bagi mikrokontroler sudah dianggap besar, sehingga pengaktifan MOC3020 dengan *active low* agar mikrokontroler tidak melakukan *sourcing*.

2.2 Perancangan Perangkat Lunak

Pada penelitian ini penulis menggunakan bahasa Basic. Perangkat lunak yang digunakan sebagai teks editor sekaligus *compiler* adalah Bascom AVR versi 1.11.7.7 buatan MCS Electronic. Bascom AVR menggunakan file dat yang berisi informasi tentang mikrokontroler (bertipe AVR) yang akan diprogram seperti alamat register internal dan alamat interupsi. Komentar program dalam Bascom AVR ditulis di sebelah kanan tanda petik (‘). Pada program ini terdapat *short cut* untuk menu tersebut yaitu menekan tombol langsung di *keyboard*. Untuk menu tersebut seperti **New** (Ctrl+N), **Open** (Ctrl+O), **Save** (Ctrl+S), **Compile** (F7) dan **Simulate** (F2). Dengan adanya *short cut* ini proses pemrograman dapat menjadi lebih cepat. Pada program tersebut terdapat beberapa menu yang biasa digunakan untuk membuat suatu program. Menu yang biasa digunakan diantaranya adalah **New**, **Open** dan **Save** yang terlatak pada menu **File**. Untuk proes kompilasi yang biasa digunakan adalah **Compile** dan **Simulate** yang terletak pada menu **Program**. Selain itu untuk terdapat menu **Help** untuk membantu dalam pembuatan program. Untuk membuat program baru maka digunakan menu **New**. Sebelum dikompilasi program harus disimpan dengan menu **Save**. Proses kompilasi dengan menu **Compile**. Program yang sudah dikompilasi dapat disimulasi dengan menu **Simulate**.

Untuk membuka dokumen yang sudah ada digunakan menu **Open**. Tampilan program Bascom AVR ditunjukkan Gambar 3.6



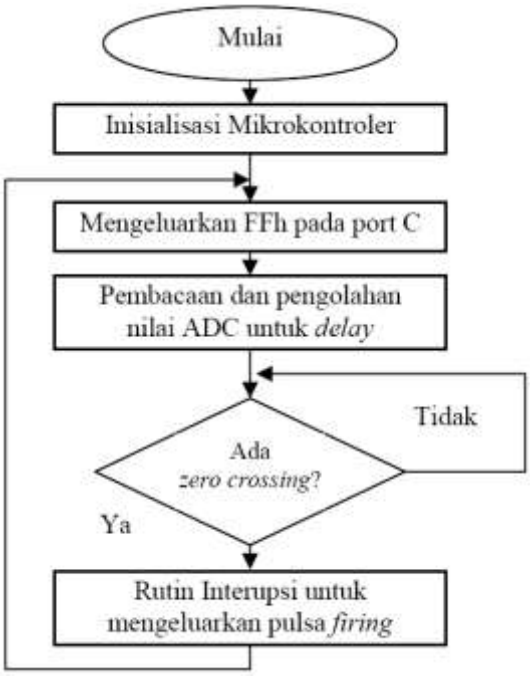
Gambar 3.6. Tampilan Bascom AVR

Perangkat lunak yang digunakan sebagai *downloader* dari komputer ke mikrokontrolernya adalah PonyProg2000 versi 2.06c Beta buatan Claudio Lanconelli. File yang di-*download* berupa file hexa (ekstensi *.hex) hasil *compile* Bascom AVR. Program ini juga terdapat beberapa menu yang biasa digunakan untuk membuat suatu program. Program ini bukan teks editor sehingga menu yang biasa digunakan adalah **Open Program** dan **Write Program Memory**. **Open Program** terlatak pada menu **File** sedangkan **Write Program Memory** terletak pada menu **Command**. Menu **Open Program** digunakan untuk membuka file hasil kompilasi Bascom AVR yang berekstensi *.hex sedangkan menu **Write Program Memory** digunakan untuk menulis program berekstensi *.hex pada flash mikrokontroler. Tampilan program PonyProg2000 terlihat di Gambar 3.7.

Gambar 3.7. Tampilan PonyProg2000

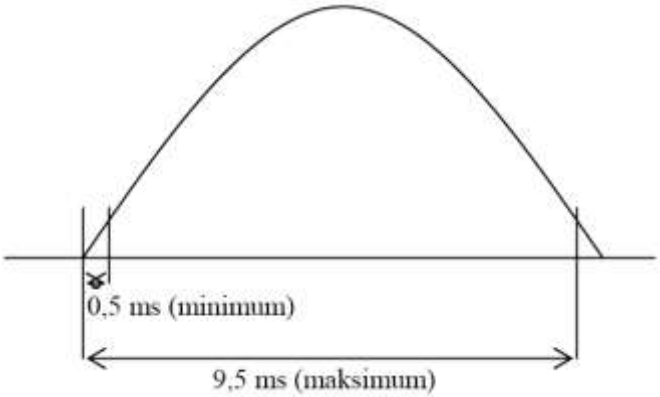
2.2.2 Program Pengendali Intensitas Lampu

Program pengendali intensitas lampu ini akan membaca tegangan pada port A dengan menggunakan ADC dan mengeluarkan pulsa *low* pada port C. Masukan tegangan ini dibatasi hanya 4 buah menggunakan port A0 – A3 dan pulsa *low* dikeluarkan pada port C0 – C3 Tujuan program ini adalah untuk mengeluarkan pulsa *active low* di port C dengan durasi 10 mikrodetik (terlihat di Gambar 3.4) yang dapat digeser-geser terhadap keadaan *zero* (saat tegangan AC = 0 volt). Besarnya pergeseran pulsa (*delay*) ini tergantung NKADC sesuai Persamaan 3.2.



Gambar 3.8. Diagram alir kendali intensitas lampu

Inisialisasi mikrokontroler meliputi pembacaan kristal, konfigurasi port C sebagai output, konfigurasi ADC, konfigurasi INT0 tipe PGT, deklarasi variabel, deklarasi subrutin, menghidupkan ADC dan mengaktifkan interupsi. Pembatasan nilai ADC maksudnya membatasi nilai yang ada dalam variabel A(x) yang baru. Pembatasan dilakukan untuk membatasi *delay firing* agar berada pada rentang yang dikehendaki karena nilai ini digunakan untuk membuat *delay* sesuai Persamaan 3.2. *Delay* minimum yang dikehendaki adalah sekitar 500 mikrodetik (0,5 ms) dan maksimum sekitar 9500 mikrodetik (9,5 ms). Gambaran *delay* minimum dan maksimum *firing* yang dikehendaki pada setengah gelombang AC terlihat di Gambar 3.9.



Gambar 3.9. *Delay* minimum dan maksimum *firing*

Berdasar nilai yang dikehendaki maka dipilih nilai minimum isi variabel A(x) yang baru beserta *delay* yang dihasilkan (berdasar Persamaan 3.2) sebagai berikut.

a. A(1) = 50 → *delay* = 50 x 10us = 500 us = 0,50 ms
b. A(2) = 51 → *delay* = 51 x 10us = 510 us = 0,51 ms
c. A(3) = 52 → *delay* = 52 x 10us = 520 us = 0,52 ms
d. A(4) = 53 → *delay* = 53 x 10us = 530 us = 0,53 ms

Pembatasan nilai maksimum isi variabel A(x) yang baru agar sesuai dengan yang dikehendaki adalah A(x) = 950. Untuk nilai maksimum nilainya dapat dibuat sama antar variabel A(x). Besar *delay* yang dihasilkan berdasar Persamaan 3.2 yaitu: *delay* = 950 x 10 us = 9500 us = 9,5 ms Variabel W, X, Y dan Z nantinya digunakan untuk menentukan keluarnya pulsa *low* pada kaki Cx. Caranya dengan mengecek isi variabel A(x) dimulai dari A(1) sampai A(4) satu persatu sebagai berikut.

a. Jika A(x) = W pulsa *low* di kaki C0
b. Jika A(x) = X pulsa *low* di kaki C1
c. Jika A(x) = Y pulsa *low* di kaki C2
d. Jika A(x) = Z pulsa *low* di kaki C3

Variabel bantu dalam program ini adalah Temp. Untuk n = 4 maka setelah operasi pengurutan akan didapat:

A(1) < A(2) < A(3) < A(4)

Program menunggu interupsi menggunakan perintah **idle**. Perintah **idle** membuat mikrokontroler menggunakan mode *power saving* yaitu mode *idle*. Pada saat *idle* CPU berhenti tetapi interupsi tetap berfungsi. Program mengeluarkan data FFh pada port C, program pembacaan dan pengolahan NKADC menggunakan *looping* diawali dengan perintah **Do** dan diakhiri dengan perintah **Loop**. Tujuan digunakan *looping* agar setelah mikrokontroler selesai melakukan rutin interupsi, program dapat berjalan kembali mulai dari awal *looping*. Berikut ini adalah program *looping* yang digunakan. Diagram alir yang terlihat di Gambar 3.8, bagian rutin pelayanan interupsinya dapat diurai lebih detail seperti terlihat di Gambar 3.10.

Gambar 3.10. Diagram Alir bagian pelayanan interupsi

Pulsa-pulsa berikutnya dikeluarkan setelah pulsa pertama dengan membandingkan isi variabel A(2), A(3) dan A(4) dengan isi W, X, Y dan Z. Pengeluaran pulsa tersebut berurutan karena program Basic bekerja secara sekuensial. Sebagai contoh, misalkan is variabel W, X, Y, Z yang memuat kondisi awal A(1), A(2), A(3) dan A(4) terlihat di bawah ini.

1) W = A(1) = 500,
2) X = A(2) = 300,
3) Y = A(3) = 900, dan
4) Z = A(4) = 700.

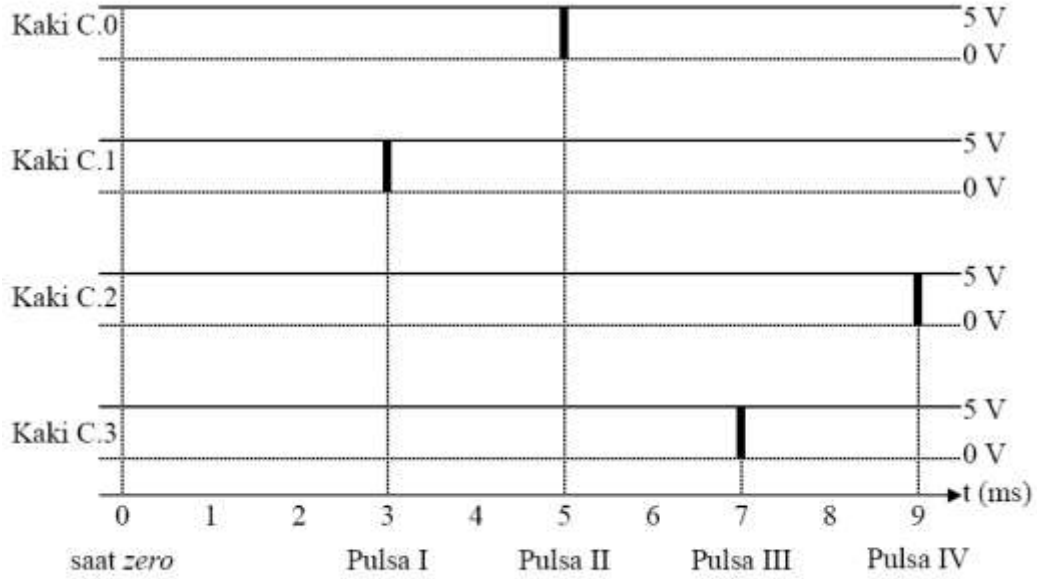
Maka di akhir operasi pengolahan ADC yang telah mengalami pembatasan dan pengurutan, maka isi variabel A(x) yang baru terlihat di bawah ini.

1) A(1) = 300 □□*delay* = 300 x 10 = 3000 us = 3 ms,

2) A(2) = 500 □□*delay* = 500 x 10 = 5000 us = 5 ms,
3) A(3) = 700 □□*delay* = 700 x 10 = 7000 us = 7 ms, dan
4) A(4) = 900 □□*delay* = 900 x 10 = 9000 us = 9 ms.

Isi variabel A(x) yang baru kemudian dicocokan dengan variabel W, X, Y dan Z dan diperoleh hasil di bawah ini.

1) A(1) = 300 = X □□pulsa keluar di kaki C1 dengan *delay* 3 ms,
2) A(2) = 500 = W □□pulsa keluar di kaki C0 dengan *delay* 5 ms,
3) A(3) = 700 = Z □□pulsa keluar di kaki C3 dengan *delay* 7 ms, dan
4) A(4) = 900 = Y □□pulsa keluar di kaki C2 dengan *delay* 9 ms.



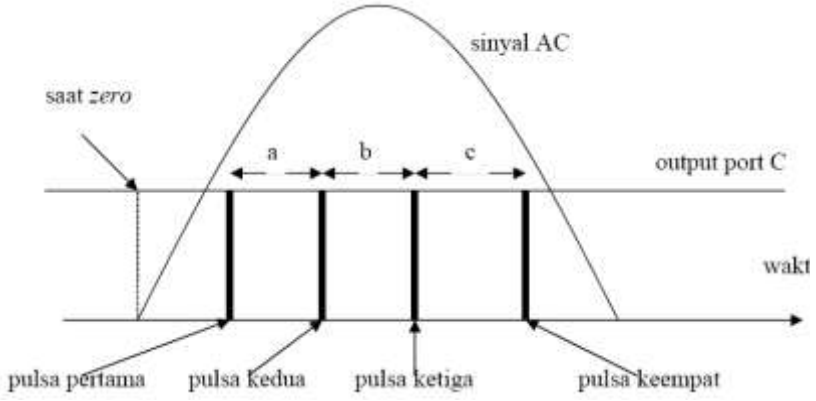
Gambar 3.11. Output Port C terhadap waktu

Dari hasil tersebut terlihat bahwa *delay* pulsa pertama adalah 3 ms pada kaki C1, kedua 5 ms pada kaki C0, ketiga 7 ms pada kaki C3 dan keempat 9 ms pada kaki C2. Acuan *delay* tersebut adalah terhadap *zero*. Hasil tersebut apabila digambar *timing diagram*-nya terlihat di Gambar 3.11.

Berdasar contoh di Gambar 3.11 maka *delay* relatif berikutnya dapat dihitung sebagai berikut.

a. pulsa kedua = 5 – 3 = 2 ms
b. pulsa ketiga = 7 – 5 = 2 ms
c. pulsa keempat = 9 – 7 = 2 ms

Delay pulsa kedua (a), ketiga (b) dan keempat (c) apabila digambar dalam satu sumbu waktu ditunjukkan dengan Gambar 3.12. Bagian program untuk mengeluarkan pulsa *low* kedua, ketiga dan keempat agar sesuai dengan masukan ADC, selengkapnya dapat dilihat di lampiran.



Gambar 3.12. Waktu *firing* untuk *half wave*

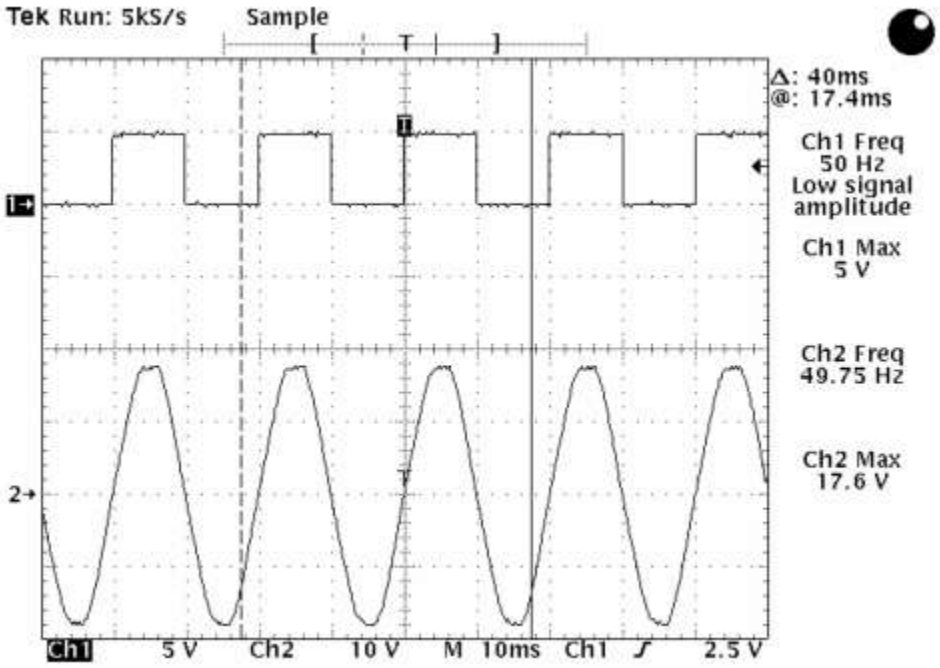
3. HASIL DAN IMPLEMENTASI

Pengujian dilakukan untuk mengetahui kinerja *zero detector* dan kinerja sistem secara keseluruhan. Hubungan input berupa masukan tegangan (maksimum sama dengan Vcc) dan output berupa pulsa *low* (untuk *firing*) yang mengalami *delay* diharapkan sesuai dengan rancangan pada Bab III. Selain itu diamati juga hubungan intensitas lampu dengan masukan tegangan ADC. *Zero detector* diberi catu daya sebesar 5 volt (sama dengan catu daya mikrokontroler). Tujuan diberi catu daya 5 volt ini agar outputnya dapat langsung dibaca oleh mikrokontroler di kaki INT0 sebagai level *low* atau *high* tergantung masukan inputnya yang berasal dari sinyal AC. Beban lampu yang berupa lampu pijar dihubungkan dengan sumber PLN yang memiliki frekuensi 50 Hz. Kinerja sistem dapat diketahui melalui bentuk gelombang yang dihasilkan pada Triac. Melalui bentuk gelombang ini akan dapat dilihat pula besarnya intensitas lampu karena besarnya intensitas lampu tergantung dari besarnya daya yang lewat Triac.

3.1 Hasil Pengujian

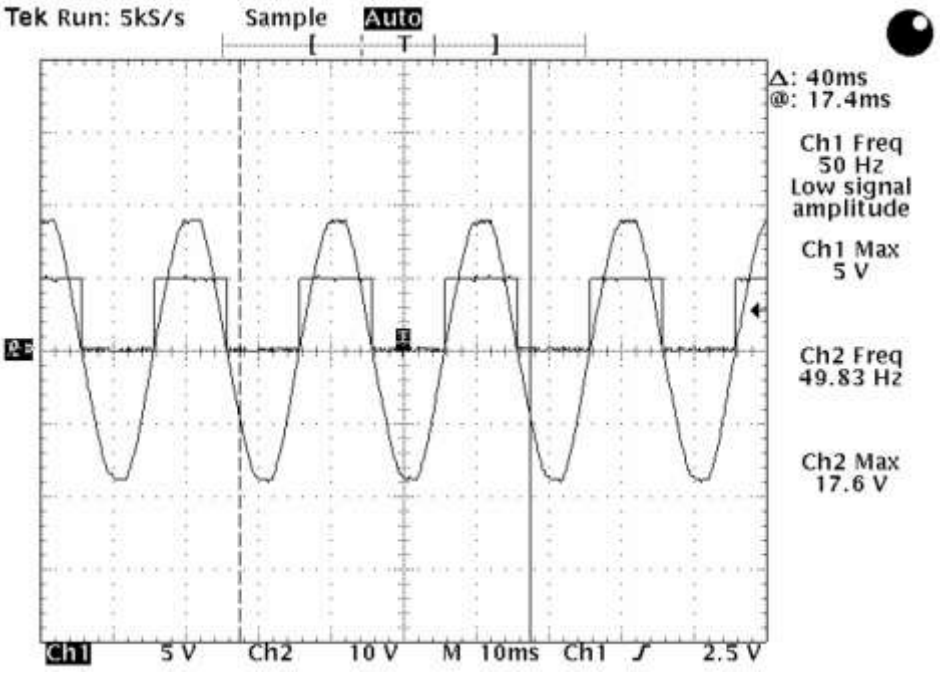
3.1.1 Pengujian *Zero Detector*

Ouput dari *zero detector* ini diamati dengan menggunakan osiloskop. Proses pengamatannya dengan cara meng-*capture* sinyal pada layar osiloskop melalui port serial dengan program Realterm. Program tersebut dapat di-*download* di http://realterm.sourceforge.net. Osiloskop yang digunakan adalah Tektronix TDS 320. Gambarnya terlihat di Gambar 4.1.



Gambar 4.1. Gelombang input dan output *zero detector*

Chanel 2 menunjukkan masukan AC yang dideteksi *zero*-nya. Outputnya terlihat pada chanel 1 yaitu berupa gelombang kotak. Secara teoritis keduanya memilki frekuensi sama yaitu 50 Hz. Perbedaan ini karena osiloskop membaca sinyal dengan frekuensi yang sebenarnya tidak tetap (sedikit berubah-ubah) namun dalam batas yang diizinkan. Apabila diamati lebih jauh, output *zero detector* ini memberikan hasil sesuai harapan yaitu akan menghasilkan output *high* (5 volt) bila sinyal masukanya positif dan menghasilkan output *low* (0 volt) bila masukannya negatif. Pada saat perubahan dari *low* ke *high* (PGT) atau *high* ke *low* (NGT) inilah saat terjadi *zero*. Apabila sinyal input dan output *zero* diamati dalam satu sumbu x maka gambarnya akan terlihat seperti di Gambar 4.2.

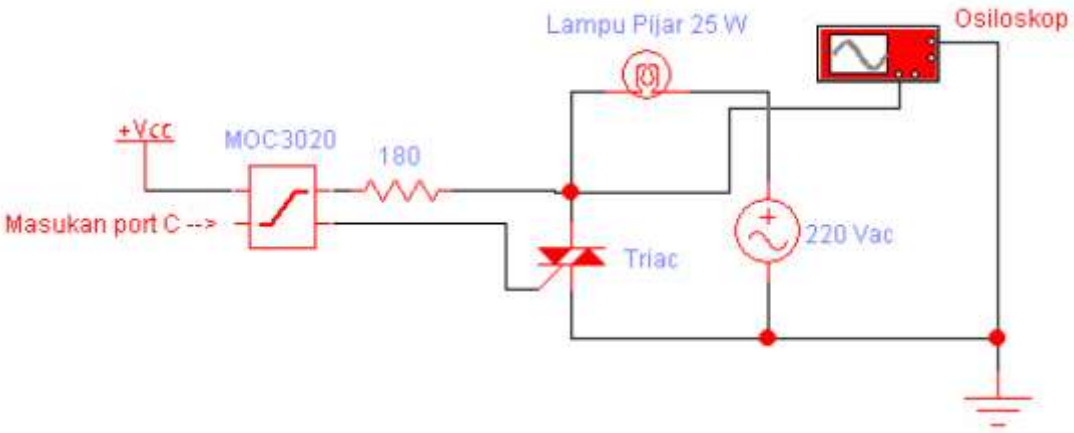


Gambar 4.2. Gelombang input-output dalam satu sumbu x

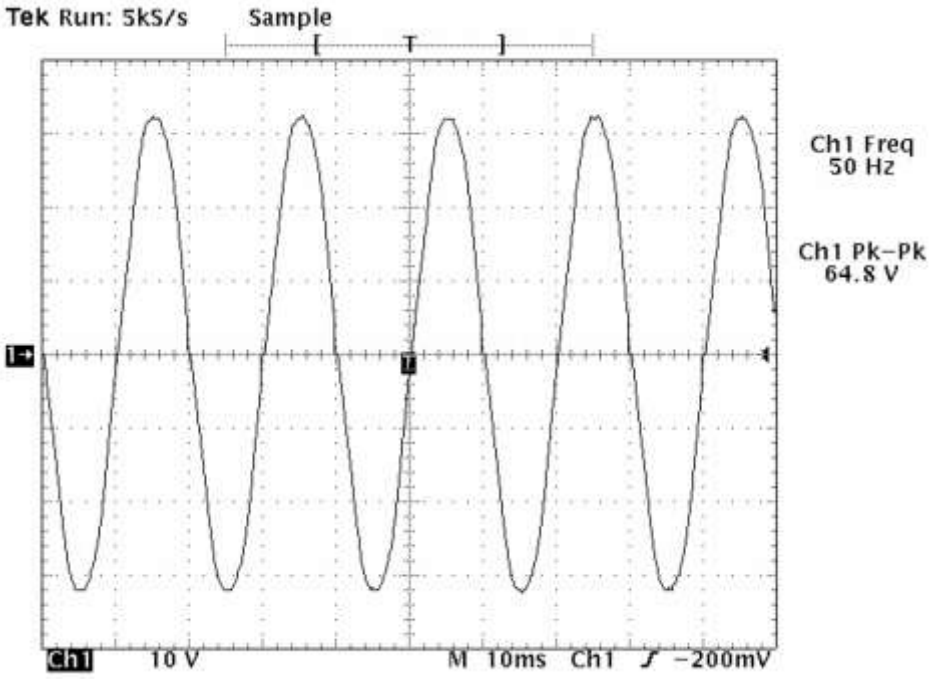
Output *zero detector* ini selanjutnya dihubungkan ke kaki INT0 (pin 16) pada mikrokontroler ATmega8535. Mikrokontroler membaca *zero* hanya pada saat PGT untuk mengaktifkan INT0 dengan mengatur programnya sehingga interupsi terjadi setiap 20 ms (sama dengan periode gelombang AC).

3.1.2 Pengujian Gelombang Output pada Triac

Pengamatan gelombang dilakukan pada kaki Triac dengan osiloskop Tektronix TDS 320 kemudian di-*capture* seperti halnya pada pengukuran *zero detector*. Letak kaki Triac yang diukur gelombangnya ditunjukkan oleh gambar 4.3.

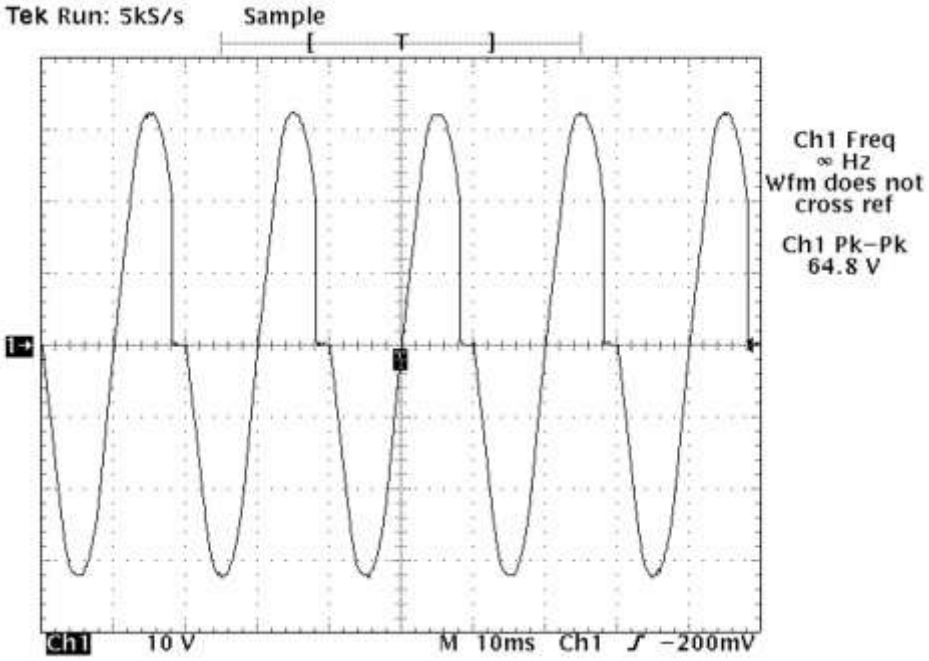


Gambar 4.3. Pengukuran gelombang output



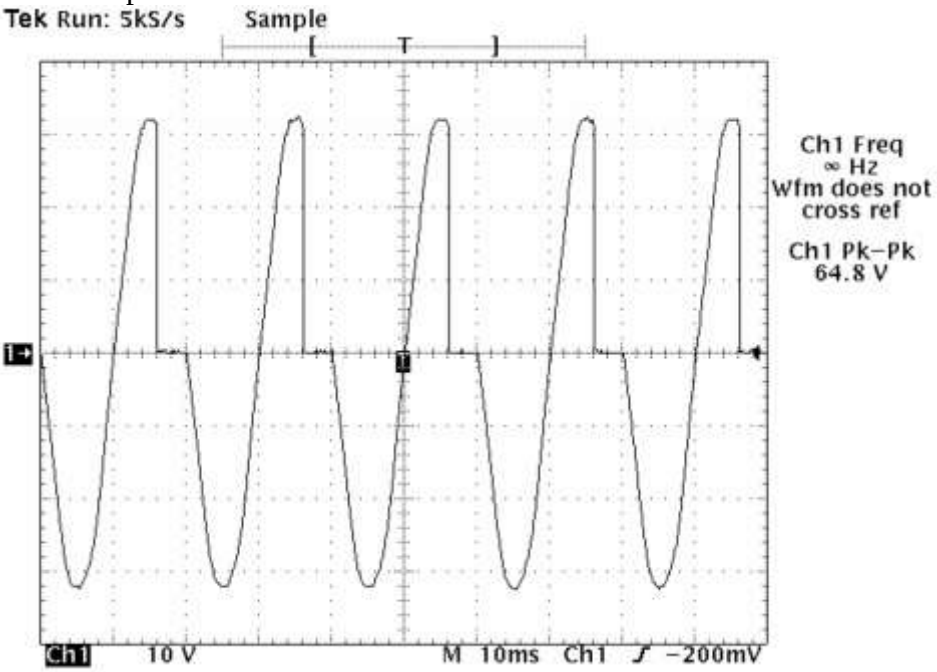
Gambar 4.4. Gelombang output masukan 0 V

Masukan 1 volt dapat dilihat di Gambar 4.5.



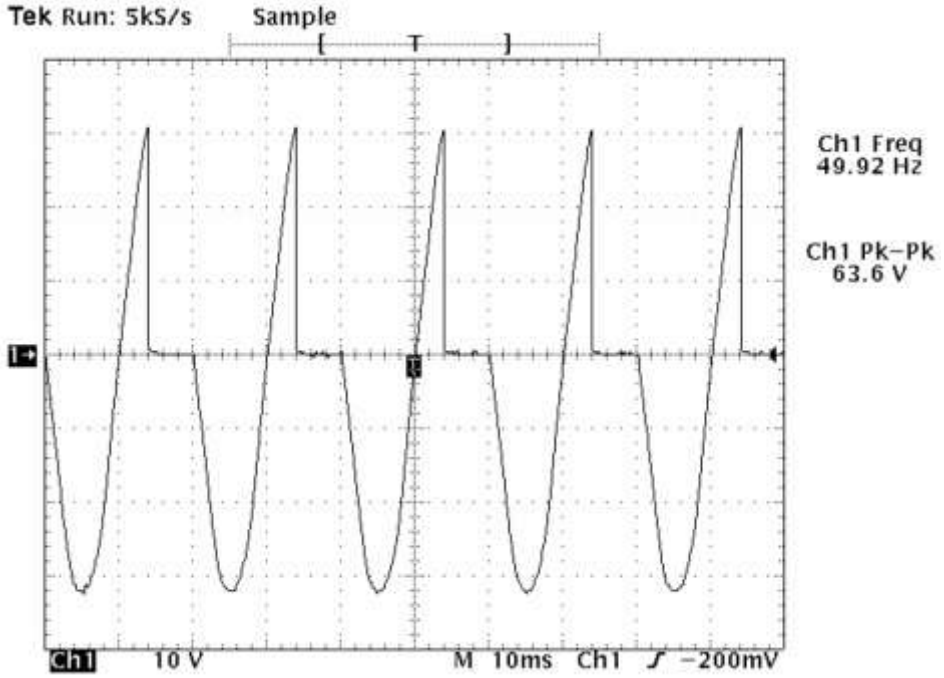
Gambar 4.5. Gelombang output masukan 1 V

Masukan 2 volt dapat dilihat di Gambar 4.6.



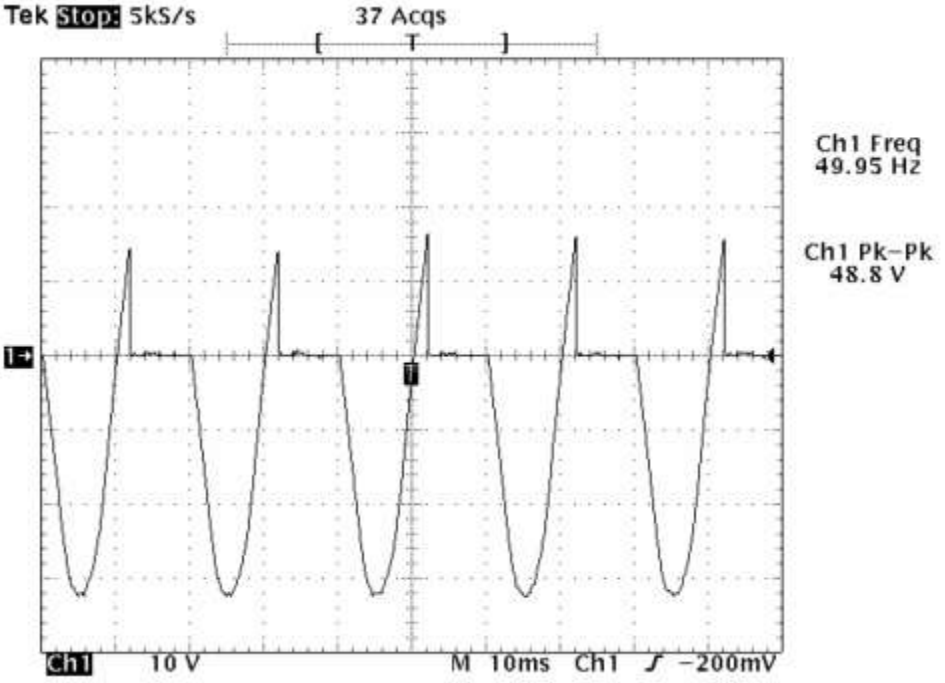
Gambar 4.6. Gelombang output masukan 2 V

Masukan 3 volt dapat dilihat di Gambar 4.7.



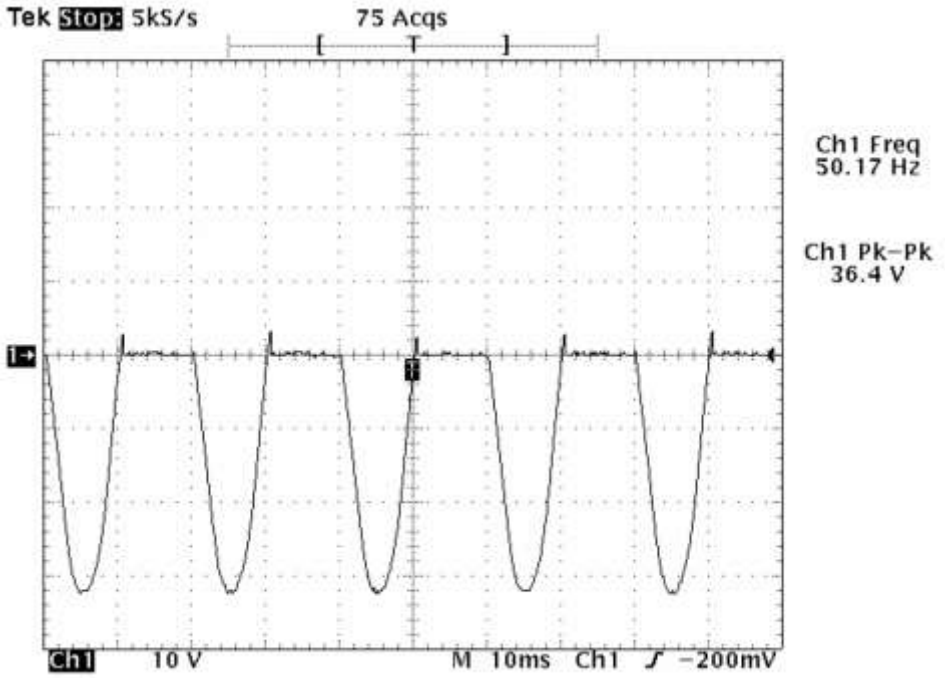
Gambar 4.7. Gelombang output masukan 3 V

Masukan 4 volt dapat dilihat di Gambar 4.8.



Gambar 4.8. Gelombang output masukan 4 V

Masukan 4,93 volt dapat dilihat di Gambar 4.9.



Gambar 4.9. Gelombang output masukan 4,93 V

3.2 Analisis Kerja Sistem

3.2.1 Pewaktuan *firing*

Ketelitian *delay firing* dalam penelitian ini mencapai 10 mikrodetik (basis waktu terkecilnya 10 mikrodetik) sehingga proses *firing*-nya bisa sangat halus. Proses *delay*-nya dengan cara mengali nilai ADC-nya dengan 10 mikrodetik bentuknya dengan me-*looping delay* 10 mikrodetik sebanyak nilai konversi ADC.

Misal nilai konversi ADC = 300 □□*delay*-nya = 300 x 10 us = 3000 us = 3 ms.

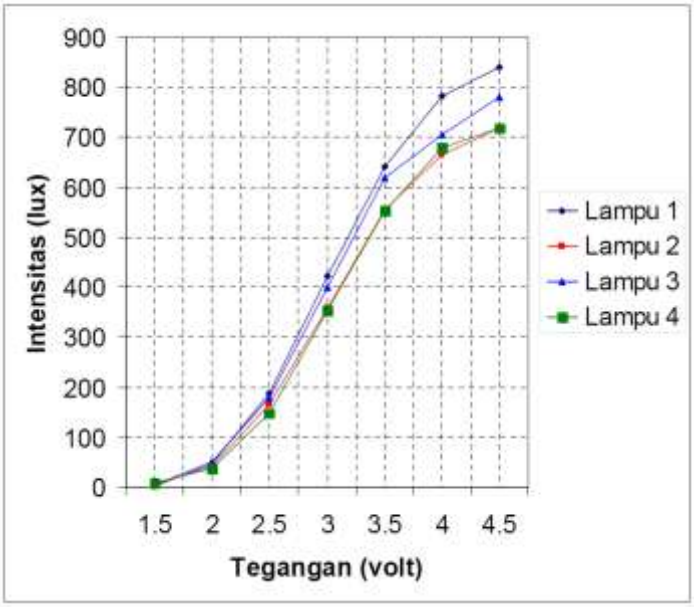
Frekuensi kristal = 8 MHz □□1 clock perlu 1/8000000 detik = 0,000000125 detik

Error = 14 x 0,000000125 detik = 0,00000175 detik = 1,75 mikrodetik

Untuk *firing* yang kedua, ketiga, dan keempat terdapat tambahan *error delay* yaitu untuk operasi selisih. Besarnya *delay* untuk operasi selisih yaitu 20 clock sehingga *delay* operasi untuk sebesar 20 x 0,000000125 detik = 0,0000025 detik = 2,5 mikrodetik. Gambaran mengenai *firing* kedua, ketiga dan keempat dapat dilihat di Gambar 3.12. pada Bab III.

Untuk pemilihan kaki tempat pengeluaran pulsa *low* terdapat *delay* sebesar 16 clock sehingga *delay* operasinya sebesar 2 mikrodetik. Terdapat operasi lain yang perlu clock lebih kecil seperti operasi men-*disable* interupsi, operasi memanggil subrutin dengan basis waktu 10 mikrodetik, operasi meng-enable interupsi yang besarnya masing-masing 3 clock (0,375 mikrodetik).

3.2.2 Kinerja Sistem



Gambar 4.10. Grafik hubungan intensitas terhadap tegangan

Hubungan antara masukan tegangan ADC dan intensitas lampu sebanding namun tidak linear. Pembatasan nilai minimum dan maksimum *delay firing* dengan pembatasan nilai konversi ADC dalam program untuk pengamatan grafik di atas adalah sebagai berikut.

1. Lampu 1 : minimum = 80, maksimum = 950, range 870.
2. Lampu 2 : minimum = 70, maksimum = 940, range 870.
3. Lampu 3 : minimum = 60, maksimum = 930, range 870.
4. Lampu 4 : minimum = 50, maksimum = 920, range 870.

Setelah melakukan pengujian beberapa kali didapat batas minimum dan maksimum nilai konversi ADC yang tidak membuat lampu berkedap-kedip.

1. Lampu 1 : minimum = 53, maksimum = 950.
2. Lampu 2 : minimum = 51, maksimum = 950.
3. Lampu 3 : minimum = 51, maksimum = 950.
4. Lampu 4 : minimum = 50, maksimum = 950.

Sistem ini sangat sensistif terhadap gamgguan luar yaitu lampu menjadi berkedip-kedip apabila *ground* osiloskop dihubungkan ke *ground* atau probe osiloskop dihubungkan ke sistem pengendali intensitas lampu. Namun bentuk gelombang ketika diamati tidak terlalu terpengaruh. Hal ini nampak saat melakukan pengamatan gelombang output pada Triac. Secara umum sistem ini bekerja bekerja dengan baik dan dapat mengendalikan intensitas lampu dengan halus. Pembatasan nilai maksimum dan minimum pada *delay firing* ini tidak terlalu berpengaruh karena mata manusia sangat sulit membedakan perubahan intensitas lampu yang kecil. Hubungan masukan tegangan ADC dan outputnya sebanding tapi tidak linear.

4. Kesimpulan

Dari hasil pengamatan, pengujian dan analisis pada hasil perancangan yang dibuat dapat diperoleh beberapa kesimpulan sebagai berikut. 1) Hasil pengamatan menunjukkan bahwa

sistem ini sudah mendekati hasil rancangan yang ditunjukkan dengan hasil pengamatan output Triac dan intensitas lampu. 2) Program pengendalian intensitas lampu harus memperhatikan *delay* setiap operasi karena inti program ini adalah mengeluarkan pulsa *low* dengan melakukan *delay* yang besarnya sesuai dengan masukan tegangan ADC. 3) Hubungan antara intensitas dan tegangan pengendali dalam sistem ini adalah sebanding namun tidak linear. 4) Sistem ini sangat sensitif terhadap gangguan dari luar terutama *ground*-nya. 5) IC komparator LM393 lebih cocok digunakan sebagai *zero detector* karena catu dayanya dapat digabung dengan catu daya mikrokontroler. 6) Tegangan yang dihasilkan resistor pembagi tegangan tidak untuk men-*supply* daya tapi hanya dibaca tegangannya. 7) Mikrokontroler berfungsi menghasilkan pulsa *low* yang digunakan untuk *firing* Triac melalui *driver* MOC3020. 8) Proses *firing* dengan mengeluarkan pulsa *low* untuk menghindari mikrokontroler mengeluakan arus (*sourcing*) yang dapat membebani mikrokontroler.

DAFTAR PUSTAKA

[1] Atmel, 2005. '*8 Bit AVR Microcontroller ATMEGA8535*' http://www.atmel.com, USA.
[2] Brown B, 1999. '*A Simple Exchange Sort Algorithm*' http://gd.tuwien.ac.at/
[3] Electronic T, 2002. '*AN1001: Fundamental Characteristic of Thyristor*' http://www.teccor.com
[4] Electronic T, 2002. '*AN1003: Phase Control Using Thyristor*' http://www.teccor.com
[5] Engdah. T, 2004. '*Light dimmer circuits*' http://www.epanorama.net
[6] Eurotherm, 2005. '*Eurotherm | Knowledge | What is a thyristor?*' http://www.eurotherm.co.uk/uk/eng/Knowledge/PowerControl/
[7] Farlex. Inc, 2004. '*Thyristor - encyclopedia article about Thyristor*' http://encyclopedia.thefreedictionary.com/Thyristor, USA.
[8] Hamonangan A, 2005. '*Thyristor*' http://www.electroniclab.com/
[9] Missirliu P, 2002. '*Universal Thyristor Driving Board Using Atmel's ATmega163 Microcontroller*' Lycee Newton-Enrea
[10] Ryan. V, 2002. *'The Thyristor*' http://www.technologystudent.com
[11] Xiaobo Tan dkk, 1998. "Characteristic and Firing Control of Thyristor Controlled Series Compensation Installations" *IEEE Press, New York.*
[12] Anonim, 2003. '*AVR Architecture*' http://www.avrbeginners.net
[13] Anonim, 1999. '*Three Quadrant Triacs bring major benefit to OEMs*' http://www.semiconductors.philips.com, Netherlands.