# Analisa Perhitungan Data

a. Rata-rata

1. Frekuensi 20 Hz

   Rata-rata = $\frac{20{,}68 + 20{,}28 + 20{,}81 + 20{,}89 + 20{,}11}{5} = 20{,}554$ Hz

2. Frekuensi 120 Hz

   Rata-rata = $\frac{120{,}09 + 120{,}55 + 120{,}58 + 120{,}84 + 122{,}91}{5} = 120{,}994$ Hz

3. Frekuensi 125 Hz

   Rata-rata = $\frac{125{,}24 + 125{,}58 + 125{,}85 + 125 + 125{,}49}{5} = 125{,}432$ Hz

4. Frekuensi 250 Hz

   Rata-rata = $\frac{257{,}35 + 258{,}36 + 257{,}73 + 256{,}6 + 258{,}96}{5} = 257{,}8$ Hz

5. Frekuensi 500 Hz

   Rata-rata = $\frac{512{,}14+507{,}88+518{,}77+525{,}52+514{,}42}{5} = 517{,}546$ Hz

6. Frekuensi 1000 Hz

   Rata-rata = $\frac{1030+1022+1025+1019+1014}{5} = 1022$ Hz

7. Frekuensi 2500 Hz

   Rata-rata = $\frac{2583+2596+2599+2590+2587}{5} = 2591$ Hz

8. Frekuensi 4000 Hz

   Rata-rata = $\frac{4006+4015+4036+4018+4080}{5} = 4031$ Hz

9. Frekuensi 8000 Hz

   Rata-rata = $\frac{8204+8210+8215+8212+8208}{5} = 8209{,}8$ Hz

10. Frekuensi 15000 Hz

    Rata-rata = $\frac{14507+15302+14289+14160+14623}{5} = 14576{,}2$ Hz

b. Simpangan

1. Frekuensi 20 Hz

   Simpangan = 20,554 – 20 = 0,554

2. Frekuensi 120 Hz

   Simpangan = 120,994 – 120 = 0,994

3. Frekuensi 125 Hz

   Simpangan = 125,432 – 125 = 0,432

4. Frekuensi 250 Hz

   Simpangan = 257,8 – 250 = 7,8

5. Frekuensi 500 Hz

   Simpangan = 517,546 – 500 = 17,546

6. Frekuensi 1000 Hz

   Simpangan = 1022– 1000 = 22

7. Frekuensi 2500 Hz

   Simpangan = 2591 – 2500 = 91

8. Frekuensi 4000 Hz

   Simpangan = 4031 – 4000 = 31

9. Frekuensi 8000 Hz

   Simpangan = 8209,8 – 8000 = 209,8

10. Frekuensi 15000 Hz

    Simpangan = 15000 – 14576,2 = 423,8

*c. Error*

1. Frekuensi 20 Hz

   $\%\ Error = \frac{0,554}{20,554} \times 100\ \% = 3\%$

2. Frekuensi 120 Hz

   $\%\ Error = \frac{0,994}{120,994} \times 100\ \% = 1\%$

3. Frekuensi 125 Hz

   $\%\ Error = \frac{0,432}{125,432} \times 100\ \% = 0\%$

4. Frekuensi 250 Hz

$$\%\ Error = \frac{7{,}8}{257{,}8} \times 100\ \% = 3\%$$

5. Frekuensi 500 Hz

$$\%\ Error = \frac{17{,}546}{517{,}546} \times 100\ \% = 3\%$$

6. Frekuensi 1000 Hz

$$\%\ Error = \frac{22}{1022} \times 100\ \% = 2\%$$

7. Frekuensi 2500 Hz

$$\%\ Error = \frac{91}{2591} \times 100\ \% = 4\%$$

8. Frekuensi 4000 Hz

$$\%\ Error = \frac{31}{4031} \times 100\ \% = 1\%$$

9. Frekuensi 8000 Hz

$$\%\ Error = \frac{209{,}8}{8209{,}8} \times 100\ \% = 3\%$$

10. Frekuensi 15000 Hz

$$\%\ Error = \frac{423{,}8}{14576{,}2} \times 100\ \% = 3\%$$

# LISTING PROGRAM

```
//Chip type               : ATmega16/32
//AVR Core Clock frequency: 8,000000 MHz


#include <mega16.h>   // mega 16/32 //library
#include <stdio.h> // library untuk print, put char
#include <delay.h> // library untuk jeda


// Alphanumeric LCD functions
#include <alcd.h> // untuk LCD 2x16


#define ADC_VREF_TYPE 0x40 // telah tersedia pada saat menyeting
program menggunakan 10 bit


#define potensio 0 // pada pin ADC 0


// stepper // perintah memutar motor
// 288YJ-48
// 1 putaran=4096 // data


#define IN1 PORTC.2 // pin stepper
#define IN2 PORTC.3 // pin stepper
#define IN3 PORTC.4 // pin stepper
#define IN4 PORTC.5 // pin stepper


#define equalizer1  PORTD.0 // pin equalizer relay pada portd.0
#define equalizer2  PORTD.1 // pin equalizer relay pada portd.1
```

```
#define equalizer3  PORTD.2 // pin equalizer relay pada portd.2

#define equalizer4  PORTD.3 // pin equalizer relay pada portd.3

#define equalizer5  PORTD.4 // pin equalizer relay pada portd.4

#define equalizer6  PORTD.5 // pin equalizer relay pada portd.5

#define equalizer7  PORTD.6 // pin equalizer relay pada portd.6

#define equalizer8  PORTD.7 // pin equalizer relay pada portd.7

#define equalizer9  PORTB.7 // pin equalizer relay pada portb.7

#define equalizer10 PORTB.6 // pin equalizer relay pada portb.6


#define mp3      PORTC.1 // pin suara

#define balance  PORTC.0 // pin balance untuk menyeimbangkan dari
headphone


#define ledkiri   PORTC.6 // pin led kiri

#define ledkanan  PORTC.7 // pin led kanan


#define volup     PINA.1 // pin tombol volume up

#define voldown   PINA.2 // pin tombol volume down

#define frequp    PINA.3 // pin tombol frequensi up

#define freqdown  PINA.4 // pin tombol frequensi down

#define start     PINA.5 // pin tombol start

#define skiri     PINA.6 // pin tombol pengaturan headphone kiri

#define skanan    PINA.7 // pin tombol pengaturan headphone kanan


unsigned char
data_out[]={0b1000,0b1100,0b0100,0b0110,0b0010,0b0011,0b0001,0b100
1}; // untuk mengolah data out kode kode dari stepper

char buffer[33]; // menampilkan desible pada LCD
```

```
unsigned char db_out[6]={10,20,30,40,50,60};   //db // mapping
untuk menampilkan nilai desible pada LCD

unsigned int db_map[6]={0,10,20,40,50,60};  //adc

unsigned int
freq_out[10]={20,120,125,250,500,1000,2500,4000,8000,15000};  //hz
// mapping untuk menampilkan nilai frequensi pada LCD


int db=0,freq=0,hz,headphone,mp3on; //variabel yang digunakan
untuk kondisi dari desible, frequensi, headphone dan MP3

float khz;


// Read the AD conversion result // fungsi dengan parameter dengan
nilai balik

unsigned int read_adc(unsigned char adc_input) // nilai balik

{

ADMUX=adc_input | (ADC_VREF_TYPE & 0xff);

// Delay needed for the stabilization of the ADC input voltage

delay_us(10);

// Start the AD conversion

ADCSRA|=0x40;

// Wait for the AD conversion to complete

while ((ADCSRA & 0x10)==0);

ADCSRA|=0x10;

return ADCW;

}

// Declare your global variables here


void putarkiri(int speed){ // fungsi untuk memutar motor stepper
putar kiri

    int i;

    for(i=0;i<8;i++){
```

```
    IN1=data_out[i]&0b0001;
    IN2=data_out[i]&0b0010;
    IN3=data_out[i]&0b0100;
    IN4=data_out[i]&0b1000;
    delay_ms(speed); // max 1000hz (1ms)
    }
    IN1=0;
    IN2=0;
    IN3=0;
    IN4=0;
}
void putarkanan(int speed){ // fungsi untuk memutar motor stepper
putar kanan
    int i;
    for(i=7;i>=0;i--){
    IN1=data_out[i]&0b0001;
    IN2=data_out[i]&0b0010;
    IN3=data_out[i]&0b0100;
    IN4=data_out[i]&0b1000;
    delay_ms(speed); // max 1000hz (1ms)
    }
    IN1=0;
    IN2=0;
    IN3=0;
    IN4=0;
}
void stop(){ //fungsi untuk memberhentikan motor stepper
    IN1=0;
```

```
    IN2=0;
    IN3=0;
    IN4=0;
}
void zerodb(){ // pada saat posisi dinyalakan pertama kali dalam
kondisi 0
int adc=0,i;
adc=read_adc(potensio);
i=adc;
lcd_clear();
while(1){
adc=read_adc(potensio);
lcd_gotoxy(0,0);
sprintf(buffer,"POS:%04d",adc);
lcd_puts(buffer);
if(adc>db_map[db]){
putarkiri(1); // memutar kekiri sampai nilai adc nya 0
}
else {
stop();
break;
}
}
}
void program_audio(){
int adc=0,i;
adc=read_adc(potensio); // membaca potensio
```

```
if(volup==0){ //tombol volume up
lcd_clear();
db++;
if(db>11)db=11;
i=adc;
while(1){
adc=read_adc(potensio);
lcd_gotoxy(0,0);
sprintf(buffer,"POS:%04d",adc);
lcd_puts(buffer);
if(adc<db_map[db]){
putarkanan(1);
}
else {
stop();
break;
}
}
}
if(voldown==0){ // tombol volume down
lcd_clear();
db--;
if(db<0)db=0;
i=adc;
while(1){
adc=read_adc(potensio);
lcd_gotoxy(0,0);
sprintf(buffer,"POS:%04d",adc);
```

```
lcd_puts(buffer);

if(adc>db_map[db]){

putarkiri(1);

}

else {

stop();

break;

}

}

}

if(start==0)mp3on=1; // ketika di tekan tombol start, suara
menyala

if(mp3on==1)mp3=0; // on dan off suara


if(skanan==0)headphone=1; // on dan off relay untuk headphone
kanan

if(skiri==0)headphone=0; // on dan off relay untuk headphone kiri


if(headphone==0){ledkiri=1; ledkanan=0; balance=1; }

else {ledkiri=0; ledkanan=1;  balance=0; }


if(frequp==0){ // frequensi up

freq++;

if(freq>9)freq=9;

}

if(freqdown==0){ // frequensi down

freq--;

if(freq<0)freq=0;

}
```

```
if(freq_out[freq]>=1000)khz=(float)freq_out[freq]/1000;
else hz=freq_out[freq];

if(freq_out[freq]==20){ // kondisi relay untuk frequensi 20 Hz
equalizer1=0;
equalizer2=1;
equalizer3=1;
equalizer4=1;
equalizer5=1;
equalizer6=1;
equalizer7=1;
equalizer8=1;
equalizer9=1;
equalizer10=1;
}
if(freq_out[freq]==120){ // kondisi relay untuk frequensi 120 Hz
equalizer1=1;
equalizer2=0;
equalizer3=1;
equalizer4=1;
equalizer5=1;
equalizer6=1;
equalizer7=1;
equalizer8=1;
equalizer9=1;
equalizer10=1;
}
if(freq_out[freq]==125){ // kondisi relay untuk frequensi 125 Hz
```

```
equalizer1=1;
equalizer2=1;
equalizer3=0;
equalizer4=1;
equalizer5=1;
equalizer6=1;
equalizer7=1;
equalizer8=1;
equalizer9=1;
equalizer10=1;
}
if(freq_out[freq]==250){ // kondisi relay untuk frequensi 250 Hz
equalizer1=1;
equalizer2=1;
equalizer3=1;
equalizer4=0;
equalizer5=1;
equalizer6=1;
equalizer7=1;
equalizer8=1;
equalizer9=1;
equalizer10=1;
}
if(freq_out[freq]==500){ // kondisi relay untuk frequensi 500 Hz
equalizer1=1;
equalizer2=1;
equalizer3=1;
equalizer4=1;
```

```
equalizer5=0;
equalizer6=1;
equalizer7=1;
equalizer8=1;
equalizer9=1;
equalizer10=1;
}
if(freq_out[freq]==1000){ // kondisi relay untuk frequensi 1000 Hz
equalizer1=1;
equalizer2=1;
equalizer3=1;
equalizer4=1;
equalizer5=1;
equalizer6=0;
equalizer7=1;
equalizer8=1;
equalizer9=1;
equalizer10=1;
}
if(freq_out[freq]==2500){ // kondisi relay untuk frequensi 2500 Hz
equalizer1=1;
equalizer2=1;
equalizer3=1;
equalizer4=1;
equalizer5=1;
equalizer6=1;
equalizer7=0;
equalizer8=1;
```

```
equalizer9=1;
equalizer10=1;
}
if(freq_out[freq]==4000){ // kondisi relay untuk frequensi 4000 Hz
equalizer1=1;
equalizer2=1;
equalizer3=1;
equalizer4=1;
equalizer5=1;
equalizer6=1;
equalizer7=1;
equalizer8=0;
equalizer9=1;
equalizer10=1;
}
if(freq_out[freq]==8000){ // kondisi relay untuk frequensi 8000 Hz
equalizer1=1;
equalizer2=1;
equalizer3=1;
equalizer4=1;
equalizer5=1;
equalizer6=1;
equalizer7=1;
equalizer8=1;
equalizer9=0;
equalizer10=1;
}
```

```
if(freq_out[freq]==15000){ // kondisi relay untuk frequensi 1500
Hz
equalizer1=1;
equalizer2=1;
equalizer3=1;
equalizer4=1;
equalizer5=1;
equalizer6=1;
equalizer7=1;
equalizer8=1;
equalizer9=1;
equalizer10=0;
}
lcd_clear();
lcd_gotoxy(0,0);
sprintf(buffer,"VOL:%d db",db_out[db]);
lcd_puts(buffer);
lcd_gotoxy(13,0);
if(mp3on==1)lcd_putsf("ON");
else lcd_putsf("OFF");
if(freq_out[freq]<1000) {
lcd_gotoxy(0,1);
sprintf(buffer,"FREQ:%d Hz",hz);
lcd_puts(buffer);
}
else {
lcd_gotoxy(0,1);
sprintf(buffer,"FREQ:%.1f KHz",khz);
```

```
lcd_puts(buffer);

}

lcd_gotoxy(13,1);

if(headphone==1)lcd_putsf("R");

else lcd_putsf("L");

delay_ms(100);

}




void main(void)

{

// Declare your local variables here


// Input/Output Ports initialization

// Port A initialization

// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
Func0=In

// State7=P State6=P State5=P State4=P State3=P State2=P State1=P
State0=T

PORTA=0xFE;

DDRA=0x00;


// Port B initialization

// Func7=Out Func6=Out Func5=In Func4=In Func3=In Func2=In
Func1=In Func0=In

// State7=0 State6=0 State5=T State4=T State3=T State2=T State1=T
State0=T

PORTB=0x00;

DDRB=0xC0;
```

```
// Port C initialization

// Func7=Out Func6=Out Func5=Out Func4=Out Func3=Out Func2=Out
Func1=Out Func0=Out

// State7=0 State6=0 State5=0 State4=0 State3=0 State2=0 State1=0
State0=0

PORTC=0x00;

DDRC=0xFF;


// Port D initialization

// Func7=Out Func6=Out Func5=Out Func4=Out Func3=Out Func2=Out
Func1=Out Func0=Out

// State7=0 State6=0 State5=0 State4=0 State3=0 State2=0 State1=0
State0=0

PORTD=0x00;

DDRD=0xFF;


// Timer/Counter 0 initialization

// Clock source: System Clock

// Clock value: Timer 0 Stopped

// Mode: Normal top=0xFF

// OC0 output: Disconnected

TCCR0=0x00;

TCNT0=0x00;

OCR0=0x00;


// Timer/Counter 1 initialization

// Clock source: System Clock

// Clock value: Timer1 Stopped

// Mode: Normal top=0xFFFF
```

```
// OC1A output: Discon.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=0x00;
TCCR1B=0x00;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

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

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

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

// USART initialization
// USART disabled
UCSRB=0x00;

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

// ADC initialization
// ADC Clock frequency: 1000,000 kHz
// ADC Voltage Reference: AVCC pin
ADMUX=ADC_VREF_TYPE & 0xff;
ADCSRA=0x83;

// SPI initialization
```

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

// TWI initialization
// TWI disabled
TWCR=0x00;

// Alphanumeric LCD initialization
// Connections are specified in the
// Project|Configure|C Compiler|Libraries|Alphanumeric LCD menu:
// RS - PORTB Bit 0
// RD - PORTB Bit 6
// EN - PORTB Bit 1
// D4 - PORTB Bit 2
// D5 - PORTB Bit 3
// D6 - PORTB Bit 4
// D7 - PORTB Bit 5
// Characters/line: 16
lcd_init(16);

equalizer1=1; // kondisi relay off
equalizer2=1; // kondisi relay off
equalizer3=1; // kondisi relay off
equalizer4=1; // kondisi relay off
equalizer5=1; // kondisi relay off
equalizer6=1; // kondisi relay off
equalizer7=1; // kondisi relay off
equalizer8=1; // kondisi relay off
```

```
equalizer9=1; // kondisi relay off
equalizer10=1; // kondisi relay off
mp3=1;
balance=1;

zerodb(); // untuk posisi awal 0

lcd_clear();
lcd_gotoxy(0,0);
lcd_putsf("AUDIO METRY"); // tampilan awal pada LCD

delay_ms(1500); // waktu jeda
lcd_clear();
while (1)
      {
      // Place your code here
      program_audio(); // untuk memanggil program
      }
}
```