2. LANDASAN TEORI

2.1. *Flowchart* Diagram

2.1.1. Definisi *Flowchart*

Flowchart adalah model yang digunakan dalam menggambarkan suatu sistem yang ada dengan lebih nyata. Dengan *flowchart* akan lebih diketahui kapan dan dimana sebuah aktivitas dilakukan, disimpan, serta digunakan lagi oleh sebuah sistem.

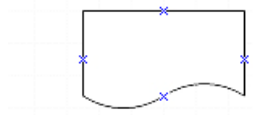
2.2.2. Simbol-simbol *Flowchart*

Simbol-simbol *flowchart* secara garis besar dapat dikelompokkan menjadi beberapa bagian, antara lain:

a. Simbol *Input/Output*

Simbol yang termasuk dalam kelompok ini adalah simbol-simbol yang digunakan untuk melambangkan proses penginputan dan pencetakan output yang dilakukan oleh suatu sistem. Daftar simbol-simbol *input/output* dapat dilihat pada tabel 2.1.

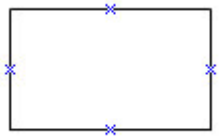
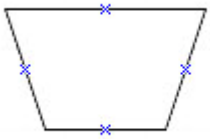
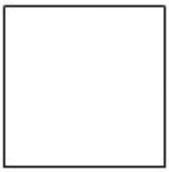
Tabel 2.1. Tabel simbol *Input/Output*

Simbol	Nama Simbol	Keterangan
	*Document*	Digunakan untuk menunjukkan dokumen atau laporan
	*Input/Output*	Digunakan untuk menunjukkan data yang dimasukkan ke proses atau data yang dihasilkan

b. Simbol Proses

Simbol-simbol yang termasuk dalam kelompok ini merupakan simbol-simbol yang digunakan unuk melambangkan suatu proses yang digunakan oleh suatu sistem. Daftar simbol-simbol proses dapat dilihat pada tabel 2.2.

Tabel 2.2 Tabel simbol Proses

Simbol	Nama Simbol	Keterangan
	Proses Komputer	Digunakan untuk menunjukkan proses yang dilakukan dengan/oleh komputer
	Proses Manual	Digunakan untuk menunjukkan proses yang dilakukan secara manual.
	*Auxiliary* Prosess	Digunakan untuk menunjukkan proses yang dilakukan dengan bantuan alat.

c. Simbol *Storage*

Simbol-simbol yang termasuk dalam kelompok ini merupakan simbol-simbol yang digunakan unuk melambangkan suatu media penyimpanan yang digunakan oleh suatu sistem. Daftar simbol-simbol s*torage* dapat dilihat pada tabel 2.3.

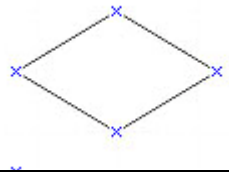
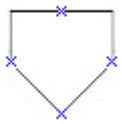
Tabel 2.3. Tabel simbol *Storage*

Simbol	Nama Simbol	Keterangan
	*Magnetic Disk*	Digunakan untuk menunjukkan data yang disimpan pada *Magnetic disk.*
	*Magnetic Tape*	Digunakan untuk menunjukkan data yang disimpan pada *Megnetic tape.*

d. Simbol *Flow*

Simbol-simbol yang termasuk dalam kelompok ini merupakan simbol-simbol yang digunakan unuk melambangkan aliran yang terjadi dalam suatu sistem. Daftar simbol-simbol *flow* dapat dilihat pada tabel 2.4.

Tabel 2.4 Tabel simbol *Flow*

Simbol	Nama Simbol	Keterangan
	Terminal	Digunakan untuk menunjukkan awal, akhir, dan interupsi dalam sistem.
	*Decision*	Digunakan untuk menunjukkan pengambilan keputusan
	*On Page Connector*	Digunakan untuk menghubungkan proses dalam satu halaman yang sama
	*Off Page Connector*	Digunakan untuk menghubungkan proses apabila berganti halaman
	Document/Processing Flow	Digunakan untuk menunjukkan arah aliran dokumen atau proses

2.2. *Discrete Fourier Transform* (DFT)

Semua sinyal periodic terbentuk dari gabungan sinyal-sinyal sinusoide yang menjadi satu dimana dalam perumusannya dapat ditulis:

$$F(t) = \sum A[k].e^{j.2.\pi.\ k.Fo.t} \qquad 0 <= t < tn,\ t\pi \qquad (2.2.1)$$

K = ~

keterangan :

F[t] = Fungsi time domain sebuah sinyal

A[k] = Amplitudo frekuensi untuk frekuensi k.Fo

Fo = Frekuensi fundamental/dasar

T = Waktu

Nilai euler dapat dijabarkan sebagai berikut:

$$e^{j.2.\pi.\ k.Fot} = \cos 2.\pi.k.Fo.t + j.\ \sin 2.\pi.k.Fo.t \qquad (2.2.2)$$

sehingga dapat ditulis bahwa nilai amplitudo untuk suatu frekuensi adalah:

$$A[k] = (1/Tp)\ .\ \int Xt.\ e^{j.2.\pi.\ k.Fot}\ dt \qquad (2.2.3)$$

keterangan:

Tp = Periode sinyal

A[k] = Amplitudo frekuensi untuk frekuensi Fs. k

Fs.k = Frekuensi sinyal

Dalam pemrosesan komputer, sinyal-sinyal kontinyu akan tersebut akan disampling ke dalam bentuk sinyal diskret dengan kecepatan sampling tertentu yang lebih umum dikenal sebagai sinyal digital. Analisa spektrum untuk siyal digital menggunakan metode *discrete fourier transform* (DFT) untuk mengubah data sinyal pada time domain ke dalam data sinyal pada frekuensi domain.

Perumusan DFT dapat digambarkan sebagai berikut:

$$X(k) = \sum_{n=0}^{N-1} x(n)\, W_N^{k.n} \qquad k = 0,1,2,\ldots,N-1 \qquad (2.2.4)$$

$W_N = e^{-j.2.\pi./N.}$

Keterangan:

X[k] = Data sinyal pada frekuensi kFo

x(n) = Data sinyal pada time domain

N = Jumlah sampel

Untuk menemukan spektrum sebuah sinyal dengan DFT diperlukan N buah sample data yang berurutan pada *time domain* yaitu dari x[m] sampai dengan x[m+N-1], dan dari N data tersebut akan ditemukan N buah frekuensi pembentuk sinyal yang sesuai.

2.3. Fast Fourier Transform metode Butterfly radix-2

Proses perhitungan secara langsung dalam komputerisasi dapat menyebabkan proses berjalan cukup lama karena dalam DFT terdapat N^2 perkalian bilangan kompleks. Dalam hal ini untuk mempercepat proses perhitungan maka dipakailah metode *Fast Fourier Transform (FFT).* Dasar dari perhitungan *Fast Fourier Transform* adalah menghilangkan proses perhitungan kembar yang ada di dalam DFT. Salah satu metode dari komputasi *Fast Fourier Transform* yang ada ialah komputasi dengan menggunakan metode *FFT Butterfly radix-2*. syarat khusus dari komputasi dengan menggunakan metode *FFT Butterfly radix-2* ini adalah bahwa jumlah data harus memenuhi bilangan 2^n buah, dalam metode *FFT Butterfly radix-2* terdapat ($N/2 \log_2 N$) buah perkalian bilangan kompleks.

Karena jumlah perkalian kompleks yang terjadi dalam *FFT Butterfly radix-2* jauh lebih sedikit daripada jumpah perkalian dalam metode DFT secara langsung, maka proses komputerisasi dengan menggunakan metode *FFT Butterfly radix-2* akan lebih cepat daripada menggunakan metode DFT secara langsung.

2.3.1 Algoritma FFT *Butterfly Radix-2*

Algoritma *FFT Butterfly Radix-2* menggunakan $N=2^v$ *point DFT* berdasarkan pendekatan membagi dan menyelesaikan. Dalam hal ini N-point data diagi menjadi dua N/2 – point data dengan urutan $f_1(n)$ dan $f_2(n)$, berdasarkan bilangan ganjil dan genap dari x(n), yaitu:

$$f_1(n) = x(2n)$$
$$f_2(n) = x(2n+1), \qquad n = 0,1,\ldots,(N/2) -1 \qquad (2.3.1.1)$$

Jadi, $f_1(n)$ dan $f_2(n)$ diperoleh dengan menguraikan x(n) dengan faktor dua. Sekarang DFT N titik dapat dinyatakan dari segi DFT pada deret yang diuraikan sebagai berikut:

$$X(k) = \sum_{n=0}^{N-1} x(n)\, W_N^{kn}, \qquad k = 0,1,\ldots$$
$$= \sum_{n\ even} x(n) W_N^{kn} + \sum_{n\ odd} x(n) W_N^{kn}$$
$$= \sum_{m=0}^{(N/2)-1} x(2m)\, W_N^{2mk} + \sum_{m=0}^{\left(\frac{N}{2}\right)-1} x(2m+1)\, W_N^{2(m+1)k} \qquad (2.3.1.2)$$

Tetapi $W^2_N = W_{N/2}$ .Dengan subsitusi ini maka dapat dinyatakan sebagai:

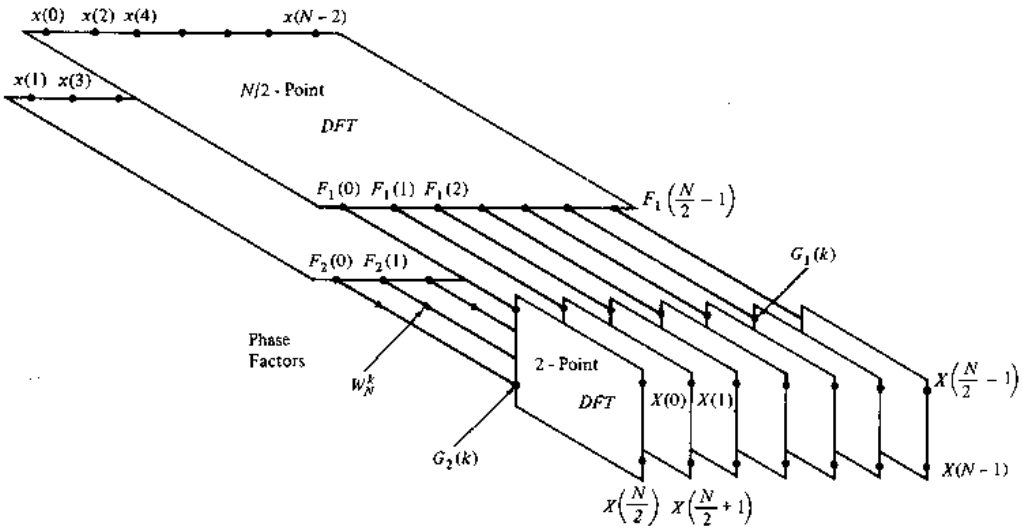
$$X(k) = \sum_{m=0}^{(N/2)-1} f_1(m)\, W_{N/2}^{km} + \sum_{m=0}^{(N/2)-1} f_2(m)\, W_{N/2}^{km}$$
$$= F_1(k) + W_N^k\, F_2(k), \qquad k = 0,1,\ldots,N-1 \qquad (2.3.1.3)$$

Dengan $F_1(k)$ dan $F_2(k)$ adalah DFT *N/2*-titik dari berturut-turut deret $f_1(m)$ dan $f_2(m)$. Karena $F_1(k)$ dan $F_2(k)$ periodik dengan periode *N/2*, maka didapatkan $F_1(k+N/2) = F_1(k)$ dan $F_2(k+N/2) = F_2(k)$. Sebagai tambahan, faktor $W_N^{k+N/2} = W_N^k$. Karena itu dapat dinyatakan sebagai:

$$X(k) = F_1(k) + W_N^k F_2(k) \qquad k = 0,1, \ldots, \frac{N}{2} - 1$$

$$X\left(k + \frac{N}{2}\right) = F_1(k) + W_N^k F_2(k) \qquad k = 0,1, \ldots, \frac{N}{2} - 1 \qquad (2.3.1.4)$$

Dapat diamati bahwa komputasi langsung *$F_1(k)$* membutuhkan perkalian kompleks $(N/2)^2$. Pemakaian yang sama terhadap komputasi $F_2(k)$. Selanjutnya terdapat perkalian kompleks tambahan *N/2* dibutuhkan untuk menghitung $W_N^k F_2(k)$. Karena itu komputasi $X(k)$ membutuhkan erkalian kompleks $2(N/2)^2 + N/2 = N^2/2 + N/2$. Langkah pertama ini menghasilkan reduksi jumlah perkalian dari N^2 hingga $N^2/2 + N/2$, yang kira-kira adalah faktor 2 untuk *N* besar.



Gambar 2.1. Langkah pertama penguraian algoritma dalam waktu

Sumber: Prolaksis (1995, p.437)

Dengan menghitung DFT *N/4*-titik, dapat diperoleh DFT *N/2*-titik *$F_1(k)$ dan F2(k)* dari hubungan-hubungan

$$F_1(k) = F\{f_1(2n)\} + W_{N/2}^k F\{f_1(2n+1)\}, k = 0,1,\ldots,\frac{N}{4}-1 \quad n = 0,1,\ldots,\frac{N}{4}-1$$

$$F_1\left(k+\frac{N}{4}\right) = F\{f_1(2n)\} + W_{N/2}^k F\{f_1(2n+1)\}, k = 0,1,\ldots,\frac{N}{4}-1 \quad n = 0,1,\ldots,\frac{N}{4}-1 \qquad (2.3.1.5)$$

$$F_2(k) = F\{f_2(2n)\} + W_{N/2}^k F\{f_2(2n+1)\}, k = 0,1,\ldots,\frac{N}{4}-1 \quad n = 0,1,\ldots,\frac{N}{4}-1$$

$$F_2\left(k+\frac{N}{4}\right) = F\{f_2(2n)\} + W_{N/2}^k F\{f_2(2n+1)\}, k = 0,1,\ldots,\frac{N}{4}-1 \quad n = 0,1,\ldots,\frac{N}{4}-1 \qquad (2.3.1.6)$$

F(x) represent Fourier Transform

Penguraian data diatas dapat diulang lagi hingga deret-deret yang dihasilkan dikurang dengan deret-deret satu titik. Untuk N = 2^v, penguraian ini dapat dilakukan v = $\log_2 N$ kali. Jadi jumlah total perkalian kompleks dalam *FFT butterfly radix-2* berkurang menjadi $(N/2)\log_2 N$ dibandingkan dengan komputasi DFT secara langsung. Penjumlahan bilangan kompleksnya adalah $N\log_2 N$.

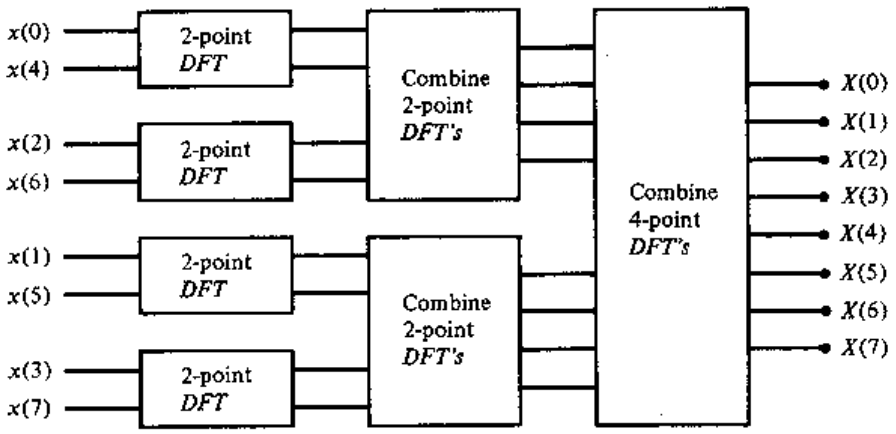
Tabel 2.5 menggambarkan perbandingan metode DFT secara langsung dengan *FFT Butterfly radix-2*.

Tabel 2.5. Tabel perbandingan kecepatan komputasi FFT dan non FFT

Jumlah data (N)	Perkalian kompleks (dalam DFT)	Perkalian kompleks (dalam FFT)	Faktor penambah kecepatan
4	16	4	4
8	64	12	5.3
16	256	32	8.0
32	1024	80	12.8
64	4096	192	21.3
128	16384	366	44.8
256	65536	1024	64
512	264144	2304	113.8

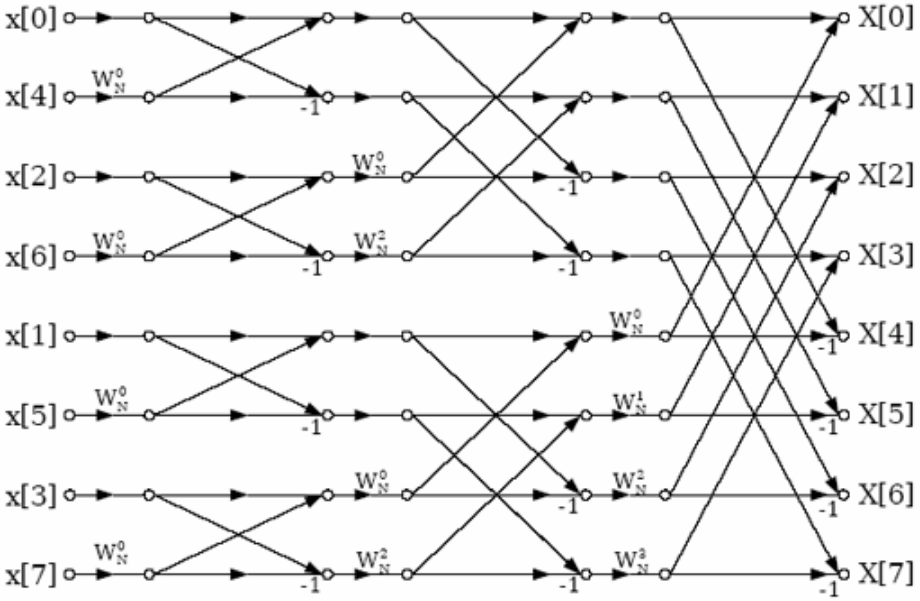
Sumber: Prolaksis (1995, p.438)

Sebagai ilustrasi, pada gambar dibawah ini memperlihatkan komputasi DFT N = 8 titik. Komputasi tersebut dilakukan dalam tiga tahap, dimulai dengan komputasi empat DFT dua titik, kemudian dua DFT empat titik, sampai pada akhirnya satu DFT delapan titik, dimana kombinasi DFT yang lebih kecil membentuk DFT yang lebih besar.



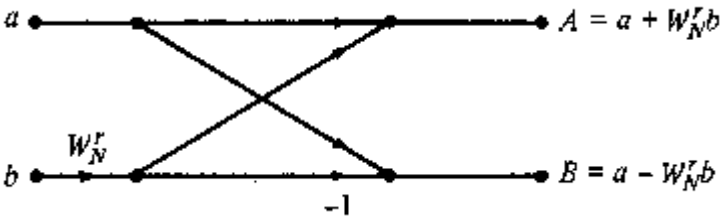
Gambar 2.2. Tiga tahap komputasi untuk DFT dengan *N*=*8*-titik.

Sumber: Prolaksis (1995, p.439)



Gambar 2.3. Algoritma FFT penguraian waktu *8*-titik.

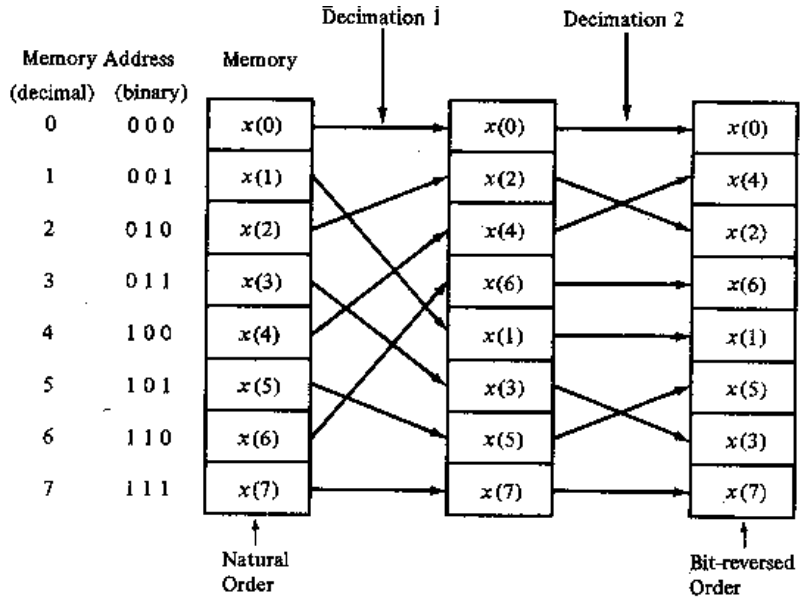
Sumber: Prolaksis (1995, p.439)



Gambar 2.4. Perhitungan kupu-kupu dasar pada algoritma penguraian waktu FFT

Sumber: Prolaksis (1995, p.440)

Pengamatan penting kedua diperlihatkan dengan orde deret data masukan setelah diuraikan (v-1) kali. Sebagai contoh, jika melihat kasus dengan jumlah data N = 8, akan diketahui bahwa pernguraian pertama menghasilkan deret x(0), x(2), x(4), x(6), x(1), x(3), x(5), x(7). Dan penguraian kedua menghasilkan deret x(0), x(4), x(2), x(6), x(1), x(5), x(3), x(7). Pergantian deret data masukan ini mempunyai suatu orde terdefinisi yang baik seperti yang dapat ditegaskan dari pengamatan Gambar 2.14. yang mengilustrasikan penguraian deret delapan titik.



(a)

$(n_2 n_1 n_0)$	→	$(n_0 n_2 n_1)$	→	$(n_0 n_1 n_2)$
(0 0 0)	→	(0 0 0)	→	(0 0 0)
(0 0 1)	→	(1 0 0)	→	(1 0 0)
(0 1 0)	→	(0 0 1)	→	(0 1 0)
(0 1 1)	→	(1 0 1)	→	(1 1 0)
(1 0 0)	→	(0 1 0)	→	(0 0 1)
(1 0 1)	→	(1 1 0)	→	(1 0 1)
(1 1 0)	→	(0 1 1)	→	(0 1 1)
(1 1 1)	→	(1 1 1)	→	(1 1 1)

(b)

Gambar 2.5. Penggantian data dan bit

Sumber: Prolaksis (1995, p.437)

Pada Gambar 2.5, proses perhitungan FFT Butterfly radix2 dimulai dengan pengurutan data FFT yang disesuaikan dengan jumlah sampel data yang digunakan. Urutan data tersebut merupakan pencerminan dari bilangan binary dari urutan data yang bersangkutan. Misalnya pada pemakaian frekuensi sample = 8 data maka urutan data yang dipakai adalah:

Tabel 2.6. Pengurutan data dalam FFT Radix-2

Urutan dalam FFT (Decimal)	Urutan dalam FFT (Hexadecimal)	Data(n) n = urutan data dalam time domain	Pencerminan urutan dalam FFT (Hexadecimal)
0	000	0	000
1	001	4	100
2	010	2	010
3	011	6	110
4	100	1	001
5	101	5	101
6	110	3	011
7	111	7	111

Algoritma FFT radiks-2 penting lainnya, yang dinamakan algoritma penguraian dalam frekuensi (*decimation-in-frequency algorithm*), diperoleh dengan menggunakan pendekatan membagi dan menyelesaikan. Untuk menurunkan algoritma tersebut, dimulai dengan menguraikan rumus DFT menjadi 2 penjumlahan, salah satu yang meliputi jumlah melalui titik-titik data *N/2* pertama dan penjumlahan kedua meliputi titik-titik data *N/2* terakhir. Dari penguraian rumus DFT tersebut akan diperoleh.

$$X(k) = \sum_{n=0}^{(N/2)-1} x(n)\, W_N^k + \sum_{n=0}^{(N/2)-1} x(n)\, W_N^k$$

$$= \sum_{n=0}^{(N/2)-1} x(n)\, W_N^k + \sum_{n=0}^{(N/2)-1} x\left(n+\frac{N}{2}\right) W_N^k \qquad (2.3.1.7)$$

Since $W_N^{kN/2} = (-1)^k$

$$X(k) = \sum_{n=0}^{(N/2)-1} \left[x(n) + (-1)^k x\left(n+\frac{N}{2}\right)\right] W_N^k \qquad (2.3.1.8)$$

Kemudian, jika rumus DFT dipisahkan (diuraikan) X(k) menjadi bagian-bagian benomor genap dan ganjil, maka didapatkan

$$X(2k+1) = \sum_{n=0}^{(N/2)-1} \left\{\left[x(n) - x\left(n+\frac{N}{2}\right)\right]\right\}, \qquad k = 0,1,\ldots,\frac{N}{2}-1 \qquad (2.3.1.9)$$

Dengan menggunakan dakta bahwa $W_N^2 = W_{N/2}$

Jika kita mendefinisikan deret-deret *N/2* titik g1(n) dan g2(n) sebagai

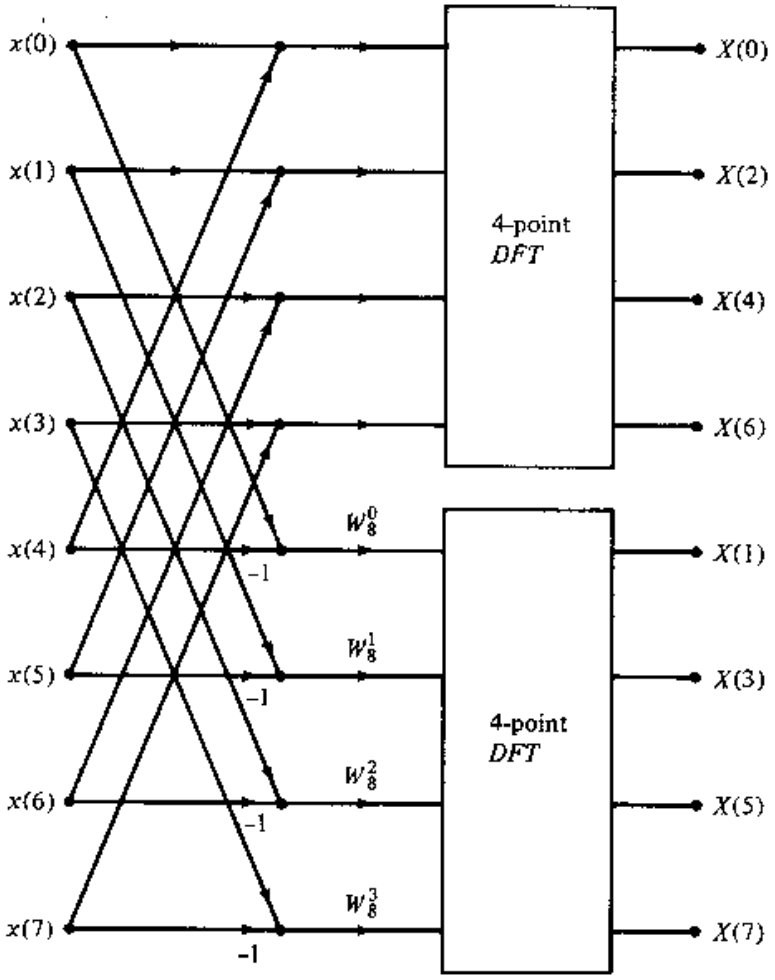
$$g_1(n) = x(n) + x\left(n+\frac{N}{2}\right)$$

$$g_2(n) = \left[x(n) - x\left(n+\frac{N}{2}\right)\right] W_N^n \qquad n = 0,1,2,\ldots,\frac{N}{2}-1 \qquad (2.3.1.10)$$

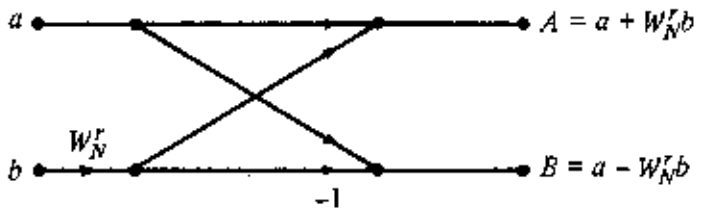
Maka

$$X(2k) = \sum_{n=0}^{\left(\frac{N}{2}\right)-1} g_1(n) W_{\frac{N}{2}}^{kn} \text{ s}$$

$$X(2k+1) = \sum_{n=0}^{\left(\frac{N}{2}\right)-1} g_2(n) W_{\frac{N}{2}}^{kn} \qquad (2.3.1.11)$$

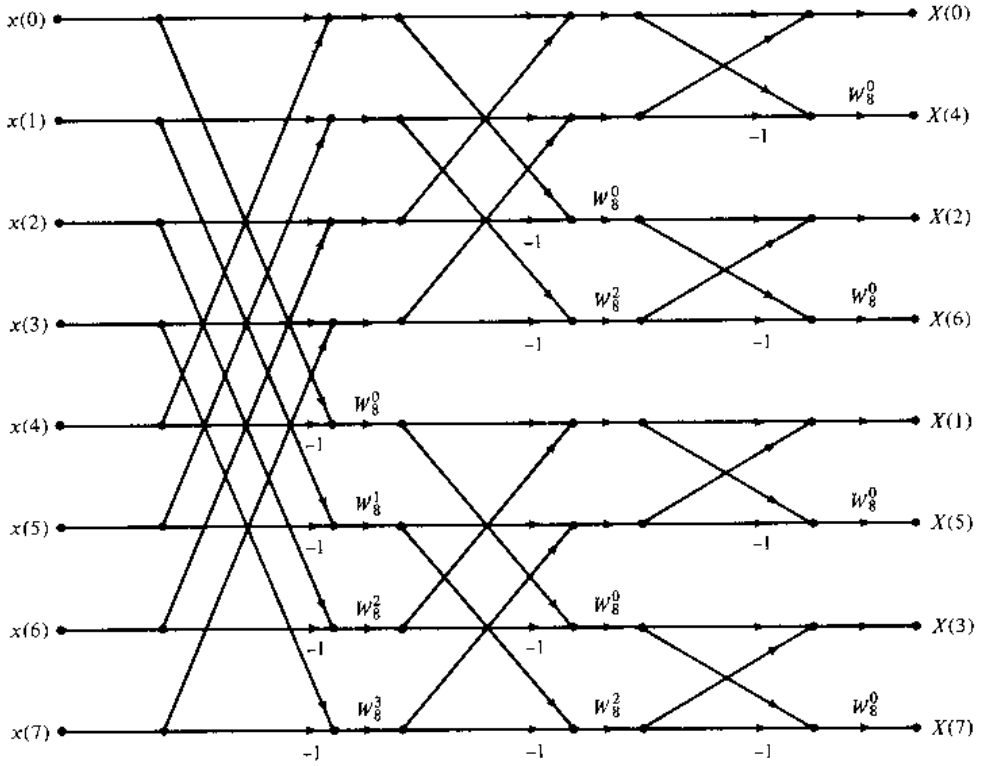


Gambar 2.6. Tahap pertama algoritma FFT peruraian frekuensi



Gambar 2.7. Perhitungan kupu-kupu dasar melalui algoritma FFT peruraian dalam frekuensi

Cara komputasi ini dapat diulangi melalui penguraian DFT *N/2*-titik, *X*(2k) dna *X*(2k +1). Keseluruhan proses meliputi V = $\log_2 N$ tahap penguraian, dimana masing-masing tahap meliputi *N/2* kupu-kupu dengan tipe yang diperlihatkan pada gambar 2.16. konsekuensinya, komputasi DFT N-titik melalui algoritna FFT penguraian dalam frekuensi, membutuhkan perkalian kompleks (*N/2*) $\log_2 N$ dan penambahan kompleks $N\log_2 N$ tepat seperti algoritma penguraian dalam waktu. Algoritma penguraian dalam frekuensi (*decimation-in-frequency algorithm*) delapan titik seperti yang ditunjukkan pada Gambar 2.16

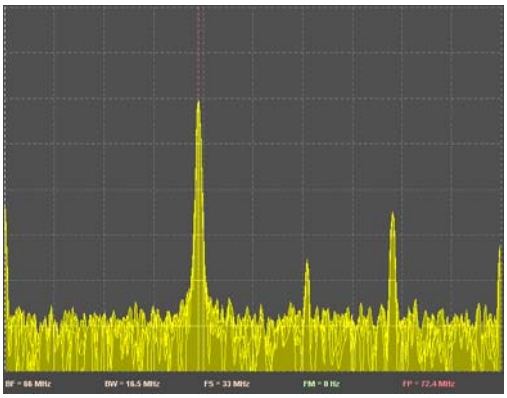


Gambar 2.8. Contoh algoritma FFT radix-2 denagn 8 sampel data

2.4. Spectrum Analyzer

Spectrum analyzer adalah suatu program untuk menampilkan spectrum frekuensi suara untuk setiap waktu tertentu. Dalam *spectrum analyzer* terdapat dua buah komponen sinyal yang dapat diteliti, yaitu frekuensi dan amplitude.

Tampilan *spectrum analyzer* yang ditunjukkan oleh Gambar 2.9:



Gambar 2.9. Tampilan *spectrum analyzer*

Keterangan :

sumbu x = frekuensi

sumbu y = Amplitudo

Pembuatan *Spectrum analyzer* dilakukan dengan melakukan perhitungan FFT untuk setiap N buah data pada suatu sinyal secara terus menerus dan berurutan. Data–data yang didapat dari perhitungan kemudian dirangkai menjadi satu kesatuan sehingga kondisi spektrum dapat dipantau setiap waktu. *Spectrum analyzer* memiliki kelebihan dalam proses analisis suara dibandingkan dengan melihat sinyal pada time domain secara langsung, hal ini dikarenakan *spectrum analyzer* bekerja pada domain frekuensi yang cenderung konstan dan posisi frekuensinya tidak terpengaruh oleh noise.

2.5. Transduser

Transduser (*transducer*) adalah sebuah alat yang mengubah satu bentuk daya menjadi bentuk daya lainnya untuk berbagai tujuan termasuk pengubahan ukuran atau informasi. Transduser bisa berupa peralatan listrik, elektronik, elektromekanik, elektromagnetik, fotonik, atau fotovoltaik. Dalam pengertian lainnya, transduser dapat juga didefinisikan sebagai suatu peralatan yang mengubah suatu bentuk sinyal menjadi bentuk sinyal lainnya.Contoh yang umum adalah pengeras suara (*audio speaker*), yang mengubah beragam sinyal listrik menjadi sinyal suara Contoh lain adalah mikrofon, yang mengubah suara kita, bunyi, atau energi akustik menjadi sinyal atau energi listrik

2.6. *Loudspeaker*

Loudspeaker merupakan suatu transduser yang mengubah sinyal elektrik menjadi sinyal suara, *loudspeaker* merupakan salah satu komponen dari sebuah sistem audio, dimana *loudspeaker* sendiri terdiri dari beberapa subsistem (*driver*), yaitu:

- *Subwoofer / Woofer* : bagian dari *loudspeaker* ini dipakai untuk menghasilkan reproduksi suara dengan rentang frekuensi 100Hz kebawah, dimana jika sebuah *woofer* dapat mereproduksi suara dengan frekuensi dibawah 40 Hz maka akan disebut *subwoofer*.
- Middle / Midwoofer : bagian dari *loudspeaker* ini dipakai untuk menghasilkan reproduksi suara pada rentang frekuensi antara 80 – 4500Hz.
- Tweeter : bagian dari *loudspeaker* ini dipakai untuk menghasilkan reproduksi suara pada rentang frekuensi antara 3500-20000 Hz
- *Full range* : speaker *full range* memiliki rentang frekuensi yang paling lebar, biasanya memiliki ukuran kecil (3-8 inci) agar dapat memproduksi suara pada frekuensi tinggi, namun juga didesain dengan sedemikian rupa agar dapat memproduksi suara pada frekuensi rendah.

2.7. *Frequency response*

Frequency Response atau responsi frekuensi adalah suatu respon dari suatu sistem terhadap nilai-nilai fekuensi yang berbeda (tetapi dengan nilai ampitudo yang konstan) yang diberikan kepada sistem tersebut.

Pengukuran responsi frekuensi dapat digunakan secara langsung untuk mengukur performa dari suatu sistem, selain itu grafik atau data dari hasil pengukuran responsi frekuensi dapat digunakan untuk mengukur tingkat ketepatan dari suatu *amplifier* atau *loudspeaker* dalam mereproduksi suara.

2.8 Microphone

Microphone atau mikrofon lebih sering disebut dengan *mic* atau *mike*, adalah sebuah transduser yang mengubah sinyal suara menjadi sinyal elektrik. *Microphone* banyak digunakan pada berbagai aplikasi, antara lain

- Telephone
- Tape Recorder
- Alat bantu pendengaran
- Dunia permusikan, dan perfileman
- *Broadcasting,* seperti pertelevisian dan radio
- Perekaman pada komputer seperti VoiP
- Hal-hal lainnya yang tidak berhubungan dengan dunia Akustik seperti pengecekan sinyal *ultrasonic.*

2.8.1. Jenis mikrofon berdasarkan material

Mikrofon selalu dihubungkan dengan alat pengeras suara (amplifier), agar keluaran mikrofon dalam bentuk sinyal listrik yang masih lemah tersebut dapat diperkuat semaksimal mungkin sesuai kebutuhan dan hasilnya dapat didengar melalui loud speaker. Ada berbagai jenis mikrofon yang terdapat di pasaran, antara lain:

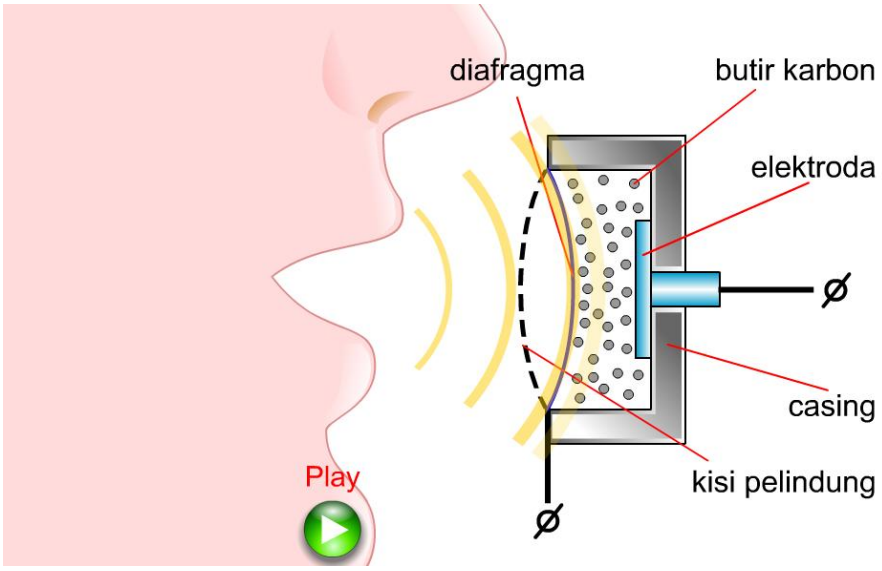
- Mikrofon Karbon

  Mikrofon ini bekerja berdasarkan pada resistansi variabel dimana konstruksinya dibuat dengan sebuah diafragma logam yang pada salah satu ujung dari sebuah kotak logam yang berbentuk silinder.

  Sebuah penghubung (contact) logam berbentuk plunyer dilekatkan pada diafragma itu sehingga gerakan diafragma dapat diteruskan melalui plunyer kepada butir-butir karbon didalam mikrofon tersebut. Sebuah kontak tetap lainnya yang terisolasi juga dibenamkan ke dalam butir-butir karbon untuk membentuk elektroda yang kedua.

  Bila gelombang suara yang menekan mengenai diafragma itu, plunyer akan terdorong dan memampatkan butir-butir karbon, sehingga menurunkan resistensi kontak diantaranya. Bila tidak ada tekanan resistansi akan naik kembali, sehingga dengan adanya getaran suara yang berubah-ubah akan menimbulkan perubahan nilai resistansi dan juga akan mengakibatkan perubahan sinyal output mikrofon.

Gambar 2.10. Mikrofon karbon

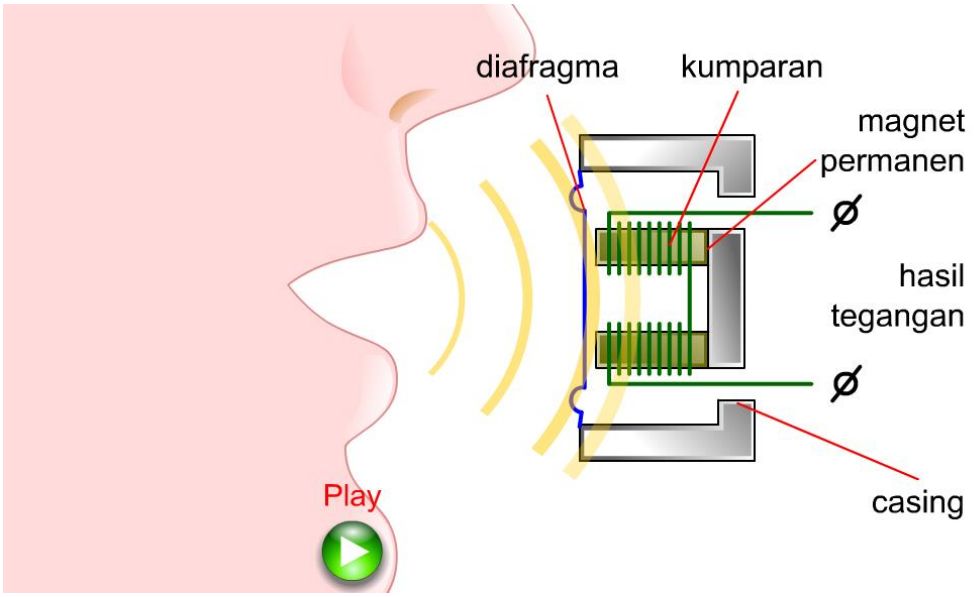


Gambar 2.11. Cara kerja mikrofon karbon

- Mikrofon Reluktansi Variabel

  Merupakan mikrofon jenis magnetic yang dibuat dengan sebuah diafragma bahan magnetic yang bergerak, seperti baja silicon yang tergantung di atas kepingan-kepingan kutub sebuah magnet permanen. Kumparan-kumparan induksi digulung pada kepingan kutub itu dan dihubungkan menurut hubungan seri yang saling memperkuat. Bila tekanan udara pada diafragma meningkat akibat getaran suara, maka celah udara dalam rangkaian magnetis tersebut akan berkurang, sehingga mengurangi reluktansi dan mengakibatkan perubahan-perubahan magnetis yang terpusat didalam struktur magnetis itu. Ketika garis-garis perubahan-perubahan (fluks) magnetis bergerak masuk, maka garis-garis akan memotong lilitan kumparan dan menginduksi suatu medan elektroinagnetik di dalamnya. Bila diafragma bergerak menjauhi kepingan-kepingan kutub, celah udara melebar, reluktansi meningkat dan garis-garis fluks bergerak keluar dari kepingan-kepingan kutub sehingga mengimbas suatu medan elektromagnetis dengan polaritas yang berlawanan didalam kumparan, maka perubahan-perubahan itu menyebabkan sinyal yang keluar dari mikrofon berubah-ubah pula.

Gambar 2.12. Mikrofon reluktansi variabel

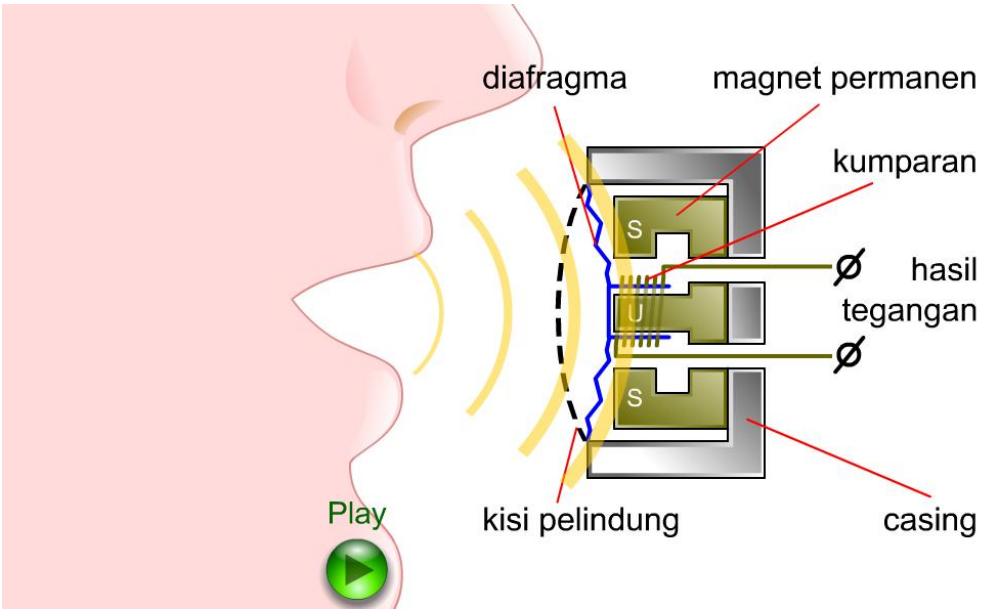


Gambar 2.13. Cara kerja mikrofon reluktansi variabel

- Mikrofon dengan Kumparan Bergerak

  Mikrofon dengan kumparan yang bergerak (*Moving coil microphone*), merupakan sebuah mikrofon dengan kumparan induksi yang digulungkan pada suatu silinder bukan magnetis yang dilekatkan pada diafragma dan dipasang di dalam celah udara berbentuk silinder dari suatu magnet permanen. Diafragma dibuat dari bahan bukan logam, sedangkan kawat-kawat penghubung listrik ke kumparan direkatkan ke permukaan diafragma. Bila gelombang suara menggerakkan diafragma, maka kumparan akan bergerak maju mundur di dalam medan magnet, sehingga terjadi perubahan-perubahan magnetik yang melewati kumparan dan menghasilkan sinyal listrik.

Gambar 2.14. Mikrofon dengan kumparan bergerak



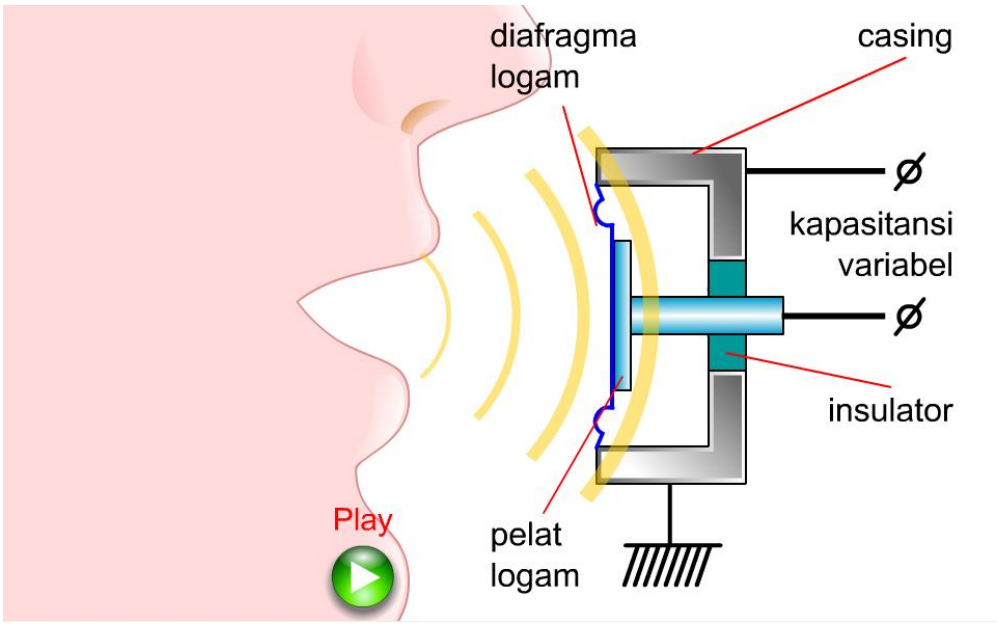
Gambar 2.15. Cara kerja mikrofon dengan kumparan bergerak

- Mikrofon Kapasitor

  Terdiri dari sebuah diafragma logam yang digantung dengan jarak yang sangat dekat terhadap sebuah pelat logam statis, dimana keduanya terisolasi sehingga menyerupai bentuk sebuah kapasitor. Diafragma akan bergerak-gerak bila terkena getaran suara, hal itu akan mengakibatkan berubah-ubahnya jarak pemisah antara diafragma dan pelat statis yang mengakibatkan berubah-ubahnya nilai kapasitansi.

Diperlukan suatu tegangan DC konstan dari luar yang dihubungkan pada diafragma dan pelat logam statis lewat sebuah resistor beban, sehingga tegangan terminal mikrofon dapat berubah-ubah seiring dengan terjadinya perubahan tekanan udara akibat getaran suara.

Gambar 2.16. Mikrofon Kapasitor



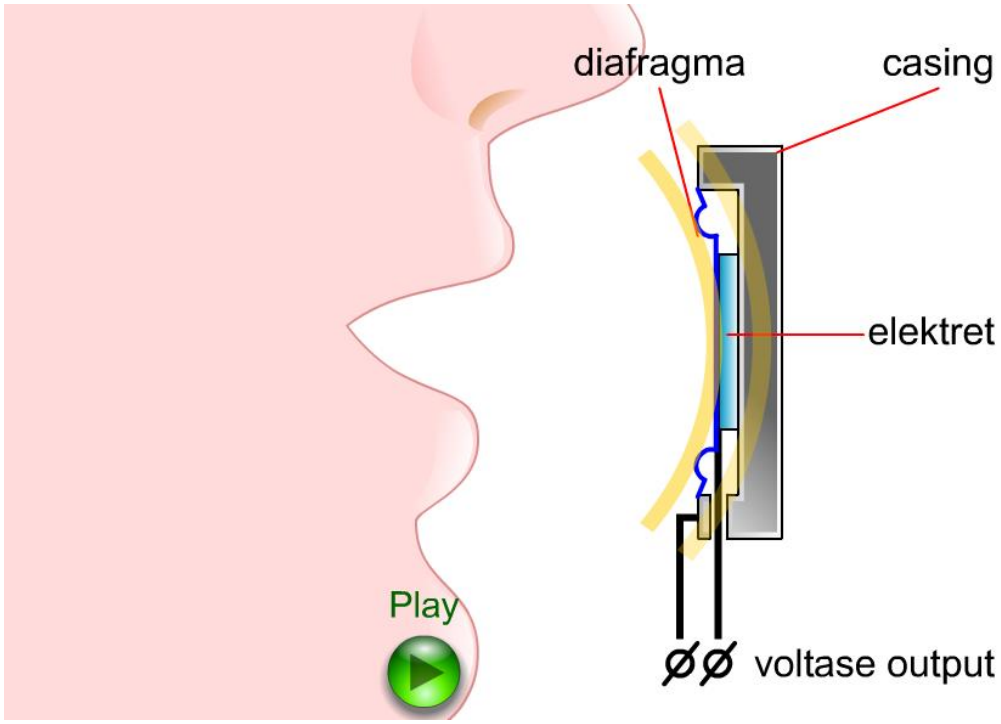
Gambar 2.17. Cara kerja mikrofon Kapasitor

- Mikrofon Elektret

  Mikrofon ini merupakan jenis khusus dari mikrofon kapasitor yang sudah mempunyai sumber muatan sendiri yang terpasang didalamnya sehingga tidak perlu pencatu daya dari luar. Sumber muatan itu sebenarnya didapat dari suatu alat penyimpan muatan berupa bahan Teflon yang diproses dengan semestinya sehingga dapat menangkap muatan-muatan tetap dalam jumlah besar dan mempertahankannya untuk waktu tak terbatas. Lapisan tipis Teflon yang dilekatkan pada pelat logam statis, mengandung sejumlah besar muatan-muatan negative yang terperangkap yang kemudian diinduksikan sebagai suatu muatan bayangan kepada pelat statis dan diafragma logam yang dihubungkan padanya melalui sebuah resistor beban luar.

  Muatan-muatan yang terperangkap pada satu sisi dan muatan bayangan pada sisi yang lain menimbulkan medan listrik pada celah yang membentuk kapasitor. Tekanan udara yang berubah-ubah akibat getaran suara akan membuat berubah-ubahnya jarak antara diafragma dan pelat logam statis, sehingga nilai kapasitansi berubah dan mengakibatkan tegangan terminal mikrofon juga turut berubah.

Gambar 2.18. Mikrofon elektret



Gambar 2.19. Cara kerja mikrofon elektret

- Mikrofon Piezoelektris

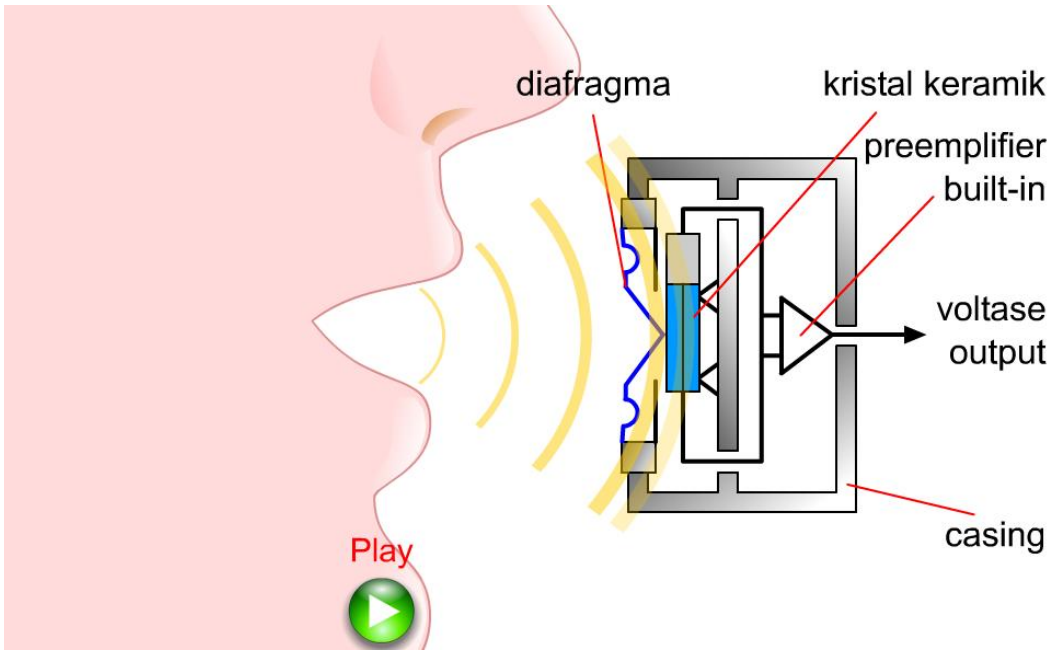
  Adalah mikrofon yang tidak memerlukan sebuah pencatu daya karena jenis mikrofon ini terbuat dari bahan kristal aktif yang dapat menimbulkan tegangan sendiri bila diberikan getaran dari luar, sehingga dapat merupakan sebuah generator.

  Kristal dipotong menurut bidang-bidang tertentu untuk membentuk suatu irisan dan dengan elektroda-elektroda / pelat lempengan dilekatkan pada kedua permukaannya sehingga akan menunjukkan sifat-sifat piezoelektris. Bila mendapat tekanan, kristal akan berubah bentuk {deform), akan terjadi perpindahan suatu muatan sesaat didalam susunan kristal tersebut sehingga dapat menimbulkan suatu beda potensial diantara kedua pelat-pelat lempengan.

  Sebaliknya bila suatu potensial listrik dikenakan antara kedua permukaan kristal itu, secara fisik kristal akan melengkung atau berubah bentuk. Kristal langsung dapat menerima getaran suara tanpa harus dibentuk menjadi sebuah diafragma, sehingga dapat diperoleh respon frekuensi yang lebih baik dari

pada mikrofon lainnya meskipun dengan suatu tingkat keluaran yang jauh lebih rendah, yaitu kurang dari 1 mV.

Gambar 2.20 Mikrofon pizzoelektris



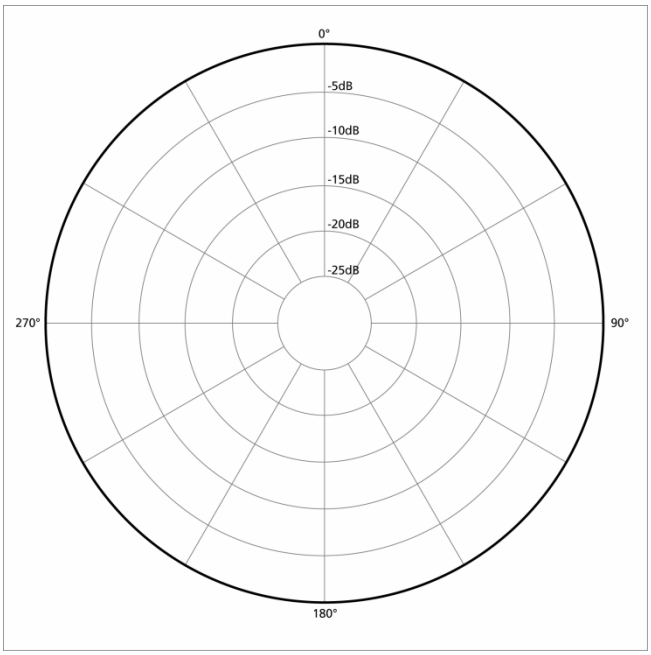
Gambar 2.21 cara kerja mikrofon pizzoelektris

2.8.2. Jenis mikrofon berdasarkan *polar pattern*

Polar Pattern adalah arah penangkapan bunyi pada sebuah mikrofon. Fungsi daripada polar pattern ini untuk mengetahui arah penangkapan suara daripada sebuah mikrofon. Berdasarkan *polar pattern* mikrofon dapat dibagi menjadi beberapa jenis, antara lain:

1. Omnidirectional

Pada jenis mikrofon ini, bunyi ditangkap secara merata pada segala arah, dan jenis mikrofon ini paling banyak ditemukan pada pasaran.



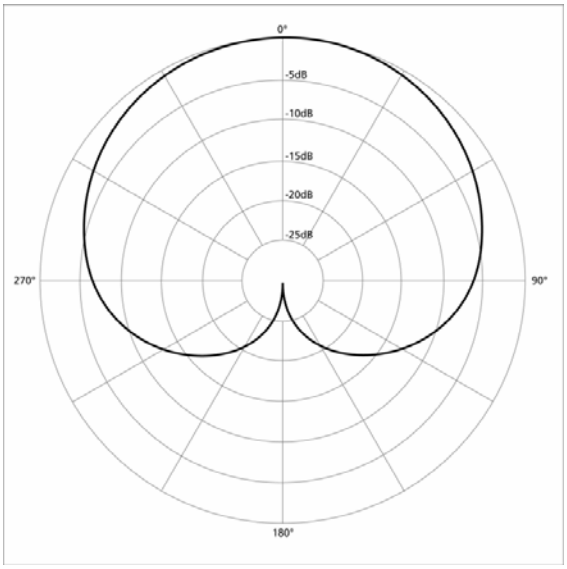
Gambar 2.22. *Omnidirectional polar pattern*

2. Unidirectional

Pada mikrofon jenis ini, arah penangkapan bunyi difokuskan pada satu arah penangkapan (biasanya dari sisi depan mikrofon), *Unidirectional Microphone* sendiri dapat dibagi menjadi beberapa bagiam menurut bentuk *polar pattern*-nya, antara lain:

- *Cardioids*

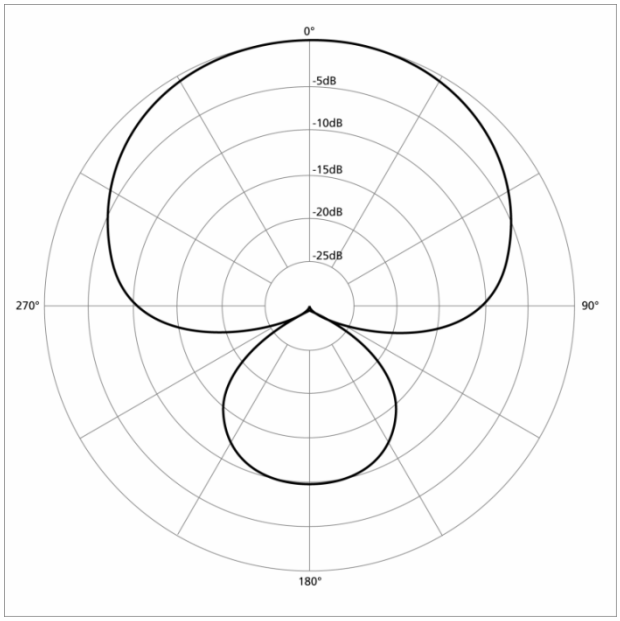
Polar pattern pada mikrofon jenis ini memiliki bentuk seperti hati, dimana arah penangkapan bunyi hanya difokuskan pada sisi depan mikrofon saja.



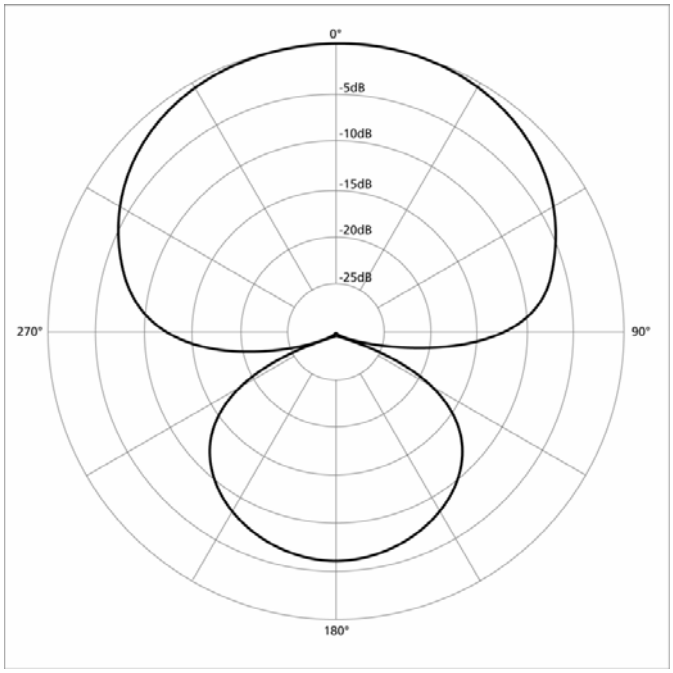
Gambar 2.23. *Cardioids polar pattern*

- *Supercardioids* dan *Hypercardioids*

  Arah penangkapan bunyi pada mikrofon *Supercardioids* hampir sama dengan mikrofon *Hypercardioids*, dimana sensifitas dari arahpenangkapan suara difokuskan pada sisi depan,dan sebagian kecil pada sisi belakang. Yang membedakan kedua jenis mikrofon ini, pada mikrofon *Supercardioids* sisi depan dari mikrofon mempunyai sensifitas yang lebih besar daripada mikrofon *Hypercardioids*, dan sensifitas pada sisi belakang mikrofon *Hypercardioids* lebih besar daripada mikrofon *Supercardioids.*



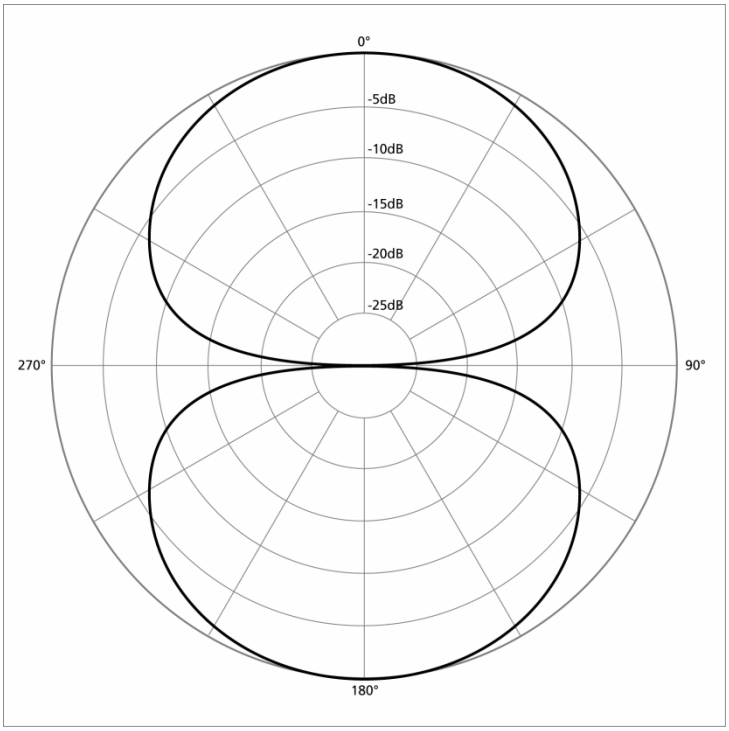
Gambar 2.24. *Supercardioids polar pattern*



Gambar 2.25. *Hypercardioids polar pattern*

3. Bi-directional

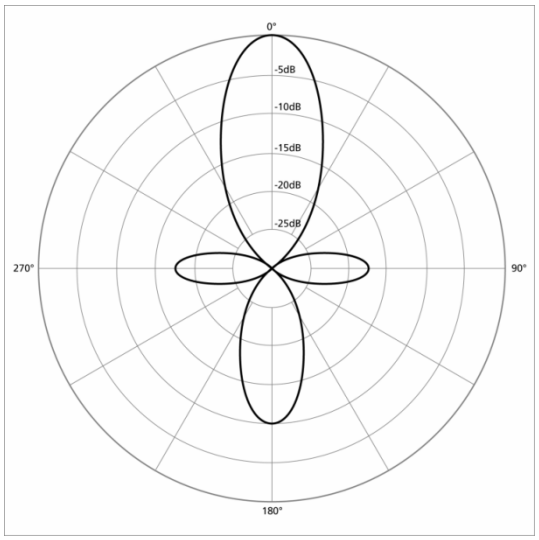
Mikrofon yang memiliki *polar pattern* seperti ini sering disebut dengan nama *Figure-Eight Microphone,* pada mikrofon jenis ini, arah penangkapan bunyi difokuskan pada sisi depan dan belakang dari mikrofon.



Gambar 2.26. *Bi-directional Polar Pattern*

4. Shotgun

Pada mikrofon jenis *Shotgun* penangkapan difokuskan pada sisi depan dan belakang mikrofon, serta sedikit sensifitas pada sisi kiri dan kanan mikrofon. Jenis mikrofon ini umum digunakkan pada studio rekaman maupun televisi, stadium, dan dipakai dalam perekaman satwa liar.



Gambar 2.27. *Shotgun Polar Pattern*

2.9 Sinyal suara digital

Suara merupakan sebuah sinyal atau gelombang yang merambat melalui perantara yaitu air, udara, dan benda padat. Pada umumnya manusia dapat mendengar suara yang memiliki rentang frekuensi antara 20 Hz – 20KHz.

Sinyal natural seperti halnya suara merupakan sinyal kontinyu yang memiliki rentang nilai tak terbatas, dan sinyal-sinyal tersebut tidak akan dapat diproses oleh komputer karena keanekaragaman nilainya.

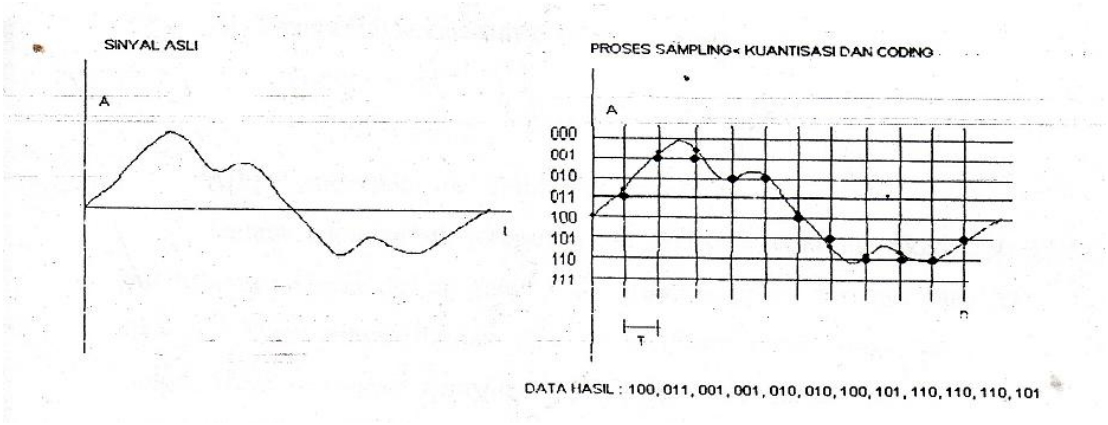
Komputerisasi sinyal natural dapat dilakukan jika sinyal tersebut diubah menjadi sebuah sinyal digital atau dapat disebut juga sinyal *discrete.* Proses digitalisasi sebuah sinyal dilakukan melalui 3 buah proses, yaitu proses sampling data, proses kuantisasi, dan proses pengkodean.

Poses sampling data ialah suatu proses pengambilan data dari suatu sinyal untuk setiap periode tertentu. Dalam proses sampling data, frekuensi sampling yang diambil harus memiliki nilai 2 kali frekuensi tertinggi dari sinyal yang akan diambil. Dan jika pengambilan sampling dilakukan dengan nilai sampling kurang dari 2 kali sinyal yang diambil, akan timbul sebuah sinyal baru yang memiliki frekuensi dengan nilai berlainan dengan frekuensi sinyal yang asli, hal ini dinamakan efek aliasing.

Proses kuantisasi adalah proses untuk membulatkan nilai data kedalam bilangan-bilangan tertentu. Semakin banyak level yang dipakai maka akan semakin akurat pula data sinyal yang akan disimpan, namun banyaknya level yang dipakai juga akan menyebabkan proses kuantisasi berjalan semakin lama dan memiliki ukuran data yang besar

Proses pengkodean adalah proses pemberian kode untuk tiap-tiap data sinyal yang telah terkuantisasi berdasarkan level yang ditempati.

Proses pembentukan sinyal digital dapat dilihat pada Gambar 2.28



Gambar 2.28. Proses pembentukan sinyal digital

2.10. *Borland Delphi*

Delphi adalah sebuah bahasa pemrograman dan lingkungan pengembangan perangkat lunak. Produk ini dikembangkan oleh "CodeGear" sebagai divisi pengembangan perangkat lunak milik "Borland".

Bahasa *Delphi*, atau dikenal pula sebagai *object pascal* (pascal dengan ekstensi pemrograman berorientasi objek (PBO/OOP) yang pada mulanya ditujukan hanya untuk *Microsoft Windows*, namun saat ini telah mampu digunakan untuk mengembangkan aplikasi untuk *Linux* dan *Microsoft .NET* framework.

Dengan menggunakan *Free Pascal* yang merupakan proyek *opensource*, bahasa ini dapat pula digunakan untuk membuat program yang berjalan di sistem operasi *Mac OS X* dan *Windows CE*

Delphi adalah sebuah bahasa pemrograman dan lingkungan pengembangan perangkat lunak. Produk ini dikembangkan oleh CodeGear sebagai divisi pengembangan perangkat lunak milik Embarcadero, divisi tersebut sebelumnya adalah milik Borland. Bahasa Delphi, atau dikenal pula sebagai *object pascal* (pascal dengan ekstensi pemrograman berorientasi objek (PBO/OOP)) pada mulanya ditujukan hanya untuk Microsoft Windows, namun saat ini telah mampu digunakan untuk mengembangkan aplikasi untuk Linux dan Microsoft .NET framework (lihat di bawah). Dengan menggunakan Free Pascal yang merupakan proyek opensource, bahasa ini dapat pula digunakan untuk membuat program yang berjalan di sistem operasi Mac OS X dan Windows CE

Keunggulan dari *Delphi* adalah keberadaan bahasanya (Bahasa pemrograman *delphi*), *Visual Component Library* (VCL/CLX), Penekanan konektifitas *database* yang sangat baik, dan banyaknya komponen-komponen pihak ketiga yang mendukungnya.

Keunggulan *delphi* tersebut, antara lain:

1. Penanganan *object* sebagai *reference*/*pointer* transparan
2. Properti sebagai bagian dari bahasa tersebut; benar, sebagai *getter* dan *setter* (atau accessor and mutator), yang secara transparan mengenkapsulasi akses pada field-field anggota dalam kelas tersebut.
3. Property index dan Default yang menyediakan akses pada data kolektif
4. Pendelegasian (*type safe method pointer*) yang digunakan untuk memproses event yang dipicu oleh *component*
5. Pendelegasian implementasi *interface* pada *Field* ataupun *property* dari *class*.
6. Implementasi penanganan *windows message* dengan cara membuat *method* dalam *class* dengan nomer/nama dari *windows message* yang akan dihandle.
7. COM bersifat sebagai *interface* yang independen dengan implementasi *class* sebagai *reference counted*
8. Komunitas pengguna yang besar pada *Usenet* maupun
9. Dapat mengkompilasi menjadi single *executable*, memudahkan distribusi dan meminimalisir masalah yang terkait dengan *versioning*
10. Banyaknya dukungan dari pihak ketiga terhadap VCL (biasanya tersedia berikut *source code*-nya) ataupun *tools* pendukung lainnya (dokumentasi, *tool debugging*)
11. Optimasi kompiler yang cukup cepat
12. Mendukung multiple platform dari source code yang sama