TUGAS AKHIR – TE 145561

PERANCANGAN SISTEM *MONITORING VOLTAGE FLICKER* BERBASIS ARDUINO DENGAN METODE *FAST FOURIER TRANSFORM* (FFT)

Adhitya Wisnu Wardhana
NRP 2213038014
Faisal Akhbar
NRP 2213038016

Dosen Pembimbing
Dr. Eng. Ardyono Priyadi, S.T., M.Eng.

PROGRAM STUDI D3 TEKNIK ELEKTRO
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember
Surabaya 2016

FINAL PROJECT – **TE 145561**

MONITORING OF VOLTAGE FLICKER WITH FAST FOURIER TRANSFORM (FFT) METHOD BASED ARDUINO

Adhitya Wisnu Wardhana
NRP 2213038014
Faisal Akhbar
NRP 2213038016

Supervisor
Dr. Eng. Ardyono Priyadi, S.T., M.Eng.

ELECTRICAL ENGINEERING D3 STUDY PROGRAM
Faculty of Industrial Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2016

PERNYATAAN KEASLIAN
TUGAS AKHIR

Dengan ini saya menyatakan bahwa isi sebagian maupun keseluruhan Tugas Akhir saya dengan judul "**Perancangan Sistem *Monitoring Voltage Flicker* Berbasis Arduino Dengan Metode *Fast Fourier Transform* (FFT)**" adalah benar-benar hasil karya intelektual mandiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diijinkan dan bukan merupakan karya pihak lain yang saya akui sebagai karya sendiri.

Semua referensi yang dikutip maupun dirujuk telah ditulis secara lengkap pada daftar pustaka.

Apabila ternyata pernyataan ini tidak benar, saya bersedia menerima sanksi sesuai peraturan yang berlaku.

Surabaya, 2 Juni 2016

Adhitya Wisnu Wardhana
NRP 2213038014

Faisal Akhbar
NRP 2213038016

PERANCANGAN SISTEM *MONITORING VOLTAGE FLICKER* BERBASIS ARDUINO DENGAN METODE *FAST FOURIER TRANSFORM* (FFT)

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar Ahli Madya Teknik
Pada
Bidang Studi Teknik Listrik
Program Studi D3 Teknik Elektro
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember

Menyetujui :
Dosen Pembimbing

Dr. Eng. Ardyono Priyadi, S.T., M.Eng.
NIP. 1973 09 27 1998 03 1004

SURABAYA
JUNI, 2016

PERANCANGAN SISTEM *MONITORING VOLTAGE FLICKER* BERBASIS ARDUINO DENGAN METODE *FAST FOURIER TRANSFORM* (FFT)

Nama Mahasiswa 1	**: Adhitya Wisnu Wardhana**
NRP	**: 2213038014**
Nama Mahasiswa 2	**: Faisal Akhbar**
NRP	**: 2213038016**
Pembimbing	**: Dr. Eng. Ardyono Priyadi, S.T., M.Eng.**
NIP	**: 1973 09 27 1998 03 1004**

ABSTRAK

PT. PLN(Persero) tiap tahunnya selalu berusaha memberi kualitas daya yang semakin baik. Salah satu faktor yang mempengaruhi adalah gangguan *Voltage Flicker*. *Voltage Flicker* adalah gangguan perubahan naik turun tegangan di sistem penyaluran tenaga listrik secara terus menerus yang diakibatkan oleh beban yang berdenyut-denyut (seperti : *arc furnace*, motor listrik, dan las listrik).

Permasalahan pada tugas akhir ini adalah bagaimana memonitoring gangguan *Voltage Flicker* agar PLN dapat memberi kualitas daya listrik yang semakin baik. *Voltage Flicker* dapat menyebabkan cahaya lampu yang berkedip yang berefek iritasi pada mata, tidak bekerjanya relay proteksi dengan baik. Tujuan tugas akhir ini adalah membuat perancangan sistem monitoring *Voltage Flicker* yang hasilnya ditampilkan di *interface* Matlab dalam bentuk grafik dan hasil data tegangan dan arus dapat disimpan pada *SD Card* guna mengaudit hasil kualitas listrik.

Alat pada Tugas Akhir ini dapat mendeteksi adanya *Voltage Flicker* lalu menghitung ΔV serta THDV dari beban yang diukur. Dari 5 macam beban yang diukur didapatkan hasil bahwa mesin Las Listrik Falcon 211GE berpotensi membangkitkan *flicker*. Alat ini juga dapat membuktikan bahwa metode *Fast Fourier Transform* (FFT) yang digunakan lebih cepat yakni membutuhkan waktu 0,000577069 detik untuk 2500 data sedangkan untuk metode DFT membutuhkan waktu 1,35219 detik. Hasil tegangan dan arus juga dapat disimpan pada *SD Card* dengan selang waktu 1 detik yang ditampilkan pada *Database*.

Kata Kunci : *Monitoring*, *voltage flicker, FFT, SD Card*

***MONITORING OF VOLTAGE FLICKER WITH FAST FOURIER TRANSFORM(FFT) METHOD BASED* ARDUINO**

Student Name 1	**: Adhitya Wisnu Wardhana**
Register Number	**: 2213038014**
Student Name 2	**: Faisal Akhbar**
Register *Number*	**: 2213038016**
Supervisor	**: Dr. Eng. Ardyono Priyadi, S.T., M.Eng.**
ID	**: 1973 09 27 1998 03 1004**

ABSTRACT

Every year PT. PLN (Persero) *is always trying to give power quality is getting better. One factor that can affect is Voltage Flicker disturbances. Voltage flicker is a fluctuation voltage fault with changes up and down amplitude in electric power distribution systems continually caused by the load pulsing (such as arc furnaces, electric motors, and electric welding).*

The main issues is how to monitoring Voltage Flicker fault so PLN *can give power quality getting better. Voltage flicker may causes changes the illumination intensity of light source , known as flicker. Flicker may produce a very unpleasant visual sensation, leading to complaints from Customers and broke the protection relay. The main purpose is to create a Voltage Flicker monitoring system design which the results are displayed in Matlab interface in the form of graphs and voltage data and current results can be stored on the SD Card to audit the results of power quality.*

This tool of final project can detect Voltage Flicker and calculate ΔV and THDV of 5 loads measured. From 5 loads measured showed that Falcon Electric 211GE welding electric machine potentially be a flicker load. This tool of final project can also prove that the method of Fast Fourier Transform (FFT) used a faster which takes 0.000577069 seconds in 2500 data, while the data for the DFT method takes 1.35219 seconds. The result of voltage and current data can store on SD Card and displayed in Database.

Keywords : *Monitoring, voltage flicker, FFT, SD Card*

KATA PENGANTAR

Puji syukur penulis panjatkan kehadirat Allah SWT yang selalu memberikan rahmat dan hidayah-Nya sehingga Tugas Akhir ini dapat terselesaikan dengan baik. Shalawat serta salam semoga selalu dilimpahkan kepada Rasulullah Muhammad SAW, keluarga, sahabat, dan umat muslim yang senantiasa meneladani beliau.

Tugas Akhir ini disusun untuk memenuhi sebagian persyaratan guna menyelesaikan pendidikan Diploma 3 pada Bidang Studi Teknik Listrik, Jurusan Teknik Elektro, Fakultas Teknologi Industri, Institut Teknologi Sepuluh Nopember Surabaya dengan judul:

PERANCANGAN SISTEM *MONITORING VOLTAGE FLICKER* BERBASIS ARDUINO DENGAN METODE *FAST FOURIER TRANSFORM* (FFT)

Dalam Tugas Akhir ini dirancang sistem *monitoring Voltage Flicker* dengan menggunakan Arduino sebagai pusat kontroler dari sistem ini, setelah mendapatkan data dari sensor yang terhubung dengan Arduino maka akan dilakukan pemrosesan data pada Matlab sehingga didapatkan grafik dari *Voltage Flicker*.

Penulis mengucapkan terima kasih kepada Ibu dan Bapak penulis yang memberikan berbagai bentuk doa serta dukungan tulus tiada henti, Bapak Dr. Eng. Ardyono Priyadi, S.T., M.Eng. atas segala bimbingan ilmu, moral, dan spiritual dari awal hingga terselesaikannya Tugas Akhir ini, Penulis juga mengucapkan banyak terima kasih kepada semua pihak yang telah membantu baik secara langsung maupun tidak langsung dalam proses penyelesaian Tugas Akhir ini.

Penulis menyadari dan memohon maaf atas segala kekurangan pada Tugas Akhir ini. Akhir kata, semoga Tugas Akhir ini dapat bermanfaat dalam pengembangan keilmuan di kemudian hari.

Surabaya, 2 Juni 2016

Penulis

DAFTAR ISI

HALAMAN

DAFTAR GAMBAR

HALAMAN

DAFTAR TABEL

HALAMAN

BAB I
PENDAHULUAN

1.1 Latar Belakang

Data terakhir Dinas ESDM Jawa Timur mencatat, rasio elektrifikasi di wilayah Jawa Timur baru mencapai 83%. Dengan kata lain, sebanyak 17% warga Jawa Timur belum memiliki dan menikmati listrik di rumahnya. Semakin tahun angka ini akan meningkat, dengan meningkatnya rasio elektrifikasi jawa timur maka kualitas daya listrik yang diperlukan harus semakin baik [1].

Perhatian masyarakat terhadap kualitas daya semakin tahun semakin meningkat seiring dengan penggunaan energi listrik oleh masyarakat. Terdapat definisi yang berbeda tentang kualitas daya listrik, tergantung kerangka acuan yang digunakan dalam mengartikan istilah tersebut sebagai contoh suatu pengguna utilitas kelistrikan dapat mengartikan kualitas daya listrik sebagai kehandalan, dimana dengan menggunakan angka statistik 99,98%, sistem tenaga listriknya mempunyai kualitas yang dapat diandalkan [2]. Kualitas daya sistem tenaga listrik dipengaruhi oleh adanya gangguan gangguan. Dengan kata lain adanya gangguan yang terjadi pada sistem tenaga listrik yang menyebabkan kualitas daya menurun. Gangguan tersebut tidak hanya faktor eksternal seperti petir, pohon tumbang dan curah hujan yang tinggi sedangkan gangguan internal dengan rentang yang luas sehingga dapat mengganggu operasi dari beban industri yang sensitif dan mengganggu jalannya produksi. Gangguan internal masalah kualitas daya ialah *short interruptions, voltage dips, voltage swells, voltage and current transients, voltage and current harmonic distortion, voltage flicker, voltage unbalance, phase angle imbalance, and/or jump* [2]. Dari gangguan gangguan internal tersebut yang menjadi fokus kami adalah gangguan *Voltage Flicker*.

Flickers adalah gangguan perubahan naik turun tegangan di sistem penyaluran tenaga listrik secara terus menerus yang diakibatkan oleh beban yang berdenyut-denyut (seperti : *arc furnace*, motor listrik, dan las listrik) [3]. Pada saat ini dari pihak Perusahaan Listrik Negara jarang melakukan me*monitoring* terhadap beban-beban yang dapat menyebabkan kualitas daya menurun, salah satunya *Voltage Flicker*. *Voltage Flicker* dapat menyebabkan cahaya lampu yang berkedip yang

berefek iritasi pada mata, tidak bekerjanya relay proteksi dengan baik, menyebabkan kerusakan pada perlatan listrik yang rentan terhadap adanya fluktuasi tegangan [4]. Alat *Voltage Flicker* meter saat ini telah tersedia, namun harga dari alat ini masih relatif mahal dan tidak digunakan oleh PLN pada setiap beban yang ada.

Dengan adanya sistem *monitoring* terhadap *Voltage Flicker* dapat membantu Perusahaan Listrik Negara (PLN) untuk memenuhi alat *Voltage Flicker* meter yang relatif murah dan melakukan langkah-langkah sejak dini ketika tanda-tanda *flicker* telah ada, sehingga pelayanan dari segi kualitas daya dari PLN terjamin mutu kualitasnya.

1.2 Permasalahan

Besarnya permintaan daya listrik oleh pelanggan yang semakin tinggi tiap tahunnya, menuntut Perusahaan Listrik Negara (PLN) untuk memberikan pelayanan terbaik bagi konsumennya, tidak hanya dari segi fasilitas yang baik namun dari segi kualitas daya yang diberikan pun menjadi tanggung jawab Perusahaan Listrik Negara (PLN). Berbicara kualitas daya, banyak faktor yang mempengaruhinya, salah satunya adanya gangguan *Voltage Flicker*. Yang menjadi rumusan masalah dalam Tugas Akhir ini adalah bagaimana me*monitoring* adanya gangguan *Voltage Flicker* pada sistem tenaga listrik agar kualitas daya yang diberikan PLN terjamin mutu kualitasnya.

1.3 Batasan Masalah

Dari perumusan masalah diatas, maka batasan masalah dari Tugas Akhir ini adalah :

1. Parameter yang diukur adalah arus dan tegangan,
2. Sensor tegangan yang digunakan untuk mengukur besaran tegangan selama proses *sampling* berlangsung adalah ZMPT101B, yang nantinya akan diolah dengan metode FFT,
3. *Sampling* tegangan dilakukan sebanyak 2500 data,
4. Pengujian menggunakan beban 1 fasa arus maksimal 16 A, dengan 5 macam beban,
5. Besaran *Voltage Flicker* akan ditampilkan dalam bentuk gelombang pada HMI,
6. Penyimpanan data pada *SD Card* hanya berupa besaran arus dan tegangan,

1.4 Tujuan

Tujuan dari Tugas Akhir Perancangan Sistem *Monitoring Voltage Flicker* ini adalah membuat perancangan sistem *monitoring Voltage Flicker,* yang hasil perhitungannya akan ditampilkan dalam bentuk grafik pada *interface* yang menggunakan Matlab, serta mengukur beban yang dapat menyebabkan terjadinya *Voltage Flicker.* Serta mebuat tampilan *interface* berupa tampilan data yang tersimpan dalam *SD Card* sehingga dapat dilakukan audit kualitas tegangan.

1.5 Metodologi Penelitian

Penelitian ini dilakukan melalui beberapa tahapan metodologi, yaitu, studi literatur, perancangan sistem, simulasi hasil desain, implementasi dan analisis data, dan yang terakhir adalah penyusunan laporan berupa buku Tugas Akhir.

Pada tahap studi literatur akan dipelajari mengenai identifikasi sensor ZMPT101B, Identifikasi sensor arus CT, Identifikasi sensor tegangan, Identifikasi Arduino Mega 2560, identifikasi rangkaian RTC DS 1307, Identifikasi rangkaian MMC *Shield*, dan sistem *interface* pada Matlab. Pada tahap perancangan sistem akan dibahas mengenai perencanaan dan pembuatan perangkat keras yang meliputi rangkaian-rangkaian, desain bangun, dan perangkat lunak yang berupa *software interface* menggunakan Matlab akan digunakan untuk menampilkan data dari *hardware*. Pada tahap pengujian dan analisa data, akan dilakukan pengukuran, pengujian, dan penganalisaan terhadap kepresisian sensor dan alat yang telah kami buat. Tahap akhir penelitian adalah penyusunan laporan penelitian.

1.6 Sistematika Laporan

Pembahasan Tugas Akhir ini akan dibagi menjadi lima Bab dengan sistematika sebagai berikut:

Bab I **Pendahuluan**

Bab ini meliputi latar belakang, permasalahan, batasan masalah, maksud dan tujuan, sistematika laporan, serta relevansi.

Bab II **Teori Dasar**

Bab ini menjelaskan tentang tijauan pustaka *Voltage Flicker*, teori *Fast Fourier Transform*, Sensor arus

ACS 712, Sensor Tegangan, Arduino yang mendukung dalam perencanaan dan pembuatan alat.

Bab III **Perancangan Alat**

Bab ini membahas tentang perencanaan dan pembuatan perangkat keras yang meliputi rangkaian-rangkaian, desain bangun, dan perangkat lunak yang meliputi program yang akan digunakan untuk mengaktifkan alat tersebut.

Bab IV **Pengujian dan Analisa Data**

Bab ini membahas tentang pengukuran, pengujian, dan penganalisaan terhadap kepresisian sensor dan alat yang telah kami buat.

Bab V **Penutup**

Bab ini berisi kesimpulan dan saran dari hasil pembahasan yang telah diperoleh.

1.7 Relevansi

Hasil yang diperoleh dari Tugas Akhir ini diharapkan menjadi referensi *monitoring Voltage Flicker* dan juga dengan adanya alat ini diharapkan mempermudah PLN dalam memberikan pelayanan terbaik kepada konsumen, salah satunya memberikan kualitas daya yang baik terhadap konsumen pengguna tenaga listrik.

BAB II
TEORI DASAR

Pada Bab II ini akan dijelaskan mengenai teori-teori dasar yang menunjang dan berhubungan dalam pengerjaan Tugas Akhir ini. Teori dasar ini diharapkan mampu membantu dalam pengerjaan Tugas Akhir dan dapat dijadikan referensi nantinya.

2.1 Kualitas Daya Listrik [5]

Terdapat berbagai definisi yang berbeda untuk kulitas daya listrik, tergantung dari acuan yang digunakan untuk mengukur kulitas tersebut. kualitas daya sebagai suatu keandalan dimana menunjukkan bahwa 99,9% sistem dalam keadaan andal.[2] Sedangkan untuk kulitas daya di PT PLN (Persero) mencakup 8 parameter yaitu :

1. Tegangan di luar standar
2. Tegangan tidak seimbang
3. Lonjakan tegangan
4. Kedip tegangan
5. Kelip (*flicker*)
6. Harmonisa
7. Frekuensi
8. Suplai listrik terputus (*interrupt*) < 10 ms

Setiap perusahaan memiliki standar mutu untuk setiap produk yang dihasilkan. Tidak terkecuali PLN sebagai perusahaan yang menghasilkan produk berupa energi listrik. Beberapa standar mutu produk ini kemudian ditetapkan dalam Keputusan Direksi PT. PLN (Persero) No. 109.K/039/DIR/1997 tentang Penggunaan Piranti Tenaga Listrik.

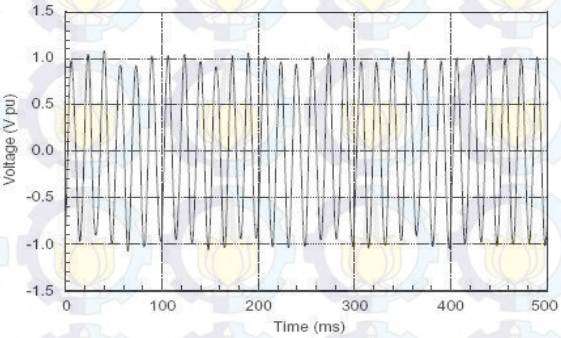
Power Quality adalah tingkat kualitas dari daya listrik yang meliputi :

1. Bentuk gelombang
2. Besar amplitudo atau *magnitude*
3. Nilai frekuensi, baik nilai periode penuh maupun nilai setengah periode

2.2 *Flicker* [3]

Flicker atau kedip tegangan adalah variasi tegangan atau fluktuasi tegangan yang disebabkan beban yang berubah sangat cepat dan terjadi terus-menerus. Fluktuasi tegangan adalah suatu perubahan tegangan

yang sistematis atau serangkaian perubahan tegangan secara acak, di mana *magnitude* dari tegangan mempunyai nilai yang tidak semestinya (Roger C. Dugan, 1996), yaitu di luar rentang tegangan ditentukan oleh ANSI C84.1 sebesar 0,9 sampai 1,1 pu. Menurut IEC 61000-2-1 salah satu fluktuasi tegangan, mempunyai karakteristik sebagai rangkaian tegangan acak yang berfluktuasi secara terus menerus. Istilah *flicker* atau kedip tegangan berasal dari dampak adanya fluktuasi tegangan terhadap lampu, yang dianggap seperti mata manusia yang berkedip. Gelombang *flicker* ditunjukkan pada Gambar 2.1



Gambar 2.1 Gelombang Tegangan *Flicker*

Gambar 2.1 adalah contoh dari gelombang tegangan yang menghasilkan *flicker* yang disebabkan oleh sebuah busur bunga api, salah satu faktor paling umum penyebab fluktuasi tegangan pada transmisi dan distribusi sistem tenaga listrik. *Flicker* tegangan diukur dengan sensitivitas mata manusia. Biasanya, *flicker* yang besarnya lebih rendah 0,5 persen dapat menyebabkan lampu nampak berkedip, jika frekuensi berada dalam kisaran antara 6 sampai 8 Hz.

Beberapa penyebab yang menimbulkan tegangan *flicker* antara lain sebagai berikut. Adanya *furnace* atau tungku api, adanya peralatan dengan busur api seperti las listrik, adanya peralatan elektronika daya yang sebagian merupakan beban non linear.

2.3 *Fast Fourier Transform* [6]

Fast Fourier Transform atau lebih populer dengan istilah FFT yang diperkenalkan oleh J.S.Bendat dan A.G.Piersol pada 1986. *Fast Fourier Transform* dalam bahasa indonesia adalah Transformasi *Fourier* Cepat adalah sumber dari suatu algoritma untuk menghitung *Discrete*

Fourier Transform (transformasi *fourier* diskrit atau DFT) dengan cepat, efisien dan *invers*nya.

Fast Fourier Transform (FFT) diterapkan dalam beragam bidang dari pengolahan sinyal digital dan memecahkan Persamaan diferensial parsial menjadi algoritma-algoritma untuk penggandaan bilangan integer dalam jumlah banyak. Ada pun kelas dasar dari algoritma FFT yaitu *decimation in time* (DIT) dan *decimation in frequency* (DIF). Garis besar dari kata *Fast* diartikan karena formulasi FFT jauh lebih cepat dibandingkan dengan metode perhitungan algoritma *Fourier Transform* sebelumnya.

Metode FFT memerlukan sekitar 10000 operasi algoritma matematika untuk data dengan 1000 observasi, 100 kali lebih cepat dibandingan dengan metode sebelumnya. Penemuan FFT dan perkembangan personal komputer, teknik FFT dalam proses analisa data menjadi populer, dan merupakan salah satu metode baku dalam analisa data. Satu bentuk transformasi yang umum digunakan untuk merubah sinyal dari domain waktu ke domain frekuensi adalah Transformasi *Fourier*: Persamaan dari bentuk sinyal x(t).

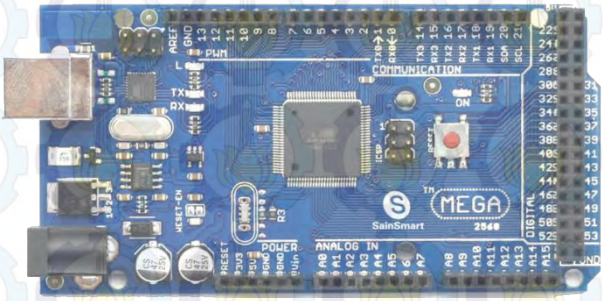
$$x(\omega) = \int_{-\infty}^{\infty} x(t) e^{-j\omega t} dt \quad \text{.....(2.1)}$$

FFT dalam pengolahan isyarat meliputi periode dan frekuensi. Secara umum periode didefinisikan sebagai waktu yang dibutuhkan untuk sebuah isyarat atau gelombang mencapai suatu gelombang penuh dan dapat menentukan nilai periodesitasnya. Perlu dicermati bahwa pengertian ini berlaku untuk isyarat monokromatis, isyarat yang dimaksud adalah gelombangnya bersifat tunggal, pasti memiliki sebuah periode. Dengan demikian isyarat itu dikenal dengan istilah periodis, pengamatan dapat dilakukan dengan memantau gelombang kita dapat mengetahui nilai nilai yang terkandung dalam isyarat serta periodenya.

Ada periode, maka ada frekuensi diartikan sebagai jumlah gelombang yang terjadi dalam 1 detik. Frekuensi didefinisikan secara sederhana sebagai kebalikan dari waktu. Sehingga waktu yang satuannya adalah detik (*second*) akan menjadi Hz (1-per *second*) hanya akan memiliki tepat satu nilai spektrum. Yang dikenal dengan spektrum frekuensi. Pengertian frekuensi ini juga berlaku untuk gelombang monokromatis.

2.4 Arduino Mega 2560 [7]

Arduino Mega 2560 adalah papan pengembangan *microcontroller* yang berbasis Arduino dengan menggunakan *chip* ATMega2560. *Board* ini memiliki pin I/O yang cukup banyak, sejumlah 54 buah digital I/O pin (15 pin diantaranya adalah PWM), 16 pin *analog input*, 4 pin UART (serial *port hardware*). Arduino Mega 2560 dilengkapi dengan sebuah *oscillator* 16 Mhz, sebuah *port* USB, *power jack* DC, ICSP *header*, dan tombol *reset*. *Board* ini sudah sangat lengkap, sudah memiliki segala sesuatu yang dibutuhkan untuk sebuah *microcontroller*. Dengan penggunaan yang cukup sederhana, anda tinggal menghubungkan *power* dari USB ke PC anda atau melalui adaptor AC/DC ke *jack* DC. Gambar Arduino Mega ditunjukkan pada Gambar 2.2



Gambar 2.2 *Board* Arduino Mega 2560

Spesifikasi Arduino Mega 2560 :

1. Menggunakan *chip microcontroller* AtMega2560.
2. Tegangan operasi 5 Volt.
3. Tegangan *input* (yang direkomendasikan, via *jack* DC) sebesar 7-12 Volt.
4. Digital I/O sebanyak 54 buah, 6 diantaranya menyediakan PWM *output*.
5. *Analog input* pin sebanyak 16 buah.
6. Arus DC per pin I/O sebesar 20 mA.
7. Arus DC pada pin 3,3 Volt sebesar 50 mA.
8. *Flash memory* sebesar 256 KB, 8 KB telah digunakan untuk *bootloader*.
9. SRAM sebesar 8 kb.
10. EEPROM sebesar 4 kb.
11. *Clock speed* sebesar 16 Mhz.

12. Dimensi Arduino Mega 2560 sebesar 101,5 mm x 53,4 mm.
13. Berat Arduino Mega 2560 sebesar 37 g.

2.4.1 Arduino IDE

Board Arduino dapat diprogram menggunakan *software open source* bawaan Arduino IDE. Arduino IDE adalah sebuah aplikasi *crossplatform* yang berbasis bahasa pemrograman *processing* dan *wiring*. Arduino IDE didesain untuk mempermudah pemrograman dengan adanya kode editor yang dilengkapi dengan *syntax highlighting, brace matching*, dan indentasi otomatis untuk kemudahan pembacaan program, serta dapat meng-*compile* dan meng-*upload* program ke *board* dalam satu klik. Jendela Arduino IDE dapat dilihat pada Gambar 2.3

Gambar 2.3 Jendela Arduino IDE

IDE Arduino adalah *software* yang sangat canggih ditulis dengan menggunakan Java. IDE Arduino terdiri dari:

1. *Editor program*, sebuah *window* yang memungkinkan pengguna menulis dan mengedit program dalam bahasa *processing*.
2. *Compiler*, sebuah modul yang mengubah kode program (bahasa *Processing*) menjadi kode biner. Bagaimanapun sebuah *microcontroller* tidak akan bisa memahami bahasa *processing*. Yang bisa dipahami oleh *microcontroller* adalah kode biner. Itulah sebabnya *compiler* diperlukan dalam hal ini.

3. *Uploader*, sebuah modul yang memuat kode biner dari komputer ke dalam *memory* di dalam papan Arduino.

2.5 *Analog to Digital Converter* (ADC) [8]

ADC adalah sebuah alat yang mengkonversi nilai tegangan menjadi nilai dalam nilai digital. Pengkonversian oleh ADC dilakukan dengan teknik *sampling*. Yaitu data yang masuk secara kontinu hanya diambil dalam interval waktu tertentu. Sehingga ADC adalah salah satu alat yang mengkonversi sinyal analog kontinu menjadi sinyal digital diskrit.

Penghitungan ADC dengan menggunakan Arduino dapat menggunakan rumus berikut :

$$ADC = \frac{Vin}{Vref} \times ADC \quad \text{.....(2.2)}$$

Keterangan:

ADC = Nilai digital dari tegangan yang terbaca

Vin = Nilai tegangan yang masuk pada pin ADC (Volt)

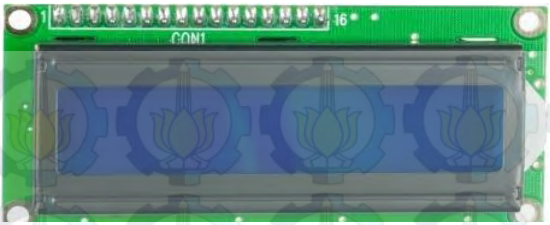
Vref = Tegangan referensi yang digunakan rangkaian ADC (Volt)

bit ADC = Bit ADC yang digunakan biasanya 8-bit atau 10-bit

Pada *board* Arduino, ADC yang digunakan adalah ADC 10-bit sehingga memiliki *range* nilai antara 0-1023 dan *range* tegangan *input* antara 0-5 Volt. Apabila dihitung, setiap kenaikan 4,89 miliVolt nilai ADC naik 1 tingkat. Sebagai catatan, nilai ADC selalu dalam bentuk bulat sehingga nilai ADC akan selalu dibulatkan ke atas, oleh karena itu ADC memiliki *error* sebesar $1/2^{\text{bit-resolusi}}$ dimana apabila menggunakan 10-bit *error*-nya adalah 1/1023 % ≈ 0,0978 %.

2.6 *Liquid Crystal Diplay* (LCD) [8]

LCD (*Liquid Crystal Diplay*) berfungsi menampilkan suatu nilai hasil sensor, menampilkan teks, atau menampilkan menu pada aplikasi *microcontroller*. Sumber cahaya di dalam sebuah perangkat LCD adalah lampu neon berwarna putih dibagian belakang susunan kristal cair tadi. Titik cahaya yang jumlahnya puluhan ribu bahkan jutaan inilah yang membentuk tampilan. Kutub kristal cair yang dilewati arus listrik akan berubah karena pengaruh polarisasi medan magnetik yang timbul dan oleh karenanya akan hanya beberapa warna. Gambar 2.4 merupakan bentuk fisik dari LCD 16x2 tipe HD44780.



Gambar 2.4 *Liquid Crystal Diplay* 16x2

LCD membutuhkan *driver* supaya bisa dikoneksikan dengan sistem minimum dalam suatu *microcontroller*. *Driver* yang disebutkan berisi rangkaian pengaman, pengatur tingkat kecerahan maupun data, serta untuk mempermudah pemasangan di *microcontroller*. Berikut daftar pin dari LCD 16x2 menurut *datasheet* pada Tabel 2.1

Tabel 2.1 Konfigurasi Pin LCD 16x2

PIN	Simbol	Fungsi
1	Vss	*Power Supply* 0 Volt (*ground*)
2	Vdd/VCC	*Power Supply* VCC
3	Vee	Seting kontras
4	RS	0: instruksi *input* / 1: data *input*
5	R/W	0: tulis ke LCD / 1: membaca dari LCD
6	E	Mengaktifkan sinyal
7	DB0	Data pin 0
8	DB1	Data pin 1
9	DB2	Data pin 2
10	DB3	Data pin 3
11	DB4	Data pin 4
12	DB5	Data pin 5
13	DB6	Data pin 6
14	DB7	Data pin 7
15	VB+	*Power* 5 Volt (VCC) Lampu latar (jika ada)
16	VB-	*Power* 0 Volt (*ground*) Lampu latar (jika ada)

2.7 *Real Time Clock* (RTC) [8]

Real Time Clock (RTC) adalah jenis pewaktu yang bekerja berdasarkan waktu yang sebenarnya atau dengan kata lain berdasarkan waktu yang ada pada jam kita. Meskipun istilah sering mengacu pada perangkat di komputer pribadi, *server* dan *embedded system,* RTC hadir di hampir semua perangkat elektronik yang perlu untuk menjaga keakuratan waktu.

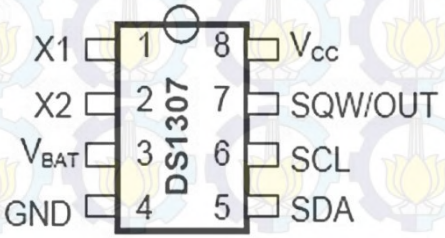
RTC memiliki sumber tenaga alternatif, sehingga mereka dapat terus menjaga waktu sementara sumber utama daya mati atau tidak tersedia. Sumber tenaga alternatif ini biasanya berupa baterai *lithium* dalam sistem lama, tetapi beberapa sistem yang lebih baru menggunakan *supercapacitor*, karena mereka dapat diisi ulang dan dapat disolder. Sumber daya alternatif juga dapat menyalurkan listrik ke *RAM* yang didukung baterai. Pada umumnya tenaga alternatif yang digunakan sebesar 3 Volt dari baterai *lithium*.

Kebanyakan RTC menggunakan osilator kristal, tetapi beberapa menggunakan frekuensi saluran listrik. Dalam banyak kasus frekuensi osilator yang digunakan adalah 32,768 kHz. Frekuensi ini sama dengan yang digunakan dalam jam kuarsa dan jam tangan, selain itu frekuensi yang dihasilkan adalah persis 215 siklus per detik, yang merupakan tingkat nyaman untuk digunakan dengan sirkuit biner sederhana. RTC ditunjukkan pada Gambar 2.5

Gambar 2.5 *Real Time Clock*

DS1307 merupakan salah satu tipe IC RTC yang dapat bekerja dalam daya listrik rendah. Di dalamnya berisi waktu jam dan kalender dalam format BCD. Waktu jam dan kalender memberikan informasi detik, menit, jam, hari, tanggal, bulan, dan tahun. Untuk bagian jam dapat berformat 24 jam atau 12 jam. Pendeteksi sumber listrik juga disediakan untuk mendeteksi kegagalan sumber listrik dan langsung mengalihkannya ke sumber baterai. DS1307 membutuhkan konsumsi

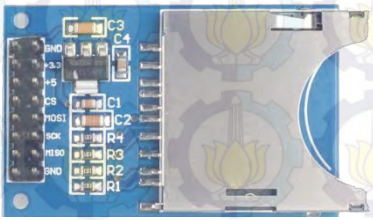
tegangan yang sangat kecil ketika tidak mendapatkan *supply*. DS1307 ditunjukkan pada Gambar 2.6



Gambar 2.6 DS1307

2.8 MMC *Shield* [8]

MMC *Shield* adalah sebuah rangkaian yang digunakan untuk menyimpan data kedalam sebuah media. Pada alat ini media yang digunakan adalah *SD Card*. Data yang akan disimpan adalah hasil pembacaan sensor yang telah diproses terlebih dahulu oleh Arduino sehingga nilai yang didapat akan disimpan dalam bentuk teks ataupun dalam *excel*. Bentuk fisik MMC *Shield* yang digunakan dapat dilihat pada Gambar 2.7



Gambar 2.7 MMC *Shield*

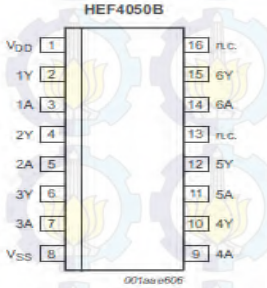
MMC *Shield* ini nanti akan menggunakan *micro sd* dengan kapasitas 8 Gb. Fungsi dari *SD Card* ini sebagai penyimpan data dari sensor tegangan dan arus yang dilengkapi dengan data waktu yang didapat dari rangkaian RTC diatas. MMC *Shield* itu sendiri memiliki beberapa pin yang nantinya akan dihubungkan dengan pin dari Arduino, diantaranya pin GND, pin 3,3 Volt, pin VCC, pin Miso, pin Mosi, pin SCK, dan pin CS.

MMC *Shield* hanya bekerja pada level tegangan 3,3 Volt, jika langsung dihubungkan dengan Arduino yang mengeluarkan tegangan 5

Volt, akan merusak *micro sd*. Maka digunakan pengkondisi dengan IC 4050 yang sering disebut *level shifter*. Konfigurasi pin IC 4050 yang digunakan ditunjukkan pada Tabel 2.2 , dan untuk gambar IC 4050 ditunjukkan pada Gambar 2.8

Tabel 2.2 Konfigurasi Pin IC 4050

PIN	Fungsi
Pin 1	Sebagai VCC
Pin 2	Sebagai *Ground*
Pin 9	Sebagai SCK Arduino
Pin 10	Sebagai SCK *SD Card*
Pin 11	Sebagai MOSI Arduino
Pin 12	Sebagai MOSI *SD Card*
Pin 14	Sebagai Pin 4 Arduino
Pin 15	Sebagai CS *SD Card*



Gambar 2.8 IC 4050

2.9 *SD Card* [8]

SD Card adalah salah satu media penyimpanan yang berupa kartu *flash*. Yang dimaksud dengan kartu memori *flash* adalah kartu memori yang data yang ditulis didalamnya dapat tersimpan tanpa membutuhkan suplai listrik.

Secara fisik, *SD Card* dilengkapi dengan sebuah kunci. Kunci ini digunakan sebagai perlindungan akses isi dari *SD Card*. Apabila dalam kondisi tidak terkunci, *SD Card* dapat dilihat isinya maupun ditambahi isi baru. Sedangkan dalam posisi terkunci, *SD Card* hanya bisa diakses untuk dilihat saja isinya. Bentuk fisik dari *SD Card* ditunjukkan pada Gambar 2.9

Gambar 2.9 Bentuk Fisik *SD Card*

Sebuah *SD Card* didalamnya terdiri atas sebuat *chip* pengontrol dan modul penyimpanan. Modul penyimpanan berfungsi sebagai tempat dimana data disimpan. Ketika kita memformat sebuah *SD Card* sebuah *file* sistem akan ditulis pada bagian ini. Sedangkan *chip* pengontrol berfungsi sebagai penghubung antara modul penyimpanan dengan alat lain misalnya *microcontroller* dengan menggunakan *SD command*.

2.10 Komunikasi Serial [8]

Untuk mengirim data dari Arduino ke PC dan menerima data dari PC kita bisa menggunakan berbagai cara salah satunya yang paling sederhana adalah komunikasi serial yang terdapat pada setiap *board* Arduino.

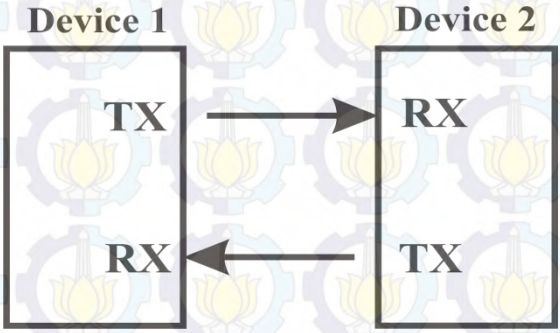
Komunikasi data serial digunakan untuk komunikasi antara *board* Arduino dengan komputer atau perangkat lain. Semua *board* Arduino mempunyai sedikitnya 1 buah *port* serial yang juga dikenal dengan nama UART atau USART. Komunikasi data serial menggunakan 2 buah pin yaitu pin RX untuk menerima data dan pin TX untuk mengirimkan data. Pada *board* Arduino pin RX terletak pada pin 0 dan pin TX terletak pada pin 1. Ketika *board* Arduino dikonfigurasikan untuk berkomunikasi secara serial, maka kedua pin 0 dan pin 1 tidak dapat digunakan sebagai pin *input* / *output* digital.

Perlu kita ketahuai pemrograman *sketch* Arduino menggunakan gaya bahasa C tapi pada pembuatan *library*-nya menggunakan C++ yang menerapkan pemrograman Objek (*Class*). Untuk itu saya sarankan anda sebaiknya mengetahuai sedikit dasar-dasar pemrograman berorientasi objek.

Pemrograman *code* Arduino (*sketch*) untuk komunikasi serial menjadi mudah karena fungsi-fungsi sudah tersedia dalam *class* yang tersedia untuk komunikasi Serial . *Instance* dari *class* untuk

komunikasi serial (objek) sudah dibuatkan namanya serial . Untuk Arduino yang mempunyai lebih dari 1 *port* serial misal Arduino Mega 2560 nama objek untuk komunikasi serialnya adalah Serial 1, Serial 2, Serial 3.

Data yang dikirim ke *port* serial akan dikirim ke *buffer* pengirim (Tx *buffer*) begitu juga data yang diterima adalah data yang diambil dari *buffer* penerima (RX *buffer*). Ilustrasi pengiriman ditunjukkan pada Gambar 2.10



Gambar 2.10 Ilustrasi Pengiriman Data

2.11 Sensor ZMPT101b [9]

Flicker adalah fluktuasi tegangan yang terjadi dalam sekejap dan dalam waktu yang berulang ulang akibat adanya beban yang tak linear. Untuk mendeteksi tegangan *flicker* yang menjadi pokok bahasan kami, sensor yang kami gunakan adalah sensor tegangan ZMPT101b. Sensor ZMPT101b ini dapat mengukur hingga tegangan 250 Volt. Maka dari itu karena kebanyakan beban tak linear yang dapat menimbulkan tegangan *flicker* adalah beban dengan daya yang besar, sehingga sensor ZMPT101b sangat cocok. Gambar sensor ZMPT101b ditunjukkan pada Gambar 2.11

Gambar 2.11 Sensor ZMPT101b

ZMPT101b merupakan sensor tegangan AC *single phase* pengguanaan sensor ini bertujuan untuk mendapatkan kondisi sinus dari tegangan AC dengan besaran tegangan yang sesuai dengan kriteria ADC yakni 0 – 5 Volt. *Output* dari sensor ini berupa gelombang sinus yang nilai *offset* sebesar 2,5 Volt dengan nilai maksimal sebesar 5 Volt. Kondisi tersebut sesuai dengan kebutuhan untuk mendeteksi *Voltage Flicker* yaitu bentuk gelombang yang sesuai dengan kondisi tegangan AC (220 Volt), serta nilai tegangan dari *output* sensor memenuhi kriteria tegangan ADC Arduino.

Sensor ZMPT101b juga memiliki rangkaian *operational amplifier* didalamnya. Sehingga dapat mengambil sampel dengan sangat akurat. Spesifikasi sensor ZMPT101b ditunjukkan pada Tabel 2.3

Tabel 2.3 Spesifikasi ZMPT101b

Parameter	Spesifikasi
Arus primer	2 mA
Arus sekunder	2 mA
Eror sudut fasa	$<20^{o}$ (50 Ohm)
Jangkauan arus	0-3 mA
Linearitas	0,1%
Akurasi 0,2	0,2
Level dielektrik	3000 VAC/Min
Resistansi DC pada suhu 20^{o} C	110 Ω

2.12 Sensor Arus Trafo CT-OD [10]

Sensor arus merupakan suatu piranti yang digunakan untuk mengukur besaran arus pada suatu sistem listrik. Sensor ini mengubah arus menjadi suatu besaran listrik (arus / tegangan) yang lebih proporsional untuk selanjutnya digunakan untuk keperluan *metering* maupun proteksi. Contoh sensor arus yang umum digunakan adalah adalah *Current Transformer* (CT) atau trafo arus.

Trafo arus digunakan untuk pengukuran arus yang besarnya ratusan Ampere lebih yang mengalir pada jaringan tegangan tinggi. Jika arus hendak diukur mengalir pada tegangan rendah dan besarnya dibawah 5 Ampere, maka pengukuran dapat dilakukan secara langsung sedangkan arus yang besar tadi harus dilakukan secara tidak langsung dengan menggunakan trafo arus sebutan trafo pengukuran arus yang besar.

Prinsip kerja dari CT sama dengan trafo daya satu fasa. Jika pada kumparan primer mengalir arus Ip, maka pada kumparan primer akan timbul gaya gerak magnet sebesar Np.Ip , gaya gerak magnet ini memproduksi fluks pada inti. Fluks ini membangkitkan gaya gerak listrik pada kumparan sekunder. Jika kumparan sekunder tertutup, maka pada kumparan sekunder mengalir arus Is. arus ini menimbulkan gaya gerak magnet Ns.Is pada kumparan sekunder.

Jenis CT yang dipakai adalah *Induction Current Sensor* seperti pada Gambar 2.12 yang memiliki spesifikasi seperti yang ditunjukkan pada Tabel 2.4

Gambar 2.12 *Induction Current Sensor*

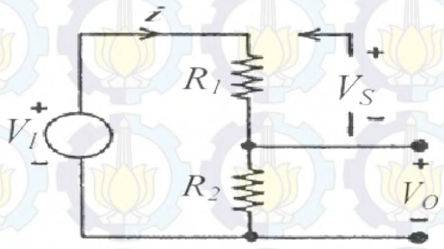
Tabel 2.4 Spesifikasi *Induction Current Sensor*

Parameter	Spesifikasi
AC/DC *Current Range*	± 60 A
Ratio	1000:1
Peak Pulse Measurement	± 60 A
Input overload limit	± 1000 A
Accuracy Offset-Error	< 0,1 %
Linearity-Error for $\|I\| \leq Imax$	< 0,5 %
Dynamic characteristics	> 1000 A/µs
Output voltage to current ratio at 50 Ω/1W in line	1 mA/1A

2.13 Sensor Tegangan [10]

Sinyal tegangan yang akan diukur dengan menurunkan tegangan dengan menggunakan transformator *step down* dan rangkaian pembagi tegangan dimana resistor disusun secara seri. Prinsip dari rangkaian pembagi tegangan sesuai dengan hukum Kirchoff tegangan yang menyatakan bahwa "Tegangan dalam rangkaian tertutup sama dengan jumlah semua tegangan di seluruh rangkaian". Dari Gambar 2.13 R_1 dan

R_2 dipasang secara seri, di mana tegangan keluaran (V*out*) adalah tegangan R_2.



Gambar 2.13 Rangkaian Pembagi Tegangan

Dari rangkaian pembagi tegangan diatas dapat dirumuskan tegangan *output* V_O. Arus (I) mengalir pada R_1 dan R_2 sehingga nilai tegangan sumber V_I adalah penjumlahan V_S dan V_O sehingga dapat dirumuskan sebagai berikut.

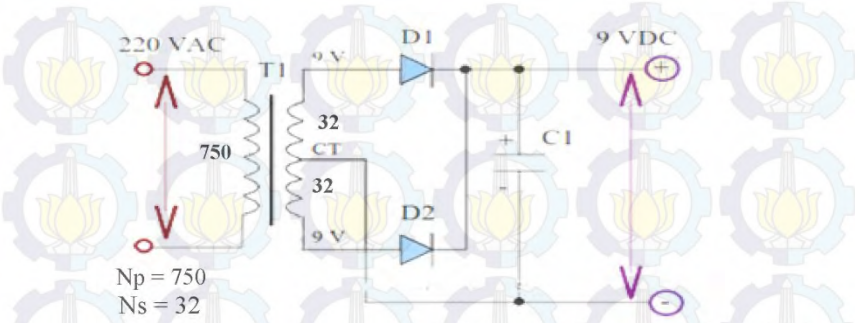
$$V_I = V_S + V_O = i.R_1 + i.R_2 \quad \text{.....(2.3)}$$

Nampak bahwa tegangan masukan terbagi menjadi dua bagian, masing-masing sebanding dengan harga resistor yang dikenai tegangan tersebut. Sehingga besarnya V_O dapat dirumuskan sebagai berikut.

$$V_O = V_I.(\frac{R_2}{R_1 + R_2}) \quad \text{.....(2.4)}$$

Sensor tegangan merupakan alat yang digunakan untuk mendeteksi besar tegangan yang melalui suatu peralatan listrik. Sensor tegangan menggunakan transformator dan rangkaian penyearah. Prinsip kerja dari sebuah transformator adalah ketika kumparan primer dihubungkan dengan sumber tegangan bolak-balik, perubahan arus listrik pada kumparan primer menimbulkan medan magnet yang berubah. Medan magnet yang berubah diperkuat oleh adanya inti besi dan dihantarkan inti besi ke kumparan sekunder, sehingga pada ujung-ujung kumparan sekunder akan timbul ggl induksi. Efek ini dinamakan induktansi timbal-balik (*mutual inductance*). Jika efisiensi sempurna, semua daya pada lilitan primer akan dilimpahkan ke lilitan sekunder. Penyearah gelombang penuh dapat dibuat dengan 2 macam yaitu, menggunakan 4 dioda dan 2 dioda. Untuk membuat penyearah gelombang penuh dengan

2 dioda menggunakan transformator CT seperti terlihat pada Gambar 2.14



Gambar 2.14 Rangkaian Penyearah Gelombang Penuh

Prinsip kerja rangkaian penyearah gelombang penuh dengan 2 dioda ini dapat bekerja karena menggunakan transformator dengan CT. Transformator dengan CT seperti pada gambar diatas dapat memberikan *output* tegangan AC pada kedua terminal *output* sekunder terhadap terminal CT dengan level tegangan yang berbeda fasa 180°. Pada saat terminal *output* transformator pada D1 memberikan sinyal puncak positif maka terminal *output* pada D2 memberikan sinyal puncak negatif, pada kondisi ini D1 pada posisi *forward* dan D2 pada posisi *reverse*. Sehingga sisi puncak positif dilewatkan melalui D1. Kemudian pada saat terminal *output* transformator pada D1 memberikan sinyal puncak negatif maka terminal *output* pada D2 memberikan sinyal puncak positif, pada kondisi ini D1 posisi *reverse* dan D2 pada posisi *forward*. Sehingga sinyal puncak positif dilewatkan melalui D2. Penggunaan transformator jenis CT dapat mengurangi penggunaan dari dioda yang digunakan. *Output* dari sensor tegangan ini masih terdapat *ripple* yang dihasilkan, *ripple* dapat diperhalus dengan cara memberikan kapasitor pada *output* dari sensor tegangan, karena kapasitor sesuai dengan karakteristiknya yang dapat menyimpan tegangan sampai Vcap = Vsumber, dan akan menjadi sumber tegangan ketika kapasitor telah terisi penuh.

2.14 *Miniature Circuit Breaker* (MCB) [10]

Mini Circuit Breaker (MCB) memiliki fungsi sebagai alat pengaman arus lebih. MCB ini memproteksi arus lebih yang disebabkan terjadinya beban lebih dan arus lebih karena adanya hubungan pendek.

Prinsip dasar kerjanya yaitu untuk pemutusan hubungan yang disebabkan beban lebih dengan relai arus lebih sesaat menggunakan elektromagnet. MCB ditunjukkan pada Gambar 2.15

Gambar 2.15 *Mini Circuit Breaker*

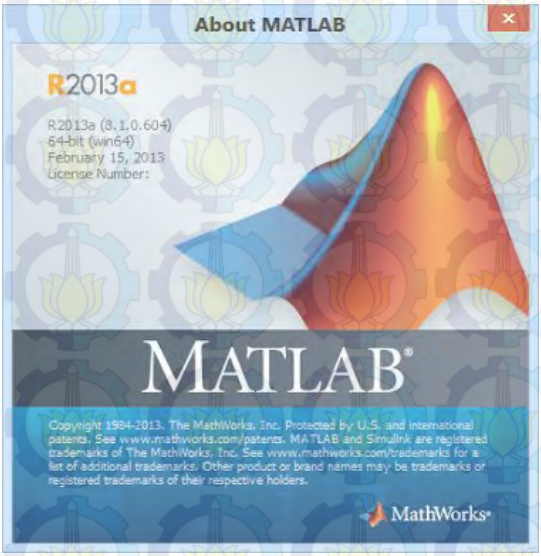
Bila elektromagnet bekerja, maka akan memutus hubungan kontak yang terletak pada pemadam busur dan membuka saklar. MCB untuk rumah seperti pada pengaman lebur diutamakan untuk proteksi hubungan pendek, sehingga pemakaiannya lebih diutamakan untuk mengamankan instalasi atau konduktornya. Arus nominal yang digunakan pada APP dengan mengenal tegangan 230/400 V ialah: 1 A, 2 A, 4 A, 6 A, 10 A, 16 A, 20 A, 25 A, 35 A, dan 50 A disesuaikan dengan tingkat VA konsumen. Adapun kemampuan membuka (*breaking capacity*) bila terjadi hubung singkat 3 KA dan 6 KA (SPLN 108-1993). MCB yang khusus digunakan oleh PLN mempunyai tombol biru. MCB pada saat sekarang ini paling banyak digunakan untuk instalasi rumah, instalasi industri maupun instalasi gedung bertingkat.

2.15 *Software* Matlab [11]

Matlab (*Matrix Laboratory*) adalah suatu bahasa tingkat tinggi yang digunakan untuk komputasi teknik. Bahasa ini mengintegrasikan proses komputasi, visualisasi, dan pemrograman dengan *environment* yang mudah digunakan dengan mengekspresikan masalah dan solusi ke dalam notasi-notasi matematika. Kegunaan umum dari Matlab diantaranya untuk Matematika dan Komputasi, Pengembangan Algoritma, Akuisisi Data, Pemodelan dan Simulasi, Pembuatan

Prototipe, Analisis Data, Eksplorasi, Visualisasi, dan Pengembangan Aplikasi termasuk GUI.

Bahasa Matlab dapat digunakan dalam sebuah sistem Matlab interaktif yang elemen data dasarnya adalah *array* yang tidak membutuhkan pengaturan dimensi. Hal ini memungkinkan penyelesaian banyak masalah komputasi teknik, terutama yang berhubungan dengan formulasi matriks dan vektor. Logo *software* Matlab ditunjukan pada Gambar 2.16



Gambar 2.16 Matlab

Sebagai sebuah sistem, Matlab tersusun dari 5 bagian utama:

Development Environment, merupakan sekumpulan perangkat dan fasilitas yang membantu kita untuk menggunakan fungsi-fungsi dan *file-file* MATLAB. Beberapa perangkat ini merupakan sebuah *Graphical User Interfaces* (GUI). Termasuk didalamnya adalah *Matlab desktop* dan *Command Window*, *Command History*, sebuah editor dan *debugger*, dan *browsers* untuk melihat *help*, *workspace, files*, dan *search path.*

Matlab Mathematical Function Library, merupakan sekumpulan algoritma komputasi mulai dari fungsi-fungsi dasar sepertri: sum, sin, cos, dan *complex arithmetic*, sampai dengan fungsi-fungsi yang lebih

komplek seperti *matrix inverse*, *matrix eigenvalues*, *bessel functions*, dan *Fast Fourier Transforms*.

Matlab *Language*, merupakan suatu *high-level matrix/array language* dengan *control flow statements, functions, data structures, input/output,* dan *fitur-fitur object-oriented programming*. Ini memungkinkan bagi kita untuk melakukan kedua hal baik "pemrograman dalam lingkup sederhana " untuk mendapatkan hasil yang cepat, dan "pemrograman dalam lingkup yang lebih besar" untuk memperoleh hasil-hasil dan aplikasi yang komplek.

Graphics, Matlab memiliki fasilitas untuk menampilkan *vector* dan *matrices* sebagai suatu grafik. Di dalamnya melibatkan *high-level functions* (fungsi-fungsi level tinggi) untuk visualisasi data dua dimensi dan data tiga dimensi, *image processing, animation*, dan *presentation graphics*. Ini juga melibatkan fungsi level rendah yang memungkinkan bagi kita untuk membiasakan diri untuk memunculkan grafik mulai dari bentuk yang sederhana sampai dengan tingkatan *graphical user interfaces* pada aplikasi Matlab.

Matlab Application Program Interface (API), merupakan suatu *library* yang memungkinkan program yang telah kita tulis dalam bahasa *C* dan *Fortran* mampu berinterakasi dengan Matlab. Ini melibatkan fasilitas untuk pemanggilan *routines* dari Matlab (*dynamic linking*), pemanggilan Matlab sebagai sebuah *computational engine*, dan untuk membaca dan menuliskan *MAT-files.*

BAB III
PERENCANAAN ALAT

Pada bab ini akan dibahas mengenai perancangan serta pembuatan "Perancangan Sistem *Monitoring Voltage Flicker* Berbasis Arduino Dengan Metode *Fast Fourier Transform* (FFT)", baik perancangan perangkat elektronik (*hadware*), perancangan dan pembuatan perangkat lunak (*software)* yang meliputi :

1. Perancangan Mekanik
2. Perancangan *Hardware* terdiri dari :
    a. Perancangan Sensor Tegangan
    b. Perancangan Sensor Arus
    c. Perancanagan LCD
    d. Perancangan *Power Supply*
    e. Perancangan Rangkaian RTC
    f. Perancangan Rangkaian *SD Card*
    g. Perancangan *Shield* Arduino Mega
3. Perancangan *Software* yang berupa *flowchart* terdiri dari :
    a. Pemograman Arduino IDE
    b. Pemograman Matlab

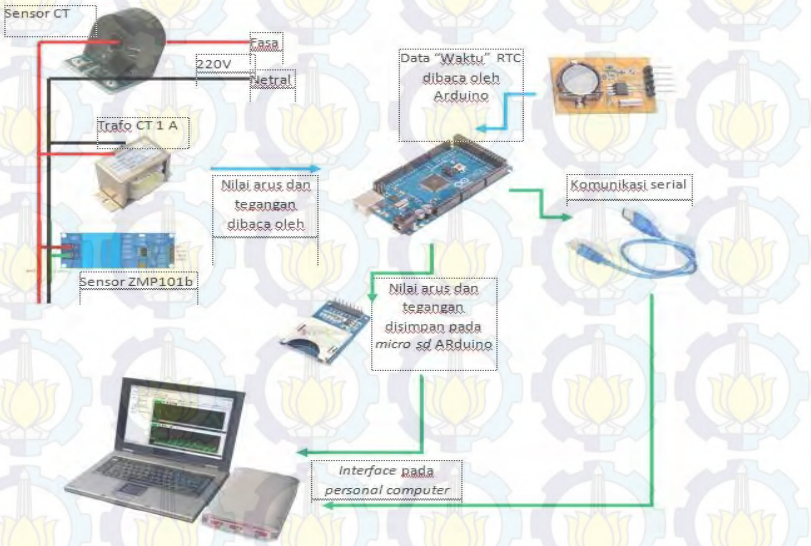
3.1 Diagram Fungsional Alat

Perencanaan Tugas Akhir "Perancangan Sistem *Monitoring Voltage Flicker* Berbasis Arduino Dengan Metode *Fast Fourier Transform* (FFT)" ini mengenai sistem kerja alat secara keseluruhan. Dimana pada alat ini mempunyai dua fungsi yaitu sebagai *monitoring* beban yang dapat meyebabkan terjadinya *Voltage Flicker* serta dapat digunakan untuk menyimpan kondisi tegangan dan arus pada kondisi normal. Kedua fungsi tidak dapat digunakan secara bersamaan, karena ketika digunakan secara bersama maka akan mengganggu proses *sampling* sensor tegangan untuk mendeteksi adanya V*oltage Flicker*.

Sistem kerja dari alat untuk *monitoring Voltage Flicker* yakni diawali dengan melakukan pembacaan sensor tegangan ZMPT101B, kemudian tegangan *output* dari sensor akan dibaca oleh ADC Arduino dan akan diolah menjadi nilai tegangan, lalu hasil nilai tegangan akan dikirim ke *personal computer* dengan komunikasi serial. Pada *personal computer* akan dilakukan pengolahan data tegangan menggunakan *software* Matlab sehingga mendapatkan garfik sinus serta grafik frekuensi yang menjadi indikasi ada atau tidaknya *Voltage Flicker.*

Sedangkan untuk sistem kerja pada kondisi normal yakni nilai tegangan dikonversi oleh sensor tegangan serta nilai arus dikonversi oleh sensor arus, kemudian *output* dari sensor akan dibaca oleh ADC Arduino dan diolah menjadi nilai tegangan dan arus yang sebenarnya. Setelah pembacaan sensor telah dilakukan maka Arduino akan mengambil data dari RTC yang berupa data waktu. Setalah itu nilai arus, nilai tegangan, serta data RTC akan disimpan dalam *SD Card* dalam bentuk file *excel*.

Diagram fungsional alat secara keseluruhan dapat dilihat pada Gambar 3.1



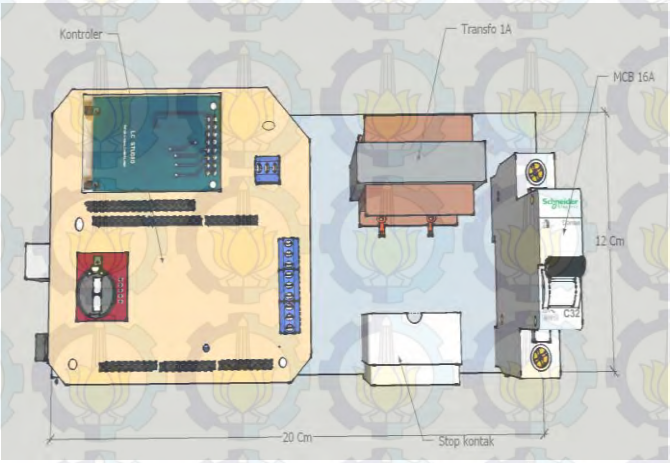
Gambar 3.1 Skema Sistem Keseluruhan

3.2 Perancangan Mekanik

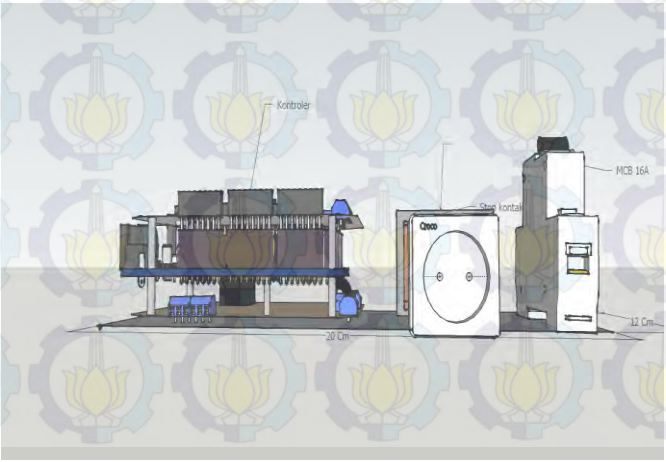
Perancangan mekanik pada Tugas Akhir ini terbagi menjadi 2 bagian, yaitu perancangan pada bagian sistem dan perancangan dari panel kontrol.

Pada perancangan sistem merupakan bagian yang terdiri dari komponen penyusun dari Tugas Akhir ini, yang diantaranya sensor tegangan, sensor arus, Arduino, rangkaian *Power Supply*, rangkaian RTC, rangkaian *SD Card,* MCB serta transformator. Seluruh komponen tersebut diletakkan pada akrilik yang ditunjukkan pada Gambar 3.2 untuk tampak atas dan Gambar 3.3 untuk tampak depan.

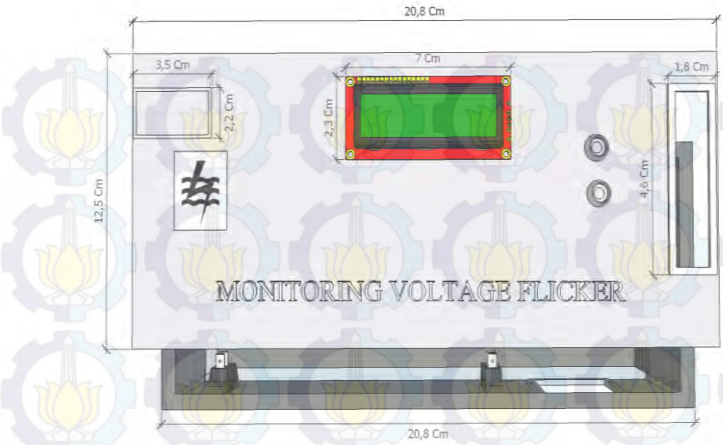
Sedangkan panel kontrol pada Tugas Akhir ini dirancang untuk meletakkan bagian sistem dan juga terdapat indikator serta saklar untuk menjalankan sistem secara keseluruhan. Rancangan panel kontrol ditunjukkan pada Gambar 3.4 untuk tampak atas, dan Gambar 3.5 untuk tampak depan.



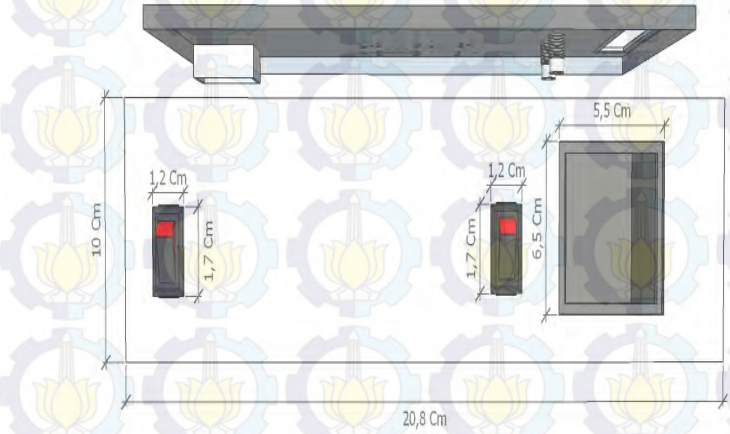
Gambar 3.2 Perancangan Alat Tampak Atas



Gambar 3.3 Perancangan Alat Tampak Depan



Gambar 3.4 Perancangan Panel Kontrol Tampak Atas

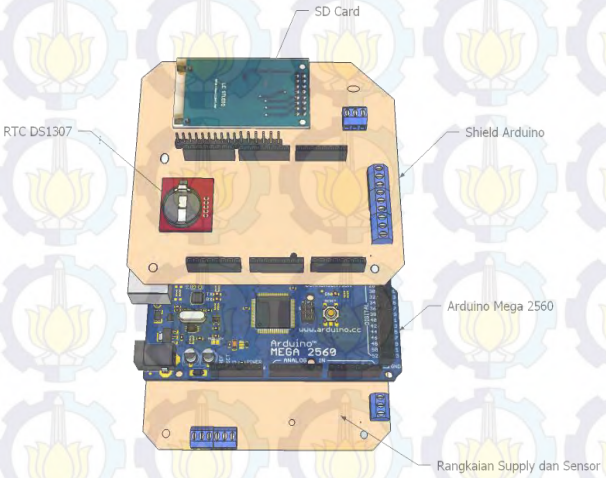


Gambar 3.5 Perancangan Panel Kontrol Tampak Depan

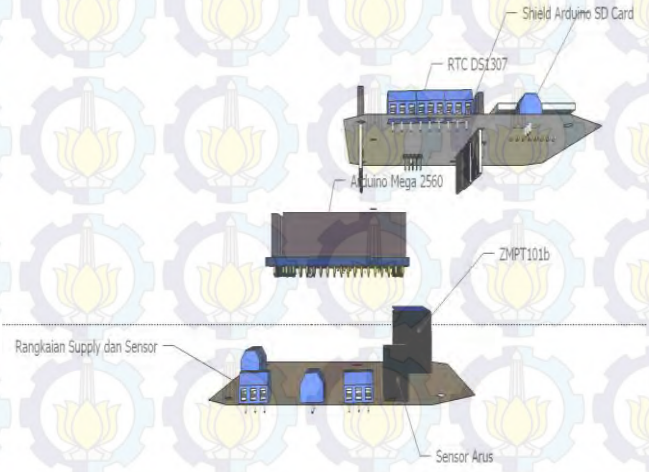
3.3 Perancangan Perangkat Elektronik

Pada perancangan perangkat elektronik (*hardware)* pada Tugas Akhir ini meliputi perancangan sensor tegangan, sensor arus CT, *Power Supply,* rangkaian RTC, rangkaian *SD Card*, rangkaian LCD 16x2,

MCB. Tampilan perancangan perangkat elektronik terdapat pada Gambar 3.6 untuk tampak atas, dan Gambar 3.7 untuk tampak samping.



Gambar 3.6 Perancangan *Hardware* Tampak Atas



Gambar 3.7 Perancangan *Hardware* Tampak Samping

Susunan dari perancangan elektronik terbagi menjadi 3 bagian, yaitu, terdapat rangkaian *Power Supply* dan sensor pada bagian bawah, sedangkan pada susunan kedua terdapat papan Arduino sebagai kontrol pada Tugas Akhir ini, pada susunan ketiga terdapat rangkaian RTC dan *SD Card* yang tersusun menjadi satu dengan *shield* dari papan Arduino.

Tata letak dari komponen-komponen sistem control ini dibuat seperti demikian untuk mempermudah dalam pengoperasiaannya. Selain itu dengan diletakkan menjadi satu *board* seperti Gambar 3.6 dan Gambar 3.7 diatas akan menjadi lebih rapi dan efisien.

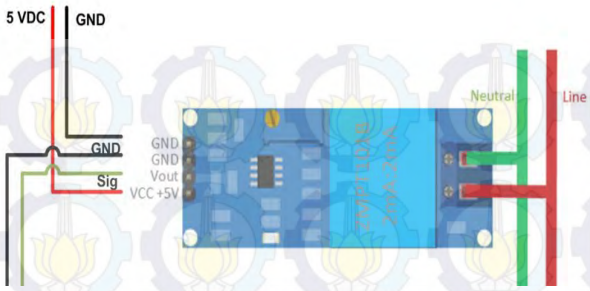
Mengenai komponen – komponen yang digunakan akan dijelaskan pada sub bab ini.

3.3.1 Perencanaan Sensor Tegangan

Tugas Akhir ini menggunakan dua sensor tegangan yakni sensor tegangan ZMPT101b serta sensor tegangan menggunakan transformator *stepdown* untuk menurunkan nilai tegangan kerja yang akan diukur besarannya. Penggunaan dua jenis sensor tegangan yakni berdasarkan fungsi dan *output* dari sensor tersebut. Untuk ZMPT101b digunakan untuk *mode sampling* untuk kondisi *monitoring Voltage Flicker, output* dari ZMPT101b berupa gelombang sinus dengan *offset* 2,5 Volt. Sedangkan untuk sensor tegangan menggunakan transformator *stepdown* digunakan untuk *monitoring* keadaan tegangan pada kondisi normal.

3.3.1.1 Sensor Tegangan ZMPT101b

ZMPT101b merupakan sensor teganga AC *single phase* pengguanaan sensor ini bertujuan untuk mendapatkan kondisi sinus dari tegangan AC dengan besaran tegangan yang sesuai dengan kriteria ADC yakni 0 – 5 Volt. *Output* dari sensor ini berupa gelombang sinus yang nilai *offset* sebesar 2,5 Volt dengan nilai maksimal sebesar 5 Volt. Kondisi tersebut sesuai dengan kebutuhan untuk mendeteksi *Voltage Flicker* yaitu bentuk gelombang yang sesuai dengan kondisi tegangan AC (220 Volt), serta nilai tegangan dari keluaran sensor memenuhi kriteria tegangan ADC Arduino. Selain itu untuk dimensi dari sensor ZMPT101b ini relatif lebih kecil yaitu 49,5 mm X 19,4 mm membuat sensor ini lebih *compact*. Sensor tegangan ZMPT101b memiliki rating arus *primary* dan arus *secondary* sebesar 2 mA serta dapat dioperasikan pada suhu -40^0 C hingga 130^0 C. Sensor tegangan ZMPT101b ditunjukkan oleh Gambar 3.8



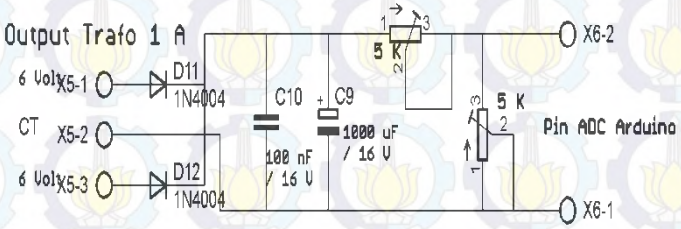
Gambar 3.8 Sensor Tegangan ZMPT101b

3.3.1.2 Sensor Tegangan Trafo *Stepdown*

Penggunaan transformator *stepdown* pada sensor tegangan ini bertujan untuk menurunkan nilai tegangan kerja (220 Volt) yang digunakan.

Nilai dari *output* transformator *stepdown* sebesar 6,45 Volt nilai tersebut masih terlalu besar untuk digunakan sebagai sensor yang akan dibaca oleh ADC Arduino maka setelah tegangan diturunkan diberi rangkaian pembagi tegangan yakni terdiri dari dua buah resistor variabel dengan besar resistansi 10 kΩ, sehingga nilai tegangan 6,45 Volt dapat diatur menjadi 4,5 Volt untuk tegangan 250 Volt.

Pada kondisi tersebut tegangan masih berupa AC dan bentuk gelombangnya sinus, kondisi tersebut perlu diberi penyearah yakni berupa rangkaian *fullwave rectifier* dengan dioda 1N4004. Sebelum data dikirim ke Arduino, *output* dari rangkaian penyearah diberi kapasitor untuk mengurangi *ripple* yang terjadi, nilai kapasitor yang digunakan yakni 1000 µF. Skema rangkaian sensor tegangan ditunjukan oleh Gambar 3.9



Gambar 3.9 Skematik Sensor Tegangan

Sensor tegangan tersebut dirancang dengan menggunakan prinsip pembagian tegangan dari tegangan sekunder pada trafo *stepdown* 350 mA.

Adapun Persamaan yang digunakan seperti pada Persamaan 3.1 berikut.

$$V_2 = \frac{R_2 \, x \, V_1}{R_1 + R_2} \quad \text{.....(3.1)}$$

R_2 = 10 kΩ
R_1 = 10 k Ω
V_2= Tegangan *output* dari Sensor (V)
V_1= Tegangan sekunder dari transformator *stepdown* (V)

Sehingga didapat V_2 sebesar 4,5 Volt ketika:

$$V_2 = \frac{5600\ \Omega}{2400\ \Omega + 5600\ \Omega} \text{ x } 6{,}45 \text{ Volt}$$

V_2= 4,515 Volt.

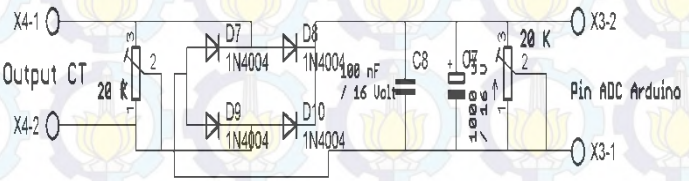
3.3.2 Perancangan Sensor Arus CT

Perancangan sensor arus pada Tugas Akhir ini menggunakan sebuah *current transformers* yang nantinya data arus yang diukur menjadi *output* berupa arus tetapi dalam satuan mA. *Current transformers* ini mampu mengukur hingga 60 A. Karena *output* dari *current transformers* ini masih dalam besaran arus maka perlu ditambahkan rangkaian pengkondisi. Gambar sensor arus *current transformers* ditunjukkan pada Gambar 2.12

Jenis sensor ini memenuhi spesifikasi untuk pengukuran pada KWH meter milik PT. PLN (Persero) dengan rating arus ± 60A. Dari dimensinya terlihat simpel dari pada sensor arus yang lain. Spesifikasi lengkap dari sensor tersebut terdapat pada Tabel 2.4

ADC dari Arduino hanya mampu membaca data sensor yang berupa besaran tegangan 0 – 5 Volt, maka diperlukan rangkaian pengkondisi agar output dari sensor arus ini dapat terbaca oleh ADC Arduino. Dengan rangkaian tersebut tegangan output yang dihasilkan oleh sensor arus current transformers akan bisa diatur oleh multitune (Variabel Resistor). Resistor yang pertama digunakan sebagai pengatur tegangan output dari sensor arus current transformers. Kemudian resistor yang kedua digunakan untuk mengatur delta (Δ) atau linieritas yang

diperlukan oleh sebuah microcontroller untuk mampu merubah nilai berapapun hingga arus tertentu. Skema rangkaian ditunjukkan pada Gambar 3.10



Gambar 3.10 Skematik Rangkaian Pengkondisi CT

Untuk resistor variabel pada Tugas Akhir ini digunakan sebesar 20 kΩ untuk *output* dari sensor, dan pada resistor variabel yang mengatur linearitas sensor sebesar 240 Ω. Jadi dalam 1 A arus yang melewati sensor maka *output* dari sensor sebesar 240 mV. Sedangkan untuk mengurangi *ripple* yang timbul maka diberi kapasitor sebesar 1000 µF, dan sebesar 100 nF. Arus yang keluar dari sensor tersebut masih dalam besaran arus AC yang dimana gelombangnya masih berupa sinus, oleh karena itu diberi rangkaian *fullwave rectifier* agar kondisi dari *output* sensor menjadi DC, dan mempermudah pembacaan sensor oleh ADC dari papan Arduino. ADC yang digunakan untuk pembacaan sensor arus menggunakan ADC A2, dimana ADC pada papan Arduino sebesar 10 bit jadi terdapat 1024 data (0 - 1023).

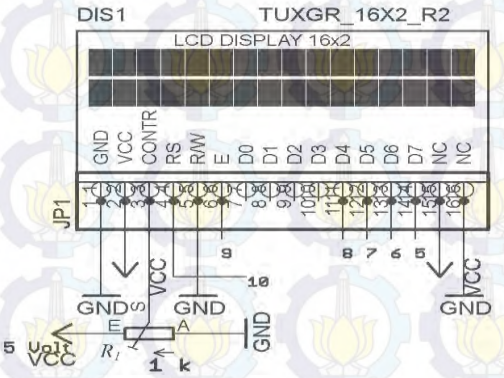
3.3.3 Perancangan LCD (*Liquid-Crystal Display*)

Pada Tugas Akhir ini menggunakan LCD yang berfungsi menampilkan karakter yang telah diproses oleh Arduino, yang bertujuan untuk menginformasikan data yang telah diproses dalam bentuk karakter.

LCD yang digunakan pada Tugas Akhir ini menggunakan LCD 16x2. Untuk 16x2 itu sendiri memiliki arti, dimana 16 mewakili kolom dan 2 baris, berarti LCD ini dapat menampilkan karakter pada sepanjang 32 buah dengan masing-masing kolom maksimal berisi 16 buah karakter.

LCD ini membutuhkan *Power Supply* sebesar 5 Volt DC yang didapat dari Arduino dan pin yang digunakan pada LCD pada alat ini yaitu pin CONTR, RS, RW, E, D4, D5, D6, D7. Pada pin LCD yang

digunakan di hubungkan dengan pin yang ada di Arduino yaitu pin CONTR dihubungkan dengan potensio 1 kΩ untuk mengatur kecerahan pada karakter yang di tampilkan LCD pin RS, E, D4, D5, D6 dan D7 di hubungkan pada pin 10, 9, 8, 7, 6 dan 5 secara berurutan dan pin RS di hubungkan dengan *ground*. Rangkaian LCD ini ditunjukkan pada Gambar 3.11 berikut.



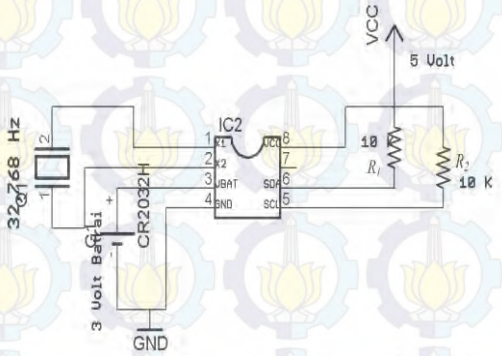
Gambar 3.11 Skema Rangkaian LCD 16x2

3.3.4 Perancangan RTC

Rangkaian RTC (*Real Time Clock)* yang digunakan pada Tugas Akhir ini ialah DS1307 yang merupakan IC untuk *real-time clock*, dimana IC ini dapat menghitung dari detik, menit, jam, hari, bulan, dan tahun. *Port* yang digunakan untuk *interface* dengan *microcontroller* juga mudah yakni menggunakan I^2C serial *interface.* Pengunaan RTC pada Arduino telah terdapat *library* pada pemrogramannya, pengguna hanya memasukkan *library* RTC kedalam program. Konsumsi baterai ketika sumber tidak aktif pun juga sangat kecil yakni kurang dari 500 nA. IC ini memiliki 8 pin jadi ukurannya pun kecil, dan dapat bekerja pada suhu -40^O C sampai $+85^O$ C. Untuk lebih jelasnya tentang IC DS1307 ditunjukkan pada Gambar 2.6

Pada Tugas Akhir ini rangkaian RTC dihubungkan dengan pin Arduino untuk SDA dan SCL pada pin 20 dan 21 sedangkan untuk VCC dihubungkan pada sumber dari Arduino yakni 5 Volt. IC ini juga memiliki pin untuk baterai dimana fungsi dari baterai tersebut digunakan ketika VCC tidak aktif maka IC DS1307 masih mendapatkan sumber dari baterai untuk aktif dan menghitung data waktu jadi data waktu

masih tersimpan meskipun Arduino dalam keadaaan tidak aktif. Untuk X1 dan X2 dihubungkan dengan kristal 32,768 KHz, untuk jelasnya ditunjukkan pada Gambar 3.12



Gambar 3.12 Rangkaian RTC DS1307

3.3.5 Perancangan *SD Card*

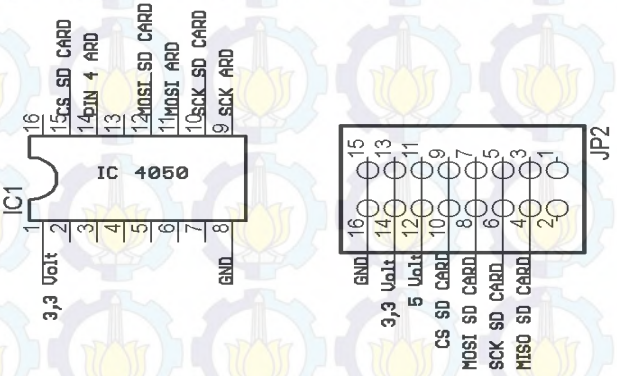
Untuk rangkaian *SD Card* pada Tugas Akhir ini menggunakan MMC *Shield*, tetapi pada dasarnya menggunkan *micro sd* dengan kapasitas 8 Gb. Fungsi dari *SD Card* ini sebagai penyimpan data dari sensor tegangan dan arus yang dilengkapi dengan data waktu yang didapat dari rangkaian RTC diatas.

MMC *Shield* itu sendiri memiliki beberapa pin yang nantinya akan dihubungkan dengan pin dari Arduino, diantaranya pin GND, pin 3,3 Volt, pin VCC, pin Miso, Pin Mosi, Pin SCK, dan pin CS. MMC *Shield* tidak dapat langsung dihubungkan dengan pin Arduino karena *SD Card* bekerja pada tegangan 3,3 Volt, sedangkan untuk pin Arduino memiliki tegangan 5 Volt. Jika tetap menghubungkan langsung ke pin Arduino maka dapat merusak *SD Card* itu sendiri. Oleh karena itu dibutuhkan rangkaian pengkondisi agar tegangan yang masuk ke *SD Card* sebesar 3,3 Volt. Konfigurasi MMC *Shield* ditunjukkan pada Gambar 2.7.

Jadi solusinya menggunakan tegangan 5 Volt sebagai *input* dan mendapatkan tegangan 3,3 Volt pada *output*, hal tersebut sering disebut sebagai *level shifter*. Dengan IC 4050 sebagai *level shifter,* memiliki 16 pin, konfigurasi pin IC 4050 ditunjukkan pada Gambar 2.8.

Dimana pin 1 (VDD) merupakan sumber tegangan, sedangkan pin 8 (VSS) merupakan GND, untuk pin 1Y sampai 6Y merupakan *output,*

sedangkan 1A sampai 6A merupakan *input.* Jadi untuk pin dari Arduino dihubungkan pada pin *input* dari IC 4050 sedangkan untuk MMC *Shield* dihubungkan pada pin *output* dari IC 4050. Untuk skema rangkaian ditunjukkan pada Gambar 3.13



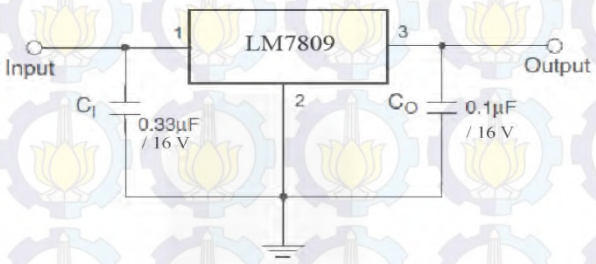
Gambar 3.13 Skematik Rangkaian *SD Card*

3.3.6 Perancangan *Power Supply*

Power Supply yang dibutuhkan pada Tugas Akhir ini memiliki *output* sebesar 9 Volt dan 5 Volt. Terdapat dua *output* yang dibutuhkan, 9 Volt digunakan untuk sumber Arduino Mega, dimana Arduino Mega direkomendasikan untuk mendapatkan sumber tegangan 7 – 12 Volt, jadi untuk tegangan 9 Volt dapat digunakan sebagai sumber Arduino. Sedangkan untuk sumber 5 Volt digunakan untuk sensor tegangan ZMPT101b, sensor tersebut memerlukan sumber 5 Volt sebagai *input* VCC pada sensor.

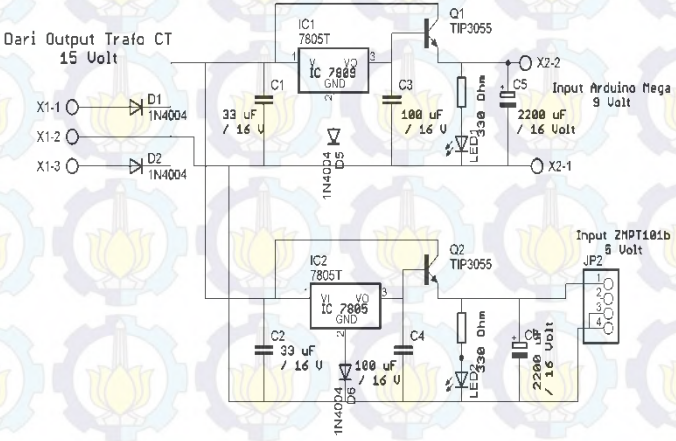
Rangkaian *Power Supply* ini menggunakan 2 buah IC regulator yakni 7805 dan 7809. 7805 memiliki *output* sebesar positif 5 Volt, dan IC 7809 memiliki *output* sebesar positif 9 Volt. Sebelum masuk ke regulator terdapat penyearah *fullwave rectifier* yakni menggunkan dua dioda 1N4004, karena *input* dioda adalah transformator *stepdown* 15 Volt. Penggunaan dua dioda karena transformator *stepdown* ini termasuk jenis CT, menggunakan tiga *input* yaitu 15 Volt, CT, 15 Volt. Selain itu perlu ditambahakan TIP 3055 agar kondisi IC regulator dapat mengalirkan arus lebih dari 1 A. Ditambahkan juga kapasitor untuk

mengurangi *ripple* yang ada. Kapasitor dipasang sebelum dan sesudah regulator seperti ditunjukan pada Gambar 3.14



Gambar 3.14 Rangkaian Regulator LM7809

Untuk rangkaian yang digunkana pada Tugas Akhir ini menggunakan kapasitor sebesar 33 µF dan 100 µF serta ditambahkan pula kapasitor sebesar 2200 µF. terdapat pula rangkaian indikator yaitu berupa led 5 mm pada setiap *output* dari regulator sebagai indikator ada atau tidaknya tegangan *output*. Trafo pada *Power Supply* ini menggunakan transformator *stepdown* jenis CT dimana hanya membutuhkan dua dioda untuk menjadi *fullwave rectifier*, skema rangkaian *Power Supply* ditunjukkan pada Gambar 3.15

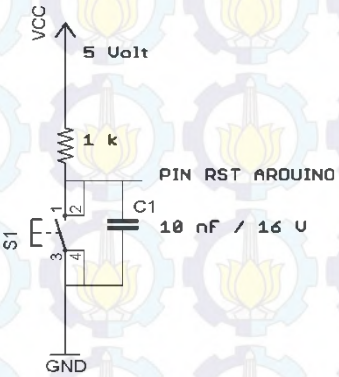


Gambar 3.15 Skema Rangkaian *Power Supply*

3.3.7 Perancangan Pin *Reset* Arduino

Penggunaan Arduino Mega pada Tugas Akhir ini tidak langsung terhubung antara pin Arduino dengan rangkaian yang lain, tetapi masih menggunakan *shield*. Oleh karena itu tombol *reset* pada papan Arduino akan tertutup oleh *shield* sehingga mempersulit keadaan ketika terjadi *error* yang membutuhkan *reset*, karena posisi dari pin *reset* terletak dibawah *shield*. Dan tidak memungkinkan pengguna untuk membuka *shield* Arduino.

Dengan kondisi tersebut maka diperlukan tombol *reset* yang terhubung dengan papan Arduino, sehingga ketika membutuhkan tombol *reset* tidak membuka *shield* yang terpasang diatas papan Arduino. Skema rangkaian *reset* ditunjukkan pada Gambar 3.16



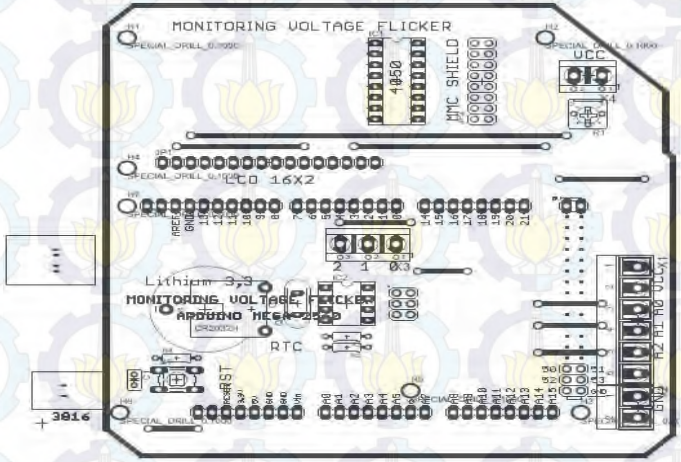
Gambar 3.16 Skema Rangkaian *Reset*

3.3.8 *Shield* Arduino Mega

Tugas Akhir ini menggunakan Arduino Mega sebagai *controller*-nya, tidak semua pin pada Arduino Mega digunakan serta pin pada Arduino Mega hanya berupa pin *head female* yang kurang kokoh apabila digunakan untuk peletakan peralatan sensor ataupun perangkat elektronik lainnya. *Wiring* kabel pun terlihat acak – acakan dan tidak rapi, apabila terjadi *error* pada peralatan maka akan susah melakukan *troubleshooting*.

Dengan alasan tersebut maka dibuat rangkaian *shield* Arduino, dalam rangkaian *shield* ini hanya beberapa pin yang digunakan, yakni pin 10, 9, 8, 7, 6, 5 untuk digunakan untuk menghubungkan LCD ke

Arduino, beberapa pin analog yakni A0, A1, A2 digunakan sebagai membaca sensor. Terdapat beberapa *ground* dan sumber 5 Volt. Selain itu pada rangkaian *shield* ini terdapat rangkaian MMC *Shield* yang dihubungkan dengan pin ISP Arduino yakni Miso, Mosi, SCK, serta CS dan pin 3,3 Volt, rangkaian RTC yang terhubung dengan SDA dan SCL pada pin Arduino, terdapat pin *reset* untuk mempermudah melakukan *reset.* Untuk rangkaian *shield* Arduino ini terletak pada sisi paling atas dari perangkat elektronik pada Tugas Akhir ini, untuk lebih jelasnnya ditunjukkan pada Gambar 3.17



Gambar 3.17 *Shield* Arduino Mega

3.3.9 Pemutus Daya (MCB 16A)

Terdapat MCB 16 A yang digunkan untuk membatasi arus yang mengalir pada peralatan ini. 16 A dipilih karena batas dari perangkat elektronik yang lain memiliki batas yang berbeda – beda, untuk sensor Arus batas yang dimiliki hanya sebesar 20 A, kabel yang digunakan ialah NYAF 2,5 mm, dan terminal pada alat ini hanya memiliki batas kemampuan 25 A. Dengan hal tersebut maka dipilih pemutus daya 16 A agar kondisi dari peralatan yang lain masih cukup kuat untuk menahannya.

Pemutus daya yang digunakan pada Tugas Akhir ini termasuk jenis CL 16 A jadi ketika mendapatkan nilai arus yang lebih dari 16 A

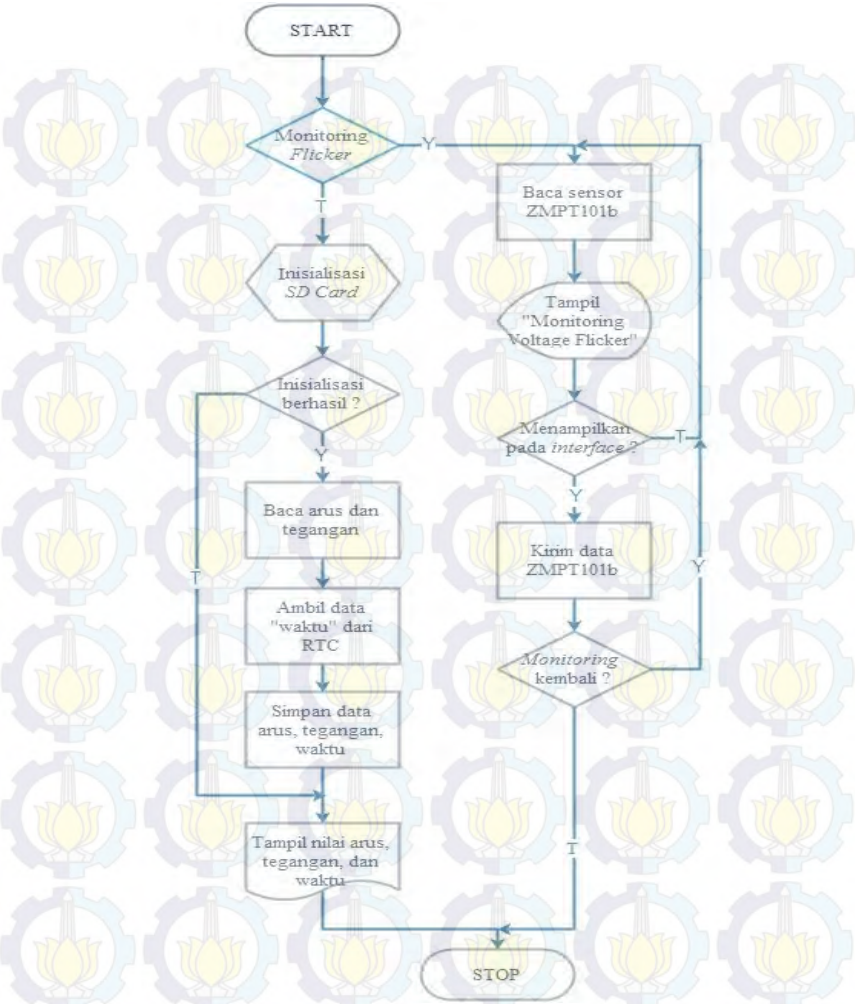
tidak langsung terputus, tetapi memiliki jeda. Maka kemampuan perangkat elektronik yang lain seperti sensor, kabel, serta terminal sambungan diberi nilai yang lebih dari batas pemutus daya (MCB), sebagai keamanan ketika perangkat ini sedang dalam keadaan beroperasi. Untuk gambar pemutus daya yang digunakan pada Tugas Akhir ini ditunjukkan pada Gambar 2.15

3.4 Perancangan Perangkat Lunak (*software)*

Perancangan perangkat lunak (*software)* menggunakan dua macam *software* yakni Arduino IDE untuk pemrograman pada papan Arduino serta menjalankan beberapa fungsi dari perangkat elektronik yang terhubung pada Arduino, sedangkan *software* Matlab digunakan untuk merancang pada sisi *interface* yang nantinya kan mengolah data yang dikirim oleh Arduino Mega melalui komunikasi serial ataupun dengan *micro SD*.

3.4.1 Pemrograman Arduino IDE

Software Arduino IDE pada Tugas Akhir digunakan untuk melakukan pemrograman papan Arduino dalam menjalankan sistem secara keseluruhan. *Software* ini menggunakan bahasa pemrograman C. Perancangan dari pemrograman ini seperti yang ditunjukkan pada Gambar 3.18, dimana Arduino akan membaca *output* dari setiap sensor, yaitu sensor arus, dan sensor tegangan yang akan diubah kedalam bentuk ADC serta dilakukan perkalian dari hasil linearisasi sensor yang nantinya akan mendapatkan nilai sebenarnya. Kemudian akan ada pemilihan *mode* yaitu melakukan *monitoring Voltage Flicker* atau melakukan *monitoring* terhadap keadaan normal dari arus dan tegangan. Ketika melakukan *monitoring Voltage Flicker* data ADC dari papan Arduino akan dikirim ke *personal computer* dengan komunikasi serial dan akan dilakukan pengamatan gelombang *flicker*. Sedangkan untuk *monitoring* keadaan normal dari arus dan tegangan akan ditampilkan pada LCD 16x2, serta disimpan dalam *SD Card* dengan format .CSV.

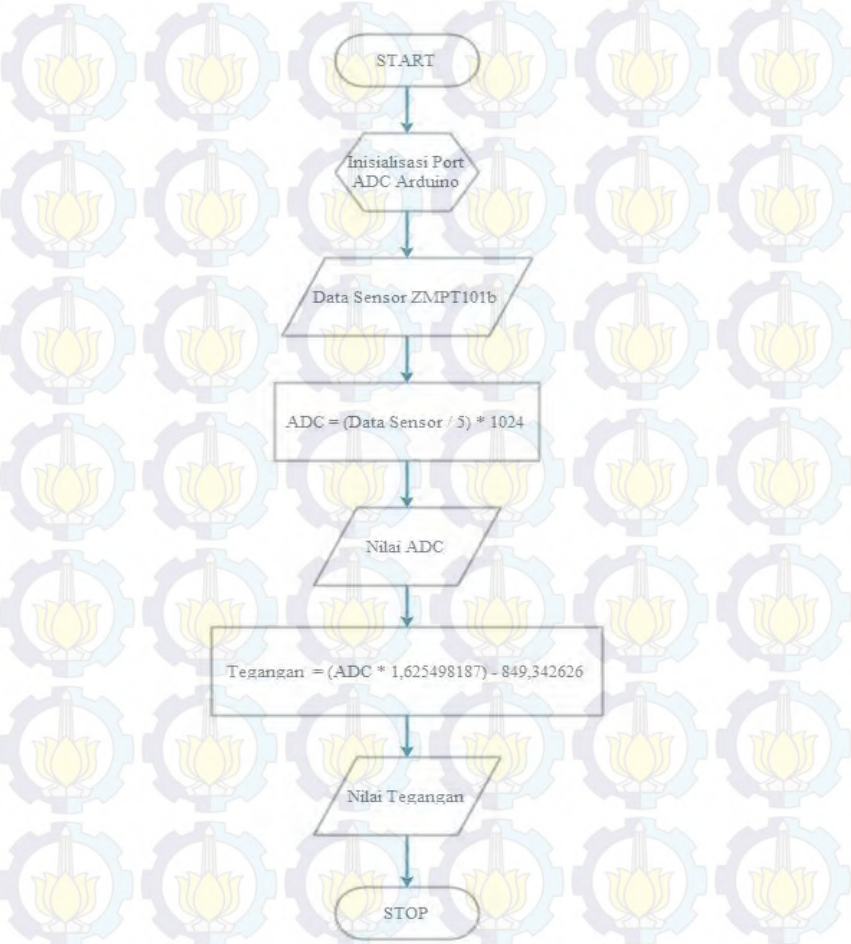


Gambar 3.18 *Flowchart* Pemrograman Arduino IDE

3.4.1.1 Pemrograman Sensor ZMPT101b

Penggunaan sensor ZMPT101b pada Tugas Akhir ini digunakan untuk mengukur tegangan yang nantinya akan ditampilkan pada *interface*. Agar sensor ZMPT101b ini dapat digunakan sebagai sensor

tegangan, dibutuhkan program yang sesuai untuk sensor ini. *Flowchart* untuk program sensor ZMPT101b ditunjukkan pada Gambar 3.19



Gambar 3.19 *Flowchart* Sensor ZMPT101b

Pada Gambar 3.19 di atas dapat dilihat perancangan *flowchart* untuk proses pembacaan tegangan melalui sensor ZMPT101b pada Arduino. Untuk urutan cara kerja *flowchart* adalah sebagai berikut :

1. *Start* adalah ketika program dimulai.

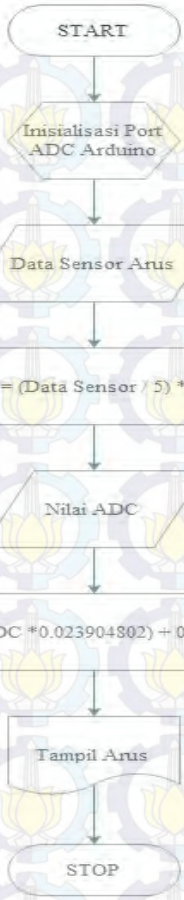
2. Pada Arduino Mega terdapat 15 buah pin ADC, 15 pin ADC ini memiliki karakteristik yang sama, sedangkan untuk sensor ZMPT101b menggunakan pin ADC 0.
3. Data dari sensor yang masuk ke Arduino berupa tegangan akan dibaca dan dikonversikan oleh Arduino, sehingga menjadi data ADC.
4. Untuk mendapatkan nilai tegangan yang sesuai dengan nilai tegangan sebenarnya maka diperlukan perhitungan yang sesuai dengan Persamaan 4.9.

3.4.1.2 Pemrograman Sensor Arus CT

Sensor arus yang digunakan pada Tugas Akhir ini adalah sensor arus *current transformers* dimana *output* dari sensor ini berupa arus dan harus dikondisikan dengan rangkaian pengkondisi agar nilai menjadi tegangan, sehingga dapat terbaca oleh Arduino. Agar sensor arus ini dapat digunakan sebagai sensor, dibutuhkan program yang sesuai untuk sensor ini. *Flowchart* untuk program sensor arus ditunjukkan pada Gambar 3.20

Pada Gambar 3.20 dapat dilihat perancangan *flowchart* untuk proses pembacaan arus melalui sensor arus *current transformers* pada Arduino. Untuk urutan cara kerja dari *flowchart* adalah sebagai berikut :

1. *Start* adalah ketika program dimulai.
2. Pada Arduino Mega terdapat 15 buah pin ADC, 15 pin ADC ini memiliki karakteristik yang sama, sedangkan untuk sensor arus menggunakan pin ADC 2.
3. Data dari sensor yang masuk ke Arduino berupa tegangan 0 – 5 Volt. Nilai arus maksimal yang dapat dibaca yakni 20 Ampere, dimana nilai tegangan sebesar 4,5 Volt. Nilai tegangan dari sensor arus *current transformers* akan dibaca dan dikonversikan oleh Arduino, sehingga menjadi data ADC.
4. Untuk mendapatkan nilai arus yang sesuai dengan nilai tegangan sebenarnya maka diperlukan perhitungan yang sesuai dengan Persamaan 4.6.
5. Setelah diolah dengan Persamaan 4.6 maka akan didapat nilai arus yang sebenarnya.
6. Nilai arus kemudian akan ditampilkan pada LCD.

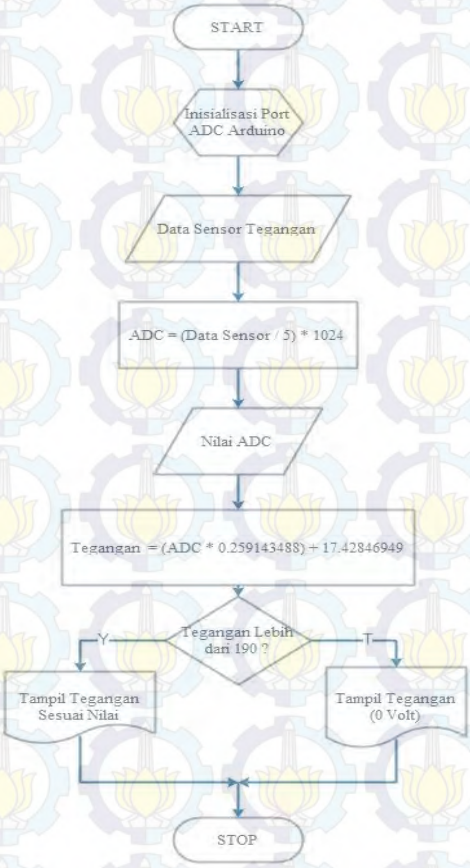


Gambar 3.20 *Flowchart* Sensor Arus

3.4.1.3 Pemrograman Sensor Tegangan

Untuk pengukuran nilai tegangan pada kondisi normal digunakan sensor tegangan berdasarkan prinsip pembagi tegangan. Dimana tegangan yang akan diukur akan disearahkan dan diturunkan nilai tegangannya sesuai kemampuan dari pin ADC pada Arduino. Pada sensor tegangan ini menggunakan transformator *step down* yang memiliki *output* 6 Volt, lalu disearahkan serta dikondisikan pada

tegangan 0 – 5 Volt. Agar sensor tegangan ini dapat digunakan sebagai sensor tegangan dengan baik maka dibutuhkan program yang sesuai untuk sensor ini. *Flowchart* untuk program sensor tegangan ditunjukkan pada Gambar 3.21.



Gambar 3.21 *Flowchart* Sensor Tegangan

Pada Gambar 3.21 dapat dilihat perancangan *flowchart* untuk proses pembacaan tegangan melalui sensor tegangan yang telah melaui

pembagi tegangan pada Arduino. Untuk urutan cara kerja dari *flowchart* adalah sebagai berikut :

1. *Start* adalah ketika program dimulai.
2. Pada Arduino Mega terdapat 15 buah pin ADC, 15 pin ADC ini memiliki karakteristik yang sama, sedangkan untuk sensor tegangan menggunakan pin ADC 1.
3. Data dari sensor yang masuk ke Arduino berupa tegangan 0 – 5 Volt. Nilai tegangan maksimal yang dapat dibaca yakni 250 Volt, dimana nilai tegangan sebesar 4,5 Volt. Nilai tegangan dari sensor tegangan akan dibaca dan dikonversikan oleh Arduino, sehingga menjadi data ADC.
4. Untuk mendapatkan nilai arus yang sesuai dengan nilai tegangan sebenarnya maka diperlukan perhitungan yang sesuai dengan Persamaan 4.8.
5. Setelah diolah dengan Persamaan 4.8 maka akan didapat nilai tegangan yang sebenarnya.
6. Nilai tegangan kemudian akan ditampilkan pada LCD. Apabila nilai tegangan lebih dari 190 Volt maka nilai akan tampil yang sebenarnya, sedangkan untuk nilai tegangan kurang dari 190 Volt maka akan tampil tegangan sebesar 0 Volt.

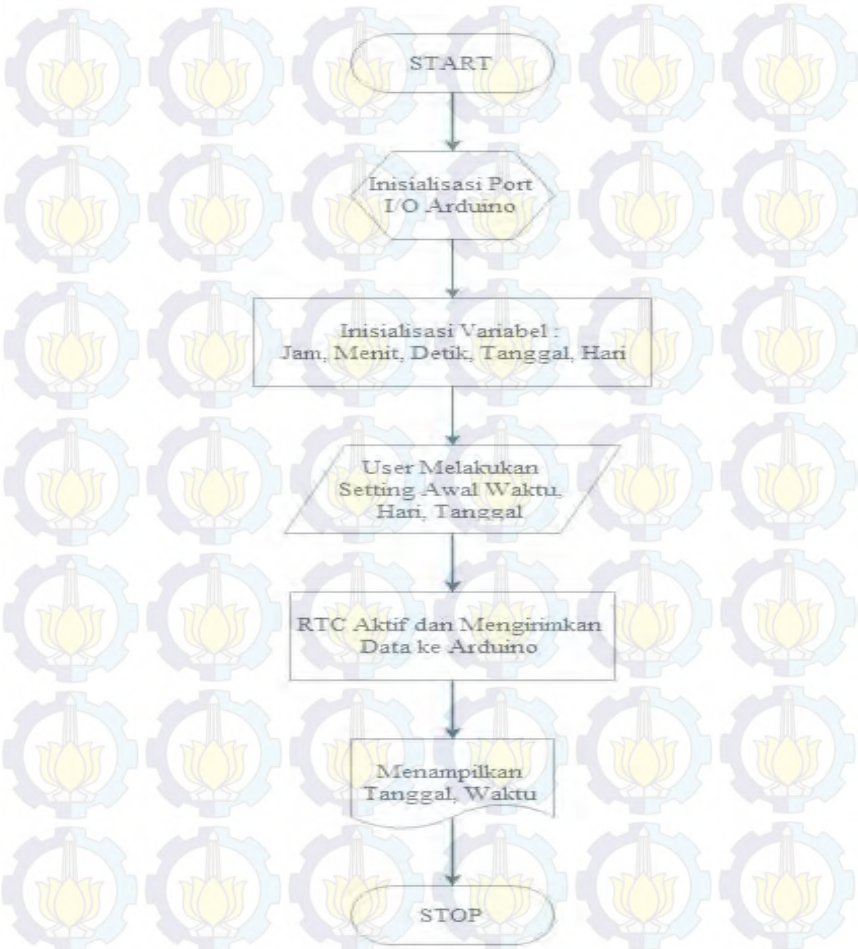
Sensor tegangan ini dibatasi pada tegangan 190 Volt untuk batas bawah, karena sesuai dengan SPLN 1 : 1995 variasi tegangan pelayanan ditetapkan maksimum + 5% dan minimum -10% terhadap tegangan nominal [12]. Dengan alasan tersebut pada pengukuran dibawah 190 Volt akan tampil 0 Volt pada LCD 16x2.

3.4.1.5 Pemrograman RTC

Real Time Clock yang digunakan pada Tugas Akhir ini, yaitu IC DS1307. Untuk pemrograman pada papan Arduino telah tersedia *library* dari RTC tersebut. *Wiring* RTC ke Arduino dengan I^2C, yaitu menghubungkan pin SDA dan SCL dari RTC ke Arduino pada pin 20 dan 21. Pengguna dapat memanggil *library* yang telah tersedia pada Arduino IDE.

Agar RTC ini dapat digunakan sebagai pemberi data waktu dengan baik maka dibutuhkan program yang sesuai untuk RTC ini. Pada RTC ini set awal untuk data hari, tanggal, serta waktu diberikan pada program dan tidak di set secara manual setelah program di*upload*. Alur

flowchart untuk program *Real Time Clock* ditunjukkan pada Gambar 3.22



Gambar 3.22 *Flowchart Real Time Clock*

Pada Gambar 3.22 dapat dilihat untuk perancangan program dari rangkaian RTC. Dimana data waktu akan didapat dari IC DS1307 yang telah diprogram sesuai *flowchart* diatas. Untuk urutan cara kerja dari *flowchart* adalah sebagai berikut :

1. *Start* adalah ketika program dimulai.
2. Untuk *wiring* RTC dengan Arduino dihubungkan pada pin SDA dan SCL jadi pada inisialisasi *port* I/O Arduino harus mengaktifkan pin SDA dan SCL. Yakni dengan mengaktifkan komunikasi I^2C.
3. Selanjutnya inisialisasi variabel yang akan digunakan untuk detik, menit, jam, tanggal, dan hari.
4. Selanjutnya *user* akan melakukan *setting* awal untuk waktu pada RTC tersebut.
5. Dengan data *setting* awal tersebut RTC akan mengirim data tersebut ke Arduino dengan komunikasi I^2C yakni melalui pin SDA dan SCL.
6. Data waktu akan ditampilkan pada LCD.

3.4.1.5 Pemrograman *SD Card*

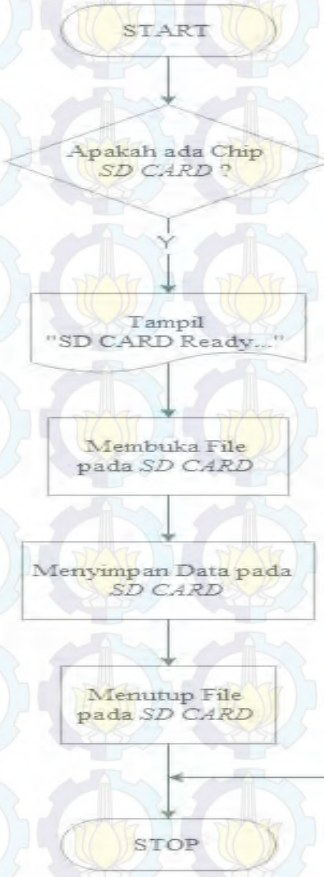
Pemrograman data *logger* juga menggunakan pemrograman Arduino IDE. Komunikasi data *logger* berbeda dengan komunikasi dengan RTC yaitu menggunakan SPI. Lebih jelasnya menggunakan pin MISO, MOSI, SCK, dan CS. Pada rancangan *hardware* pin CS terletak pada pin I/O digital 4. Pin tersebut akan digunakan sebagai acuan untuk mengaktifkan komunikasi dengan *slave*. Agar *SD Card* ini berjalan dengan baik pada Arduino, maka dibutuhkan pemrograman yang sesuai dengan kebutuhan dari *SD Card*. *Flowchart* untuk program *SD Card* ditunjukkan pada Gambar 3.23

Pada Gambar 3.23 dapat dilihat untuk perancangan program dari rangkaian *SD Card*. Untuk urutan cara kerja dari *flowchart* adalah sebagai berikut :

1. *Start* adalah ketika program dimulai.
2. Dilakukan pengecekan untuk mengetahui ada atau tidaknya *chip SD Card*.
3. Jika terdapat *chip SD Card* maka akan dimulai untuk proses penyimpanan dari data yang akan disimpan, sedangkan jika tidak ada *chip SD Card* maka program akan berhenti.
4. Tampilan “SD Card Ready” untuk proses selanjutnya.
5. Pada tahap ini akan dimulai proses penyimpanan data yang diawali dengan membuka *file* pada *SD Card*.
6. Data akan tersimpan pada *file* yang telah dibuka pada tahap sebelumnya.

7. Setelah data tersimpan, selanjutnya *file* akan ditutup dan data telah selesai tersimpan

Program tersebut digunakan untuk menulis data *String* pada *file* "log.csv". Setiap pengaksesan *SD Card* dimulai dengan perintah SD.*open*(); untuk menulis data yang tersimpan dalam memori pada *file* menggunakan dataFile.*print* (); ketika penulisan selesai maka akan ditutup dengan dataFile.*close*();.



Gambar 3.23 *Flowchart SD Card*

3.4.2 Pemrograman Matlab

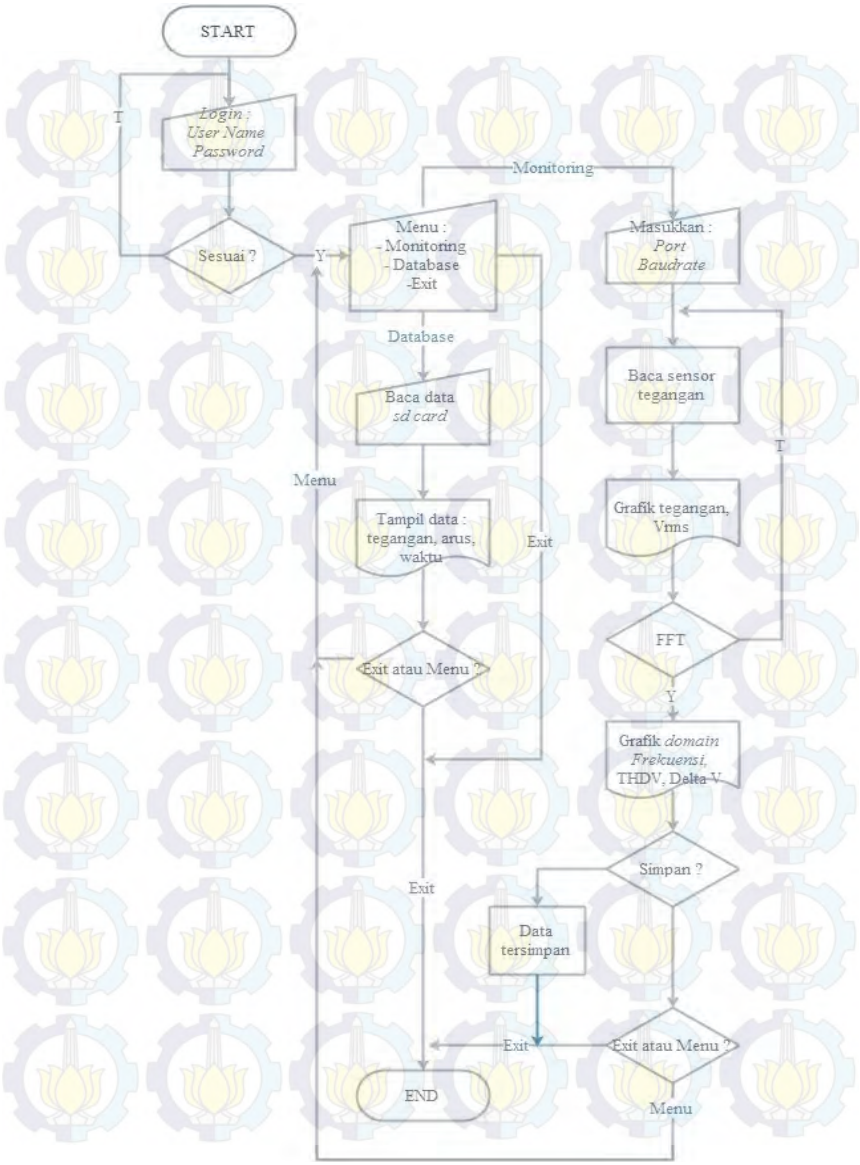
Software Matlab digunakan sebagai perancangan *interface* yang nantinya akan menerima data yang dikirim dari papan Arduino. Matlab adalah sebuah lingkungan komputasi *numerical* dan bahasa pemrograman komputer generasi keempat. Dikembangkan oleh The MathWorks, Matlab memungkinkan manipulasi matriks, *plotting* fungsi dan data, implementasi algoritma, pembuatan antarmuka pengguna, dan *interface* dengan program dalam bahasa lainnya.

Dari penjelasan tersebut *software* Matlab dapat digunakan pada Tugas Akhir ini, yaitu melakukan antar muka dengan program lain serta menampilkan grafik dari data yang dikirim. Komputasi matematik pada *software* ini pun mendukung terhadap pengolahan data untuk menjadi grafik yang nantinya akan ditampilkan pada *personal computer*. Terdapat dua menu dalam perancangan ini yaitu ketika melakukan *monitoring flicker* serta melakukan pengamatan pada keadaaan normal dari arus dan tegangan dengan melihat data dari *SD Card*.

Perancangan dari *interface* yang akan dibuat di Matlab yakni, pada *monitoring Voltage Flicker* data dari sensor ZMPT101b akan dikirim ke *personal computer* melalui komunikasi serial, kemudian akan diatampilkan pada layar monitor setelah melakukan beberapa *setting* yang tertera pada tampilan *interface.* Setelah data gelombang telah diterima, maka kita dapat melakukan pengamatan dari hasil yang telah di *plotting* tetapi masih berupa tampilan sinus. Dari data gelombang sinus tersebut kita akan mendapatkan nilai tegangan RMS yang akan tampil pada layar monitor. Akan terlihat apabila terjadi naik turunnya tegangan dari grafik yang telah ditampilkan. Pada proses selanjutnya akan dilakukan pengamatan yang dilihat dari frekuensi dari data yang telah didapatkan. Untuk mengubah ke bentuk domain frekuensi maka digunakan persamaan FFT (*Fast Foriour Transform)*. Setelah diubah Data tersebut dapat disimpan kedalam format .JPG dan .mat.

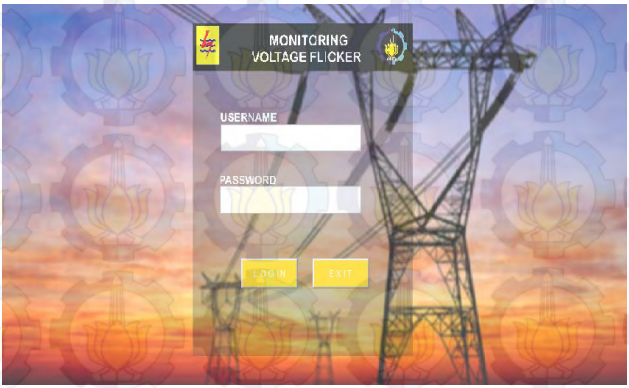
Sedangkan untuk tampilan pada menu *monitoring* keadaan normal tegangan dan arus, dapat ditampilkan pada Matlab. Berbeda dengan *monitoring flicker*, untuk menu ini menggunakan data yang telah disimpan pada *SD Card*. Hanya membuka *file* yang ada pada *SD Card* maka nilai arus, tegangan, serta waktu pengambilan akan ditampilkan.

Mengenai alur pemrograman *interface* dengan menggunakan *software* Matlab ini ditunjukkan pada Gambar 3.24



Gambar 3.24 *Flowchart* Pemrograman Matlab

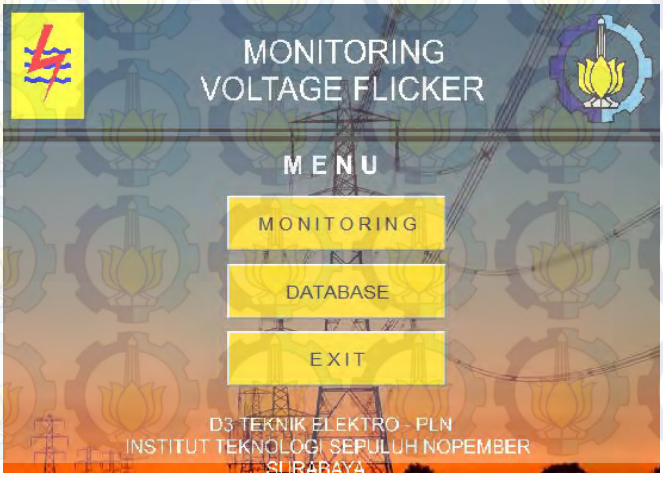
Tampilan *login* awal pada *interface* yang disusun menggunakan Matlab ditunjukkan pada Gambar 3.25



Gambar 3.25 Tampilan *Login*

Kegunaan dari *login* yakni membatasi pengguna dari aplikasi yang akan digunakan. Tidak semua orang dapat mengakses aplikasi ini, pengguna dari aplikasi ini harus mempunyai *username* serta *password* untuk mengaksesnya.

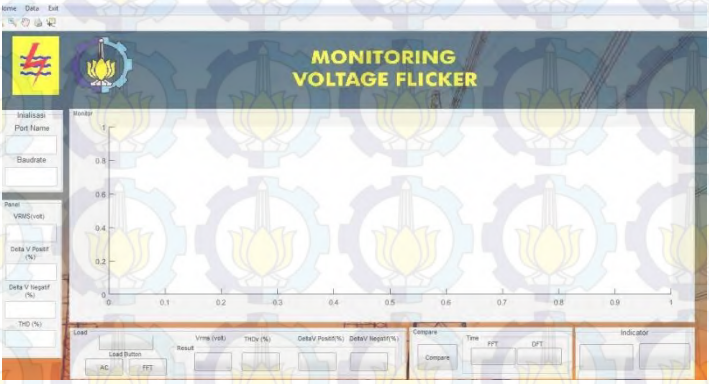
Tampilan menu awal setelah berhasil memasukkan *username* dan *password* ditunjukkan pada Gambar 3.26



Gambar 3.26 Menu Utama Aplikasi *Monitoring Voltage Flicker*

Pada menu tersebut pengguna dapat memilih sesuai dengan kebutuhan, yakni menu *database* untuk pembacaan data arus dan tegangan yang telah tersimpan pada *micro sd*, sedangkan *monitoring* digunakan untuk menganalisa gelombang ada atau tidaknya *flicker*.

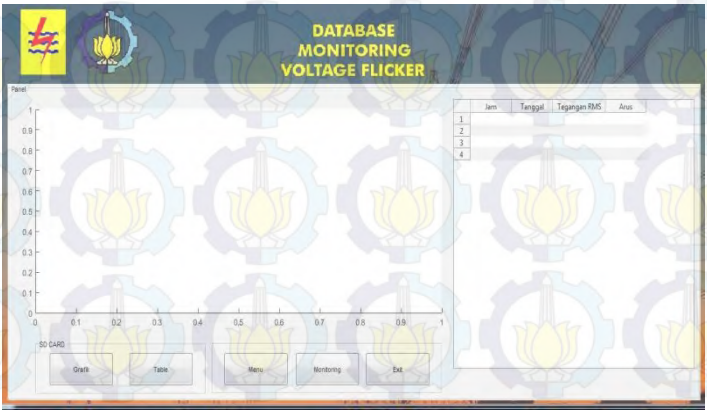
Tampilan lembar *monitoring* ditunjukkan pada Gambar 3.27



Gambar 3.27 Tampilan Lembar *Monitoring Voltage Flicker*

Pada Gambar 3.27 tersebut pengguna dapat menampilkan gelombang dari sensor tegangan, akan terlihat kondisi tegangan dalam keadaan normal ataupun terjadi fluktuasi. Serta dapat diubah ke domain frekuensi dengan mengguakan metode FFT.

Tampilan lembar *database* ditunjukkan pada Gambar 3.28



Gambar 3.28 Tampilan Lembar *Database*

Data yang telah tersimpan dapat dibaca melalui lembar yang ditunjukkan pada Gambar 3.28, nilai arus, tegangan, serta waktu pengambilan data akan tampil pada *form* sebelah kanan, sedangkan untuk grafik perubahan tegangan dapat dilihat pada *form* sebelah kiri. Dari data tersebut pengguna dapat menggetahui kondisi tegangan tiap detiknya, karena data yang tersimpan tiap detik.

BAB IV
PENGUJIAN DAN ANALISA DATA

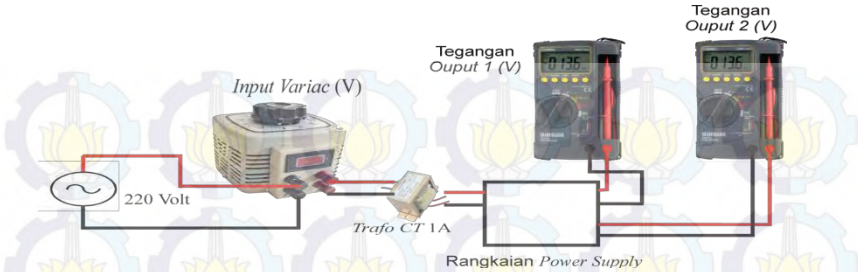
Untuk mengetahui kinerja dari peralatan dan pembuatan sistem yang telah dirancang dan direncanakan sedemikian rupa pada BAB III, maka diperlukan pengujian dan analisa data dari setiap komponen pendukung yang dibuat agar sistem dapat berjalan dengan baik sesuai dengan yang diharapkan. Pada bab ini akan dibahas tentang pengujian dan analisa data *hardware* dan *software* yang telah dibuat. Adapun bagian – bagian yang akan diuji pada alat ini adalah :

1. Pengujian *Power Supply*
2. *Input/Output* Arduino
3. Pembacaan RTC
4. Memori *SD Card*
5. Tampilan LCD 16x2
6. Pembacaan Sensor Arus
7. Pembacaan Sensor Tegangan
8. Pembacaan Sensor ZMPT101b
9. Pengujian Komunikasi Serial
10. Pengujian *Software* Matlab
11. Pengujian Metode FFT
12. Pengujian Alat Keseluruhan

4.1 Pengujian *Power Supply*

Pengujian ini dilakukan pada *Power Supply* yang nantinya akan disambungkan ke Arduino dan sensor ZMPT101b. pengujian ini menggunakan variac sebagai *input* dan multimeter "SANWA CD800a" untuk mengukur besar tegangan *output Power Supply*. *Input* pada variac diatur dari tegangan 200 Volt hingga tegangan 248 Volt, hal ini dilakukan untuk memastikan tegangan *output* dari *Power Supply* stabil pada tegangan antara 200 Volt hingga 228 Volt. Pada rangkaian *Power Supply* ini terdapat dua *output* yakni 9 Volt untuk *output* 1 dan 5 Volt untuk *output* 2. Skema pengujian untuk pengukuran tegangan *output* 1 dan *output 2* ditunjukkan pada Gambar 4.1.

Data yang didapat untuk pengujian *Power Supply* pada tegangan *output* 1 dan tegangan *output* 2 terdapat pada Tabel 4.1



Gambar 4.1 Skema Pengujian *Power Supply*

Tabel 4.1 Pengujian *Power Supply*

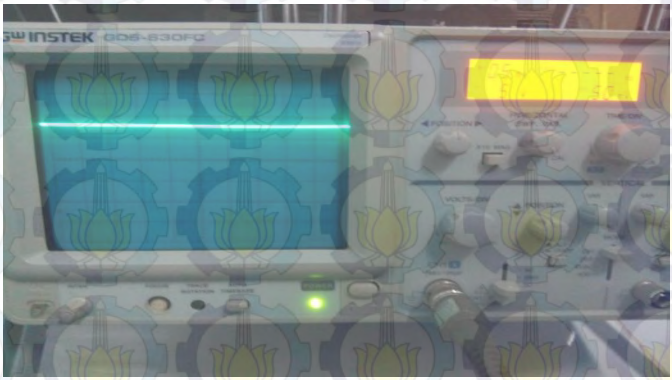
No.	*Input* Variac (V)	Tegangan *Output* 1 (V)	Tegangan *Output* 2 (V)
1.	200	8,97	5,01
2.	202	8,97	5,01
3.	204	8,97	5,01
4.	206	8,97	5,01
5.	208	8,97	5,01
6.	210	8,97	5,01
7.	212	8,97	5,01
8.	214	8,97	5,01
9.	216	8,97	5,01
10.	218	8,97	5,01
11.	220	8,97	5,01
12.	222	8,97	5,01
13.	224	8,97	5,01
14.	226	8,97	5,01
15.	228	8,97	5,01
16.	230	8,97	5,01
17.	232	8,97	5,01
18.	234	8,97	5,01
19.	236	8,97	5,01
20.	238	8,97	5,01
21.	240	8,97	5,01
22.	242	8,97	5,01
23.	244	8,97	5,01
24.	246	8,97	5,01
25.	248	8,97	5,01

Dari Tabel 4.1 didapat bahwa tegangan *output* pada *Power Supply* yang akan digunakan memiliki tegangan *output* yang stabil meskipun tegangan yang diberikan bervariasi.

Pada tegangan *output Power Supply* yang pertama yaitu 8,97 Volt. *Output* tersebut akan digunakan sebagai sumber Arduino, nilai tersebut sudah memenuhi kebutuhan dari Arduino dan aman digunakan karena kebutuhan dari Vin Arduino berkisar 7 – 12 Volt.

Pada tegangan *output Power Supply* yang kedua yaitu 5,01 Volt digunakan untuk memenuhi kebutuhan dari sensor ZMPT101b. Kebutuhan dari sensor ZMPT101b berkisar 5 Volt. Dengan menggunakan sumber dari *Power Supply* ini maka kebutuhan dari sensor telah terpenuhi dan aman digunakan karena tegangannya 5,01 Volt.

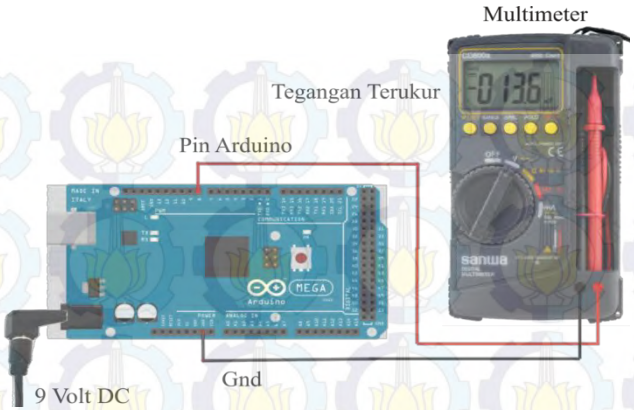
Power Supply ini juga diuji dengan osiloskop untuk melihat gelombang dari tegangan *output* yang dihasilkan, apakah terdapat *ripple* atau tidak. Untuk pengujian dengan menggunakan osiloskop rangkaian *output Power Supply* dihubungkan dengan osiloskop yang telah dikalibrasi sebelumnya. Selanjutnya akan muncul tampilan pada layar osiloskop tegangan yang telah terukur, hasilnya terdapat pada Gambar 4.2



Gambar 4.2 *Output Power Supply* 8,97 V

4.2 *Input/Output* Arduino

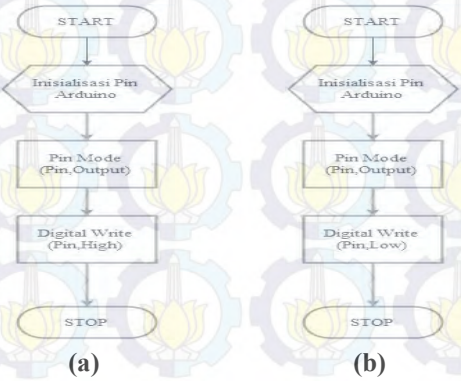
Pegujian ini dilakukan terhadap *board* Arduino yang digunakan yakni Arduino Mega, untuk mengetahui bahwa kondisi Arduino Mega dapat digunakan dengan baik untuk Tugas Akhir ini. Skema pengujian ditunjukkan pada Gambar 4.3



Gambar 4.3 Skema Pengujian I/O Arduino Mega

Dilakukan pengujian pada pin Arduino Mega yang akan digunakan saja, yaitu pada pin digital Arduino dari pin 0 – 21 dan pada pin analog Arduino dari pin A0 – A15. Pengujian dilakukan dengan cara memberikan program pada Arduino yakni memberikan perintah *HIGH* dan *LOW* atau logika 0 dan 1 pada setiap pin Arduino yang akan diuji sesuai dengan *flowchart* yang ditunjukkan pada Gambar 4.4, kemudian mengukur besaran tegangan yang keluar dari pin tersebut seperti yang ditunjukkan pada Gambar 4.3

Hasil pengukuran menggunakan multimeter "SANWA CD800a" data yang didapat pada Tabel 4.2



Gambar 4.4 *Flowchart* Pemograman Pengujian I/O Arduino Mega

Tabel 4.2 Pengujian I/O Arduino Mega

No.	Nomer Pin Arduino	*Logic*	Tegangan Terukur (V)	*Logic*	Tegangan Terukur (V)
1.	0	1	5,08	0	0,100
2.	1	1	5,08	0	0,04
3.	2	1	5,08	0	0,01
4.	3	1	5,08	0	0,01
5.	4	1	5,08	0	0,01
6.	5	1	5,08	0	0,01
7.	6	1	5,08	0	0,00
8.	7	1	5,08	0	0,00
9.	8	1	5,08	0	0,00
10.	9	1	5,08	0	0,00
11.	10	1	5,08	0	0,00
12.	11	1	5,08	0	0,00
13.	12	1	5,08	0	0,00
14.	13	1	5,08	0	0,00
15.	14	1	5,08	0	0,00
16.	15	1	5,08	0	0,00
17.	16	1	5,08	0	0,00
18.	17	1	5,08	0	0,00
19.	18	1	5,08	0	0,00
20.	19	1	5,08	0	0,00
21.	20	1	5,08	0	0,01
22.	21	1	5,08	0	0,01
23.	A0	1	5,08	0	0,00
24.	A1	1	5,08	0	0,00
25.	A2	1	5,08	0	0,00
26.	A3	1	5,08	0	0,00

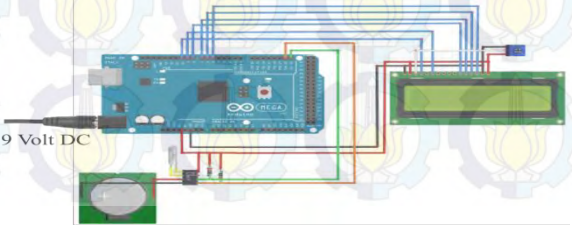
No.	Nomer Pin Arduino	*Logic*	Tegangan Terukur (V)	*Logic*	Tegangan Terukur (V)
27.	A4	1	5,08	0	0,00
28.	A5	1	5,08	0	0,00
29.	A6	1	5,08	0	0,00
30.	A7	1	5,08	0	0,00
31.	A8	1	5,08	0	0,00
32.	A9	1	5,08	0	0,00
33.	A10	1	5,08	0	0,00
34.	A11	1	5,08	0	0,00
35.	A12	1	5,08	0	0,00
36.	A13	1	5,08	0	0,00
37.	A14	1	5,08	0	0,00
38.	A15	1	5,08	0	0,00

Dengan pengujian tersebut, dapat disimpulkan bahwa board Arduino tersebut layak dipakai karena sesuai dengan *datasheet* yang terlampir.

4.3 Pembacaan RTC

Pengujian terhadap RTC dilakukan langsung dengan membaca data pada RTC, sebelum dilakukan pengujian maka Arduino diberi program sesuai rancangan *flowchart* pada Gambar 3.22. Skema pengujian ditunjukkan pada Gambar 4.5

Pengujian dilakukan dengan membandingkan tampilan jam pada LCD dan tampilan jam pada komputer, yang ditunjukkan pada Gambar 4.6



Gambar 4.5 Skema Pengujian RTC

Gambar 4.6 Pengujian RTC

Hasil pengamatan RTC yang telah dibandingkan dengan jam yang terdapat pada komputer ditunjukkan pada Tabel 4.3

Tabel 4.3 Pengujian RTC

No.	Tampilan LCD	Tampilan Komputer	Selisih
1.	2:31	2:31	0
2.	2:32	2:32	0
3.	2:33	2:33	0
4.	2:34	2:34	0
5.	2:35	2:35	0

Berdasarkan Tabel 4.3 didapatkan bahwa selisih antara tampilan pada LCD dan tampilan pada jam *computer* adalah tetap, yaitu 0 detik. Sehingga dapat disimpulkan bahwa RTC dapat digunakan sebagai acuan karena selisihnya tetap dan tidak berubah.

Dengan acuan tersebut maka *error* pembacaan yang didapat nilainya 0% dengan alasan bahwa semua alat penunjuk waktu yang tidak terhubung secara *online* dapat menunjukkan waktu yang sama. Tetapi alat penunjuk waktu tersebut dapat digunakan karena memiliki selisih waktu yang sama tidak terlalu cepat dan tidak terlalu lambat.

4.4 Memori *SD Card*

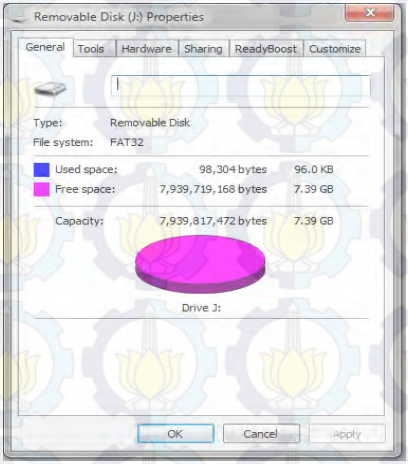
Pada pengujian *SD Card* dilakukan untuk mengetahui kapasitas yang dapat ditampung oleh *SD Card*. *Memory* yang digunakan adalah SanDisk Ultra dengan kapasitas penyimpanan 8 Gb.

Pengujian dilakukan dengan pembacaan kapasitas kartu pada komputer dalam kondisi kosong. Pengujian dilakukan untuk memastikan *SD Card* memiliki ruang penyimpanan yang benar-benar kosong, yang nantinya akan diisi data arus dan tegangan. Hasil pengujian *SD Card* ditunjukkan pada Gambar 4.7

Dari Gambar 4.7 dapat dilihat bahwa kondisi *SD Card* dalam keadaan kosong. Dan terbagi menjadi beberapa bagian yang seperti padaTabel 4.4

Tabel 4.4 Pengujian *SD Card*

Nama Ruang	Ukuran (*bytes*)	Ukuran
Used Space	98304	96, 0 KB
Free Space	7939719168	7,39 GB
Capacity	7939817472	7,39 GB

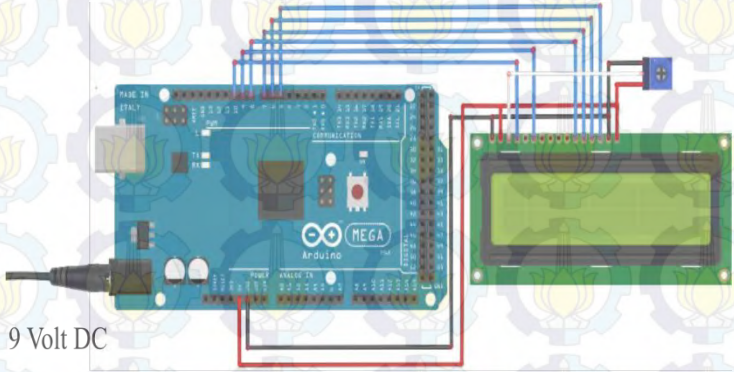


Gambar 4.7 Pengujian *SD Card*

Dengan demikian maka kondisi *SD Card* dapat digunakan pada Tugas Akhir ini untuk menyimpan nilai arus, dan tegangan pada saat kondisi normal.

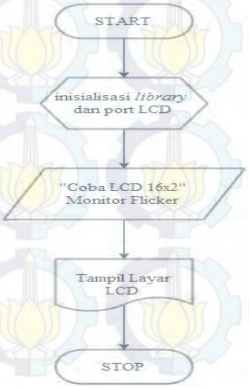
4.5 Tampilan LCD 16x2

Pengujian pada LCD 16x2 dengan tujuan untuk mengetahui kondisi dari LCD 16x2 dalam keadaan baik atau tidak. Karena LCD ini terhubung dengan Arduino maka pengujian dilakukan dengan memberikan program untuk menampilkan beberapa karakter pada LCD tersebut. LCD akan terhubung dengan pin Arduino yakni pin 10, 9, 8, 7, 6, 5. LCD ini hanya terbatas sampai 32 karakter (16 x2). Skema pengujian LCD ditunjukkan pada Gambar 4.8



Gambar 4.8 Skema Pengujian LCD

Pengujian ini dilakukan dengan memberikan program pada Arduino Mega untuk menampilkan karakter pada LCD, yang telah dirancang sesuia dengan *flowchart* yang ditunjukkan pada Gambar 4.9

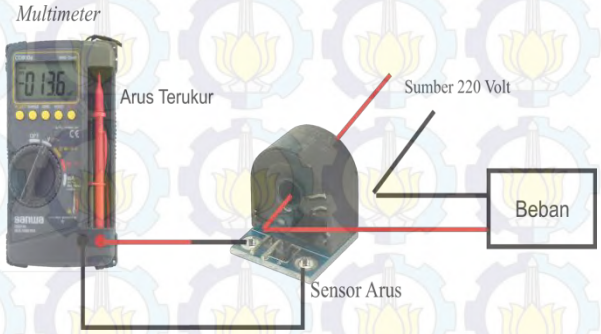


Gambar 4.9 Pengujian terhadap LCD

Setelah diberi program untuk menampilkan karakter pada LCD maka tampilan pada LCD akan tampil pada baris atas "Coba LCD 16x2" dan pada baris kedua akan tampil "Monitor Flicker. Dengan demikian maka LCD yang telah diuji dapat digunakan pada tugas Akhir ini.

4.6 Pembacaan Sensor Arus

Pada pembahasan di bab ini, pengambilan data arus yang diinduksikan oleh sensor arus CT-OD dan data linieritas tegangan dari beberapa beban daya yang berbeda – beda dengan kenaikan yang linier pula untuk menguji baik tidaknya sensor yang digunakan. Setelah melakukan pengambilan data arus seperti nilai pada Tabel 4.5, data akan dibandingkan dengan spesifikasi sensor sendiri yang menyebutkan bahwa perbandingan lilitan adalah 1000:1. Maka sensor dapat dikatakan bisa digunakan karena mampu menginduksikan nilai yang sepadan saat dilalui nilai sebesar 0,45 A akan menginduksikan arus sebesar 0,45 mA. Skema pengujian spesifikasi sensor ditunjukkan pada Gambar 4.10



Gambar 4.10 Pengujian Spesifikasi CT-OD

Dari Gambar 4.10 akan didapat data berupa arus terukur (mA) yang didapatkan dari multimeter, dan arus rumus yang didapatkan dari nilai daya pada beban, yang dibagi dengan nilai tegangan. Pada pengukuran ini tegangan yang digunakan yakni sebesar 220 Volt.

Dengan menggunakan rumus di Persamaan 4.1 dapat diambil kesimpulan nilai persen *error* dari setiap nilai yang diambil dari sensor arus CT-OD.

$$\%Error = \frac{Nilaisebenarnya - Nilaiterukur}{Nilaisebenarnya} x100\% \quad \text{.....(4.1)}$$

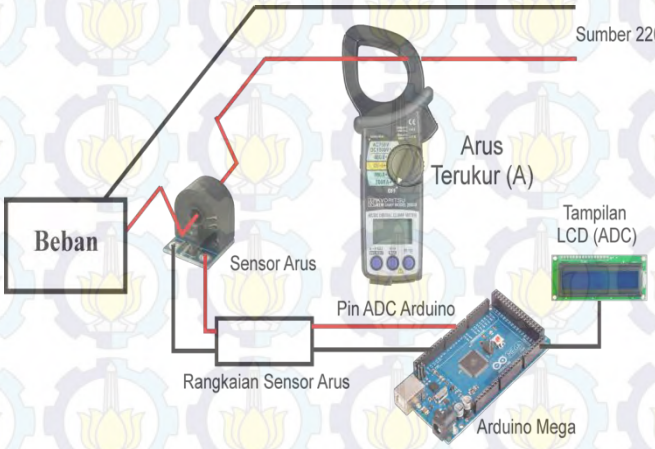
Hasil pengukuran yang telah untuk mengetahui spesifikasi sensor arus ini terdapat pada Tabel 4.5

Tabel 4.5 Pengujian *Output* CT-OD

No.	Beban (Watt)	Arus Terukur (mA)	Arus Rumus (A)	*%Error*
1.	40	0,17	0,17	0
2.	60	0,25	0,26	3,84
3.	80	0,34	0,34	0
4.	100	0,42	0,43	2,32
5.	120	0,5	0,52	3,84
6.	140	0,6	0,61	1,63
7.	160	0,67	0,69	2,89
8.	180	0,8	0,78	2,56

Dari Tabel 4.5 *output* sensor arus CT-OD memiliki rasio 1000:1 dimana untuk arus terukur sebesar 0,17 A maka *output* dari sensor CT-OD sebesar 0,17 mA.

Dilakukan pengambilan data sensor dan diubah menjadi data ADC menggunakan Arduino, skema pengambilan data ditunjukkan pada Gambar 4.11 Hal ini dilakukan untuk mendapatkan Persamaan yang nantinya akan digunakan pada pemrograman pada Arduino, data yang didapat pada Tabel 4.6

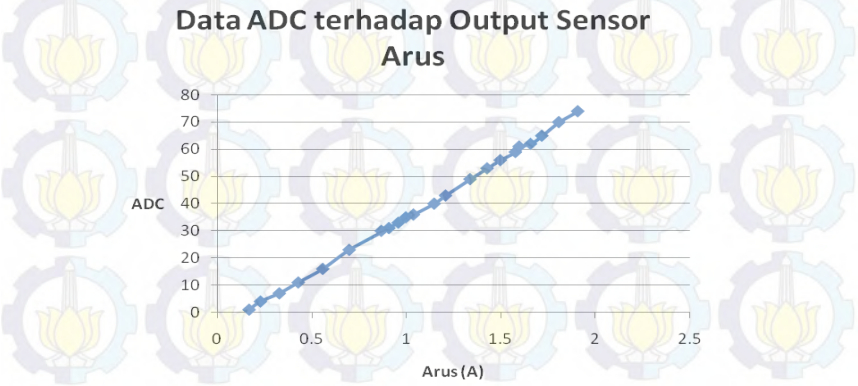


Gambar 4.11 Skema Pengambilan Data ADC

Tabel 4.6 Data Sensor Arus dan ADC

No.	Beban (Watt)	Arus Terukur (A)	Tampilan LCD (ADC)
1.	40	0,17	1
2.	60	0,23	4
3.	75	0,33	7
4.	100	0,43	11
5.	125	0,56	16
6.	160	0,7	23
7.	180	0,91	31
8.	200	0,87	30
9.	215	0,96	33
10.	225	1	35
11.	240	1,04	36
12.	260	1,15	40
13.	275	1,21	43
14.	280	1,34	49
15.	295	1,43	53
16.	350	1,5	56
17.	365	1,58	59
18.	375	1,6	61
19.	390	1,66	62
20	410	1,72	65
21.	425	1,81	70

Setelah mendapatkan data ADC dari *output* sensor dilakukan linearisasi data ADC yang ditunjukkan pada Gambar 4.12



Gambar 4.12 Data ADC terhadap Sensor Arus

Dari Gambar 4.12 tersebut dapat terlihat bahwa perubahan ADC sensor memiliki pergerakan yang linier terhadap perubahan arus. Sehingga dapat disimpulkan bahwa sensor arus CT-OD layak digunakan sebagai pendeteksi perubahan arus, karena memiliki perubahan *output* yang linier terhadap perubahan *input*.

Dari data pengujian dan grafik pengukuran, dapat dihasilkan suatu persamaan karakteristik sensor arus yang akan digunakan dalam pembacaaan Arduino agar didapatkan nilai arus yang sesuai dengan arus pengukuran pada clamp meter. Berikut ini adalah persamaan linier yang digunakan untuk mendapatkan pembacaan Arduino :

$$y = mx + b \quad \text{.....,,,,(4.1)}$$

$$m = \frac{n\sum(xy) - \sum x \sum y}{n\sum(x^2) - (\sum x)^2} \quad \text{.....(4.2)}$$

$$b = \frac{\sum y}{n} - m\frac{\sum x}{n} \quad \text{.....(4.3)}$$

Keterangan :

y = Arus AC yang diharapkan
x = ADC Arduino
m = *Slope*
b = *Intercept*
n = Jumlah data

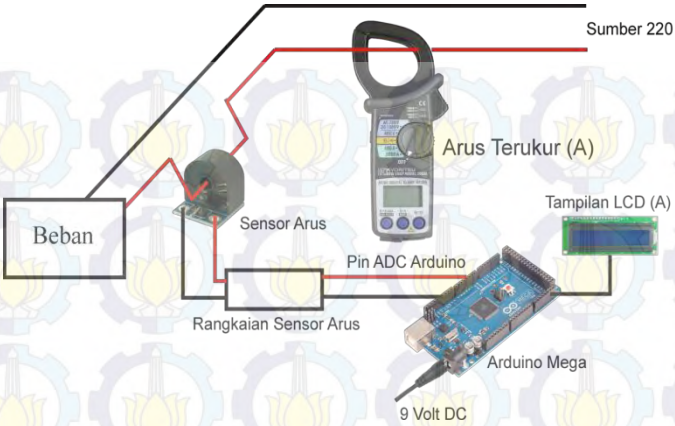
Setelah dilakukan perhitungan didapatkan sebuah persamaan untuk pembacaan Arduino :

$$y = 0{,}023904802 * x + 0{,}162535228 \quad \text{.....(4.5)}$$

Kemudian dilakukan penggantian variabel y menjadi nilai arus yang diinginkan dan x menjadi ADC, sehingga didapatkan Persamaan sebagai berikut :

$$Iout = 0{,}023904802 * ADC + 0{,}162535228 \quad \text{.....(4.6)}$$

Setelah menambahkan Persamaan (4.8) kedalam program Arduino selanjutnya dilakukan pengujian hasil perhitungan dengan cara membandingkan pembacaan sensor arus dengan clamp meter "kyoritsu KEW 2117R". Skema pengambilan data ditunjukkan pada Gambar 4.13



Gambar 4.13 Skema Perbandingan Arus LCD

Pengujian yang dilakukan terhadap sensor arus CT-OD bertujuan untuk mengetahui kesesuaian arus yang melewati beban. Pada pengujian ini akan dibandingkan nilai arus yang terukur dengan hasil pada tampilan LCD. Beban yang digunakan pada pengujian ini adalah beban beban yang biasanya menjadi kebutuhan sehari hari, seperti lampu dengan berbagai variasi daya lalu juga berbagai peralatan rumah tangga.

Hasil pengujian dengan beban lampu terdapat pada Tabel 4.7

Tabel 4.7 Pengujian Beban Lampu

No.	Beban (Watt)	Arus Terukur (A)	Tampilan LCD (A)	*%Error*
1.	40	0,2	0,22	9,09
2.	60	0,3	0,31	3,22
3.	75	0,37	0,37	0
4.	100	0,47	0,47	0
5.	115	0,56	0,53	5,66
6.	125	0,59	0,55	7,27
7.	140	0,65	0,62	4,83
8.	160	0,74	0,7	5,71
9.	175	0,81	0,76	6,57
10.	200	0,92	0,84	9,52

Dari Tabel 4.7 maka dapat disimpulkan untuk *error* yang tertinggi ditunjukkan pada beban lampu 200 Watt yaitu 9,52 % untuk rata – rata *error* dari pengujian arus dengan beban lampu yakni sebesar 5,19 %.

Pengujian dengan menggunakan beban pada peralatan rumah tangga menggunakan skema pengujian yang ditunjukkan pada Gambar 4.13, hanya perbedaan terlektak pada beban yang digunakan.

Hasil pengujian dengan beban peralatan rumah tangga ditunjukkan pada Tabel 4.8

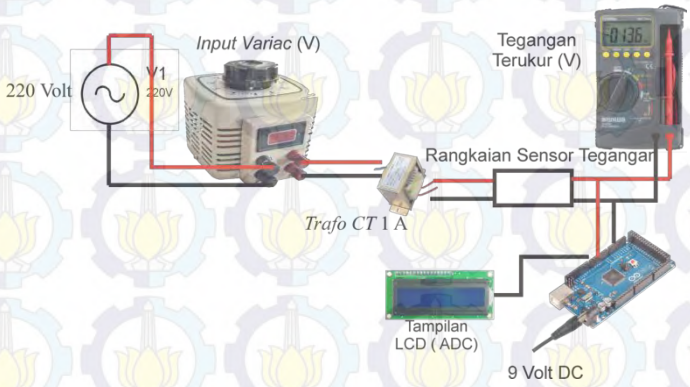
Tabel 4.8 Pengujian Beban Peralatan Rumah Tangga

Peralatan	Daya (Watt)	Arus Terukur (A)	Tampilan LCD (A)	*Error* (%)
Setrika	350	1,65	1,67	1,212
Heater	300	1,4	1,42	1,429
Magic Com	350	1,65	1,67	1,212
Dispenser	385	1,7	1,71	0,588
Setrika dan *Heater*	650	3,04	2,91	4,276
Setrika dan *Magic Com*	700	1,87	1,87	0
Setrika dan Dispenser	735	3,35	3,24	3,284
Dispenser dan *Magic Com*	735	3,35	3,24	3,284
Dispenser dan *Heater*	685	3,11	2,95	5,145
Magic Com dan *Heater*	650	3,04	2,95	2,961
Heater, Magic Com dan Dispenser	1035	4,72	4,48	5,085
Setrika, *Magic Com* dan Dispenser	1085	4,97	4,7	5,433
Heater, Setrika dan *Magic Com*	950	4,68	4,39	6,197
Heater, Setrika, *Magic Com*, dan Dispenser	1335	6,3	5,9	6,349

Dari hasil pengujian dengan beban berbagai peralatan listrik rumah tangga dapat ditarik kesimpulan bahwa rata rata *error* sebesar 3,318 % masih dapat diterima. *Error* tertinggi terdapat pada beban setrika, dispenser, *magic com* dan *heater* diukur secara bersamaan dengan *error* sebesar 6,349 %.

4.7 Pengujian Sensor Tegangan

Sensor tegangan digunakan untuk mengukur besar tegangan yang ada. Sensor tegangan yang dipakai adalah rangkaian pembagi tegangan yang sebelumnya melalui trafo *stepdown*. Sensor Tegangan diuji dengan menggunakan variac sebagai *input* yang dihubungkan dengan trafo dan multimeter "SANWA CD800a" untuk mengukur tegangan *output* dari sensor tegangan, seperti yang terlihat pada Gambar 4.14. Pengujian awal dilakukan dengan membaca *output* dari transformator yang dikonversikan menjadi ADC pada Arduino, yang ditampilkan pada LCD.



Gambar 4.14 Pengujian Sensor Tegangan

Data yang didapat dari pengujian tersebut terletak pada Tabel 4.9

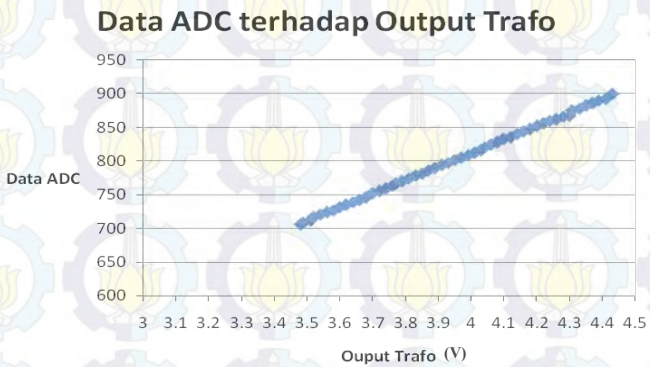
Tabel 4.9 Pengujian *Output* Transformator

No.	*Input* Variac (V)	Tegangan Terukur (V)	Tampilan LCD (ADC)
1.	200,6	3,48	706
2.	201,1	3,49	709
3.	202,2	3,51	712
4.	203	3,52	717
5.	204	3,54	721
6.	205	3,56	724
7.	206	3,58	727
8.	207	3,60	732
9.	208	3,62	735

No.	*Input* Variac (V)	Tegangan Terukur (V)	Tampilan LCD (ADC)
10.	209	3,64	739
11.	210	3,66	742
12.	211	3,68	746
13.	212	3,70	752
14.	213	3,72	755
15.	214	3,74	759
16.	215	3,76	763
17.	216	3,77	766
18.	217	3,79	770
19.	218	3,81	774
20.	219	3,83	778
21.	220	3,85	781
22.	221	3,87	786
23.	222	3,89	789
24.	223	3,91	794
25.	224	3,93	797
26.	225	3,95	802
27.	226	3,97	805
28.	227	3,99	809
29.	228	4,01	812
30.	229	4,03	816
31.	230	4,04	820
32.	231	4,06	824
33.	232	4,08	828
34.	233	4,10	833
35.	234	4,12	835
36.	235	4,14	840
37.	236	4,16	844
38.	237	4,18	847
39.	238	4,20	852
40.	239	4,22	856
41.	240	4,24	860
42.	241	4,26	862
43.	242	4,28	866

No.	*Input* Variac (V)	Tegangan Terukur (V)	Tampilan LCD (ADC)
44.	243	4,30	867
45.	244	4,31	875
46.	245	4,33	878
47.	246	4,35	882
48.	247	4,37	886
49.	248	4,39	890
50.	249	4,41	892
51.	250	4,43	899

Setelah mendapatkan data ADC dari *output* transformator dilakukan linearisasi data ADC yang ditunjukkan pada Gambar 4.15



Gambar 4.15 Perubahan Data ADC terhadap *Output* Trafo

Dari Gambar 4.15 tersebut dapat terlihat bahwa perubahan ADC sensor memiliki pergerakan yang linier terhadap perubahan tegangan *output* trafo. Sehingga dapat disimpulkan bahwa sensor tegangan dengan pembagi tegangan layak digunakan sebagai pendeteksi perubahan tegangan, karena memiliki perubahan *output* yang linier terhadap perubahan *input*.

Dari data pengujian dan grafik pengukuran, dapat dihasilkan suatu persamaan karakteristik sensor tegangan yang akan digunakan dalam pembacaaan Arduino agar didapatkan tegangan yang sesuai dengan tegangan pengukuran pada Voltmeter.

Setelah dilakukan perhitungan dengan menggunakan Persamaan 4.1, 4.2, 4.3 maka didapatkan sebuah persamaan untuk pembacaan Arduino :

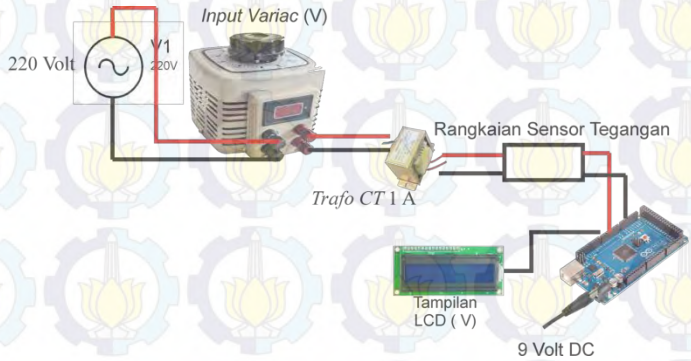
$$y = 0{,}259\ 434888 * x + 17{,}42846949 \quad \text{.....(4.7)}$$

Kemudian dilakukan penggantian variabel y menjadi tegangan yang diinginkan dan x menjadi ADC, sehingga didapatkan persamaan sebagai berikut :

$$Vout = 0{,}259\ 434888 * ADC + 17{,}42846949 \quad \text{.....(4.8)}$$

Setelah menambahkan Persamaan (4.8) kedalam program Arduino selanjutnya dilakukan pengujian hasil perhitungan dengan cara membandingkan pembacaan sensor tegangan yang ditampilkan pada LCD dengan tegangan *input* dari variac. Skema pengambilan data ditunjukkan pada Gambar 4.16

Data perbandingan pembacaan sensor tegangan dengan *input* variac ditunjukkan pada Tabel 4.10



Gambar 4.16 Skema Pengujian LCD dengan *Input* Variac

Tabel 4.10 Perbandingan Sensor Tegangan dengan *Input* Variac

No.	*Input* Variac (V)	Tampilan LCD (V)	*Error (%)*
1.	200,2	200,27	0,03
2.	201,0	200,79	0,10
3.	202,1	202,09	0,00
4.	203,2	203,12	0,03
5.	204,1	204,16	0,02
6.	205,2	204,94	0,12

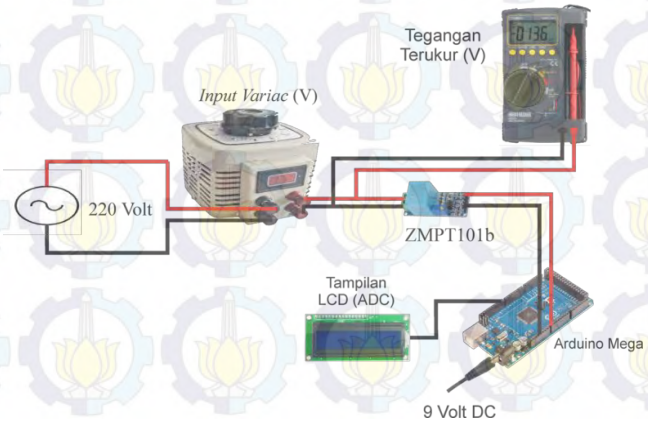
No.	*Input* Variac (V)	Tampilan LCD (V)	*Error (%)*
7.	206,3	206,23	0,03
8.	207,4	207,53	0,06
9.	208,5	208,3	0,09
10.	209,4	209,08	0,15
11.	210,4	210,12	0,13
12.	211,1	210,89	0,09
13.	212,2	212,19	0,00
14.	213,2	213,22	0,00
15.	214,1	214,0	0,04
16.	215,1	215,0	0,04
17.	216,5	216,59	0,04
18.	217,3	217,37	0,03
19.	218,3	218,40	0,04
20.	219,6	219,96	0,16
21.	220,3	220,22	0,03
22.	221,3	220,99	0,14
23.	222,1	221,77	0,14
24.	223,3	223,07	0,10
25.	224,2	224,1	0,04
26.	225,3	225,14	0,07
27.	226,2	225,91	0,12
28.	227,4	227,21	0,08
29.	228,4	228,25	0,06
30.	229,1	228,76	0,14
31.	230,1	230,06	0,01
32.	231,2	231,1	0,04
33.	232,1	231,87	0,09
34.	233,1	232,65	0,19
35.	234,7	234,46	0,10
36.	235,4	235,24	0,06
37.	236	235,76	0,10
38.	237	236,79	0,08
39.	238,1	237,83	0,11
40.	239,3	239,12	0,07
41.	240,2	239,90	0,12

No.	*Input* Variac (V)	Tampilan LCD (V)	*Error (%)*
42.	241	240,68	0,13
43.	242,2	241,71	0,20
44.	243,5	243,27	0,09
45.	244,6	244,30	0,12
46.	245,3	245,08	0,08
47.	246,3	245,86	0,17
48.	247,1	246,89	0,08
49.	248,2	247,93	0,10
50.	249,3	249,23	0,02
51.	250,3	250,00	0,11

Dari data hasil perbandingan pada Tabel 4.10 dapat dilihat bahwa *error* terbesar terjadi pada data ke-43 yaitu sebesar 0,20% dan *error* terkecil terjadi pada data ke-3 yaitu sebesar 0,00%.Sedangkan nilai *error* rata-rata adalah 0,08%. *Error* yang terjadi masih berada dalam batas yang aman. Dan sensor tegangan ini dapat digunakan.

4.8 Pembacaaan Sensor ZMPT101b

Pada pengujian sensor tegangan ZMPT101b menggunakan variac yang dihubungkan ke sumber PLN sebagai *input* dan multimeter "SANWA CD800a" sebagai pembaca tegangan terukur. *Variac* dapat mengeluarkan *output* tegangan AC yang nilainya dapat diatur sesuai kebutuhan. Skema pengambilan data ditunjukkan pada Gambar 4.17



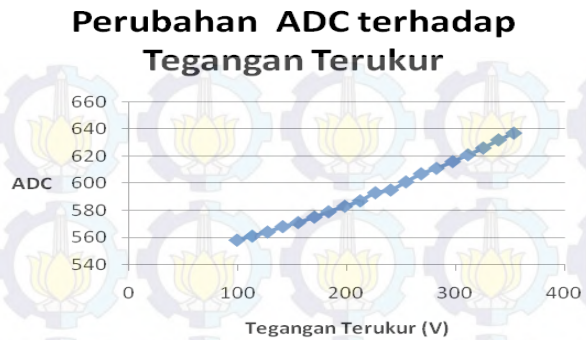
Gambar 4.17 Skema Pengujian ADC ZMPT101b

Pada pengujian ini diambil 19 data, yaitu dari 250 V – 70 V. Pengambilan data ini untuk mendapatkan nilai ADC dari setiap perubahan nilai *input* yang diberikan. Data yang didapat pada pengujian ini ditunjukkan pada Tabel 4.11

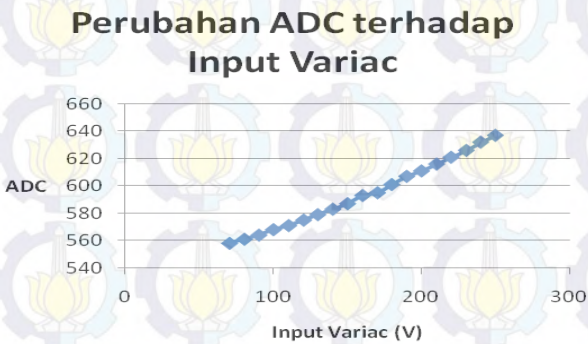
Tabel 4.11 Data ADC ZMPT101b

No.	*Input* Variac (V)	Tegangan Terukur (V)	Tampilan LCD (ADC)
1.	250	250	637
2.	240	240,3	632
3.	230	230,4	626
4.	220	220,2	621
5.	210	210,3	616
6.	200	200,1	611
7.	190	190,1	607
8.	180	180,5	601
9.	170	170,4	595
10.	160	160,3	593
11.	150	150,1	587
12.	140	140,2	583
13.	130	130	579
14.	120	120,4	575
15	110	110,3	571
16.	100	100	568
17.	90	90,2	564
18.	80	80,4	561
19.	70	70,3	558

Dari data pengujian sensor dibuat grafik perubahan tegangan *input* terhadap ADC pada Gambar 4.18 dan Gambar 4.19. Dari gambar tersebut dapat terlihat bahwa perubahan ADC sensor memiliki pergerakan yang linier terhadap perubahan tegangan *input*. Sehingga dapat disimpulkan bahwa sensor ZMPT101b layak digunakan sebagai pendeteksi perubahan tegangan, karena memiliki perubahan *output* yang linier terhadap perubahan *input*.



Gambar 4.18 Perubahan ADC terhadap Tegangan Terukur



Gambar 4.19 Perubahan ADC terhadap *Input* Variac

Dari data pengujian dan grafik pengukuran, dapat dihasilkan suatu persamaan karakteristik sensor tegangan yang akan digunakan dalam pembacaaan Arduino agar didapatkan tegangan yang sesuai dengan tegangan pengukuran pada Voltmeter.

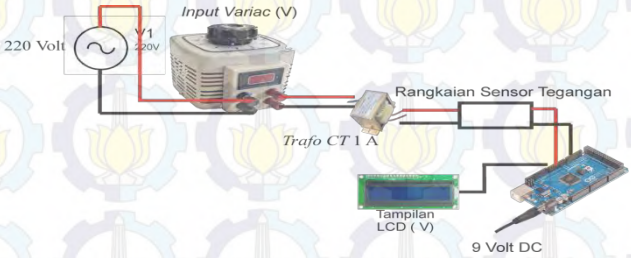
Setelah dilakukan perhitungan dengan menggunakan Persamaan 4.1, 4.2, 4.3 maka didapatkan sebuah persamaan untuk pembacaan Arduino :

$$y = 1{,}625498187 * x - 849{,}342626 \quad \text{.....(4.9)}$$

Kemudian dilakukan penggantian variabel y menjadi tegangan RMS yang diinginkan dan x menjadi ADC, sehingga didapatkan persamaan sebagai berikut :

$$Vrms = 1{,}625498187 * ADC - 849{,}342626 \quad \text{.....(4.10)}$$

Setelah menambahkan Persamaan (4.10) kedalam program Arduino selanjutnya dilakukan pengujian hasil perhitungan dengan cara membandingkan pembacaan sensor ZMPT101b pada LCD dengan *input* tegangan dari variac. Skema pengujian ditunjukkan pada



Gambar 4.20 Skema Pengujian *Output* ZMPT101b

Data perbandingan pembacaan sensor tegangan dengan *input* variac ditunjukkan pada Tabel 4.12

Tabel 4.12 Perbandingan LCD dengan Multimeter

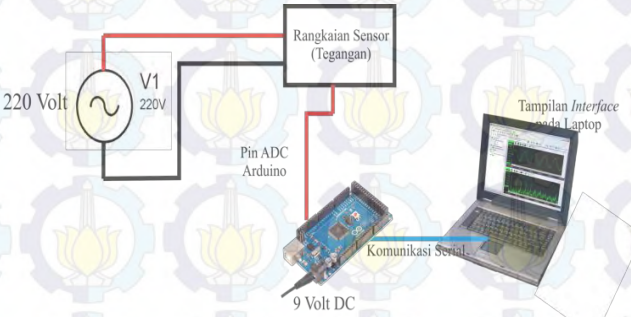
No.	*Input* Variac (V)	Tampilan LCD (V)	*Error (%)*
1.	201,7	201,6	0,04
2.	207,0	207,1	0,04
3.	211,1	211,6	0,23
4.	214,6	215,2	0,27
5.	218,4	218,8	0,18
6.	221,0	221,0	0
7.	223,2	223,6	0,17
8.	227,1	227,7	0,26
9.	230,1	230,0	0,04
10.	233,6	233,4	0,08
11.	237,2	236,8	0,16
12.	243,1	242,8	0,12
13.	244,0	243,7	0,12
14.	244,5	243,8	0,28
15.	248,5	248,0	0,20

Dari data hasil perbandingan pada Tabel 4.12 dapat dilihat bahwa *error* terbesar terjadi pada data ke-1 yaitu sebesar 0,68% dan *error* terkecil terjadi pada data ke-9 yaitu sebesar 0,06%. Sedangkan nilai *error* rata-rata adalah 0,35%. *Error* yang terjadi masih berada dalam batas toleransi.

4.9 Pengujian Komunikasi Serial

Pengujian ini dilakukan dengan menghubungkan Arduino dengan komputer. Hal ini dilakukan agar memastikan Arduino terhubung dengan baik pada komputer, karena data yang dikirim oleh Arduino merupakan data *sampling* tegangan yang natinya akan dianalisa pada Matlab.

Pengujian ini dilakukan dengan menggunakan *software* Arduino IDE dengan memanfaatkan fasilitas serial *monitor* sebagai tampilan dari data yang coba dikirim oleh Arduino Mega 2560 dengan menggunakan kabel USB. Sebelum melakukan pengujian ini Arduino diberi program untuk membaca sensor tegangan yang terhubung dengan ADC Arduino serta pengiriman data sensor tegangan ke laptop dengan mengaktifkan komunikasi serial dengan perintah "Serial.println", skema pengujian ditunjukkan pada Gambar 4.21 dan hasil pengujian pada



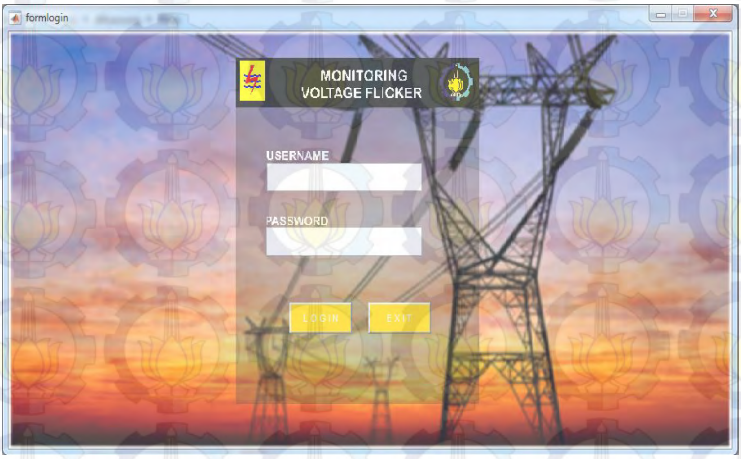
Gambar 4.21 Skema Pengujian Komunikasi Serial

Gambar 4.22 Hasil Pengujian Komunikasi Serial

Pada Gambar 4.22 terlihat tampilan pada serial monitor yang menandakan bahwa Arduino telah terhubung dengan komputer. Dan komunikasi ini nantinya akan dihubungkan dengan *software* Matlab.

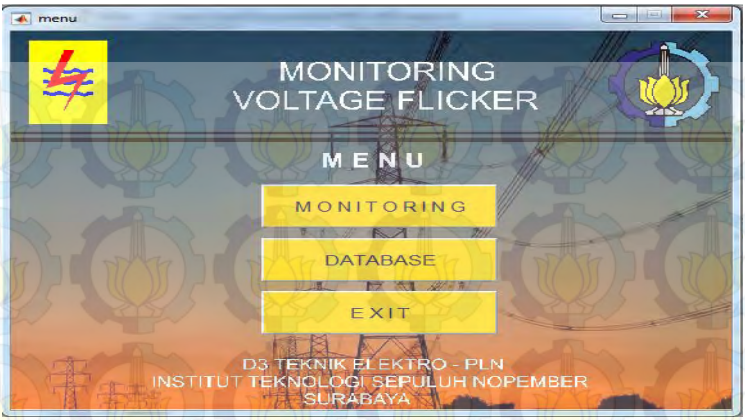
4.10 Pengujian *Software* Matlab

Untuk bisa mengetahui ada atau tidaknya atau mengetahui bentuk gelombang dari *Voltage Flicker*, diperlukan *interface* guna menghubungkan alat kita dengan manusia. *Interface* yang kita gunakan adalah dengan *software* Matlab. Pengujian Matlab ini digunakan untuk mengetahui Matlab sudah bekerja sesuai dengan sistem yang kita inginkan atau belum. Sebelum memasuki *form* utama *monitoring,* terlebih dahulu harus memasuki *form login*. *Form login* ini berfungsi sebagai perlindungan agar yang tidak berkepentingan dan tidak berizin tidak bisa menggunakan aplikasi *monitoring Voltage Flicker*. Gambar 4.23 diperlihatkan *form login* yang kami gunakan.



Gambar 4.23 Tampilan *Form Login*

Untuk bisa memasuki *form menu* dan *form monitoring,* pengguna harus memasukan *username* dan *password* yang sesuai. Jika tidak sesuai pengguna tidak akan bisa mengakses *form* inti *monitoring*. Setelah sukses melakukan *login,* maka pengguna akan masuk ke *form* selanjutnya yaitu *form menu.* Tampilan *form menu* sebagai Gambar 4.24 berikut.



Gambar 4.24 Tampilan *Form* Menu

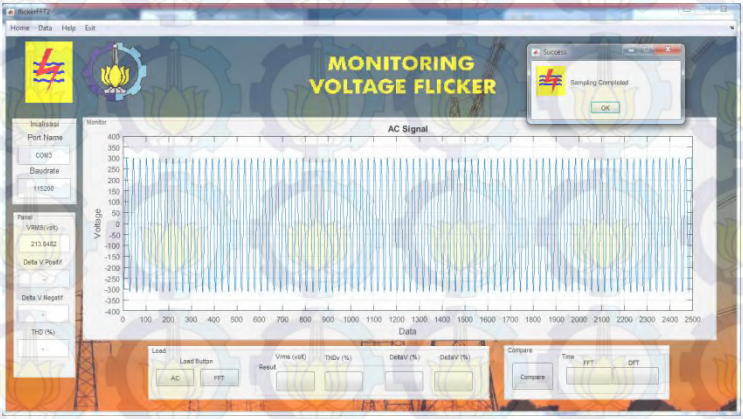
Dari *form menu*, terdapat pilihan pilihan berupa tombol *monitoring*, *database, dan exit.* Tombol tombol tersebut yang mengantar kita ke *form monitoring* dan ke *form database* atau ingin keluar dari aplikasi *monitoring Voltage Flicker*. Ketika kita memilih tombol *monitoring,* maka kita akan masuk ke *form monitoring.* Tampilan *form monitoring* dapat dilihat pada Gambar 4.25

Gambar 4.25 Tampilan *Form Monitoring*

Dalam *form monitoring* kita dapat mengetahui bentuk gelombang dari *Voltage Flicker* dan juga besaran besaran yang bersangkutan. Untuk

pengujian dari Matlab tampilan *form monitoring* , dapat memalui beberapa tahap sebagai berikut.

1. Menghubungkan kabel USB dari alat ke laptop untuk mengetahui apakah perangkat *hardware* dan *software* sudah terhubung dengan baik atau belum.
2. Banyak data yang diambil yaitu 2500 data, dilakukan dalam 1 kali *sampling* dalam waktu 1,8 detik. Yang nantinya akan ditampilkan pada grafik gelombang untuk dilakukan pengamatan terhadap tegangan tersebut, ditunjukkan pada Gambar 4.26



Gambar 4.26 Form Monitoring Setelah Sukses Berkomunikasi

Dari gambar tampilan diatas, grafik sinus diperoleh dari *sampling* olahan *ADC* dari perangkat *hardware.* Dengan terlihatnya grafik sinus tersebut menandakan bahwa antara perangkat *hardware* dan *software* tersambung dengan baik.

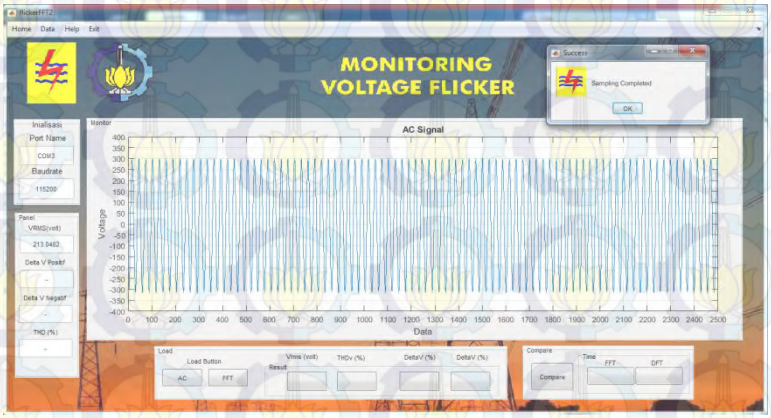
3. Setelah berhasil berkomunikasi dan menampilkan *plotting* hasil *sampling,* selanjutnya adalah transformasi ke metode FFT. Matlab dituntut mampu melakukan perhitungan FFT guna mentransformasikan sinyal tegangan dari domain waktu menjadi domain frekuensi. Sesuai Persamaan 2.1.

Dari Persamaan 2.1, dalam penggunaan di *software* Matlab menggunakan fungsi FFT yang sudah disediakan oleh *software* Matlab. Program ditunjukkan pada Gambar 4.27

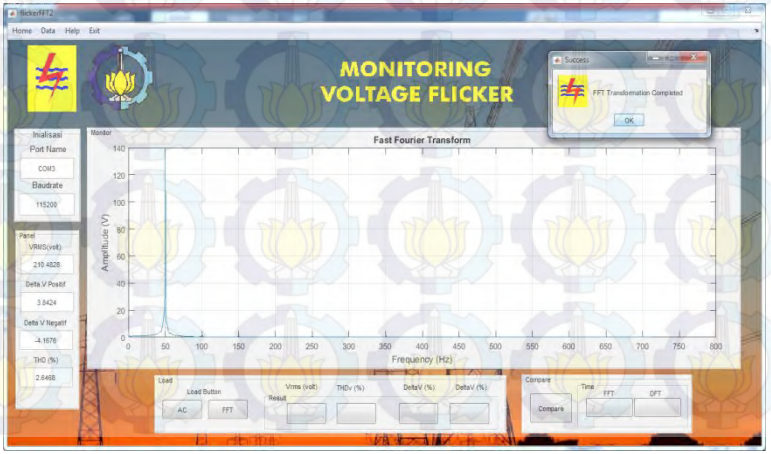
```
sketch_jun13a §
xfft10 = fft(sample,nfft);
x10 = xfft10(1:nfft/2);
magx10a = abs(x10)/nfft;
```

Gambar 4.27 Fungsi FFT Pada *Software* Matlab

Untuk transformasi dari gelombang sinus menuju FFT dapat dilihat melalui Gambar 4.28 dan Gambar 4.29 berikut.

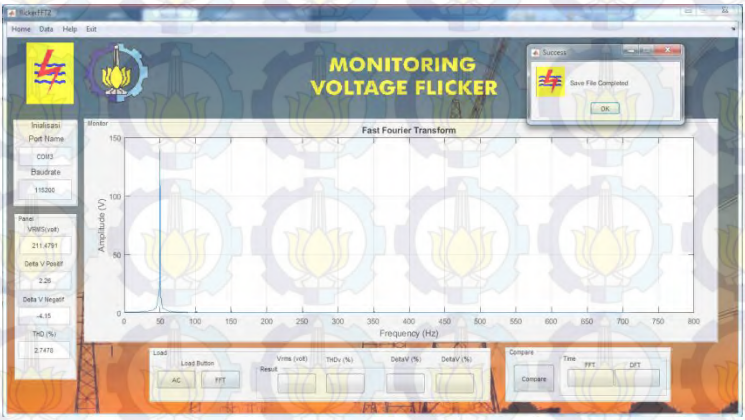


Gambar 4.28 Gelombang Sinus Setelah Sukses Melakukan *Sampling*

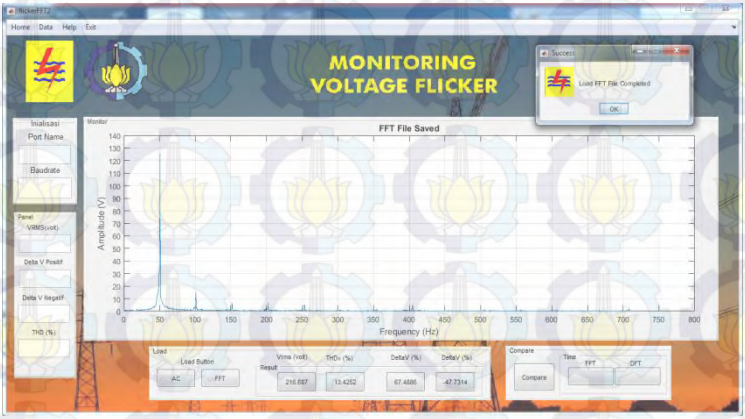


Gambar 4.29 Hasil Transformasi FFT

4. Setelah melakukan *plotting* hasil *sampling* data dan berhasil transformasi melalui metode FFT, maka hasil dapat disimpan dengan hasil *.mat. setelah disimpan dapat ditampilan hasil penyimpanan tersebut. Lebih jelasnya dapat melihat pada Gambar 4.30 dan Gambar 4.31 berikut.



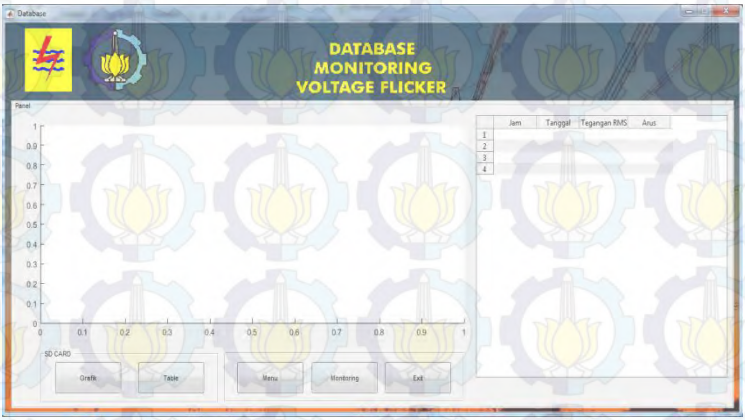
Gambar 4.30 Setelah Sukses Menyimpan



Gambar 4.31 Setelah Sukses Menampilkan Hasil Penyimpanan

Setelah dari tampilan *monitoring*, kita dapat memanfaatkan fitur lainnya dari *form menu*, yaitu *database*. Pada *database* akan diperlihatkan hasil data keselurahan berupa tanggal *sampling*, besar tegangan RMS dan juga besar arus. Selain itu kita juga bisa melihat

bentuk gelombang dari tegangan RMS, melalui *plotting*. Hasil tampilan *form database* seperti Gambar 4.32 berikut.



Gambar 4.32 Hasil *Form Database*

4.11 Pengujian Metode FFT

Pengujian ini dilakukan untuk membuktikan metode FFT yang digunakan pada Tugas Akhir ini sesuai dengan beberapa kondisi yang telah benar adanya, seperti pengujian komparasi waktu transformasi antara metode FFT dan DFT, pengujian hasil transformasi yang dibandingkan dengan alat ukur Fluke, dan yang terahir yakni pengujian indikator yang terdapat pada *form monitoring Voltage flicker*.

4.11.1 Komparasi Waktu Transformasi Metode FFT dan DFT

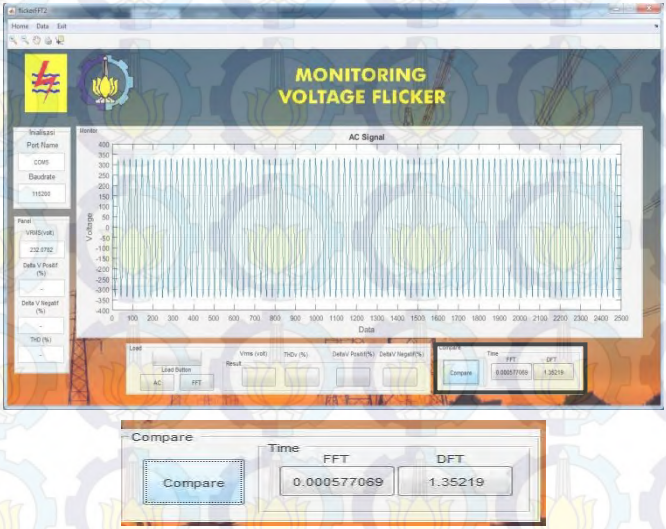
Pengujian ini dilakukan untuk membuktikan bahwa kebutuhan waktu transformasi dari kedua metode ini berbeda, sesuai dengan adanya metode FFT bahwa untuk mempersingkat proses transformasi data, dengan kata lain bahwa metode FFT lebih cepat dari pada metode DFT.

Persamaan untuk metode FFT dan DFT dalam melakukan satu kali proses transformasi hingga mendapatkan hasil perubahan grafik seperti di bawah ini,

$$n^2 \quad \text{.....(4.11)}$$

$$n \log n \quad \text{.....(4.12)}$$

Dari Persamaan 4.11 yang merupakan persamaan dari proses tranformasi DFT dan 4.12 merupakan proses transformasi dari FFT, *n* merupakan banyak data *sampling*, jadi apabila kita melakukan *sampling* sebnyak 2500 data maka proses transformasi pada DFT sesuai dari Persamaan 4.11 maka membutuhkan waktu proses sebanyak 2500^{2500}, sedangkan apabila menggunakan FFT maka diperlukan waktu sekitar $2500\log 2500$ yakni sekitar 8494 kali proses perhitungan, jadi proses FFT lebih cepat daripada metode DFT, hal ini juga kami buktikan dalam *form monitoring flicker* yang ditunjukkan pada Gambar 4.33



Gambar 4.33 Hasil Komparasi

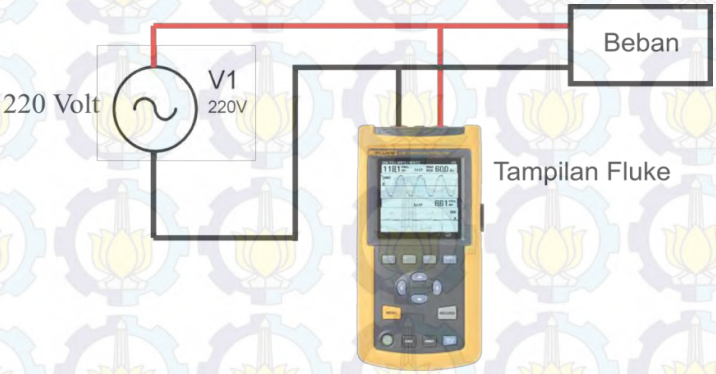
Dari Gambar 4.33 terlihat perbedaan waktu antara FFT dan DFT. FFT hanya membutuhkan 0,000577069 detik untuk satu kali proses transformasi, sedangkan untuk DFT membutuhkan waktu 1,35219 detik untuk satu kali proses transformasi.

Dengan hasil tersebut maka metode FFT terbukti lebih cepat dalam proses transformasinya.

4.11.2 Pengujian dengan *Power System Analyzer*

Dilakukan pengujian ini untuk membandingkan hasil transformasi dari Tugas Akhir ini dengan *Power System Analyzer*. Hal bertujuan untuk mengetahui hasil trasformasi dengan metode FFT pada Tugas

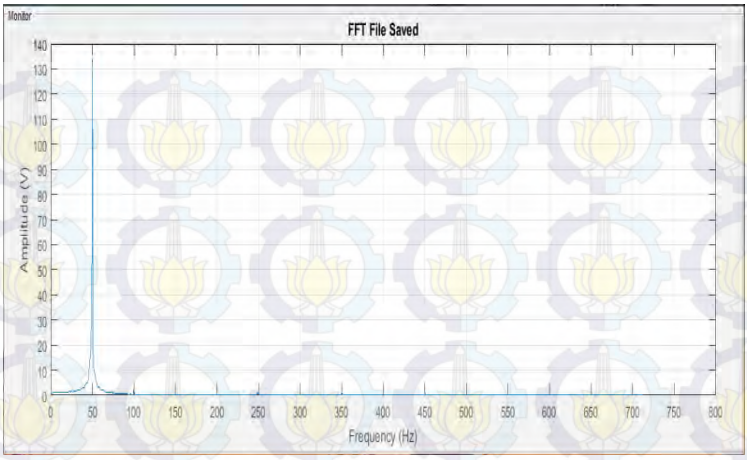
Akhir ini apakah sudah sesuai dengan alat ukur yang telah diuji kebenarannya. Pada pengujian ini menggunakan alat ukur *Power System Analyzer* yang terhubung dengan sumber tegangan, dan diparalel dengan beban yang nantinya akan diambil datanya. Skema pengujian ditunjukkan pada Gambar 4.34



Gambar 4.34 Skema Pengujian *Power System Analyzer*

Pada pengujian ini digunakan beban gerinda tangan maktec, pengambilan data dilakukan ketika gerinda ini aktif. Hasil pengujian ditunjukkan pada Gambar 4.35 merupakan hasil dari pengukuran *Power System Analyzer* dan Gambar 4.36 merupakan hasil dari Tugas Akhir ini.

Gambar 4.35 Hasil *Power System Analyzer*



Gambar 4.36 Hasil Transformasi FFT

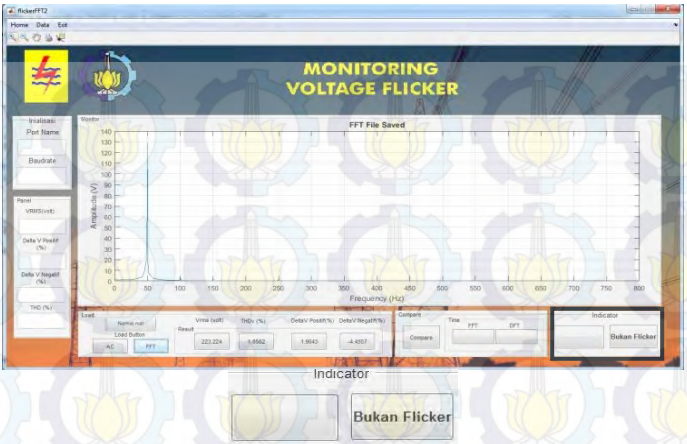
Dari Gambar 4.35 dan Gambar 4.36 terlihat hasil transformasi untuk beban dalam keadaan aktif, yang hasilnya menunjukkan pada frekuensi fundamental 50 Hz akan muncul, sedangkan pada kelipatannya tidak ada puncak yang muncul dari kedua gambar diatas.

Dengan demikian dapat disimpulkan bahwa hasil dari metode FFT yang digunakan pada Tugas Akhir ini telah sesuai dengan hasil alat ukur yang telah teruji kebenarannya, dan dapat diartikan bahwa alat ini dapat digunakan untuk pengambilan data *Voltage Flicker.*

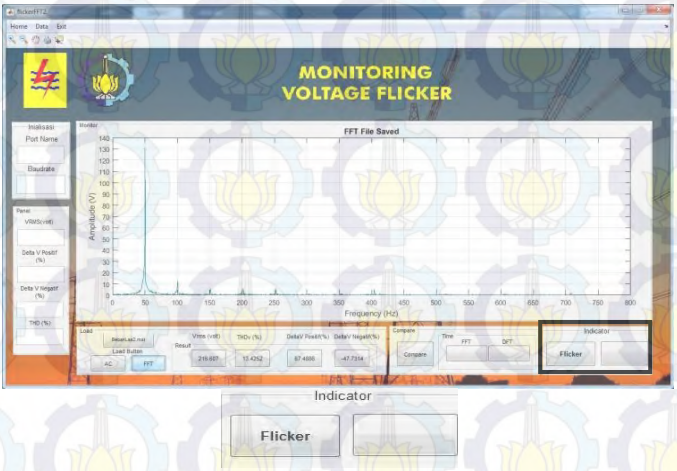
4.11.3 Pengujian Indikator

Tampilan pada *interface* pada Tugas Akhir ini dilengkapi dengan indikator untuk menandakan ada atau tidaknya *flicker*. Untuk mengetahui apakah fungsi dari indikator ini dapat berjalan dengan baik maka diperlukan pengujian pada tampilan *interface*. Untuk tampilan indikator pada *interface* ditunjukkan pada Gambar 4.25.

Dilakukan pengujian terhadap data yang menunjukkan adanya *flicker* dan tidak adanya gangguan *flicker*. Hasil dari pengujian ditunjukkan pada Gambar 4.37 untuk hasil normal, sedangkan untuk Gambar 4.38 merupakan hasil transformasi yang menunjukkan adanya gangguan *flicker.*



Gambar 4.37 Indikator Tidak Adanya *Flicker*



Gambar 4.38 Indikator Adanya *Flicker*

4.12 Pengujian Alat Keseluruhan

Pengujian keseluruhan dilakukan untuk mengetahui apakah alat *Monitoring Voltage Flicker* pada Tugas Akhir ini berjalan baik atau tidak. Pengujian dilakukan sebanyak dua macam yaitu, pengujian terhadap beban dan pengujian pembacaan data yang tersimpan pada *SD Card*.

Pegujian dengan menggunakan beban dilakukan sebanyak lima kali pada lima macam variasi beban yang berbeda – beda. Hal ini dilakukan untuk mendapatkan hasil yang akurat sehingga Tugas Akhir ini bisa dikatakan berhasil. Beban yang digunakan merupakan beban yang biasa digunakan dilingkungan sekitar seperti mesin las listrik, gerinda tangan dan sebagianya. Untuk skema pengujian ditunjukkan pada Gambar 4.39

Gambar 4.39 Skema Pegujian Keseluruhan

Dari skema pada Gambar 4.39 akan didapatkan hasil pengukuran yang akan ditampilkan pada layar laptop yang telah dibuat tampilan *interface* yang telah ditunjukkan pada Gambar 4.25 data yang didapatkan berupa nilai Vrms (V), ΔV Positif (V), ΔV Negatif (V), dan THDV (%). Selanjutnya akan ditranformasikan dengan menggunakan metode FFT, dan hasil tersebut akan dibandingkan dengan kondisi tegangan yang normal yakni ketika beban tidak aktif.

4.12.1 Pengujian Las Listrik Falcon 211GE

Pada pegujian ini menggunakan mesin las listrik Falcon 211GE dengan spesifikasi sebagai berikut [13]:

Voltage	: 160 - 250 Volt
Daya listrik	: 1300 - 2200 Watt
Arus *output*	: 20 - 200 Ampere
Diameter kawat las	: 2,0 - 5,0 mm
Ukuran soket	: 25 mm
Pendingin	: kipas
Duty cycle	: 60 % (pada 200 A), 100 % (pada 160 A)

Skema pengujian sesuai dengan Gambar 4.39. Pengujian ini untuk mengetahui pengaruh mesin las Falcon 211GE terhadap kualitas tegangan terutama pada saat mesin las tersebut dalam keadaan aktif. Pengujian ditunjukkan pada Gambar 4.40 Nilai yang dibandingkan pada pengujian ini diantaranya : grafik dari tegangan ketika mesin las aktif dan ketika tidak adanya beban, selain itu kondisi ΔV dan THDV juga akan dilihat dan dibandingkan nilainya.

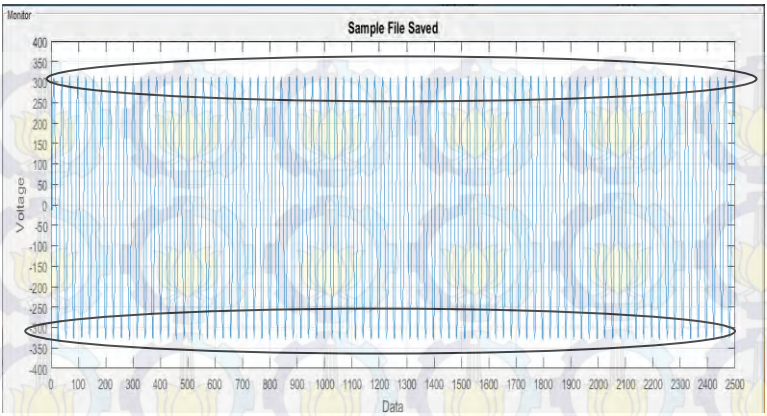
Gambar 4.40 Pengujian Las Listrik Falcon 211GE

Setelah dilakukan pengujian, maka data yang tampil pada *interface* berupa hasil pengujian terhadap las listrik Falcon 211GE yang terdapat pada Tabel 4.13

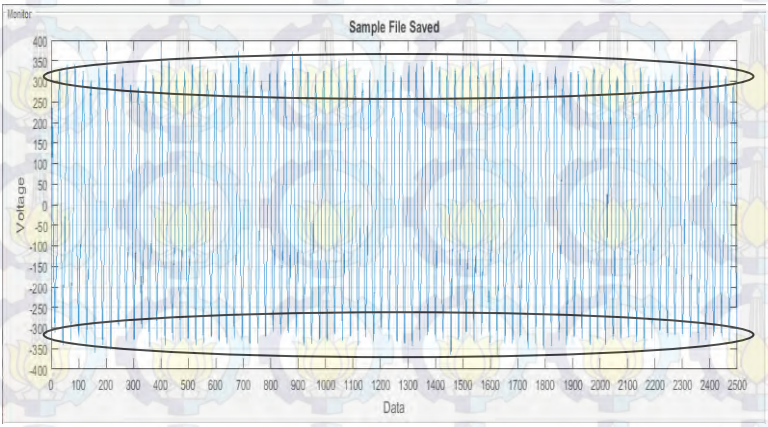
Tabel 4.13 Data Pengujian Las Listrik Falcon 211GE

No.	Kondisi	Vrms (V)	ΔV Positif (V)	ΔV Negatif (V)	THDV (%)
1.	Tanpa Beban	223,514	2,0093	-4,3907	2,8625
2.	Las Aktif	216,607	67,4886	-47,7314	13,4252

Dari Tabel 4.13 terlihat perbedaan antara kondisi tidak berbeban (mesin las tidak aktif) dan ketika mesin las aktif. Dari kondisi Vrms yang turun ketika mesin las aktif, adanya pengaruh terhadap tegangan ketika mesin las aktif. Selain itu ΔV yang naik drastis dengan perbedaan yang sangat besar, mengindikasikan bahwa kondisi tegangan yang tidak stabil seperti pada kondisi tanpa beban. Data yang diambil dengan *sampling* tegangan sebanyak 2500 data *sampling*. Grafik gelombang dalam keadaan normal dan kondisi mesin las aktif ditunjukkan pada Gambar 4.41 dan pada Gambar 4.42



Gambar 4.41 Gelombang Ketika Tanpa Beban



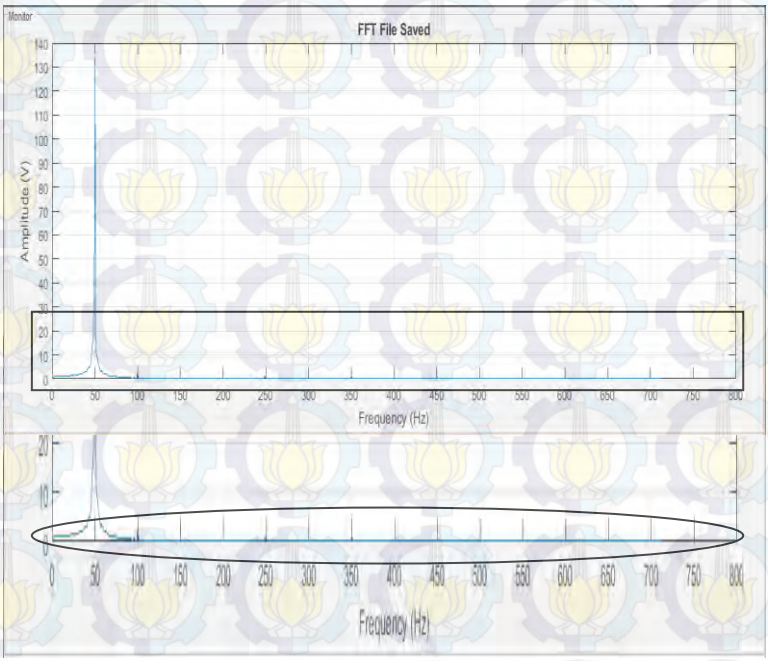
Gambar 4.42 Gelombang Ketika Mesin Las Aktif

Dari kedua Gambar 4.41 dan Gambar 4.42 terlihat perubahan kondisi gelombang tegangan ketika mesin las diaktifkan. Mesin las akan menarik tegangan yang besar ketika diaktifkan. Pada Gambar 4.41 ketika kondisi tanpa beban terlihat gelombang yang stabil tanpa adanya perubahan naik atau turunnya tegangan. Hal tersebut yang menyebabkan ΔV yang tidak terlalu tinggi perubahannya.

Pada gelombang yang ditunjukkan pada Gambar 4.42 terlihat adanya perubahan naik dan turunnya tegangan secara tidak beraturan. Hal tersebut berpengaruh terhadap kondisi ΔV yang berubah sangat

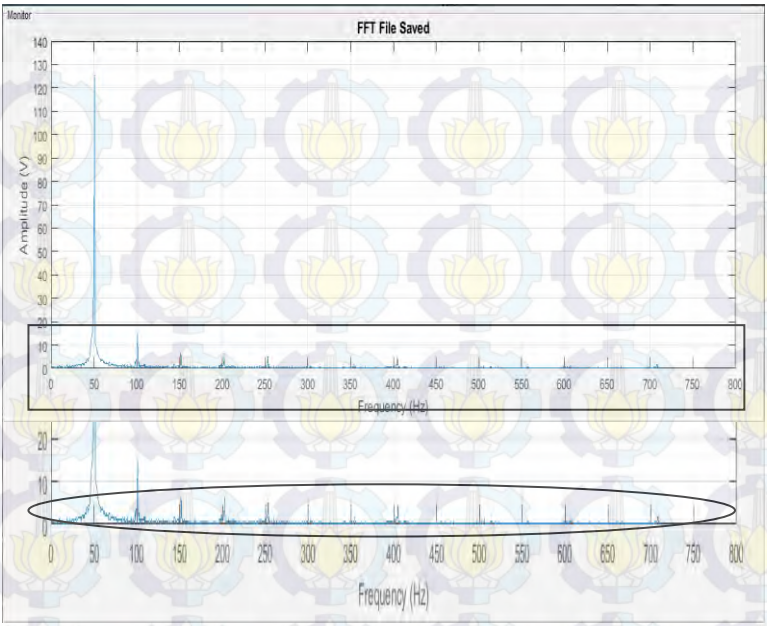
tinggi, berarti terjadi perbedaan tegangan dalam waktu yang terus menerus selama mesin las diaktifkan.

Selanjutnya akan dibandingkan grafik hasil transformasi dari domain waktu menjadi domain frekuensi menggunakan metode FFT. Gambar 4.43 adalah grafik hasil transformasi gelombang tegangan tanpa beban, pada Gambar 4.44 adalah grafik hasil transformasi gelombang saat melakukan pengelasan.



Gambar 4.43 Transformasi Gelombang Tanpa Beban

Dari Gambar 4.43 tidak terlihat adanya riak pada sekitar frekuensi 50 Hz. Dan pada frekuensi kelipantannya yaitu 100 Hz, 150 Hz, 200 Hz, tidak terlihat adanya lonjakan, berarti kondisi tegangan dalam keadaan normal tidak ada harmonisa tegangan yang terjadi pada kondisi tanpa beban.



Gambar 4.44 Transformasi Kondisi Mesin Las Aktif

Pada Gambar 4.44 terlihat adanya lonjakan pada frekuensi selain frekuensi fundamental yaitu pada 50 Hz. Terdapat riak disekitar frekuensi 50 Hz berbeda dengan kondisi tanpa beban yang tidak ada riak disekitar frekuensi fundamental. Selain itu pada frekuensi kelipatan 50 Hz yaitu, 100 Hz adanya lonjakan yang sangat tinggi sekitar 13 Volt, pada frekuensi 150 Hz juga terdapat lonjakan sekitar 5 Volt, dan pada kelipatanya yakni 200 Hz, 250 Hz terdapat lonjakan tetapi sudah menurun. Dari hasil tersebut yang berpengaruh terhadap nilai THDV yang sangat tinggi yakni sekitar 13,4252 % .

Setelah dilakukan pengujian terhadap mesin las listrik Falcon 211GE yang didapatkan data-data diatas dapat disimpulkan, ketika mesin las aktif maka berpengaruh terhadap kondisi tegangan sekitar dimana hal tersebut dapat mempengaruhi peralatan yang ada disekitar. Selain itu mesin las Falcon 211GE berpotensi menghasilkan *flicker* yang dibuktikan dengan adanya gelombang yang tidak stabil yaitu pada Gambar 4.42 dimana terjadi naik dan turunya tegangan yang tak beraturan. Serta terjadi lonjakan frekuensi yang ditunjukkan pada

Gambar 4.44 yang seharusnya tidak ada lonjakan selain frekuensi fundamental.

4.12.2 Pengujian *Drill Press*

Pada pegujian ini menggunakan *Drill Press* atau biasa disebut bor duduk dengan spesifikasi sebagai berikut [14]:

Voltage : 220 Volt
Daya listrik : 223,71 Watt
Arus : 1,8 A
Kecepatan putar : 1400 r/min
Frekuensi : 50 Hz
Jumlah kutub : 4

Skema pengujian sesuai dengan Gambar 4.39. Pengujian ini bertujuan untuk mengetahui seberapa besar pengaruh *Drill Press* terhadap kulitas daya ketika peralatan ini beroperasi. Pengambilan data tegangan dilakukan dua kali yakni ketika kondisi *Drill Press* belum aktif dan ketika *Drill Press* aktif. Pengambilan *sampling* tegangan sebanyak 2500 data dalam waktu 1,8 detik. Pengujian *Drill Press* ditunjukkan pada Gambar 4.45. Data yang akan dibandingkan pada pengujian ini yakni grafik tegangan, ΔV, dan THDV.

Gambar 4.45 Pengujian *Drill Press*

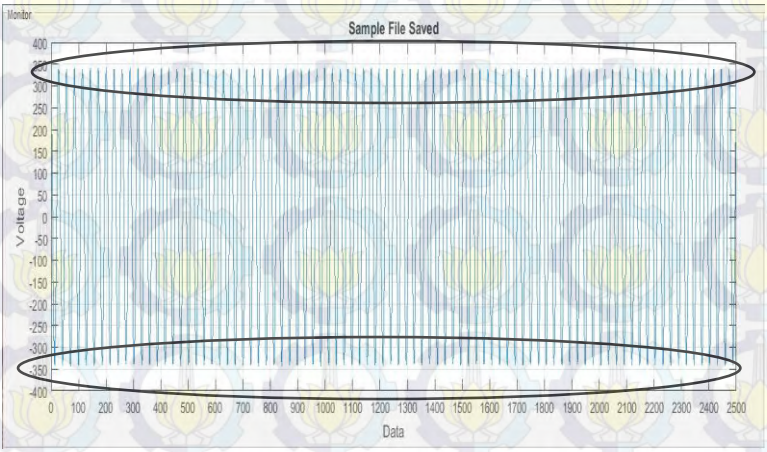
Setelah dilakukan pengujian, maka data yang tampil pada *interface* berupa hasil pengujian *Drill Press* yang terdapat pada Tabel 4.14

Tabel 4.14 Data Pengujian *Drill Press*

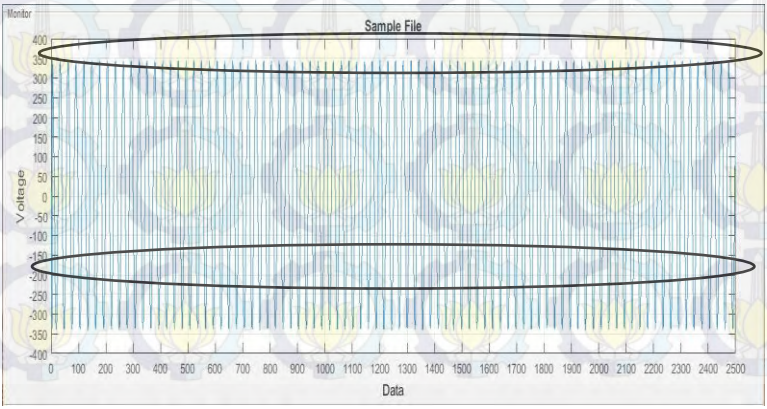
No.	Kondisi	Vrms (V)	ΔV Positif (V)	ΔV Negatif (V)	THDV (%)
1.	Tanpa Beban	236,11	0,85	-0,74	1,24

No.	Kondisi	Vrms (V)	ΔV Positif (V)	ΔV Negatif (V)	THDV (%)
2.	Bor Aktif	238,04	1,69	-3,10	2,19

Dari Tabel 4.14 terlihat perbedaan data antara kedua kondisi, tetapi perubahan nilai tidak terlalu jauh dari kondisi beban tidak aktif. Gelombang tegangan yang ditunjukkan pada Gambar 4.46 merupakan kondisi beban *Drill Press* tidak aktif, sedangkan untuk gelombang tegangan yang ditunjukkan pada Gambar 4.47 merupkan kondisi *Drill Press* aktif.



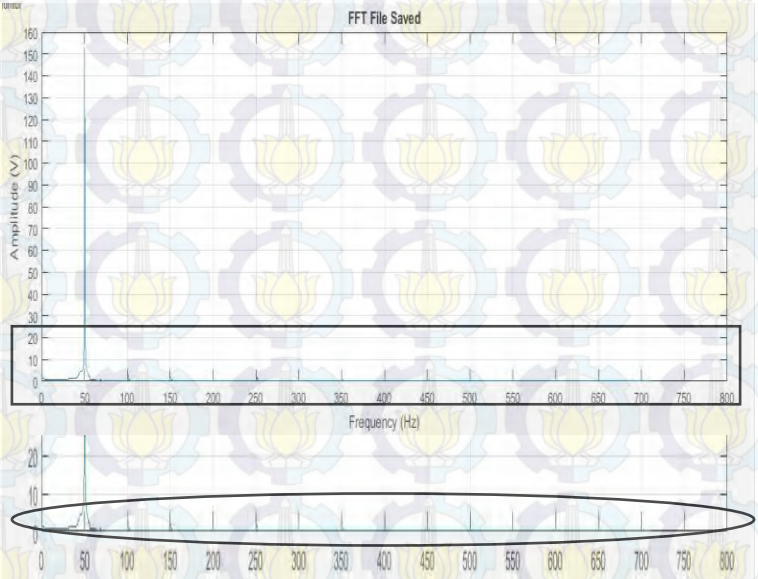
Gambar 4.46 Kondisi *Drill Press* Tidak Aktif



Gambar 4.47 Kondisi *Drill Press* Aktif

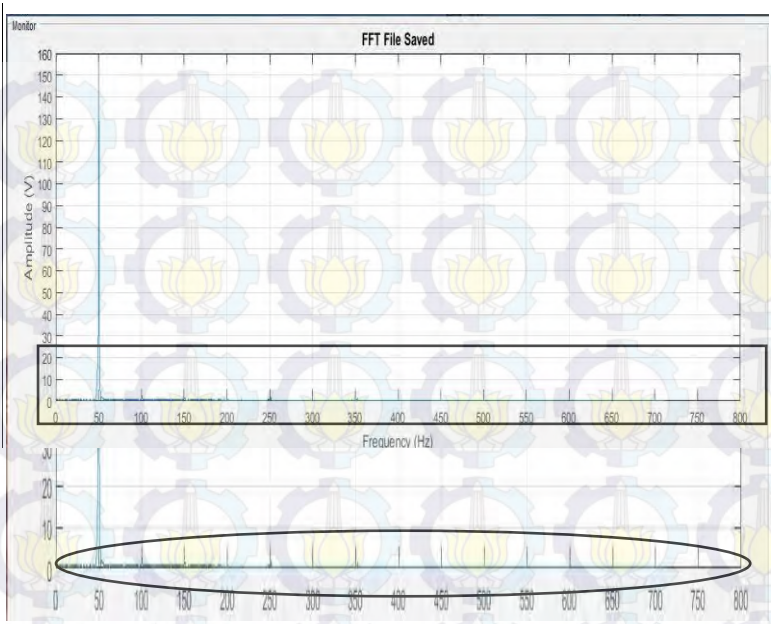
Pada Gambar 4.46 dan Gambar 4.47 tidak terlihat adanya perubahan dari dua kondisi beban. Kondisi tegangan dari kedua gambar tersebut terlihat stabil untuk puncak dan lembahnya. Hanya ada riak pada puncak dan lembah dari kedua gambar tersebut yang menyebabkan nilai THDV meningkat.

Sedangkan untuk transformasi gelombang tersebut ditunjukkan pada Gambar 4.48 untuk beban *Drill Press* tidak aktif, dan Gambar 4.49 untuk beban *Drill Press* aktif.



Gambar 4.48 Transformasi Kondisi *Drill Press* Tidak Aktif

Hasil transformasi pada Gambar 4.48 tidak adanya gangguan pada gelombang tegangan, yang ditandai dengan tidak adanya puncak selain frekuensi fundamental yakni 50 Hz. Disekitar frekuensi 50 Hz tidak terbangkit riak yang menimbulkan adanyan gelombang pada frekuensi lain, untuk kondisi diatas termasuk kondisi dimana keadaan tegangan normal. Untuk membandingkan kondisi ini dengan *Drill Press* aktif maka dilakukan transformasi pada data gelombang ketika *Drill Press* aktif, yang ditunjukkan pada Gambar 4.49.



Gambar 4.49 Transformasi Kondisi *Drill Press* Aktif

Setelah dilakukan transformasi dengan menggunakan FFT, bentuk grafik berubah kedalam domain frekuensi, dan mempermudah pengamatan dari gelombang tegangan tersebut. Pada Gambar 4.49 terlihat kondisi gelombang setelah dilakukan trasnformasi frekuensi fundamental pada 50 Hz. Pada frekuensi kelipatan 50 Hz tidak muncul harmonisa, menandakan beban ini tidak menimbulkan harmonisa tegangan.

Dari data yang telah didapat pada pengujian *Drill Press* pada kondisi aktif dan tidak aktif dapat disimpulkan untuk beban *Drill Press* tidak berpotensi membangkitkan *flicker* yang berpengaruh terhadap kondisi tegangan.

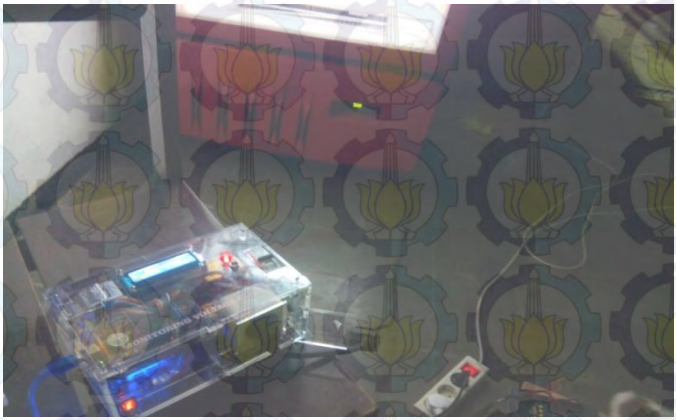
4.12.3 Pengujian Kulkas

Pada pegujian ini menggunakan beban kulkas yang biasanya banyak digunakan untuk mendinginkan minuman bersoda, dan spesifikasi dari kulkas sebagai berikut [15] :

Voltage : 220 Volt

Daya listrik : 223,71 Watt
Arus : 1,8 A
Kecepatan putar : 1400 r/min
Frekuensi : 50 Hz
Jumlah kutub : 4

Skema pengujian sesuai dengan Gambar 4.39. Pengujian ini dilakukan untuk mengetahui apakah kulkas atau lemari pendingin berpengaruh terhadap kondisi tegangan. Pengambilan data dilakukan pada saat kondisi kulkas baru menyala. Yang dibandingkan dengan kondisi beban normal. Pengujian beban kulkas ditunjukkan pada Gambar 4.50.



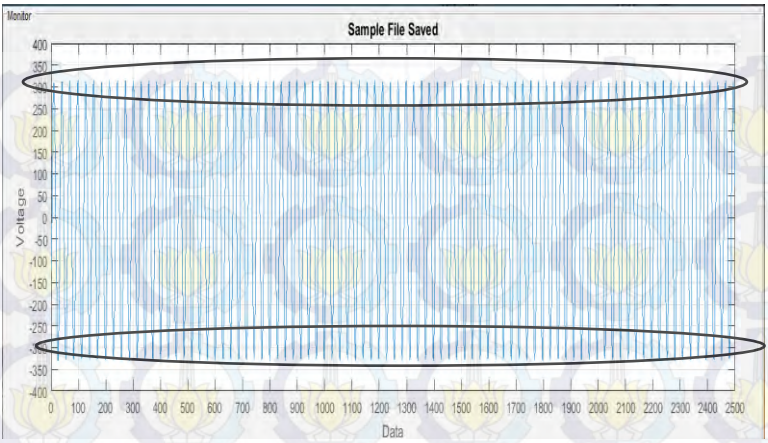
Gambar 4.50 Pengujian Kulkas

Setelah dilakukan pengujian, maka data yang tampil pada *interface* berupa hasil pengujian yang ditunjukkan pada Tabel 4.15

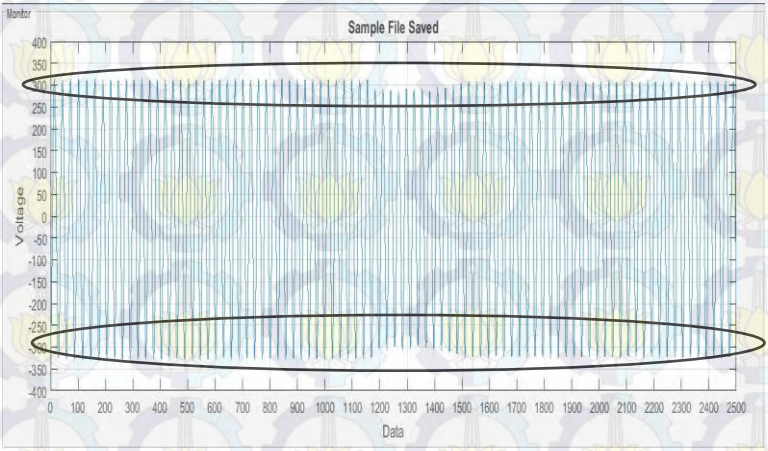
Tabel 4.15 Data Pengujian Beban Kulkas

No.	Kondisi	Vrms (V)	ΔV Positif (V)	ΔV Negatif (V)	THDV (%)
1.	Tanpa Beban	236,11	0,85	-0,74	1,24
2.	Kulkas	219,76	3,485	-4,47	3,35

Dari Tabel 4.15 terlihat kondisi THDV dari beban kulkas sebesar 3,35 ketika kulkas aktif. Kondisi tersebut diambil ketika kulkas baru menyala. Untuk data dari gelombang tegangan ditunjukkan pada Gambar 4.51 untuk kondisi normal dan Gambar 4.52 untuk kondisi kulkas mulai menyala.



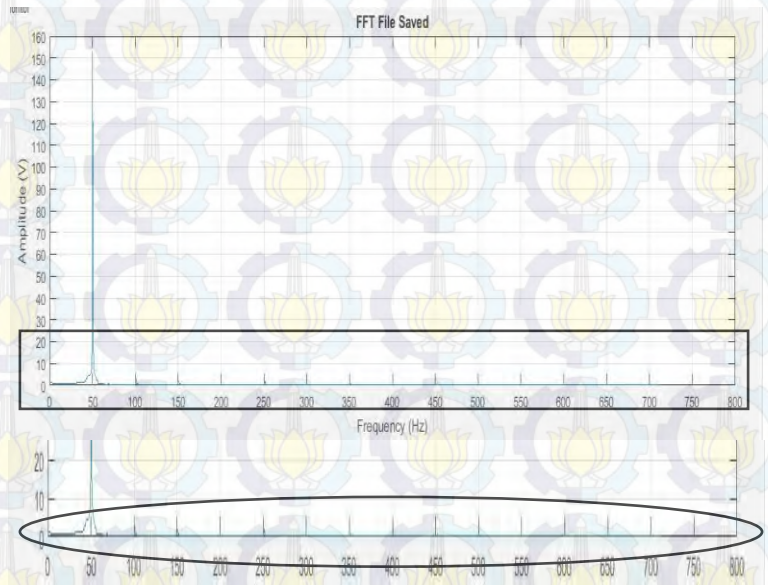
Gambar 4.51 Kondisi Kulkas Tidak Menyala



Gambar 4.52 Kondisi Kulkas Menyala

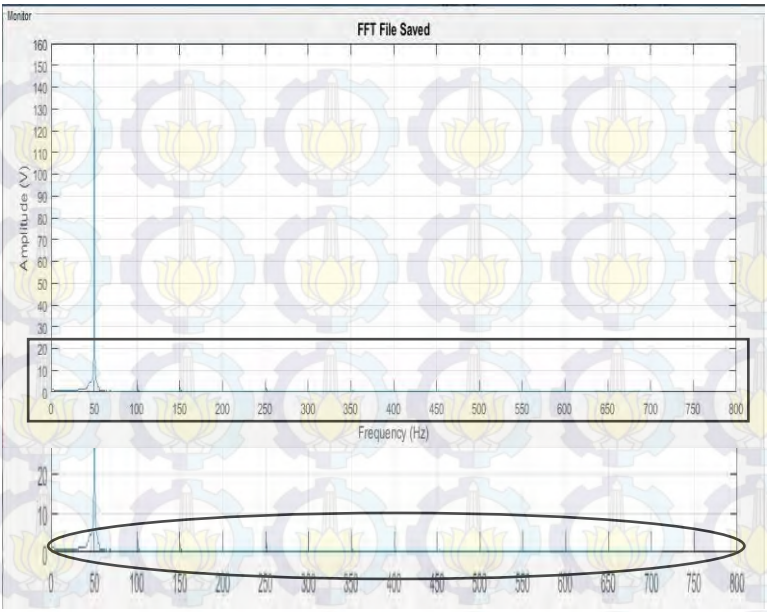
Dari kedua gambar diatas terlihat adanya perbedaan gelombang tegangan. Ketika kondisi kulkas mulai menyala akan terjadi *drop* tegangan tetapi tidak berlangsung lama. Kondisi tersebut terlihat pada Gambar 4.52 dimana adanya gelombang tegangan yang tidak rata sedikit turun. Hal tersebut dipengaruhi oleh *starting* awal kulkas ketika dari kondisi belum menyala dan kemudian kulkas mulai menyala terjadi proses *switching* beban yang berpengaruh terhadap kondisi tegangan.

Sedangkan untuk pengamatan yang lebih jelas megenai beban ini, langkah selanjutnya dilakukan transformasi data gelombang dengan metode FFT, yang hasilnya ditunjukkan pada Gambar 4.53



Gambar 4.53 Transformasi Kondisi Kulkas Tidak Menyala

Hasil transformasi pada Gambar 4.53 tidak adanya gangguan pada gelombang tegangan, yang ditandai dengan tidak adanya puncak selaian frekuensi fundamental yakni 50 Hz. Disekitar frekuensi 50 Hz tidak terbangkit riak yang menimbulkan adanya gelombang pada frekuensi lain, untuk kondisi diatas termasuk kondisi dimana keadaan tegangan normal. Untuk membandingkan kondisi ini dengan kulkas pada kondisi menyala maka dilakukan transformasi pada data gelombang ketika kondisi kulkas menyala yang ditunjukkan pada Gambar 4.54



Gambar 4.54 Transformasi Kondisi Kulkas Menyala

Kondisi grafik setelah dilakukan transformasi terlihat pada Gambar 4.54 dimana kondisi tersebut hampir sama dengan kondisi beban normal, hanya terdapat satu puncak yang menandakan frekuensi fundamental pada 50 Hz.

Setelah dilakukan pengambilan data pada kulkas tidak didapat adanya potensi *flicker*. Kondisi *starting* kulkas tidak berpengaruh terhadap grafik setelah dilakukan transformasi FFT. *Starting* kulkas belangsung sebentar dan tidak berulang pada waktu yang lain. Hal tersebut yang berpengaruh terhadap kondisi tegangan apabila kondisi pada Gambar 4.52 terjadi berulang.

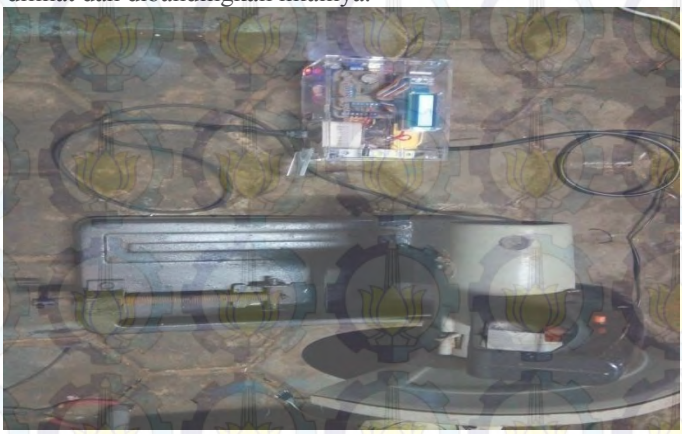
4.12.4 Pengujian Gerinda potong D28710

Pada pegujian ini menggunakan gerinda potong D28710 dengan spesifikasi sebagai berikut [16] :

Arus	: 15 Ampere
Kecepatan awal	: 3800 rpm
Daya	: 2200 Watt
Diameter roda	: 355 mm/14 inchi

Berat alat : 36,5 Lbs

Skema pengujian sesuai dengan Gambar 4.39. Pengujian ini dilakukan untuk mengetahui pengaruh gerinda potong D28710 terhadap kualitas tegangan terutama pada saat gerinda tersebut dalam keadaan aktif. Pengujian ditunjukkan pada Gambar 4.55 Nilai yang dibandingkan pada pengujian ini diantaranya : grafik dari tegangan ketika gerinda aktif dan ketika dalam keadaan normal, selain itu kondisi ΔV dan THDV juga akan dilihat dan dibandingkan nilainya.



Gambar 4.55 Pengujian Terhadap Gerinda Potong D28710

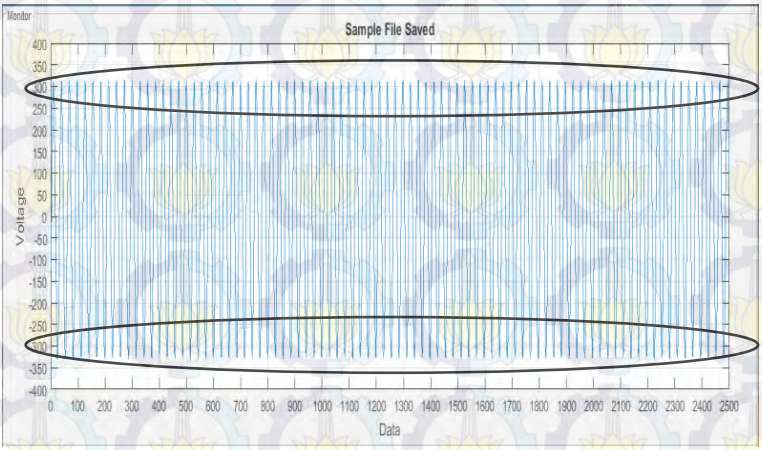
Setelah dilakukan pengujian, maka data yang tampil pada *interface* berupa hasil pengujian terhadap gerinda potong D28710 yang terdapat pada Tabel 4.16

Tabel 4.16 Pengujian Terhadap Gerinda Potong D28710

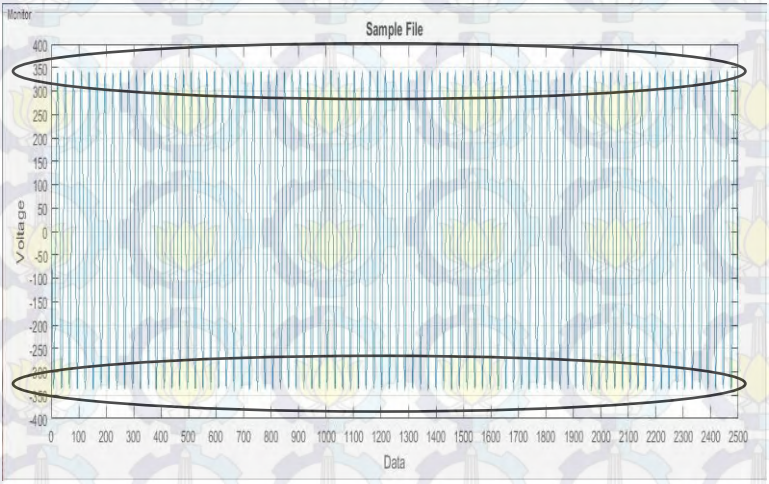
No.	Kondisi	Vrms (V)	ΔV Positif (V)	ΔV Negatif (V)	THDV (%)
1.	Tanpa Beban	234,878	1,618	-3,181	2,645
2.	Gerinda aktif	230,526	10,392	-4,028	3,08

Dari Tabel 4.16 terlihat perbedaan antara kondisi tidak berbeban (gerinda tidak aktif) dan ketika gerinda potong aktif. Dari kondisi Vrms yang turun meskipun tidak banyak ketika gerinda potong aktif, adanya pengaruh terhadap tegangan ketika gerinda potong aktif. Selain itu ΔV yang naik dengan meskipun perbedaan tidak signifikan, mengindikasikan bahwa kondisi tegangan yang tidak stabil, tidak seperti pada kondisi tanpa beban. Data yang diambil dengan *sampling* tegangan

sebanyak 2500 data *sampling*. Grafik gelombang dalam keadaan normal ditunjukkan pada Gambar 4.56 dan kondisi Gerinda aktif ditunjukkan pada Gambar 4.57



Gambar 4.56 Gelombang Ketika Tanpa Beban



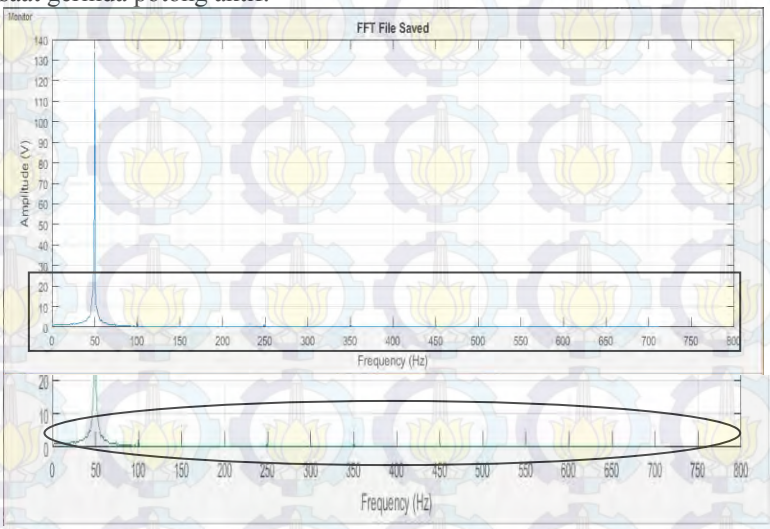
Gambar 4.57 Gelombang Ketika Gerinda Aktif

Dari kedua Gambar 4.56 dan Gambar 4.57 tidak begitu terlihat perubahan kondisi gelombang tegangan ketika gerinda potong diaktifkan. Hal tersebut dikarenakan gerinda potong memiliki arus yang

tidak begitu besar sehingga tidak terlalu menarik tegangan yang besar ketika diaktifkan. Pada Gambar 4.56 ketika kondisi tanpa beban terlihat gelombang yang stabil tanpa adanya perubahan naik atau turunnya tegangan. Hal tersebut yang menyebabkan ΔV yang tidak terlalu tinggi perubahannya.

Pada gelombang yang ditunjukkan pada Gambar 4.57 juga tidak begitu terlihat adanya perubahan naik dan turunnya tegangan secara tidak beraturan, meskipun terdapat perbedaan ΔV pada gelombang tanpa beban dan gelombang ketika gerinda aktif. Kondisi ΔV terjadi perubahaan meskipun tidak tinggi.dapat diartikan bahwa tidak terjadi perbedaan tegangan dalam waktu yang terus menerus selama gerinda potong diaktifkan.

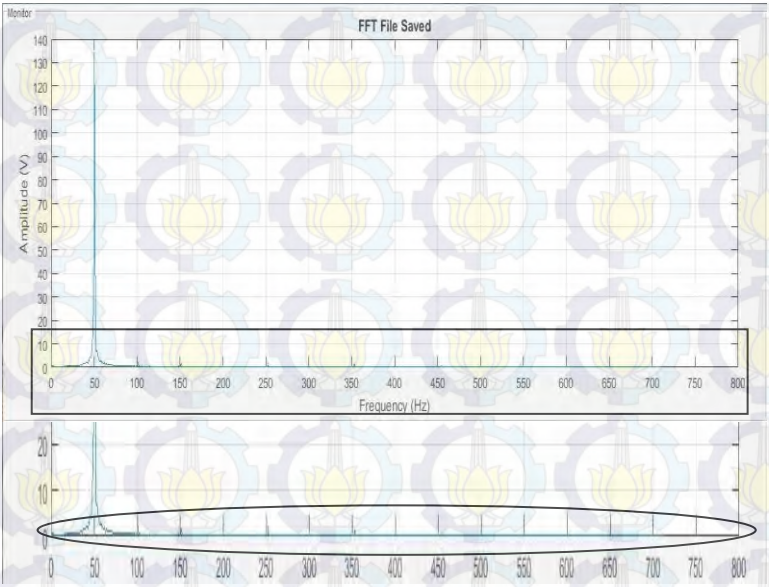
Selanjutnya akan dibandingkan grafik hasil transformasi dari domain waktu menjadi domain frekuensi menggunakan metode FFT. Gambar 4.58 adalah grafik hasil transformasi gelombang tegangan tanpa beban, pada Gambar 4.59 adalah grafik hasil transformasi gelombang saat gerinda potong aktif.



Gambar 4.58 Transformasi Gelombang Tanpa Beban

Dari Gambar 4.58 tidak terlihat adanya riak pada sekitar frekuensi 50 Hz. Dan pada frekuensi kelipantannya yaitu 100 Hz, 150 Hz, 200 Hz, tidak terlihat adanya lonjakan, berarti kondisi tegangan dalam keadaan

normal tidak ada harmonisa tegangan yang terjadi pada kondisi tanpa beban.



Gambar 4.59 Transformasi Kondisi Gerinda Potong Aktif

Pada Gambar 4.59 tidak terlihat adanya lonjakan pada frekuensi selain frekuensi fundamental yaitu pada 50 Hz. Hasil tranformasi kondisi gerinda potong aktif sekilas tidak jauh beda dengan kondisi normal. Hal itu dikarenakan arus beban gerinda potong yang tidak besar. Sehingga nilai THDV juga termasuk rendah yakni sebesar 3.08 % .

Setelah dilakukan pengujian terhadap gerinda potong D28710 didapatkan data-data diatas dan dapat disimpulkan, bahwa gerinda potong D28710 tidak berpotensi menimbulkan *flicker* dikarenakan arus beban yang kecil.

4.12.5 Pengujian Gerinda Tangan Maktec

Pada pegujian ini menggunakan gerinda tangan maktec dengan spesifikasi sebagai berikut [17] :

Kecepatan awal : 12000 rpm
Daya : 570 Watt
Diameter roda : 110 mm/ 14 inchi
Berat alat : 3,3 Lbs

Panjang alat : 256 mm

Skema pengujian sesuai dengan Gambar 4.39. Pengujian ini dilakukan untuk mengetahui pengaruh gerinda tangan maktec terhadap kualitas tegangan terutama pada saat gerinda tersebut dalam keadaan aktif. Pengujian ditunjukkan pada Gambar 4.60 Nilai yang dibandingkan pada pengujian ini diantaranya : grafik dari tegangan ketika gerinda aktif dan ketika dalam keadaan normal, selain itu kondisi ΔV dan THDV juga akan dilihat dan dibandingkan nilainya.

Gambar 4.60 Pengujian Terhadap Gerinda Tangan Maktec

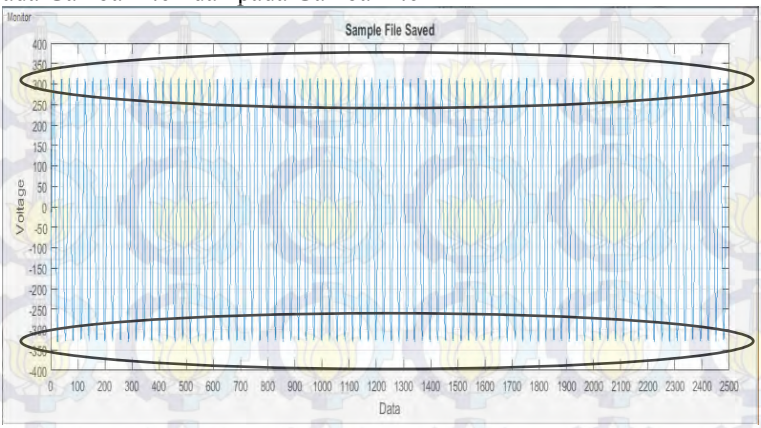
Setelah dilakukan pengujian, maka data yang tampil pada *interface* berupa hasil pengujian terhadap gerinda tangan maktec yang terdapat pada Tabel 4.17

Tabel 4.17 Pengujian Terhadap Gerinda Tangan Maktec

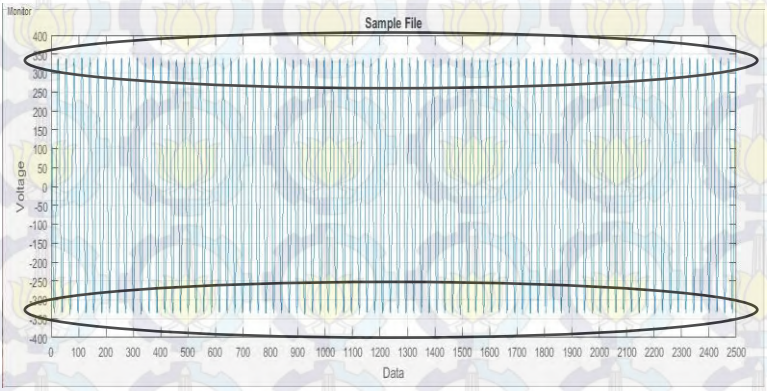
No.	Kondisi	Vrms (V)	ΔV Positif (V)	ΔV Negatif (V)	THDV (%)
1.	Tanpa Beban	230,539	0,8	-2,4	2,457
2.	Gerinda aktif	230,526	1,097	-3,702	2,668

Dari Tabel 4.17 terlihat perbedaan antara kondisi tidak berbeban (gerinda tidak aktif) dan ketika gerinda tangan maktec aktif. Dari kondisi Vrms yang turun meskipun tidak banyak, ketika gerinda tangan aktif. Selain itu ΔV yang naik, meskipun perbedaan tidak signifikan, mengindikasikan bahwa kondisi tegangan masih stabil, Data yang diambil dengan *sampling* tegangan sebanyak 2500. Grafik gelombang

dalam keadaan normal dan gerinda tangan maktec aktif ditunjukkan pada Gambar 4.61 dan pada Gambar 4.62



Gambar 4.61 Gelombang Ketika Tanpa Beban

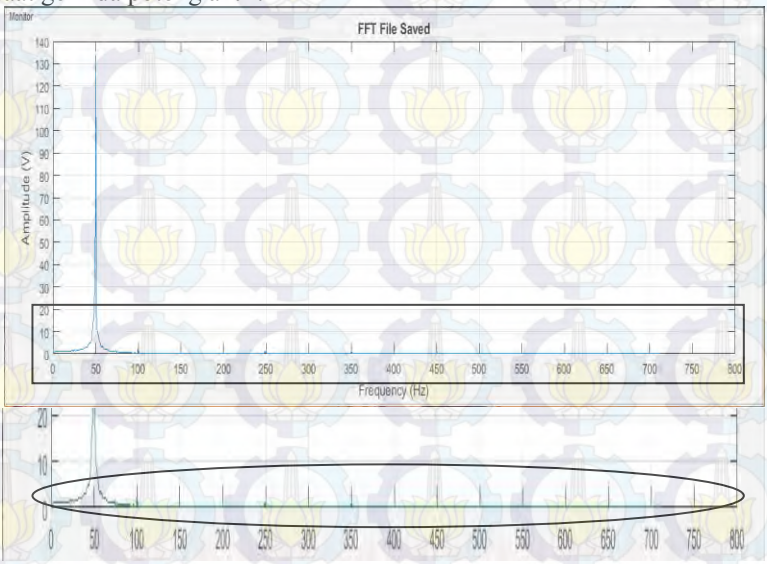


Gambar 4.62 Gelombang Ketika Gerinda Tangan Aktif

Dari kedua Gambar 4.61 dan Gambar 4.62 tidak begitu terlihat perubahan kondisi gelombang tegangan ketika gerinda tangan Maktec diaktifkan. Hal itu diakibatkan gerinda tangan memiliki arus yang kecil sehingga tidak terlalu menarik tegangan ketika diaktifkan. Pada Gambar 4.61 ketika kondisi tanpa beban terlihat gelombang yang stabil tanpa adanya perubahan naik atau turunnya tegangan. Hal tersebut yang menyebabkan ΔV yang tidak terlalu tinggi perubahannya.

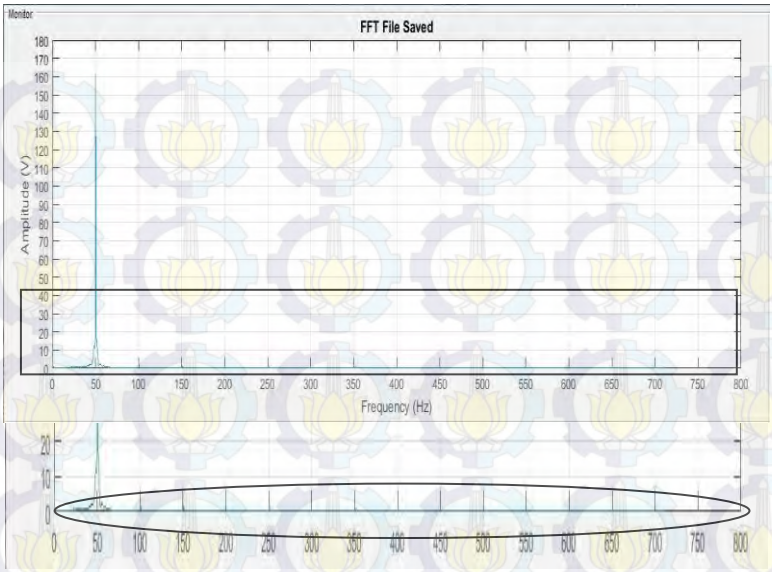
Pada gelombang yang ditunjukkan pada Gambar 4.62 juga tidak begitu terlihat adanya perubahan naik dan turunnya tegangan secara tidak beraturan, meskipun terdapat peningkatan ΔV pada gelombang ketika gerinda tangan aktif. Kondisi ΔV terjadi peningkatan meskipun tidak tinggi.

Selanjutnya akan dibandingkan grafik hasil transformasi dari domain waktu menjadi domain frekuensi menggunakan metode FFT. Gambar 4.63 adalah grafik hasil transformasi gelombang tegangan tanpa beban, pada Gambar 4.64 adalah grafik hasil transformasi gelombang saat gerinda potong aktif.



Gambar 4.63 Transformasi Gelombang Tanpa Beban

Dari Gambar 4.63 tidak terlihat adanya riak pada sekitar frekuensi 50 Hz. Dan tidak timbul harmonisa yang biasa terjadi pada frekuensi kelipatan 50 Hz. Hal ini karena tranformasi berasal dari gelombang murni jala jala tanpa beban.



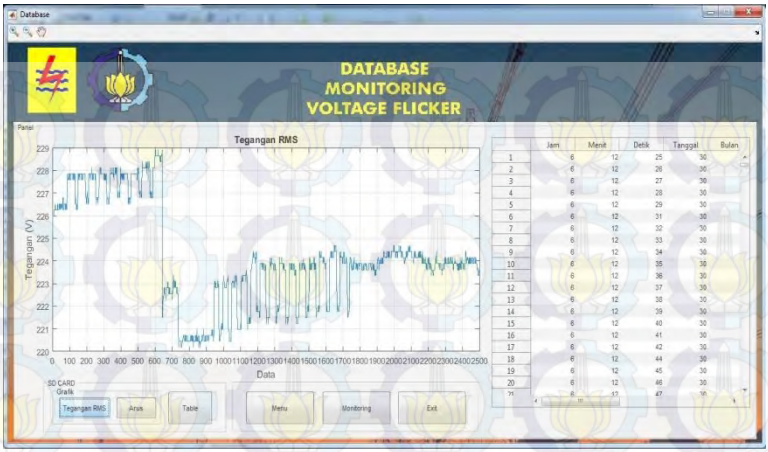
Gambar 4.64 Transformasi Kondisi Gerinda Tangan Aktif

Pada Gambar 4.64 tidak terlihat adanya lonjakan pada frekuensi selain frekuensi fundamental yaitu pada 50 Hz. Dan juga hasil tranformasi menunjukkan bahwa beban gerinda tangan Maktec tidak menimbulkan harmonisa. Hasil tranformasi kondisi gerinda potong aktif sekilas tidak jauh beda dengan kondisi normal. Hal itu dikarenakan arus beban gerinda potong yang tidak besar. Sehingga nilai THDV juga termasuk rendah yakni sebesar 2,6 % .

Setelah dilakukan pengujian terhadap gerinda tangan Maktec didapatkan data-data diatas dan dapat disimpulkan, bahwa gerinda tangan Maktec tidak termasuk beban yang menimbulkan *flicker* dikarenakan arus beban yang relatif kecil dan tidak menarik tegangan yang besar.

4.12.6 Pengujian *Database*

Pada pengujian *database* dilakukan untuk mengetahui tegangan dan arus dapat tersimpan dengan baik atau tidak. Pengujian dilakukan pata tanggal 30 April 2016, pada jam 6:12:25, pegambilan data dilakuka dengan jeda waktu tiap satu detik. Yang ditunjukkan pada Gambar 4.65



Gambar 4.65 Pengujian *Database*

Pada Gambar 4.65 terdapat grafik tegangan yang tersimpan pada *SD Card* yang ditampilkan pada *form database.* Selain itu nilai tegangan dan arus juga tampil dalam bentuk tabel pada sisi kanan *form database.*

Setelah dilakukan pengujian *database* dapat disimpulkan alat ini dapat menyimpan data arus dan tegangan serta menampilkannya pada *form database.*

4.13 Analisa Relevansi

Alat ini merupakan perancangan awal untuk *monitoring Voltage Flicker*. Apabila akan diimplementasikan pada kehidupan sehari-hari perlu adanya penyesuaian dengan beban yang akan diukur, karena pada alat ini kemampuan pengukuran hanya dibatasi pada beban dengan batas arus maksimal 16 A dan hanya dapat digunkan untuk beban 1 fasa.

Voltage Flicker banyak terjadi pada beban 3 fasa, untuk melakukan pengukuran pada beban 3 fasa maka perlu dilakukan penambahan komponen pada alat yang telah dibuat pada Tugas Akhir ini. Proses *monitoring* tidak berlangsung secara terus menerus, hanya dilakukan sebanyak 2500 data *sampling*, ketika diimplementasikan pada kehidupan sehari – hari, sebaiknya proses *sampling* dilakukan secara terus-menerus, sehingga data yang didapat lebih baik dari data yang hanya terbatas pada 2500 data.

Diharapkan dengan adnya alat *monitoring Voltage Flicker* maka dapat membantu PT PLN (Persero) dalam memberikan pelayanan yang

terbaik kepada konsumen. Serta dapat mencegah adanya kerusakan pada peralatan konsumen karena penggunaan peraltan yang dapat menimbulkan *Voltage Flicker* dapat dihindari ataupun diberi peralatan tambahan untuk mengurangi dampak dari *Voltage Flicker*.

BAB V
PENUTUP

Setelah melakukan perencanaan dan pembuatan alat, dan telah dilakukan pengujian serta analisa data maka dapat ditarik kesimpulan dan saran dari kegiatan tersebut

4.14 Kesimpulan

Berdasarkan hasil pengujian pada Tugas Akhir ini, maka diperoleh beberapa kesimpulan bahwa alat ini dapat digunakan untuk mendeteksi adanya *Voltage Flicker* dengan jumlah *sampling* sebanyak 2500 data, selain itu kondisi tegangan dan arus yang terukur saat tidak monitoring flicker dapat disimpan pada *SD Card* dengan jeda 1 detik. Hasil penyimpanan data ini sangat perlu untuk mengaudit kondisi tegangan dan juga arus.

Pada Tugas akhir ini terdapat komponen RTC yang pembacaannya memiliki kesalahan 0 %. RTC ini diperlukan untuk mengetahui waktu sekarang dan waktu saat penyimpanan data tegangan dan arus pada *SD Card*. Kesimpulan lain bahwa, dengan metode *Fast Fourier Transform* (FFT) perhitungan dapat dilakukan lebih cepat membutuhkan waktu 0,000577069 detik untuk 2500 data. Pada monitoring *flicker* juga dapat mendeteksi nilai dari ΔV serta THDV dari tegangan terukur untuk mengetahui beban yang terukur tersebut tergolong beban potensi *flicker* dan harmonisa atau tidak. Untuk pengambilan data percobaan pada Tugas Akhir ini, menggunakan 5 macam jenis beban yang diuji dan di dapatkan hasil bahwa beban Las Listrik Falcon 211GE berpotensi menimbulkan *Voltage Flicker*.

4.15 Saran

Berdasarkan proses dalam pembuatan alat ini ada beberapa hal yang perlu diperhatikan bahwa ground antara sensor dan ADC harus sama Karena apabila berbeda, nilai yang ditampilkan menjadi tidak akurat. Lalu karena waktu RTC tidak selalu sama dengan waktu setempat (WIB). Perlu dilakukan pengesetan dahulu agar RTC dapat digunakan dengan baik. Karena ruang penyimpanan terbatas, maka interval pengambilan data dapat diatur agar hemat pada ruang penyimpanan. Selanjutnya pada Tugas Akhir ini, sebaiknya pada kontroler lebih baik diberi sumber tegangan sendiri, karena pada beberapa kasus serial komunikasi terganggu akibat sumber mejadi satu.

-----Halaman ini sengaja dikosongkan-----

DAFTAR PUSTAKA

[1] Sukarelawanto, Ema, **Rasio Elektrifikasi Jawa Timur**, http://surabaya.bisnis.com/, 11 Februari 2015.

[2] Dugan, Roger C., **Electrical Power Systems Quality**, The McGraw-Hill Companies, 2004.

[3] Ruiz , J., Gutierrez, J. J., Lazkano, A., A Review of Flicker Severity Assessment by the IEC Flickermeter, **IEEE Transactions On Instrumentation And Measurement**, VOL. 59, NO. 8, Agustus 2010.

[4] Chang, G.W., Chen, Cheng-I, and Huang, Ya-Lun, A Digital Implementation of Flickermeter in the Hybrid Time and Frequency Domains, **IEEE Transactions On Power Delivery**, VOL. 24, NO. 3, July 2009.

[5] …., **Mutu Produk Menuju World Class Services 2015**,PT PLN (Persero) Distribusi Jawa Timur, Surabaya, 2013.

[6] Putra, R.M., Monitoring Harmonisa Arus Listrik Menggunakan Mikrokontroler, **Tugas Akhir**, Program D3 Teknik Elektro FTI-ITS, Surabaya, 2014.

[7] Syahwil, M., **Panduan Mudah Simulasi & Praktek Mikrokontroler Arduino,** ANDI, Yogyakarta, 2013.

[8] Kurniyanto, F.W. dan Madina, N.F., Purwarupa KWH Meter Digital yang dilengkapi Data Logger sebagai Alat Bantu melakukan Audit Energi, **Tugas Akhir**, Program D3 Teknik Elektro FTI-ITS, Surabaya, 2015.

[9] …., **ZMPT101b Voltage Transformer**, Datasheet, 2015.

[10] Pamungkas, T.S.P., Prototipe Prediksi KWH tak Terjual oleh PLN Berbasis Mikrokontroler menggunakan Wifi, **Tugas Akhir**, Program D3 Teknik Elektro FTI-ITS, Surabaya, 2014.

[11] Widiarsono, Teguh, **Tutorial Praktis Belajar Matlab**, Jakarta, 2005.

[12] …., **Tegangan - Tegangan Standart**, SPLN 1 : 1995, PT PLN (Persero), Jakarta, 1995.

[13] …., **Falcon 211GE**, http://www.perkakasku.com/, 11 Maret 2014

[14] ….., **Drill Press Models JDP-15M/MF**, Operating Instructions and Parts Manual, WALTER MEIER (Manufacturing) Inc., Tennessee, Mei 2013.

[15] ….., **Lemari ES LG SXS**, Buku Petunjuk Pemakaian, LG Electronics Inc., China, 2012.

[16] ….., **D28710 Heavy-Duty 14" (355mm) Chop Saw**, Operating Instructions and Parts Manual, DEWALT Industrial Tool, Baltimore, April 2011.

[17] ….., **Gerinda Cakram**, Petunjuk penggunaan, Makita Corporation, Japan, Januari 2003.

LAMPIRAN A

***Listing* Program Arduino**

```
#include <Wire.h>
#include "RTClib.h"
#include <LiquidCrystal.h>
#include <SD.h>
#include <SPI.h>
LiquidCrystal lcd(10,9,8,7,6,5);

RTC_DS1307 rtc;

char daysOfTheWeek[7][12] =
{"Minggu", "Senin", "Selasa", "Rabu", "Kamis", "Jumat",
"Sabtu"};

float vrms = A0;
int pin = A14;

void setup ()
{
 lcd.begin(16, 2);
 pinMode (pin, INPUT_PULLUP);
 Serial.begin(115200);
 lcd.print ("Voltage Flicker");
 pinMode(14,OUTPUT);
}

void loop ()
{
 while (digitalRead (pin) == LOW)
{
 float Vrms = analogRead(vrms)*1.625498187 - 849.342626;
 Serial.println(Vrms);
 }
  lcd.clear();
  tampil ();
  lcd.clear();
  lcd.setCursor(3,0);
```

```
  lcd.print ("Monitoring");
  lcd.setCursor(0,1);
  lcd.print ("Voltage  Flicker");
 }

void tampil()
{
  const int CS_PIN = 4;
  const int POW_PIN = 8;
  float voltage = A1;
  float current = A2;

//SD Card

  lcd.setCursor(2,0);
  lcd.print ("Read SD Card");
  delay (1000);
  lcd.clear ();

  pinMode(CS_PIN, OUTPUT);
  pinMode(POW_PIN, OUTPUT);
  digitalWrite(POW_PIN, HIGH);

  if (!SD.begin(CS_PIN))
   {
    lcd.clear ();
    delay (100);
   }
  else
   {
    lcd.print ("SC Card Ready...");
    delay (1000);
   }

//End of SD CARD

//RTC

 if (! rtc.begin())
```

```
 {
  lcd.print("Couldn't find RTC");
  while (1);
 }

 if (! rtc.isrunning())
  {
  lcd.print("RTC is NOT running!");
  rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
  rtc.adjust(DateTime(2016, 5, 29, 21, 53, 59));
  }

//End RTC

while (digitalRead (pin) == HIGH)
{
 DateTime now = rtc.now();
 float Volt = analogRead(voltage) * 0.259143488+ 17.42846949;
 Volt=Volt;
  if (Volt<190)
  {
  Volt=0;
  lcd.clear();
  }

  if (Volt<210 || Volt>230)
   {
    digitalWrite(14,HIGH);
    delay (500);
    digitalWrite(14,LOW);
    }

 float Amp = analogRead(current) * 0.023904802 + 0.162535228;
 Amp=Amp;
  if (Amp<0.17)
  {
   Amp=0;
   lcd.clear();
  }
```

```
lcd.clear();

//tampilan LCD atas

 lcd.setCursor(12,0);
 lcd.print (now.year(),DEC);
 lcd.setCursor(11,0);
 lcd.print ("/");
 lcd.setCursor(9,0);
 lcd.print(now.month(), DEC);
 lcd.setCursor (8,0);
 lcd.print("/");
 lcd.setCursor(6,0);
 lcd.print(now.day(), DEC);
 lcd.setCursor(0,0);
 lcd.print(now.hour(), DEC);
 lcd.setCursor(2,0);
 lcd.print(':');
 lcd.setCursor(3,0);
 lcd.print(now.minute(), DEC);

 //end tampilan LCD atas

 //tampilan LCD bawah

 lcd.setCursor(0,1);
 lcd.print(Volt);
 lcd.setCursor(7,1);
 lcd.print("V");
 lcd.setCursor(10,1);
 lcd.print(Amp);
 lcd.setCursor(15,1);
 lcd.print("A");

 //end tampilan LCD bawah

  delay(1000);

//end of lcd
```

```
//Penyimpanan data

  File dataFile = SD.open("log.csv", FILE_WRITE);
  if (dataFile)
   {
    dataFile.print(now.hour());
    dataFile.print(",");
    dataFile.print(now.minute());
    dataFile.print(",");
    dataFile.print(now.second());
    dataFile.print(",");
    dataFile.print(now.day());
    dataFile.print(",");
    dataFile.print(now.month());
    dataFile.print(",");
    dataFile.print(now.year());
    dataFile.print(",");
    dataFile.print(Volt);
    dataFile.print(",");
    dataFile.println(Amp);
    dataFile.close();
   }

//End Penyimpanan data

 }

}
```

***Listing* Program Matlab**

Form Login

```
function varargout = formlogin(varargin)
% FORMLOGIN MATLAB code for formlogin.fig
%      FORMLOGIN, by itself, creates a new FORMLOGIN or raises the existing
%      singleton*.
%
%      H = FORMLOGIN returns the handle to a new FORMLOGIN or the handle to
%      the existing singleton*.
%
%      FORMLOGIN('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in FORMLOGIN.M with the given input arguments.
%
%      FORMLOGIN('Property','Value',...) creates a new FORMLOGIN or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before formlogin_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to formlogin_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help formlogin

% Last Modified by GUIDE v2.5 01-Jan-2011 01:48:16
```

```
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @formlogin_OpeningFcn, ...
                   'gui_OutputFcn',  @formlogin_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before formlogin is made visible.
function formlogin_OpeningFcn(hObject, eventdata, handles,
varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to formlogin (see
VARARGIN)

% Choose default command line output for formlogin
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);
movegui(hObject, 'center');

guidata(hObject, handles);
```

```
hback = axes('units','normalized','position',[0 0 1 1]);
uistack(hback,'bottom');
% menampilkan background
[back map]=imread('login4.jpg');
image(back)
colormap(map)
% handlevisibility off agar axes tidak terlihat
% dan gambar background saja yang muncul.
set(hback,'handlevisibility','off','visible','off')



% UIWAIT makes formlogin wait for user response (see
UIRESUME)
% uiwait(handles.figure1);



% --- Outputs from this function are returned to the command line.
function varargout = formlogin_OutputFcn(hObject, eventdata,
handles)
% varargout  cell array for returning output args (see
VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;



function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
        % Hints: get(hObject,'String') returns contents of edit1 as text
        %        str2double(get(hObject,'String')) returns contents of edit1 as
a double


        % --- Executes during object creation, after setting all properties.
        function edit1_CreateFcn(hObject, eventdata, handles)
        % hObject    handle to edit1 (see GCBO)
        % eventdata  reserved - to be defined in a future version of
MATLAB
        % handles    empty - handles not created until after all CreateFcns
called

        % Hint: edit controls usually have a white background on
Windows.
        %       See ISPC and COMPUTER.
        if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
            set(hObject,'BackgroundColor','white');
        end



        function edit2_Callback(hObject, eventdata, handles)
        % hObject    handle to edit2 (see GCBO)
        % eventdata  reserved - to be defined in a future version of
MATLAB
        % handles    structure with handles and user data (see GUIDATA)

        % Hints: get(hObject,'String') returns contents of edit2 as text
        %        str2double(get(hObject,'String')) returns contents of edit2 as
a double


        % --- Executes during object creation, after setting all properties.
        function edit2_CreateFcn(hObject, eventdata, handles)
        % hObject    handle to edit2 (see GCBO)
        % eventdata  reserved - to be defined in a future version of
MATLAB
```

```
    % handles    empty - handles not created until after all CreateFcns
called

    % Hint: edit controls usually have a white background on
Windows.
    %       See ISPC and COMPUTER.
    if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end


    % --- Executes on button press in pushbutton1.
    function pushbutton1_Callback(hObject, eventdata, handles)
    % hObject    handle to pushbutton1 (see GCBO)
    % eventdata  reserved - to be defined in a future version of
MATLAB
    % handles    structure with handles and user data (see GUIDATA)
    UN = get(handles.edit1,'string');
    ID = get(handles.edit2,'userdata');
    if strcmp(UN,'admin') && strcmp(ID,'flicker')
       close(formlogin);
       (menu);
    else
       errordlg('USERNAME/PASSWORD SALAH');
    end
    myicon = imread('pln.jpg');
    h=msgbox('Anda Telah Berhasil
Masuk','Success','custom',myicon);

    % --- Executes on button press in pushbutton2.
    function pushbutton2_Callback(hObject, eventdata, handles)
    % hObject    handle to pushbutton2 (see GCBO)
    % eventdata  reserved - to be defined in a future version of
MATLAB
    % handles    structure with handles and user data (see GUIDATA)
    close
```

```
% --- Executes on key press with focus on edit2 and none of its
controls.
function edit2_KeyPressFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  structure with the following fields (see
UICONTROL)
%  Key: name of the key that was pressed, in lower case
%  Character: character interpretation of the key(s) that was
pressed
%  Modifier: name(s) of the modifier key(s) (i.e., control, shift)
pressed
% handles    structure with handles and user data (see GUIDATA)
password = get(handles.edit2,'userdata');
key = get(handles.output,'currentkey');
switch key
   case 'backspace'
      password = password(1:end-1);
      SizePass = size(password);
      if SizePass(2) > 0
         asterisk(1,1:SizePass(2)) = '•';
         set(handles.edit2,'String',asterisk)
      else
         set(handles.edit2,'String','')
      end
      set(handles.edit2,'Userdata',password) %
   case 'escape'
   case 'insert'
   case 'delete'
   case 'home'
   case 'pageup'
   case 'pagedown'
   case 'end'
   case 'rightarrow'
   case 'downarrow'
   case 'leftarrow'
   case 'uparrow'
   case 'shift'
   case 'return'
   case 'alt'
```

```
        case 'control'
        case 'windows'
        otherwise
          password = [password get(handles.output,'currentcharacter')];
          SizePass = size(password);
          if SizePass(2) > 0
            asterisk(1:SizePass(2)) = '*';
            set(handles.edit2,'string',asterisk)
          else
            set(handles.edit2,'String','');
          end
          set(handles.edit2,'Userdata',password) ;
    end


    % --- Executes on button press in checkbox1.
    function checkbox1_Callback(hObject, eventdata, handles)
    % hObject    handle to checkbox1 (see GCBO)
    % eventdata  reserved - to be defined in a future version of
MATLAB
    % handles    structure with handles and user data (see GUIDATA)

    % Hint: get(hObject,'Value') returns toggle state of checkbox1
```

***Form* Menu**

```
function varargout = menu(varargin)
% MENU MATLAB code for menu.fig
%      MENU, by itself, creates a new MENU or raises the existing
%      singleton*.
%
%      H = MENU returns the handle to a new MENU or the handle
to
%      the existing singleton*.
%
%      MENU('CALLBACK',hObject,eventData,handles,...) calls
the local
%      function named CALLBACK in MENU.M with the given
input arguments.
%
%      MENU('Property','Value',...) creates a new MENU or raises
the
%      existing singleton*.  Starting from the left, property value
pairs are
%      applied to the GUI before menu_OpeningFcn gets called.
An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to menu_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI
allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help menu

% Last Modified by GUIDE v2.5 08-May-2016 01:26:41

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
```

```
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @menu_OpeningFcn, ...
                   'gui_OutputFcn',  @menu_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
    if nargin && ischar(varargin{1})
       gui_State.gui_Callback = str2func(varargin{1});
    end

    if nargout
       [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
    else
       gui_mainfcn(gui_State, varargin{:});
    end
    % End initialization code - DO NOT EDIT


    % --- Executes just before menu is made visible.
    function menu_OpeningFcn(hObject, eventdata, handles, varargin)
    % This function has no output args, see OutputFcn.
    % hObject    handle to figure
    % eventdata  reserved - to be defined in a future version of
MATLAB
    % handles    structure with handles and user data (see GUIDATA)
    % varargin   command line arguments to menu (see VARARGIN)

    % Choose default command line output for menu
    handles.output = hObject;

    % Update handles structure


    guidata(hObject, handles);
    hback = axes('units','normalized','position',[0 0 1 1]);
    uistack(hback,'bottom');
    % menampilkan background
    [back map]=imread('MENU1.jpg');
    image(back)
    colormap(map)
```

```
% handlevisibility off agar axes tidak terlihat
% dan gambar background saja yang muncul.
set(hback,'handlevisibility','off','visible','off')
% UIWAIT makes menu wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = menu_OutputFcn(hObject, eventdata,
handles)
% varargout  cell array for returning output args (see
VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in monitoring.
function monitoring_Callback(hObject, eventdata, handles)
% hObject    handle to monitoring (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
close(menu);
   (flickerFFT2);

% --- Executes on button press in database.
function database_Callback(hObject, eventdata, handles)
% hObject    handle to database (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
close(menu);
   (Database);
```

```
% --- Executes on button press in exit.
function exit_Callback(hObject, eventdata, handles)
% hObject    handle to exit (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
close(menu);
```

Form Monitoring

```
function varargout = flickerFFT2(varargin)
% FLICKERFFT2 MATLAB code for flickerFFT2.fig
%      FLICKERFFT2, by itself, creates a new FLICKERFFT2 or raises the existing
%      singleton*.
%
%      H = FLICKERFFT2 returns the handle to a new FLICKERFFT2 or the handle to
%      the existing singleton*.
%
%      FLICKERFFT2('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in FLICKERFFT2.M with the given input arguments.
%
%      FLICKERFFT2('Property','Value',...) creates a new FLICKERFFT2 or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before flickerFFT2_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to flickerFFT2_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help flickerFFT2

% Last Modified by GUIDE v2.5 01-Jan-2011 00:22:14

% Begin initialization code - DO NOT EDIT
```

```
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @flickerFFT2_OpeningFcn, ...
                   'gui_OutputFcn',  @flickerFFT2_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before flickerFFT2 is made visible.
function flickerFFT2_OpeningFcn(hObject, eventdata, handles,
varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to flickerFFT2 (see
VARARGIN)

% Choose default command line output for flickerFFT2
handles.output = hObject;

% Update handles structure
% IKI LOH PROGRAM E
guidata(hObject, handles);
hback = axes('units','normalized','position',[0 0 1 1]);
uistack(hback,'bottom');
% menampilkan background
```

```
[back map]=imread('monitoring1.jpg');
image(back)
colormap(map)
% handlevisibility off agar axes tidak terlihat
% dan gambar background saja yang muncul.
set(hback,'handlevisibility','off','visible','off')

% UIWAIT makes gambar wait for user response (see
UIRESUME)
% uiwait(handles.figure1);

% UIWAIT makes flickerFFT2 wait for user response (see
UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = flickerFFT2_OutputFcn(hObject, eventdata,
handles)
% varargout  cell array for returning output args (see
VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --------------------------------------------------------------------
function sampling1_Callback(hObject, eventdata, handles)
% hObject    handle to sampling1 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

port = get(handles.port, 'String');
baudrate = str2num(get(handles.baudrate,'String'));
com = serial(port);
```

```
set(com, 'InputBufferSize', 256);
set(com, 'BaudRate', baudrate);
set(com, 'DataBits', 8);
set(com, 'Parity', 'none');
set(com, 'StopBit', 1);
set(com, 'FlowControl', 'hardware');
set(com, 'Timeout', 10);

%sampling data from ardino
fopen(com); % opening the port
t=1;
% disp('start')
% clear global sample;
clear global Vrms;
clear q;

while(t<2501)
    data = fscanf(com, '%f'); %storing the received data to variable
"data"
    if(~isempty(data) && isfloat(data))
        sample(t) = data;
        t=t+1;
    end
end

handles.axes6;
plot(sample);
set(gca,'xtick', -1000:100:25000);
set(gca,'ytick', -1000:50:1000);
grid on;
title('AC Signal');
xlabel ('Data');
ylabel ('Voltage');
myicon = imread('pln.jpg');
h=msgbox('Sampling Completed','Success','custom',myicon);

%mencari VRMS
q = sample.*sample;
Vrms = sqrt (mean(mean(q)));
```

```
        Vrms = num2str (Vrms);
        set (handles.vrms, 'string', Vrms);
        set (handles.deltavpos, 'string', '-');
        set (handles.deltavneg, 'string', '-')
        set (handles.thd, 'string','-');


        fclose(com); %closing the port
        % clear com ; %clear the variable s memory, just for refreshment
of your PC
        global sample;
        global Vrms;


        % --------------------------------------------------------------------
        function fft1_Callback(hObject, eventdata, handles)
        % hObject    handle to fft1 (see GCBO)
        % eventdata  reserved - to be defined in a future version of
MATLAB
        % handles    structure with handles and user data (see GUIDATA)
        %% Mendapatkan Data
        % v=xlsread('serial.xls');
        %load1 Matlab.mat
        %% Frekuensi Sampling
        global sample;
        global Vrms;
        global THDv;
        global freq;
        global DeltaVPos;
        global DeltaVNeg;
        global magx10a;
        %clear THDv;
        clear DeltaV;

        %frekuensi sampling
        fs =1425;
        nfft =2500;
        freq = (0:nfft/2-1)*fs/nfft;
```

```
%%mencari FFT
%% LOADFFT1
xfft10 = fft(sample,nfft);
x10 = xfft10(1:nfft/2);
magx10a = abs(x10)/nfft;
% ampspec10a = 20*log10(magx10a);

handles.axes6;
plot (freq,magx10a)
set(gca,'xtick', -1000:50:25000);
ylabel('Amplitude (V)');
xlabel('Frequency (Hz)');
title('Fast Fourier Transform');
grid on;
myicon = imread('pln.jpg');
h=msgbox('FFT Transformation
Completed','Success','custom',myicon);

%%mencari DeltaV
for blok=1:86
   nilaimax (blok) = 0;
   for n=(29*blok)-27 : (29*blok)+1
      Var = sample(n);
      if ( Var > nilaimax (blok))
         nilaimax (blok) = Var;
      end
   end
end

Vprata2 = mean(nilaimax);
Vpmax = max (nilaimax);
Vpmin = min (nilaimax);

DeltaVPos = Vpmax - Vprata2;
DeltaVNeg = Vpmin - Vprata2;


%% mengambil nilai THDV
for blok=1:14
```

```
        nilaimax (blok) = 0;
        for n=(87*blok)-4 : (87*blok)+6
            z = magx10a(n);
            if ( z > nilaimax (blok))
                nilaimax (blok) = z;
            end
        end
    end

    AmplitudoHarmonisa = nilaimax (2:14);
    AmplitudoHarmonisa =
AmplitudoHarmonisa.*AmplitudoHarmonisa;
    JumlahAmpHarmonisa = sum(AmplitudoHarmonisa);
    THDv =
((sqrt(JumlahAmpHarmonisa)/nilaimax(1))*100)*0.47173;
    THDv = num2str (THDv);

    DeltaVPos = num2str (DeltaVPos);
    DeltaVNeg = num2str (DeltaVNeg);

    set (handles.thd, 'string',THDv);
    set (handles.deltavpos, 'string', DeltaVPos);
    set (handles.deltavneg, 'string', DeltaVNeg);

    global THDv;
    global DeltaV;


    % --------------------------------------------------------------------
    function save1_Callback(hObject, eventdata, handles)
    % hObject    handle to save1 (see GCBO)
    % eventdata  reserved - to be defined in a future version of
MATLAB
    % handles    structure with handles and user data (see GUIDATA)

    global Vrms;
    global THDv;
    global DeltaVPos;
    global DeltaVNeg;
```

```
global sample;
global freq;
global magx10a;


VoltageRMS = str2num (Vrms);
Harmonisa = str2num (THDv);
deltavpos = str2num (DeltaVPos);
deltavneg = str2num (DeltaVNeg);
jam = clock;

data(1) = VoltageRMS;
data(2) = deltavpos;
data(3) = deltavneg;
data(4) = Harmonisa;

pengulanga = 5;
pengulangjam = 1;
while (pengulanga  <= 10)
   data(pengulanga)=jam (pengulangjam); %datafile ke 4 - 9 =
data jam 1-6
   pengulanga=pengulanga+1;
   pengulangjam=pengulangjam+1;
end
pengulangsample=1;
while (pengulanga <= (2510))
   data(pengulanga) = sample(pengulangsample); %datafile ke 10 -
2509 = data sampling (hasil sampling)(2500 data)
   pengulanga=pengulanga+1;
   pengulangsample=pengulangsample+1;
end
pengulangfreq=1;
while (pengulanga<= (3760)) %datafile ke 2510 - 3759 = data
frequency (1250 data)
   data(pengulanga) = freq(pengulangfreq);
   pengulanga=pengulanga+1;
   pengulangfreq=pengulangfreq+1;
end
pengulangmagx10a=1;
```

```
        while (pengulanga <= (5010)) %datafile ke 3760 - 5009 = data Mx
(1250 data)
            data(pengulanga) = magx10a(pengulangmagx10a);
            pengulanga=pengulanga+1;
            pengulangmagx10a=pengulangmagx10a+1;
        end

        handles.hdata = data;

        [filename,pathname] = uiputfile ('*.mat','Save Workspace As');

        if isequal ([filename,pathname],[0,0])
            return
        else
            File=fullfile(pathname,filename);
            save(File,'data')
            guidata(hObject,handles);
        end
        myicon = imread('pln.jpg');
        h=msgbox('Save File Completed','Success','custom',myicon);

        global magx10afile;
        global freqfile;
        plot(freqfile,magx10afile);
        grid on;
        set(gca,'xtick', -1000:50:25000);
        set(gca,'ytick', -1000:10:25000);
        ylabel('Amplitude (V)');
        xlabel('Frequency (Hz)');
        title('FFT File Saved');
        myicon = imread('pln.jpg');
        h=msgbox('Load FFT File Completed','Success','custom',myicon);

        function loadfile1_Callback(hObject, eventdata, handles)
        % hObject    handle to loadfile1 (see GCBO)
        % eventdata  reserved - to be defined in a future version of
MATLAB
        % handles    structure with handles and user data (see GUIDATA)
        %clear samplefile;
```

```
%clear freqfile;
%clear magx10afile;

global samplefile;
global freqfile;
global magx10afile;
global aa;

[filename,pathname]=uigetfile('*.mat');
if isequal([filename,pathname],[0,0])
   return
else
   File =fullfile(pathname,filename);
   load(File,'data')
   guidata(hObject,handles);
end
set(handles.Vrmsfile,'String',data(1));
set (handles.deltavfilepos, 'string', data(2));
set (handles.deltavfileneg, 'string', data(3));
set (handles.thdfile, 'string', data(4));

samplefile = data (11:2510);
freqfile = data (2511:3760);
magx10afile = data (3761:5010);
title(filename);
aa = filename;
aa = num2str (aa);
set (handles.nameload, 'string', aa);


% --- Executes on button press in loadsample1.
function loadsample1_Callback(hObject, eventdata, handles)
% hObject    handle to loadsample1 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
global samplefile;
global aa;
plot(samplefile);
```

```
grid on;
 set(gca,'xtick', -1000:100:2500);
 set(gca,'ytick', -1000:50  :2500);
 ylabel('Voltage');
xlabel('Data');
title('Sample File');
myicon = imread('pln.jpg');
h=msgbox('Load Sample File
Completed','Success','custom',myicon);


% --- Executes on button press in loadfft1.
function loadfft1_Callback(hObject, eventdata, handles)
% hObject    handle to loadfft1 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
global magx10afile;
global freqfile;
plot(freqfile,magx10afile);
grid on;
set(gca,'xtick', -1000:50:25000);
set(gca,'ytick', -1000:10:25000);
ylabel('Amplitude (V)');
xlabel('Frequency (Hz)');
title('FFT File Saved');
myicon = imread('pln.jpg');
h=msgbox('Load FFT File Completed','Success','custom',myicon);


% --------------------------------------------------------------------
function load sampling _Callback(hObject, eventdata, handles)
% hObject    handle to load sampling (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
global samplefile;
plot(samplefile);
grid on;
```

```
        set(gca,'xtick', -1000:100:2500);
        set(gca,'ytick', -1000:50  :2500);
        ylabel('Voltage');
       xlabel('Data');
       title('Sample File');
       myicon = imread('pln.jpg');
       h=msgbox('Load Sample File
Completed','Success','custom',myicon);


       % --------------------------------------------------------------------
       function loadingfft_Callback(hObject, eventdata, handles)
       % hObject    handle to loadingfft (see GCBO)
       % eventdata  reserved - to be defined in a future version of
MATLAB
       % handles    structure with handles and user data (see GUIDATA)
       global magx10afile;
       global freqfile;
       plot(freqfile,magx10afile);
       grid on;
       set(gca,'xtick', -1000:50:25000);
       set(gca,'ytick', -1000:10:25000);
       ylabel('Amplitude (V)');
       xlabel('Frequency (Hz)');
       title('FFT File Saved');
       myicon = imread('pln.jpg');
       h=msgbox('Load FFT File Completed','Success','custom',myicon);


       % --- Executes on button press in compare.
       function compare_Callback(hObject, eventdata, handles)
       % hObject    handle to compare (see GCBO)
       % eventdata  reserved - to be defined in a future version of
MATLAB
       % handles    structure with handles and user data (see GUIDATA)
       global sample;
       global freq;

       fs =1425;
```

```
nfft =2500;
freq = (0:nfft/2-1)*fs/nfft;


%% LOADFFT1
tic;
xfft10 = fft(sample,nfft);
x10 = xfft10(1:nfft/2);
magx10a = abs(x10)/nfft;
% ampspec10a = 20*log10(magx10a);
timefft=toc;
set(handles.waktufft, 'string', timefft);

tic;
for k=1:1250
  a=0;b=0;
  for j=1:2500
    Xr(k)=a+sample(j)*cos(2*3.14*k*j/2500);
    Xi(k)=b-sample(j)*sin(2*3.14*k*j/2500);
    a=Xr(k); b=Xi(k);
  end;
    Mx(k)=sqrt(((Xr(k))^2)+((Xi(k))^2))/2500;
end;
timedft=toc;
set(handles.waktudft, 'string', timedft);


% --------------------------------------------------------------------
function axesjpg_Callback(hObject, eventdata, handles)
% hObject    handle to axesjpg (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
F = getframe(handles.figure1);
Image = frame2im(F);
imwrite(Image, 'flicker.jpg')


% --------------------------------------------------------------------
```

```
function txt_Callback(hObject, eventdata, handles)
% hObject    handle to txt (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
global sample;
save('SAMPLE.txt','sample','-ascii')


% --------------------------------------------------------------------
function menu_Callback(hObject, eventdata, handles)
% hObject    handle to menu (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
close(flickerFFT2);
   (menu);


% --------------------------------------------------------------------
function exit1_Callback(hObject, eventdata, handles)
% hObject    handle to exit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
close(flickerFFT2);

function baudrate_Callback(hObject, eventdata, handles)
% hObject    handle to baudrate (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of baudrate as text
%        str2double(get(hObject,'String')) returns contents of
baudrate as a double


% --- Executes during object creation, after setting all properties.
```

```
function baudrate_CreateFcn(hObject, eventdata, handles)
% hObject    handle to baudrate (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on
Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function port_Callback(hObject, eventdata, handles)
% hObject    handle to port (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of port as text
%        str2double(get(hObject,'String')) returns contents of port as
a double


% --- Executes during object creation, after setting all properties.
function port_CreateFcn(hObject, eventdata, handles)
% hObject    handle to port (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on
Windows.
%       See ISPC and COMPUTER.
```

```
    if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end


    function vrms_Callback(hObject, eventdata, handles)
    % hObject    handle to vrms (see GCBO)
    % eventdata  reserved - to be defined in a future version of
MATLAB
    % handles    structure with handles and user data (see GUIDATA)

    % Hints: get(hObject,'String') returns contents of vrms as text
    %        str2double(get(hObject,'String')) returns contents of vrms as
a double


    % --- Executes during object creation, after setting all properties.
    function vrms_CreateFcn(hObject, eventdata, handles)
    % hObject    handle to vrms (see GCBO)
    % eventdata  reserved - to be defined in a future version of
MATLAB
    % handles    empty - handles not created until after all CreateFcns
called

    % Hint: edit controls usually have a white background on
Windows.
    %       See ISPC and COMPUTER.
    if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end


    function deltavpos_Callback(hObject, eventdata, handles)
    % hObject    handle to deltavpos (see GCBO)
    % eventdata  reserved - to be defined in a future version of
MATLAB
    % handles    structure with handles and user data (see GUIDATA)
```

```
% Hints: get(hObject,'String') returns contents of deltavpos as text
%        str2double(get(hObject,'String')) returns contents of
deltavpos as a double


% --- Executes during object creation, after setting all properties.
function deltavpos_CreateFcn(hObject, eventdata, handles)
% hObject    handle to deltavpos (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on
Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function thd_Callback(hObject, eventdata, handles)
% hObject    handle to thd (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of thd as text
%        str2double(get(hObject,'String')) returns contents of thd as a
double


% --- Executes during object creation, after setting all properties.
function thd_CreateFcn(hObject, eventdata, handles)
% hObject    handle to thd (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
```

```
    % handles    empty - handles not created until after all CreateFcns
called

    % Hint: edit controls usually have a white background on
Windows.
    %       See ISPC and COMPUTER.
    if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end


    % --- Executes on button press in ac.
    function ac_Callback(hObject, eventdata, handles)
    % hObject    handle to ac (see GCBO)
    % eventdata  reserved - to be defined in a future version of
MATLAB
    % handles    structure with handles and user data (see GUIDATA)


    % --------------------------------------------------------------------
    function home_Callback(hObject, eventdata, handles)
    % hObject    handle to home (see GCBO)
    % eventdata  reserved - to be defined in a future version of
MATLAB
    % handles    structure with handles and user data (see GUIDATA)


    % --------------------------------------------------------------------
    function data_Callback(hObject, eventdata, handles)
    % hObject    handle to data (see GCBO)
    % eventdata  reserved - to be defined in a future version of
MATLAB
    % handles    structure with handles and user data (see GUIDATA)


    % --------------------------------------------------------------------
    function open1_Callback(hObject, eventdata, handles)
    % hObject    handle to open1 (see GCBO)
```

```
    % eventdata  reserved - to be defined in a future version of
MATLAB
    % handles    structure with handles and user data (see GUIDATA)


    % --------------------------------------------------------------------
    function exit_Callback(hObject, eventdata, handles)
    % hObject    handle to exit (see GCBO)
    % eventdata  reserved - to be defined in a future version of
MATLAB
    % handles    structure with handles and user data (see GUIDATA)


    % --------------------------------------------------------------------
    function help_Callback(hObject, eventdata, handles)
    % hObject    handle to help (see GCBO)
    % eventdata  reserved - to be defined in a future version of
MATLAB
    % handles    structure with handles and user data (see GUIDATA)


    % --------------------------------------------------------------------
    function about_Callback(hObject, eventdata, handles)
    % hObject    handle to about (see GCBO)
    % eventdata  reserved - to be defined in a future version of
MATLAB
    % handles    structure with handles and user data (see GUIDATA)


    % --------------------------------------------------------------------
    function saveas_Callback(hObject, eventdata, handles)
    % hObject    handle to saveas (see GCBO)
    % eventdata  reserved - to be defined in a future version of
MATLAB
    % handles    structure with handles and user data (see GUIDATA)


    % --------------------------------------------------------------------
    function load1_Callback(hObject, eventdata, handles)
```

```
    % hObject    handle to load1 (see GCBO)
    % eventdata  reserved - to be defined in a future version of
MATLAB
    % handles    structure with handles and user data (see GUIDATA)


    % --- Executes on button press in Vrmsfile.
    function Vrmsfile_Callback(hObject, eventdata, handles)
    % hObject    handle to Vrmsfile (see GCBO)
    % eventdata  reserved - to be defined in a future version of
MATLAB
    % handles    structure with handles and user data (see GUIDATA)


    % --- Executes on button press in deltavfileneg.
    function deltavfileneg_Callback(hObject, eventdata, handles)
    % hObject    handle to deltavfileneg (see GCBO)
    % eventdata  reserved - to be defined in a future version of
MATLAB
    % handles    structure with handles and user data (see GUIDATA)


    % --- Executes on button press in thdfile.
    function thdfile_Callback(hObject, eventdata, handles)
    % hObject    handle to thdfile (see GCBO)
    % eventdata  reserved - to be defined in a future version of
MATLAB
    % handles    structure with handles and user data (see GUIDATA)


    % --------------------------------------------------------------------
    function acsignal_Callback(hObject, eventdata, handles)
    % hObject    handle to acsignal (see GCBO)
    % eventdata  reserved - to be defined in a future version of
MATLAB
    % handles    structure with handles and user data (see GUIDATA)
    % global sample;
    %
    % interv = 1000;
```

```
        % passo = 1;
        % t=1;
        % x=0;
        % while(t<interv)
        % b=sample.analogRead(1);
        % x=[x,b];
        % plot(x);
        % axis([0,interv,0,1024]);
        % grid on;
        % t=t+passo;
        % end

        %global sample;
        %global THDv;
        %global deltavpos;
        %handles.axes6;
        %plot(sample);
        %title('Sine Wave');
        %xlabel ('Data');
        %ylabel ('Voltage (Volt)');
        %grid on;


        function deltavneg_Callback(hObject, eventdata, handles)
        % hObject    handle to deltavneg (see GCBO)
        % eventdata  reserved - to be defined in a future version of
MATLAB
        % handles    structure with handles and user data (see GUIDATA)

        % Hints: get(hObject,'String') returns contents of deltavneg as text
        %        str2double(get(hObject,'String')) returns contents of
deltavneg as a double


        % --- Executes during object creation, after setting all properties.
        function deltavneg_CreateFcn(hObject, eventdata, handles)
        % hObject    handle to deltavneg (see GCBO)
        % eventdata  reserved - to be defined in a future version of
MATLAB
```

```
    % handles    empty - handles not created until after all CreateFcns
called

    % Hint: edit controls usually have a white background on
Windows.
    %       See ISPC and COMPUTER.
    if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end


    % --- Executes on button press in deltavfilepos.
    function deltavfilepos_Callback(hObject, eventdata, handles)
    % hObject    handle to deltavfilepos (see GCBO)
    % eventdata  reserved - to be defined in a future version of
MATLAB
    % handles    structure with handles and user data (see GUIDATA)


    % --- Executes on button press in waktufft.
    function waktufft_Callback(hObject, eventdata, handles)
    % hObject    handle to waktufft (see GCBO)
    % eventdata  reserved - to be defined in a future version of
MATLAB
    % handles    structure with handles and user data (see GUIDATA)


    % --- Executes on button press in waktudft.
    function waktudft_Callback(hObject, eventdata, handles)
    % hObject    handle to waktudft (see GCBO)
    % eventdata  reserved - to be defined in a future version of
MATLAB
    % handles    structure with handles and user data (see GUIDATA)


    % --- Executes on button press in comparegrafik.
    function comparegrafik_Callback(hObject, eventdata, handles)
    % hObject    handle to comparegrafik (see GCBO)
```

```
    % eventdata  reserved - to be defined in a future version of
MATLAB
    % handles    structure with handles and user data (see GUIDATA)
    global magx10afile;
    global freqfile;
    global freq;
    global magx10a;


    handles.axes6;
    plot(freq,magx10a,freqfile,magx10afile, 'r-');
    set(gca,'xtick',0:25:1200);
    set(gca,'ytick',0:10:1200);
    grid on;
    title('Compare');
    xlabel ('Frequency (Hz)');
    ylabel ('Amplitude (Volt)');
    legend ('baru diambil','dari file');
    % set (handles.monitor, 'string','Compare');



    % --- Executes on button press in comparesample.
    function comparesample_Callback(hObject, eventdata, handles)
    % hObject    handle to comparesample (see GCBO)
    % eventdata  reserved - to be defined in a future version of
MATLAB
    % handles    structure with handles and user data (see GUIDATA)
    global samplefile;
    global sample;
    handles.axes6;
    plot(sample,samplefile, 'r-');
    set(gca,'xtick', -1000:100:2500);
    set(gca,'ytick', -1000:50  :2500);
    grid on;
    title('Compare');
    xlabel ('Frequency (Hz)');
    ylabel ('Amplitude (Volt)');
    legend ('baru diambil','dari file');
```

```
\% --- Executes on button press in nameload.
function nameload_Callback(hObject, eventdata, handles)
% hObject    handle to nameload (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

Form Database

```
function varargout = Database(varargin)
    % DATABASE MATLAB code for Database.fig
    %      DATABASE, by itself, creates a new DATABASE or raises
the existing
    %      singleton*.
    %
    %      H = DATABASE returns the handle to a new DATABASE
or the handle to
    %      the existing singleton*.
    %
    %      DATABASE('CALLBACK',hObject,eventData,handles,...)
calls the local
    %      function named CALLBACK in DATABASE.M with the
given input arguments.
    %
    %      DATABASE('Property','Value',...) creates a new
DATABASE or raises the
    %      existing singleton*.  Starting from the left, property value
pairs are
    %      applied to the GUI before Database_OpeningFcn gets called.
An
    %      unrecognized property name or invalid value makes property
application
    %      stop.  All inputs are passed to Database_OpeningFcn via
varargin.
    %
    %      *See GUI Options on GUIDE's Tools menu.  Choose "GUI
allows only one
    %      instance to run (singleton)".
    %
    % See also: GUIDE, GUIDATA, GUIHANDLES

    % Edit the above text to modify the response to help Database

    % Last Modified by GUIDE v2.5 26-May-2016 15:27:04

    % Begin initialization code - DO NOT EDIT
    gui_Singleton = 1;
```

```
    gui_State = struct('gui_Name',       mfilename, ...
                       'gui_Singleton',  gui_Singleton, ...
                       'gui_OpeningFcn', @Database_OpeningFcn, ...
                       'gui_OutputFcn',  @Database_OutputFcn, ...
                       'gui_LayoutFcn',  [] , ...
                       'gui_Callback',   []);
    if nargin && ischar(varargin{1})
        gui_State.gui_Callback = str2func(varargin{1});
    end

    if nargout
        [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
    else
        gui_mainfcn(gui_State, varargin{:});
    end
    % End initialization code - DO NOT EDIT


    % --- Executes just before Database is made visible.
    function Database_OpeningFcn(hObject, eventdata, handles,
varargin)
    % This function has no output args, see OutputFcn.
    % hObject    handle to figure
    % eventdata  reserved - to be defined in a future version of
MATLAB
    % handles    structure with handles and user data (see GUIDATA)
    % varargin   command line arguments to Database (see
VARARGIN)

    % Choose default command line output for Database
    handles.output = hObject;

    % Update handles structure
    guidata(hObject, handles);
    guidata(hObject, handles);
    hback = axes('units','normalized','position',[0 0 1 1]);
    uistack(hback,'bottom');
    % menampilkan background
    [back map]=imread('database1.jpg');
```

```
image(back)
colormap(map)
% handlevisibility off agar axes tidak terlihat
% dan gambar background saja yang muncul.
set(hback,'handlevisibility','off','visible','off')

% UIWAIT makes Database wait for user response (see
UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = Database_OutputFcn(hObject, eventdata,
handles)
% varargout  cell array for returning output args (see
VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in plottabel.
function plottabel_Callback(hObject, eventdata, handles)
% hObject    handle to plottabel (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
% global X;
% % filename = uigetfile('*.xls') ;
% % if isequal(filename,0)
% %     return
% % end
% X=xlsread('Book1.xls');
% set(handles.tabelsd,'Data', X);
```

```
        [file,path] = uigetfile({'*.csv;*.xls;*.xlsx','Excel
Files'},'FluidesInternes');
         filename = strcat(path,file);
         %set(handles.upload);
         data = xlsread(filename);
        %    data1 = cell(data,4);
         set(handles.tabelsd,'Data',data)


        % --- Executes on button press in plotgrafik.
        function plotgrafik_Callback(hObject, eventdata, handles)
        % hObject    handle to plotgrafik (see GCBO)
        % eventdata  reserved - to be defined in a future version of
MATLAB
        % handles    structure with handles and user data (see GUIDATA)
         global Tegangan;
        % filename = uigetfile('*.csv;*.xls') ;
        % if isequal(filename,0)
        %     return
        % end
        % sheet = 1;
        %
        % Tegangan = xlsread(filename,sheet,'G1:G2500');
         [file,path] = uigetfile({'*.csv;*.xls;*.xlsx','Excel
Files'},'FluidesInternes');
          filename = strcat(path,file);
          %set(handles.upload);
          Tegangan = xlsread(filename,'G1:G2500');
        handles.axes1;
        plot (Tegangan);
        set(gca,'xtick', -1000:100:2500);
        ylabel('Tegangan (V)');
        xlabel('Data');
        title('Tegangan RMS');
        grid on;


        % --- Executes on button press in monitoring.
        function monitoring_Callback(hObject, eventdata, handles)
```

```
    % hObject    handle to monitoring (see GCBO)
    % eventdata  reserved - to be defined in a future version of
MATLAB
    % handles    structure with handles and user data (see GUIDATA)
    close(Database);
      (flickerFFT2);

    % --- Executes on button press in menu.
    function menu_Callback(hObject, eventdata, handles)
    % hObject    handle to menu (see GCBO)
    % eventdata  reserved - to be defined in a future version of
MATLAB
    % handles    structure with handles and user data (see GUIDATA)
    close(Database);
      (menu);


    % --- Executes on button press in keluar.
    function keluar_Callback(hObject, eventdata, handles)
    % hObject    handle to keluar (see GCBO)
    % eventdata  reserved - to be defined in a future version of
MATLAB
    % handles    structure with handles and user data (see GUIDATA)
    close(Database);


    % --- Executes on button press in arus.
    function arus_Callback(hObject, eventdata, handles)
    % hObject    handle to arus (see GCBO)
    % eventdata  reserved - to be defined in a future version of
MATLAB
    % handles    structure with handles and user data (see GUIDATA)
    global Arus;
    % filename = uigetfile('*.csv;*.xls') ;
    % if isequal(filename,0)
    %     return
    % end
    % sheet = 1;
    %
```

```
        [file,path] = uigetfile({'*.csv;*.xls;*.xlsx','Excel
Files'},'FluidesInternes');
         filename = strcat(path,file);
         %set(handles.upload);

        Arus = xlsread(filename,'H1:H2500');
        handles.axes1;
        plot (Arus);
        set(gca,'xtick', -1000:100:25000);
        ylabel('Arus (I)');
        xlabel('Data');
        title('Arus');
        grid on;
```

LAMPIRAN B

Datasheet

Datasheet ATMega 2560

Features

- High Performance, Low Power AVR® 8-Bit Microcontroller
- Advanced RISC Architecture
  - 135 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
- Non-volatile Program and Data Memories
  - 64K/128K/256K 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
  - 4K Bytes EEPROM
    Endurance: 100,000 Write/Erase Cycles
  - 8K Bytes Internal SRAM
  - Up to 64K Bytes Optional External Memory Space
  - 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 Prescaler and Compare Mode
  - Four 16-bit Timer/Counter with Separate Prescaler, Compare- and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Four 8-bit PWM Channels
  - Six/Twelve PWM Channels with Programmable Resolution from 2 to 16 Bits (ATmega1281/2561, ATmega640/1280/2560)
  - Output Compare Modulator
  - 8/16-channel, 10-bit ADC (ATmega1281/2561, ATmega640/1280/2560)
  - Two/Four Programmable Serial USART (ATmega1281/2561,ATmega640/1280/2560)
  - Master/Slave SPI Serial Interface
  - Byte Oriented 2-wire Serial Interface
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analog Comparator
  - Interrupt and Wake-up on Pin Change
- Special Microcontroller Features
  - Power-on Reset and Programmable Brown-out Detection
  - Internal Calibrated 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
  - 54/86 Programmable I/O Lines (ATmega1281/2561, ATmega640/1280/2560)
  - 64-pad QFN/MLF, 64-lead TQFP (ATmega1281/2561)
  - 100-lead TQFP, 100-ball CBGA (ATmega640/1280/2560)
  - RoHS/Fully Green
- Temperature Range:
  - -40°C to 85°C Industrial
- Ultra-Low Power Consumption
  - Active Mode: 1 MHz, 1.8V: 510 µA
  - Power-down Mode: 0.1 µA at 1.8V
- Speed Grade (see "Maximum speed vs. VCC" on page 377):
  - ATmega640V/ATmega1280V/ATmega1281V:
    0 - 4 MHz @ 1.8 - 5.5V, 0 - 8 MHz @ 2.7 - 5.5V
  - ATmega2560V/ATmega2561V:
    0 - 2 MHz @ 1.8 - 5.5V, 0 - 8 MHz @ 2.7 - 5.5V
  - ATmega640/ATmega1280/ATmega1281:
    0 - 8 MHz @ 2.7 - 5.5V, 0 - 16 MHz @ 4.5 - 5.5V
  - ATmega2560/ATmega2561:
    0 - 16 MHz @ 4.5 - 5.5V

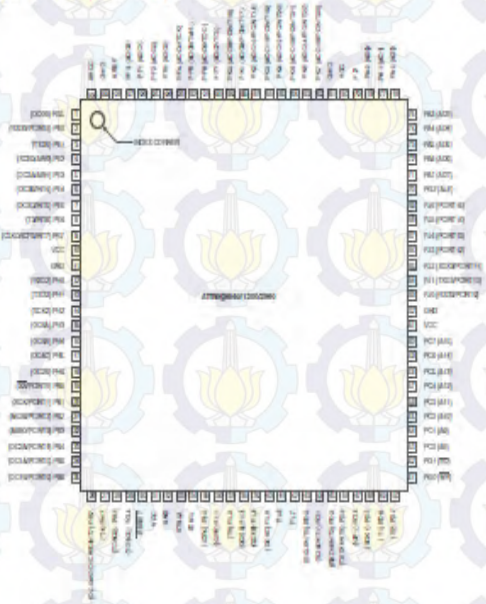
8-bit AVR®
Microcontroller
with
64K/128K/256K
Bytes In-System
Programmable
Flash

ATmega640/V
ATmega1280/V
ATmega1281/V
ATmega2560/V
ATmega2561/V

Preliminary

Pin Configurations

Figure 1. TQFP-pinout ATmega640/1280/2560

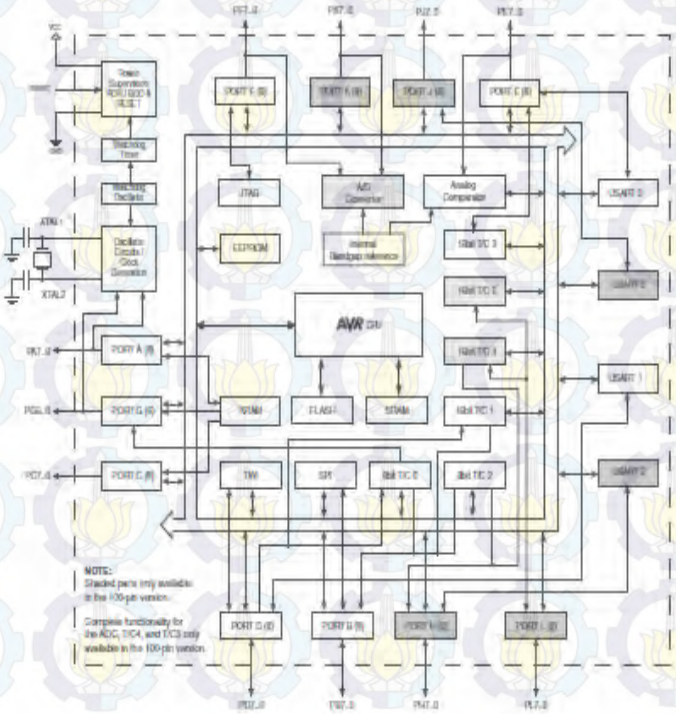


Overview

The ATmega640/1280/1281/2560/2561 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 ATmega640/1280/1281/2560/2561 achieves throughputs approaching 1 MIPS per MHz allowing the system designer to optimize power consumption versus processing speed.

Block Diagram

Figure 4. Block Diagram



Comparison Between ATmega1281/2561 and ATmega640/1280/2560

Each device in the ATmega640/1280/1281/2560/2561 family differs only in memory size and number of pins. Table 2 summarizes the different configurations for the six devices.

Table 2. Configuration Summary

Device	Flash	EEPROM	RAM	General Purpose I/O pins	16 bits resolution PWM channels	Serial USARTs	ADC Channels
ATmega640	64KB	4KB	8KB	86	12	4	16
ATmega1280	128KB	4KB	8KB	86	12	4	16
ATmega1281	128KB	4KB	8KB	54	6	2	8
ATmega2560	256KB	4KB	8KB	86	12	4	16
ATmega2561	256KB	4KB	8KB	54	6	2	8

Pin Descriptions

VCC — Digital supply voltage.

GND — Ground.

Port A (PA7..PA0) — Port A is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port A output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port A pins that are externally pulled low will source current if the 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 A also serves the functions of various special features of the ATmega640/1280/1281/2560/2561 as listed on page 91.

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 has better driving capabilities than the other ports.

Port B also serves the functions of various special features of the ATmega640/1280/1281/2560/2561 as listed on page 92.

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.

Port C also serves the functions of special features of the ATmega640/1280/1281/2560/2561 as listed on page 95.

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 ATmega640/1280/1281/2560/2561 as listed on page 97.

Port E (PE7..PE0)

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

Port E also serves the functions of various special features of the ATmega640/1280/1281/2560/2561 as listed on page 99.

Port F (PF7..PF0)

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

Port F 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 F output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port F pins that are externally pulled low will source current if the pull-up resistors are activated. The Port F 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 PF7(TDI), PF5(TMS), and PF4(TCK) will be activated even if a reset occurs.

Port F also serves the functions of the JTAG interface.

Port G (PG5..PG0)

Port G is a 6-bit I/O port with internal pull-up resistors (selected for each bit). The Port G output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port G pins that are externally pulled low will source current if the pull-up resistors are activated. The Port G pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port G also serves the functions of various special features of the ATmega640/1280/1281/2560/2561 as listed on page 105.

Port H (PH7..PH0)

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

Port H also serves the functions of various special features of the ATmega640/1280/2560 as listed on page 107.

Port J (PJ7..PJ0)

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

Port J also serves the functions of various special features of the ATmega640/1280/2560 as listed on page 109.

Port K (PK7..PK0)

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

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

Port K also serves the functions of various special features of the ATmega640/1280/2560 as listed on page 111.

Port L (PL7..PL0) Port L is a 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port L output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port L pins that are externally pulled low will source current if the pull-up resistors are activated. The Port L pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port L also serves the functions of various special features of the ATmega640/1280/2560 as listed on page 113.

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 26 on page 58. 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 F 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 This is the analog reference pin for the A/D Converter.

Resources

A comprehensive set of development tools and application notes, and datasheets are available for download on http://www.atmel.com/avr.

About Code Examples

This documentation contains simple code examples that briefly show how to use various parts of the device. 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.

These code examples assume that the part specific header file is included before compilation. For I/O registers located in extended I/O map, "IN", "OUT", "SBIS", "SBIC", "CBI", and "SBI" instructions must be replaced with instructions that allow access to extended I/O. Typically "LDS" and "STS" combined with "SBRS", "SBRC", "SBR", and "CBR".

***Datasheet* DS1307**

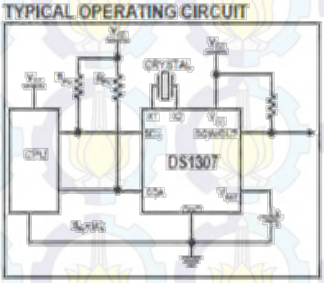
maxim integrated

DS1307
64 x 8, Serial, I²C Real-Time Clock

GENERAL DESCRIPTION

The DS1307 serial real-time clock (RTC) is a low-power, full binary-coded decimal (BCD) clock/calendar plus 56 bytes of NV SRAM. Address and data are transferred serially through an I²C, bidirectional bus. The clock/calendar provides seconds, minutes, hours, day, date, month, and year information. The end of the month date is automatically adjusted for months with fewer than 31 days, including corrections for leap year. The clock operates in either the 24-hour or 12-hour format with AM/PM indicator. The DS1307 has a built-in power-sense circuit that detects power failures and automatically switches to the backup supply. Timekeeping operation continues while the part operates from the backup supply.

TYPICAL OPERATING CIRCUIT



BENEFITS AND FEATURES

- Completely Manages All Timekeeping Functions
  - Real-Time Clock Counts Seconds, Minutes, Hours, Date of the Month, Month, Day of the Week, and Year with Leap-Year Compensation Valid Up to 2100
  - 56-Byte, Battery-Backed, General-Purpose RAM with Unlimited Writes
  - Programmable Square-Wave Output Signal
- Simple Serial Port Interfaces to Most Microcontrollers
  - I²C Serial Interface
- Low Power Operation Extends Battery-Backup Run Time
  - Consumes Less than 500nA in Battery-Backup Mode with Oscillator Running
  - Automatic Power-Fail Detect and Switch Circuitry
- 8-Pin DIP and 8-Pin SO Minimizes Required Space
- Optional Industrial Temperature Range: -40°C to +85°C Supports Operation in a Wide Range of Applications
- Underwriters Laboratories® (UL) Recognized

PIN CONFIGURATIONS

ORDERING INFORMATION

PART	TEMP RANGE	VOLTAGE (V)	PIN-PACKAGE	TOP MARK*
DS1307+	0°C to +70°C	5.0	8 PDIP (300 mils)	DS1307
DS1307N+	-40°C to +85°C	5.0	8 PDIP (300 mils)	DS1307N
DS1307Z+	0°C to +70°C	5.0	8 SO (150 mils)	DS1307
DS1307ZN+	-40°C to +85°C	5.0	8 SO (150 mils)	DS1307N
DS1307Z+T&R	0°C to +70°C	5.0	8 SO (150 mils) Tape and Reel	DS1307
DS1307ZN+T&R	-40°C to +85°C	5.0	8 SO (150 mils) Tape and Reel	DS1307N

+Denotes a lead-free/RoHS-compliant package.
*A "+" anywhere on the top mark indicates a lead-free package. An "N" anywhere on the top mark indicates an industrial temperature range device.
Underwriters Laboratories, Inc. is a registered certification mark of Underwriters Laboratories, Inc.

REV: 3/15

ABSOLUTE MAXIMUM RATINGS

Voltage Range on Any Pin Relative to Ground ..... -0.5V to +7.0V
Operating Temperature Range (Noncondensing)
Commercial ..... 0°C to +70°C
Industrial ..... -40°C to +85°C
Storage Temperature Range ..... -55°C to +125°C
Soldering Temperature (DIP, leads) ..... +260°C for 10 seconds
Soldering Temperature (surface mount) ..... Refer to the JPC/JEDEC J-STD-020 Specification.

Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated in the operational sections of the specifications is not implied. Exposure to the absolute maximum rating conditions for extended periods may affect device reliability.

RECOMMENDED DC OPERATING CONDITIONS

(T_A = 0°C to +70°C, T_A = -40°C to +85°C.) (Notes 1, 2)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
Supply Voltage	V_{CC}		4.5	5.0	5.5	V
Logic 1 Input	V_{IH}		2.2		V_{CC} + 0.3	V
Logic 0 Input	V_{IL}		-0.3		+0.8	V
V_{BAT} Battery Voltage	V_{BAT}		2.0	3	3.5	V

DC ELECTRICAL CHARACTERISTICS

(V_{CC} = 4.5V to 5.5V; T_A = 0°C to +70°C, T_A = -40°C to +85°C.) (Notes 1, 2)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
Input Leakage (SCL)	I_{LI}		-1		1	µA
I/O Leakage (SDA, SQW/OUT)	I_{LO}		-1		1	µA
Logic 0 Output (I_{OL} = 5mA)	V_{OL}				0.4	V
Active Supply Current (f_{SCL} = 100kHz)	I_{CCA}				1.5	mA
Standby Current	I_{CCS}	(Note 3)			200	µA
V_{BAT} Leakage Current	I_{BATLKG}			5	50	nA
Power-Fail Voltage (V_{BAT} = 3.0V)	V_{PF}		1.216 x V_{BAT}	1.25 x V_{BAT}	1.264 x V_{BAT}	V

DC ELECTRICAL CHARACTERISTICS

(V_{CC} = 0V, V_{BAT} = 3.0V; T_A = 0°C to +70°C, T_A = -40°C to +85°C.) (Notes 1, 2)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
V_{BAT} Current (OSC ON); SQW/OUT OFF	I_{BAT1}			300	500	nA
V_{BAT} Current (OSC ON); SQW/OUT ON (32kHz)	I_{BAT2}			480	800	nA
V_{BAT} Data-Retention Current (Oscillator Off)	I_{BATDR}			10	100	nA

WARNING: Negative undershoots below -0.3V while the part is in battery-backed mode may cause loss of data.

AC ELECTRICAL CHARACTERISTICS

(V_{CC} = 4.5V to 5.5V; T_A = 0°C to +70°C, T_A = -40°C to +85°C.)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
SCL Clock Frequency	f_{SCL}		0		100	kHz
Bus Free Time Between a STOP and START Condition	t_{BUF}		4.7			µs
Hold Time (Repeated) START Condition	$t_{HD:STA}$	(Note 4)	4.0			µs
LOW Period of SCL Clock	t_{LOW}		4.7			µs
HIGH Period of SCL Clock	t_{HIGH}		4.0			µs
Setup Time for a Repeated START Condition	$t_{SU:STA}$		4.7			µs
Data Hold Time	$t_{HD:DAT}$		0			µs
Data Setup Time	$t_{SU:DAT}$	(Notes 5, 6)	250			ns
Rise Time of Both SDA and SCL Signals	t_R				1000	ns
Fall Time of Both SDA and SCL Signals	t_F				300	ns
Setup Time for STOP Condition	$t_{SU:STO}$		4.7			µs

CAPACITANCE

(T_A = +25°C)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
Pin Capacitance (SDA, SCL)	$C_{I/O}$				10	pF
Capacitance Load for Each Bus Line	C_B	(Note 7)			400	pF

Note 1: All voltages are referenced to ground.

Note 2: Limits at -40°C are guaranteed by design and are not production tested.

Note 3: I_{LKG} specified with V_{CC} = 5.0V and SDA, SCL = 5.0V.

Note 4: After this period, the first clock pulse is generated.

Note 5: A device must internally provide a hold time of at least 300ns for the SDA signal (referred to the $V_{IH(MIN)}$ of the SCL signal) to bridge the undefined region of the falling edge of SCL.

Note 6: The maximum $t_{HD:DAT}$ only has to be met if the device does not stretch the LOW period (t_{LOW}) of the SCL signal.

Note 7: C_B—total capacitance of one bus line in pF.

TIMING DIAGRAM

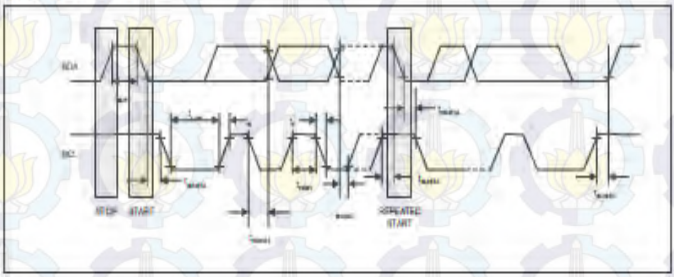
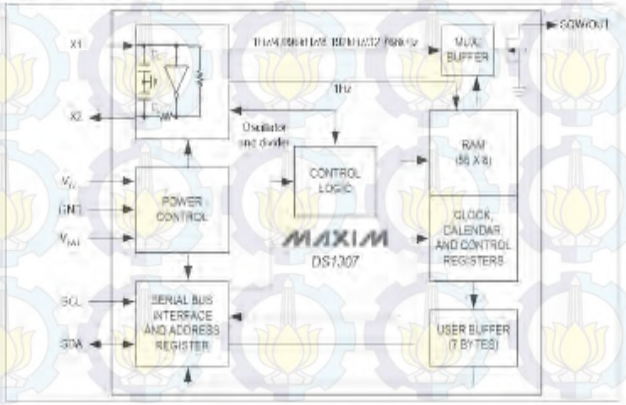


Figure 1. Block Diagram



PIN DESCRIPTION

PIN	NAME	FUNCTION
1	X1	Connections for Standard 32.768kHz Quartz Crystal. The internal oscillator circuitry is designed for operation with a crystal having a specified load capacitance (C_L) of 12.5pF. X1 is the input to the oscillator and can optionally be connected to an external 32.768kHz oscillator. The output of the internal oscillator, X2, is floated if an external oscillator is connected to X1. **Note:** For more information on crystal selection and crystal layout considerations, refer to *Application Note 58: Crystal Considerations with Dallas Real-Time Clocks.*
2	X2	
3	V_{BAT}	Backup Supply Input for Any Standard 3V Lithium Cell or Other Energy Source. Battery voltage must be held between the minimum and maximum limits for proper operation. Diodes in series between the battery and the V_{BAT} pin may prevent proper operation. If a backup supply is not required, V_{BAT} must be grounded. The nominal power-fail trip point (V_{PF}) voltage at which access to the RTC and user RAM is denied is set by the internal circuitry as 1.25 x V_{BAT} nominal. A lithium battery with 48mAh or greater will back up the DS1307 for more than 10 years in the absence of power at +25°C. UL recognized to ensure against reverse charging current when used with a lithium battery. Go to: www.maxim-ic.com/qa/info/ul/.
4	GND	Ground
5	SDA	Serial Data Input/Output. SDA is the data input/output for the I²C serial interface. The SDA pin is open drain and requires an external pullup resistor. The pullup voltage can be up to 5.5V regardless of the voltage on V_{CC}.
6	SCL	Serial Clock Input. SCL is the clock input for the I²C interface and is used to synchronize data movement on the serial interface. The pullup voltage can be up to 5.5V regardless of the voltage on V_{CC}.
7	SQW/OUT	Square Wave/Output Driver. When enabled, the SQWE bit set to 1, the SQW/OUT pin outputs one of four square-wave frequencies (1Hz, 4kHz, 8kHz, 32kHz). The SQW/OUT pin is open drain and requires an external pullup resistor. SQW/OUT operates with either V_{CC} or V_{BAT} applied. The pullup voltage can be up to 5.5V regardless of the voltage on V_{CC}. If not used, this pin can be left floating.
8	V_{CC}	Primary Power Supply. When voltage is applied within normal limits, the device is fully accessible and data can be written and read. When a backup supply is connected to the device and V_{CC} is below V_{TP}, read and writes are inhibited. However, the timekeeping function continues unaffected by the lower input voltage.

DETAILED DESCRIPTION

The DS1307 is a low-power clock/calendar with 56 bytes of battery-backed SRAM. The clock/calendar provides seconds, minutes, hours, day, date, month, and year information. The date at the end of the month is automatically adjusted for months with fewer than 31 days, including corrections for leap year. The DS1307 operates as a slave device on the I²C bus. Access is obtained by implementing a START condition and providing a device identification code followed by a register address. Subsequent registers can be accessed sequentially until a STOP condition is executed. When V_{CC} falls below 1.25 x V_{BAT}, the device terminates an access in progress and resets the device address counter. Inputs to the device will not be recognized at this time to prevent erroneous data from being written to the device from an out-of-tolerance system. When V_{CC} falls below V_{BAT}, the device switches into a low-current battery-backup mode. Upon power-up, the device switches from battery to V_{CC} when V_{CC} is greater than V_{BAT} +0.2V and recognizes inputs when V_{CC} is greater than 1.25 x V_{BAT}. The block diagram in Figure 1 shows the main elements of the serial RTC.

OSCILLATOR CIRCUIT

The DS1307 uses an external 32.768kHz crystal. The oscillator circuit does not require any external resistors or capacitors to operate. Table 1 specifies several crystal parameters for the external crystal. Figure 1 shows a functional schematic of the oscillator circuit. If using a crystal with the specified characteristics, the startup time is usually less than one second.

CLOCK ACCURACY

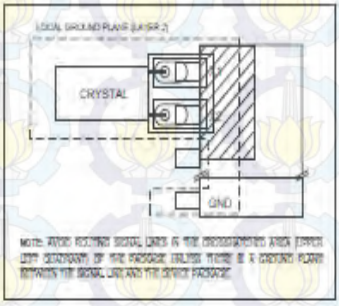
The accuracy of the clock is dependent upon the accuracy of the crystal and the accuracy of the match between the capacitive load of the oscillator circuit and the capacitive load for which the crystal was trimmed. Additional error will be added by crystal frequency drift caused by temperature shifts. External circuit noise coupled into the oscillator circuit may result in the clock running fast. Refer to Application Note 58: *Crystal Considerations with Dallas Real-Time Clocks* for detailed information.

Table 1. Crystal Specifications*

PARAMETER	SYMBOL	MIN	TYP	MAX	UNITS
Nominal Frequency	f_O		32.768		kHz
Series Resistance	ESR			45	kΩ
Load Capacitance	C_L		12.5		pF

**The crystal, traces, and crystal input pins should be isolated from RF generating signals. Refer to Application Note 58: Crystal Considerations for Dallas Real-Time Clocks for additional specifications.*

Figure 2. Recommended Layout for Crystal



RTC AND RAM ADDRESS MAP

Table 2 shows the address map for the DS1307 RTC and RAM registers. The RTC registers are located in address locations 00h to 07h. The RAM registers are located in address locations 08h to 3Fh. During a multibyte access, when the address pointer reaches 3Fh, the end of RAM space, it wraps around to location 00h, the beginning of the clock space.

FAIRCHILD
SEMICONDUCTOR™

October 1987
Revised January 1999

CD4049UBC • CD4050BC
Hex Inverting Buffer • Hex Non-Inverting Buffer

General Description

The CD4049UBC and CD4050BC hex buffers are monolithic complementary MOS (CMOS) integrated circuits constructed with N- and P-channel enhancement mode transistors. These devices feature logic level conversion using only one supply voltage (V_{CC}). The input signal high level (V_{IN}) can exceed the V_{CC} supply voltage when these devices are used for logic level conversions. These devices are intended for use as hex buffers, CMOS to DTL/TTL converters, or as CMOS current drivers, and at V_{CC} = 5.0V, they can drive directly two DTL/TTL loads over the full operating temperature range.

Features

- Wide supply voltage range: 3.0V to 15V
- Direct drive to 2 TTL loads at 5.0V over full temperature range
- High source and sink current capability
- Special input protection permits input voltages greater than V_{CC}

Applications

- CMOS hex inverter/buffer
- CMOS to DTL/TTL hex converter
- CMOS current "sink" or "source" driver
- CMOS HIGH-to-LOW logic level converter

Ordering Code:

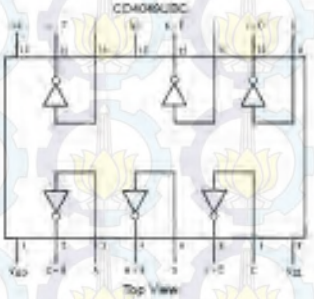
Order Number	Package Number	Package Description
CD4049UBCM	M16A	16-Lead Small Outline Integrated Circuit (SOIC), JEDEC MS-012, 0.150" Narrow
CD4049UBCN	N16E	16-Lead Plastic Dual-In-Line Package (PDIP), JEDEC MS-001, 0.300" Wide
CD4050BCM	M16A	16-Lead Small Outline Integrated Circuit (SOIC), JEDEC MS-012, 0.150" Narrow
CD4050BCN	N16E	16-Lead Plastic Dual-In-Line Package (PDIP), JEDEC MS-001, 0.300" Wide

Devices also available in Tape and Reel. Specify by appending the suffix letter "X" to the ordering code.

Connection Diagrams

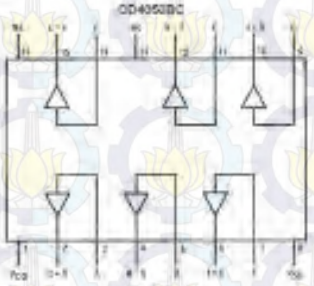
Pin Assignments for DIP

CD4049UBC



Top View

CD4050BC



Top View

© 1999 Fairchild Semiconductor Corporation

Absolute Maximum Ratings (Note 1)

(Note 2)

Supply Voltage (V_{DD})	−0.5V to +18V
Input Voltage (V_{IN})	−0.5V to +18V
Voltage at Any Output Pin (V_{OUT})	−0.5V to V_{DD} + 0.5V
Storage Temperature Range (T_S)	−65°C to +150°C
Power Dissipation (P_D)	
Dual-In-Line	700 mW
Small Outline	500 mW
Lead Temperature (T_L)	
(Soldering, 10 seconds)	260°C

Recommended Operating Conditions (Note 2)

Supply Voltage (V_{DD})	3V to 15V
Input Voltage (V_{IN})	0V to 15V
Voltage at Any Output Pin (V_{OUT})	0 to V_{DD}
Operating Temperature Range (T_A)	
CD4049UBC, CD4050BC	−40°C to +85°C

Note 1: "Absolute Maximum Ratings" are those values beyond which the safety of the device cannot be guaranteed. They are not meant to imply that the devices should be operated at these limits. The tables of "Recommended Operating Conditions" and "Electrical Characteristics" provide conditions for actual device operation.

Note 2: V_{SS} = 0V unless otherwise specified.

DC Electrical Characteristics (Note 3)

Symbol	Parameter	Conditions	−40°C		+25°C			+85°C		Units
			Min	Max	Min	Typ	Max	Min	Max	
I_{DD}	Quiescent Device Current	V_{DD} = 5V		5		0.03	5.0		30	µA
		V_{DD} = 10V		10		0.05	10.0		60	µA
		V_{DD} = 15V		20		0.07	15.0		120	µA
V_{OL}	LOW Level Output Voltage	V_{IN} = V_{DD}, V_{IL} = 0V, $\|I_O\|$ < 1 µA								
		V_{DD} = 5V		0.05		0	0.05		0.05	V
		V_{DD} = 10V		0.05		0	0.05		0.05	V
		V_{DD} = 15V		0.05		0	0.05		0.05	V
V_{OH}	HIGH Level Output Voltage	V_{IN} = V_{DD}, V_{IL} = 0V, $\|I_O\|$ < 1 µA								
		V_{DD} = 5V	4.95		4.95	5		4.95		V
		V_{DD} = 10V	9.95		9.95	10		9.95		V
		V_{DD} = 15V	14.95		14.95	15		14.95		V
V_{IL}	LOW Level Input Voltage (CD4050BC Only)	$\|I_O\|$ < 1 µA, V_O = 0.5V		1.5		2.25	1.5		1.5	V
		V_{DD} = 10V, V_O = 1V		3.0		4.5	3.0		3.0	V
		V_{DD} = 15V, V_O = 1.5V		4.0		6.75	4.0		4.0	V
V_{IL}	LOW Level Input Voltage (CD4049UBC Only)	$\|I_O\|$ < 1 µA, V_O = 4.5V		1.0		1.5	1.0		1.0	V
		V_{DD} = 10V, V_O = 9V		2.0		2.5	2.0		2.0	V
		V_{DD} = 15V, V_O = 13.5V		3.0		3.5	3.0		3.0	V
V_{IH}	HIGH Level Input Voltage (CD4050BC Only)	$\|I_O\|$ < 1 µA, V_O = 4.5V	3.5		3.5	2.75		3.5		V
		V_{DD} = 10V, V_O = 9V	7.0		7.0	5.5		7.0		V
		V_{DD} = 15V, V_O = 13.5V	11.0		11.0	8.25		11.0		V
V_{IH}	HIGH Level Input Voltage (CD4049UBC Only)	$\|I_O\|$ < 1 µA, V_O = 0.5V	4.0		4.0	3.5		4.0		V
		V_{DD} = 10V, V_O = 1V	8.0		8.0	7.5		8.0		V
		V_{DD} = 15V, V_O = 1.5V	12.0		12.0	11.5		12.0		V
I_{OL}	LOW Level Output Current (Note 4)	V_{IN} = V_{DD}, V_{IL} = 0V, V_{DD} = 5V, V_O = 0.4V	48		40	80		32		mA
		V_{DD} = 10V, V_O = 0.5V	9.6		8.0	16		6.4		mA
		V_{DD} = 15V, V_O = 1.5V	29		25	50		20		mA
I_{OH}	HIGH Level Output Current (Note 4)	V_{IN} = V_{DD}, V_{IL} = 0V, V_{DD} = 5V, V_O = 4.6V	−10		−8.0	−16		−6.4		mA
		V_{DD} = 10V, V_O = 9.5V	−2.1		−1.6	−3.2		−1.6		mA
		V_{DD} = 15V, V_O = 13.5V	−7.1		−5.6	−11		−4		mA
I_{IN}	Input Current	V_{DD} = 15V, V_{IN} = 0V	−0.3		−0.3	-10^{-5}			−1.0	µA
		V_{DD} = 15V, V_{IN} = 15V	0.3		0.3	10^{-5}			1.0	µA

Note 3: V_{SS} = 0V unless otherwise specified.

DC Electrical Characteristics (Continued)

Note 6: These are peak output current capabilities. Continuous output current is rated at 12 mA maximum. The output current should not be allowed to exceed this value for extended periods of time. I_{OH} and I_{OL} are tested one output at a time.

AC Electrical Characteristics (Note 5)

CD4048UBC

$T_A = 25°C$, $C_L = 50$ pF, $R_L = 200k$, $t_r = t_f = 20$ ns, unless otherwise specified

Symbol	Parameter	Conditions	Min	Typ	Max	Units
t_{PHL}	Propagation Delay Time	$V_{DD} = 5V$		30	65	ns
	HIGH-to-LOW Level	$V_{DD} = 10V$		20	40	ns
		$V_{DD} = 15V$		15	30	ns
t_{PLH}	Propagation Delay Time	$V_{DD} = 5V$		45	85	ns
	LOW-to-HIGH Level	$V_{DD} = 10V$		25	45	ns
		$V_{DD} = 15V$		20	35	ns
t_{THL}	Transition Time	$V_{DD} = 5V$		30	60	ns
	HIGH-to-LOW Level	$V_{DD} = 10V$		20	40	ns
		$V_{DD} = 15V$		15	30	ns
t_{TLH}	Transition Time	$V_{DD} = 5V$		60	120	ns
	LOW-to-HIGH Level	$V_{DD} = 10V$		30	60	ns
		$V_{DD} = 15V$		25	45	ns
C_{IN}	Input Capacitance	Any Input		7.5	15	pF

Note 5: AC Parameters are guaranteed by DC correlated testing.

AC Electrical Characteristics (Note 6)

CD4050BC

$T_A = 25°C$, $C_L = 50$ pF, $R_L = 200k$, $t_r = t_f = 20$ ns, unless otherwise specified

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Units
t_{PHL}	Propagation Delay Time	$V_{DD} = 5V$		60	110	ns
	HIGH-to-LOW Level	$V_{DD} = 10V$		25	55	ns
		$V_{DD} = 15V$		20	30	ns
t_{PLH}	Propagation Delay Time	$V_{DD} = 5V$		60	120	ns
	LOW-to-HIGH Level	$V_{DD} = 10V$		30	55	ns
		$V_{DD} = 15V$		25	45	ns
t_{THL}	Transition Time	$V_{DD} = 5V$		30	60	ns
	HIGH-to-LOW Level	$V_{DD} = 10V$		20	40	ns
		$V_{DD} = 15V$		15	30	ns
t_{TLH}	Transition Time	$V_{DD} = 5V$		60	120	ns
	LOW-to-HIGH Level	$V_{DD} = 10V$		30	60	ns
		$V_{DD} = 15V$		25	45	ns
C_{IN}	Input Capacitance	Any Input		5	7.5	pF

Note 6: AC Parameters are guaranteed by DC correlated testing.

Datasheet SD Card

SanDisk

SanDisk Ultra® microSDHC™_microSDXC™ UHS-I Card
SanDisk Branded Card in Trays, Bulk Delivered
Set-up Sheet

Specifications/Features

- Available capacities: 8GB, 16 GB, 32 GB and 64 GB*
- Class 10
- Record more in Full HD with enabled Class 10 video recording capability†
- Transfer more files with faster speeds up to 48MB/sec**
- Tested under extreme conditions. Water, Temperature, X-ray, Shock Proof‡

Product Configuration

- Branded Ultra microSDHC™ and Ultra microSDXC™ Cards placed in Trays that hold 120 units.
- 10 Trays are stacked to deliver 1,200 units One Case Pack that can become an inner pack.
- 6 inner packs are built into a Master Case Pack that holds 7,200 units.

Part No. Info

Product Name	Sales Item No
SanDisk Ultra® microSDHC™ UHS-I 8GB Card	SDSDQU-008G-BMUT
SanDisk Ultra® microSDHC™ UHS-I 16GB Card	SDSDQU-016G-BMUT
SanDisk Ultra® microSDHC™ UHS-I 32GB Card	SDSDQU-032G-BMUT
SanDisk Ultra® microSDXC™ UHS-I 64GB Card	SDSDQU-064G-BMUT

Set-Up Info

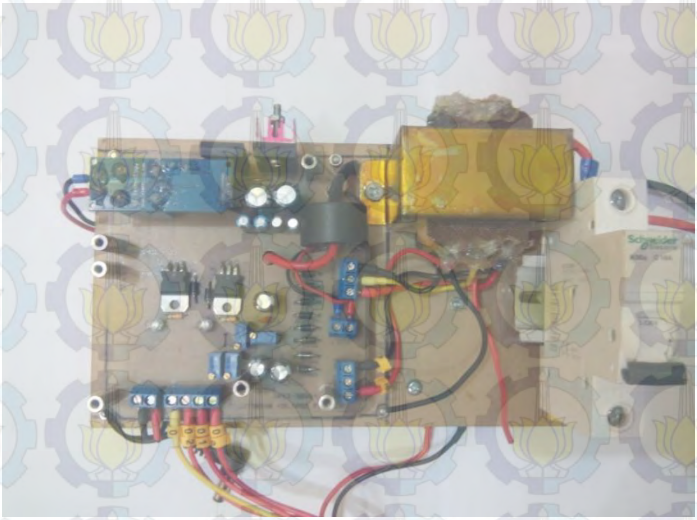
	Weight	Dimensions	Inner Qty	Master Qty	Master Pack Qty	Country of Origin
Single Inners Package	1,840 g	354 mm x 153 mm x 89 mm	10 trays	1,200	7,200	China
	4.05 lbs	13.93 in x 6.02 in x 3.50 in				
Master Case	11,900 g	385 mm x 330 mm x 290 mm	1,200 units			
	26.23 lbs	15.16 in x 13.0 in x 11.41 in				

Not all devices support microSDXC™. Contact your device manufacturer for details.
* 1GB = 1,000,000,000 bytes. Actual user storage less.
** Up to 48MB/sec read speed. Write speed lower. Based on internal testing; performance may be lower depending on host device, interface, usage conditions and other factors.
† Compatible device required. Full HD (1920x1080) video support may vary based upon host device, file attributes and other factors. See www.sandisk.com/HD.
‡ Card only. Water proof, Shock proof, Temperature proof, X-ray proof. www.sandisk.com/proof

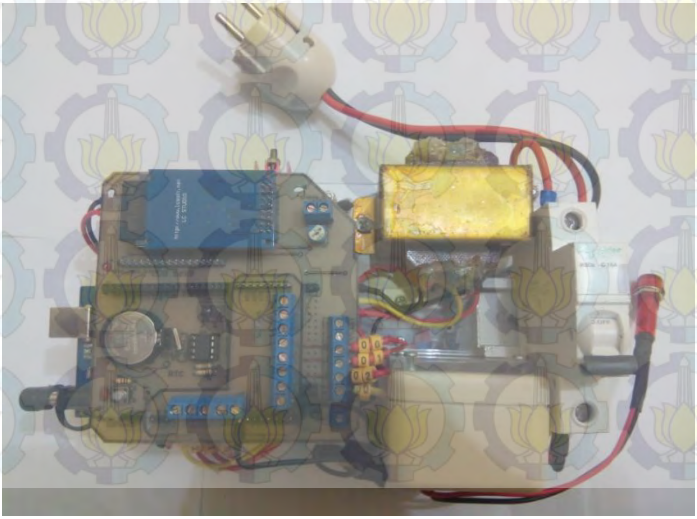
SanDisk and the SanDisk logo are trademarks of SanDisk Corporation, registered in the United States and other countries. SanDisk Ultra is a trademark of SanDisk Corporation. The microSDHC and microSDXC marks and logos are trademarks of SD-3C, LLC. Other brand names mentioned herein are for identification purposes only and may be the trademarks of their respective holder(s).

LAMPIRAN C

Realisasi Alat



Rangkaian Sensor dan *Power Supply*



Rangkaian Keseluruhan

Realisasi Kontrol Panel

Pengujian Alat pada Beban *Drill Press*

Pengujian Alat pada Beban Gerinda tangan

Pengujian Alat pada Beban Las Listrik

RIWAYAT HIDUP PENULIS

Nama	: Adhitya Wisnu W.
TTL	: Mojokerto, 29 Desember 1993
Jenis Kelamin	: Pria
Agama	: Islam
Alamat Rumah	: Jalan Lawu Raya Wates Mojokerto
Telp/HP	: 085655300273
E-mail	adhit23wewe@gmail.com
Hobi	: Membaca, Olahraga

RIWAYAT PENDIDIKAN

1999 – 2001	: TK Dharma Wanita
2001 – 2007	: SDN Wates 1 Mojokerto
2007 – 2010	: SMPN 1 Mojokerto
2010 – 2013	: SMAN 1 Sooko Mojokerto
2013 – 2016	: Bidang Studi Teknik Listrik, Program D3 Teknik Elektro, ITS - PLN

PENGALAMAN KERJA

- *On The Job Training* PT. PLN (Persero) Rayon Darmo Permai

RIWAYAT HIDUP PENULIS

Nama	: Faisal Akhbar
TTL	: Probolinggo, 11 Mei 1995
Jenis Kelamin	: Pria
Agama	: Islam
Alamat Rumah	: Dsn. Krajan Desa Sumberanyar RT 024 RW 07 Kec. Paiton, Kab. Probolinggo
Telp/HP	: 085746171287
E-mail	: faisalakhbar@yahoo.com
Hobi	: *Traveling*

RIWAYAT PENDIDIKAN

1999 – 2001	: TK Eka Prastiwi
2001 – 2007	: SDN Sukodadi 1
2007 – 2010	: SMP Bhakti Pertiwi
2010 – 2013	: SMA Tunas Luhur
2013 – 2016	: Bidang Studi Teknik Listrik, Program D3 Teknik Elektro, ITS - PLN

PENGALAMAN KERJA

- *On The Job Training* PT. PLN (Persero) Rayon Darmo Permai