BAB II

LANDASAN TEORI

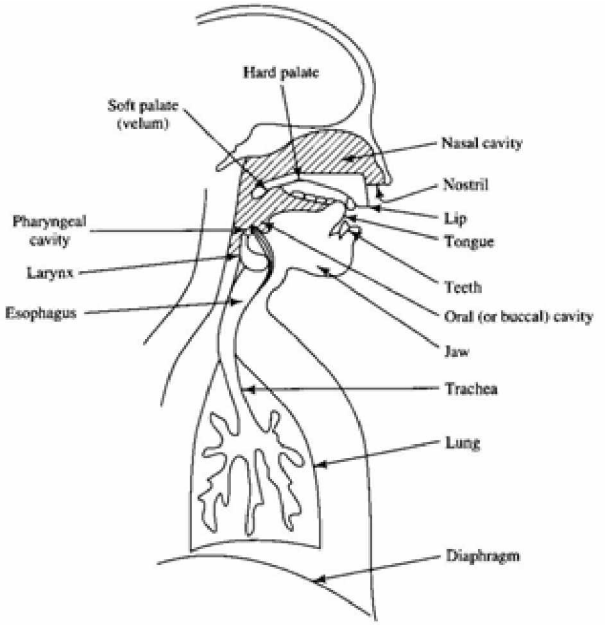
II.1. Pengenalan Suara Pada Manusia

Pada sistem pengenalan suara oleh manusia terdapat tiga organ penting yang saling berhubungan yaitu : telinga yang berperan sebagai transduser dengan menerima sinyal masukan suara dan mengubahnya menjadi sinyal syaraf, jaringan syaraf yang berfungsi mentransmisikan sinyal ke otak, dan otak yang akan mengklasifikasi dan mengidentifikasi informasi yang terkandung dalam sinyal masukan.

II.1.1. Proses Produksi Suara

Proses produksi suara pada manusia dapat dibagi menjadi tiga buah proses fisiologis, yaitu : pembentukan aliran udara dari paru-paru, perubahan aliran udara dari paru-paru menjadi suara, baik *voiced*, maupun *unvoiced* yang dikenal dengan istilah *phonation*, dan artikulasi yaitu proses modulasi atau pengaturan suara menjadi bunyi yang spesifik.

Organ tubuh yang terlibat pada proses produksi suara adalah : paru-paru, tenggorokan (*trachea*), laring (*larynx*), faring (*pharynx*), pita suara (*vocal cord*), rongga mulut (*oral cavity*), rongga hidung (*nasal cavity*), lidah (*tongue*), dan bibir (*lips*), seperti dapat dilihat pada gambar berikut ini.



Gambar II.1. Organ Sistem Suara Pada Manusia

Organ tubuh ini dapat dikelompokkan menjadi tiga bagian utama, yaitu : *vocal tract* (berawal di awal bukaan pita suara atau glottis, dan berakhir di bibir), *nasal tract* (dari velum sampai nostril), dan *source generator* (terdiri dari paru-paru, tenggorokan, dan *larynx*). Ukuran *vocal tract* bervariasi untuk setiap individu, namun untuk laki-laki dewasa rata-rata panjangnya sekitar 17 cm. Luas dari *vocal tract* juga bervariasi antara 0 (ketika seluruhnya tertutup) hingga sekitar 20 cm^2. Ketika *velum*, organ yang memiliki fungsi sebagai pintu penghubung antara *vocal tract* dengan *nasal tract*, terbuka, maka secara akustik *nasal tract* akan bergandengan dengan *vocal tract* untuk menghasilkan suara *nasal*.

Aliran udara yang dihasilkan dorongan otot paru-paru bersifat konstan. Ketika pita suara dalam keadaan berkontraksi, aliran udara yang lewat membuatnya bergetar. Aliran udara tersebut dipotong-potong oleh gerakan pita

suara menjadi sinyal pulsa yang bersifat *quasi-periodik*. Sinyal pulsa tersebut kemudian mengalami modulasi frekuensi ketika melewati *pharynx,* rongga mulut ataupun pada rongga hidung. Sinyal suara yang dihasilkan pada proses ini dinamakan sinyal *voiced*. Namun, apabila pita suara dalam keadaan relaksasi, maka aliran udara akan berusaha melewati celah sempit pada permulaan *vocal tract* sehingga alirannya menjadi turbulen, proses ini akan menghasilkan sinyal *unvoiced*. Ketika sumber suara melalui *vocal tract*, kandungan frekuensinya mengalami modulasi sehingga terjadi resonansi pada *vocal tract* yang disebut *formants*. Apabila sinyal suara yang dihasilkan adalah sinyal *voiced*, terutama vokal, maka pada selang waktu yang singkat bentuk *vocal tract* relative konstan (berubah secara lambat) sehingga bentuk *vocal tract* dapat diperkirakan dari bentuk spektral sinyal *voiced*.

Aliran udara yang melewati pita suara dapat dibedakan menjadi *phonation*, bisikan, *frication*, kompresi, vibrasi ataupun kombinasi diantaranya. *Phonated excitation* terjadi bila aliran udara dimodulasi oleh pita suara. *Whispered excitation* dihasilkan oleh aliran udara yang bergerak cepat masuk ke dalam lorong bukaan segitiga kecil antara *arytenoids cartilage* di belakang pita suara yang hampir tertutup. *Frication excitation* dihasilkan oleh desakan di *vocal tract*. *Compression excitation* dihasilkan akibat pelepasan udara melalui v*ocal tract* yang tertutup dengan tekanan tinggi. *Vibration excitation* disebabkan oleh udara yang dipaksa memasuki rusang selain pita suara, khususnya lidah. Suara yang dihasilkan oleh *phonated excitation* disebut *voiced.* Suara yang dihasilkan oleh *phonated excitation* ditambah *frication* disebut *mixed voiced,* sedangkan

yang dihasilkan oleh selain itu disebut *unvoiced*. Karakteristik suara tiap individu bersifat unik karena terdapat perbedaan dalam hal panjang maupun bentuk *vocal tract*.

II.1.2. Karakteristik Telinga

Telinga terbagi menjadi tiga bagian, yaitu bagian luar, tengah, dan dalam. Pinna, sebagai bagian luar telinga, berfungsi sebagai corong, untuk mengumpulkan sinyal suara menuju *auditory canal* sehingga dapat memberikan kesan arah sinyal suara yang diterima.

Auditory canal adalah struktur berbentuk pipa lurus sepanjang 2,7 cm, dengan diameter sekitar 0,7 cm, yang pada bagian ujungnya terdapat selaput membrane, yaitu gendang telinga. Membran ini merupakan pintu masuk telinga bagian tengah, yaitu ruangan berisi udara dengan *volume* sebesar 2 cm^3, yang terdiri dari tiga buah tulang, yaitu *malleus* (martil), *incus* (landasan), dan *stapes* (sanggurdi). Bagian ini terhubung dengan tenggorokan melalui *Eustachian tube*. Getaran pada gendang telinga ditransmisikan ke *malleus* melalui *incus*, dan *stapes*, yaitu membentuk *oval window*.

Telinga bagian dalam (*labyrinth*) memiliki tiga bagian, yaitu *vestibule* (ruang pintu masuk), *semicular canal*, dan *cochlea*. *Vestibule* terhubung dengan telinga bagian tengah melalui dua jalur, yaitu *oval window*, dan *round window*. Keduanya tertutup untuk mencegah keluarnya cairan yang mengisi telinga telinga bagian dalam. Pada *cochlea*, yang berstruktur seperti rumah siput, terdapat syaraf pendengaran. Syaraf ini memanjang sampai ke *basilar membrane*. Pada bagian

atas *basilar membrane* terdapat *organ of corty* yang memiliki empat baris sel rambut (sekitar 3 x 10^4 sel seluruhnya).

II.1.3. Proses Pendengaran

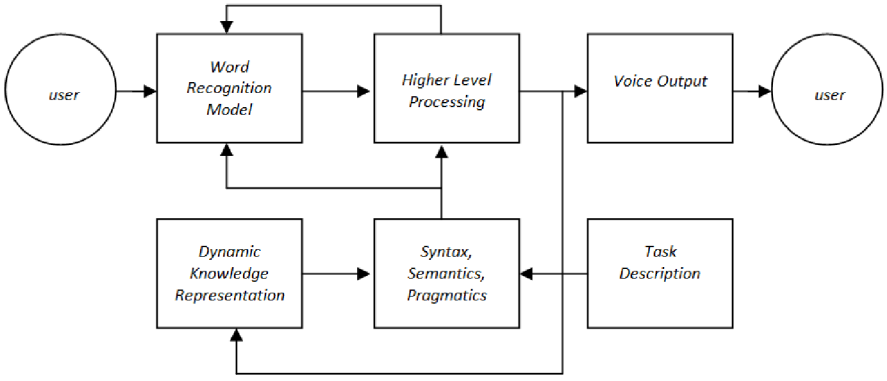
Proses pendengaran pada telinga manusia dijelaskan sebagai berikut :

1) Sinyal suara memasuki saluran telinga dan variasi tekanan yang dihasilkannya menekan gendang telinga. Karena sisi bagian dalam dari gendang telinga mempunyai tekanan yang nilainya dijaga konstan maka gendang telinga akan bergetar.
2) Getaran dari gendang telinga disalurkan pada tiga rangkaian tulang yaitu; *martil, incus* dan *stapes*. Mekanisme ini dirancang untuk mengkopel variasi suara dari udara luar ke telinga bagian dalam. Karena luas permukaan penampang yang ditekan *stapes* lebih kecil dari luas penampang gendang telinga maka tekanan suara yang sampai ke telinga bagaian dalam bertambah besar.
3) Cairan pada *cochlea* bergetar dengan frekuensi yang sama dengan gelombang yang datang. *Basilar membrane* kemudian memisahkan sinyal berdasarkan frekuensinya. *Basilar membrane* berstruktur kuat dan panjang di daerah sekitar *oval window* namun bersifat lentur pada bagian ujungnya. Frekuensi resonansi yang dihasilkan membran tersebut berbeda sepanjang dimensi *basilar membrane*. Dimana resonansi frekuensi tinggi terjadi pada bagian bagian *basilar membrane* yang berada dekat dengan *oval window,* sedangkan resonansi frekuensi rendah terjadi pada daerah ujung lainnya. Syaraf yang berada pada membran kemudian mendeteksi posisi terjadinya resonansi yang

juga akan menentukan frekuensi suara yang datang. Ukuran dari *basilar membrane* rata-rata sekitar 35 mm. Dari ukuran panjang tersebut dapat dihasilkan 10 resolusi frekuensi, sehingga pada setiap 3.5 mm panjang membran terdapat 1 oktaf frekuensi resonansi.

II.2. Proses Pengenalan Suara Secara Umum

Secara umum, model untuk proses pengenalan suara diperlihatkan pada gambar II.2 :



Gambar II.2. Diagram Blok Sistem Pengenalan Suara Secara Umum

Prinsip kerja dari sistem ini diawali ketika sinyal suara berupa ucapan diberikan oleh pengguna (*user*) untuk memerintahkan suatu tugas tertentu, kemudian sinyal suara ini akan dikodekan dalam serangkaian kode-kode informasi yang disusun berdasarkan suatu acuan yang sudah ditentukan. Acuan tersebut disebut dengan *template.*

Makna kata dapat dikenali dengan bantuan representasi pengetahuan dinamis (*dynamic knowledge representation*) yang akan melakukan modifikasi pada sintaks, semantik dan *pragmatic* yang mengacu pada konteks yang telah

dikenali sebelumnya. Dengan cara ini, hal-hal yang mengganggu dapat dihilangkan sehingga mengurangi resiko kesalahpahaman. Umpan balik dari bagian prosessor tingkat tinggi (*high level processing*) mengurangi kompleksitas model pengenalan dengan membatasi pelacakan untuk ucapan atau kalimat yang dapat diterima (*valid*) dari pengguna.

Sistem pengenalan suara akan memberikan respon terhadap pengguna dalam bentuk keluaran (*output*) yang berbeda-beda, baik berupa suara maupun melakukan tindakan lain. Dalam skripsi ini, keluaran yang diinginkan berupa tindakan mengaktifkan peralatan elektronik pada konsep rumah pintar seperti telah dijelaskan sebelumnya.

II.3. Pendekatan Sistem Pengenalan Suara Otomatis (*Automatic Speech Recognition*)

Secara umum pendekatan terhadap sistem pengenalan suara ada 3 macam yaitu :

1) Pendekatan *artificial intelegence*,
2) Pendekatan *accoustic-phonetic*,
3) Pendekatan *pattern recognition*,

II.3.1. Pendekatan Kecerdasan Buatan (Artificial Intelligence Approach)

Ide dasar dari pendekatan ini adalah mengkompilasi dan menggabungkan pengetahuan dari berbagai sumber pengetahuan dan menggunakannya untuk mengatasi masalah yang dihadapi. Sumber-sumber pengetahuan yang digunakan pada pendekatan kecerdasan buatan dalam sistem pengenalan ucapan antara lain adalah sebagai berikut :

1) Pengetahuan akustik (*Accoustic Knowledge*)

   Pengetahuan akustik merupakan keterangan mengenai sinyal dari bunyi yang di ucapkan dalam basis pengukuran spektral dan untuk mengetahui ada atau tidaknya sifat-sifat akustik.

2) Pengetahuan leksikal (*Lexical Knowledge*)

   Pengetahuan leksikal merupakan keterangan mengenai kombinasi sinyal-sinyal akustik yang menyatakan suatu kata yang dispesifikasikan berdasarkan leksikon yang memetakan suara dalam kata-kata (memecah kata dalam bunyi-bunyi ucapan).

3) Pengetahuan sintaksis (*Syntactic Knowledge*)

   Pengetahuan sintaksis merupakan keterangan mengenai kombinasi kata-kata yang membentuk suatu kalimat atau frasa secara gramatikal.

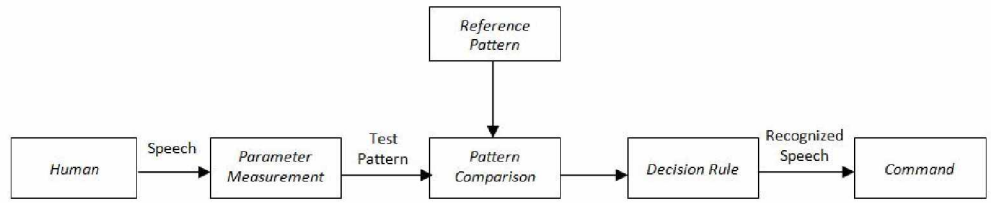
II.3.2. Pendekatan Akustik-Fonetik (Accoustic-Phonetic Approach)

Pendekatan akustik-fonetik berdasarkan pada teori akustik-fonetik dimana teori tersebut mengatakan bahwa terdapat unit fonetik yang terbatas dan khusus pada bahasa yang dipakai untuk berbicara dan unit tersebut mempunyai karakteristik umum yang ditunjukkan oleh sekumpulan *properties* yang terdapat pada sinyal suara seperti spektrum. Pendekatan ini memliki beberapa tahapan.

Tahapan-tahapan tersebut antara lain adalah menganalisa ucapan, mendeteksi sifat-sifat ucapan, melakukan segmentasi dan *labelling* (yang merupakan bagian utama dalam sistem pengenalan ucapan akustik fonetik dan akhirnya ucapan tersebut dikenali dengan mencari sifat-sifat yang cocok dari kosakata yang ada.

II.3.3. Pendekatan Pengenalan Pola (*Pattern Recognition Approach*)

Pendekatan pengenalan pola ini dapat diperlihatkan oleh gambar II.3 :



Gambar II.3. Diagram Blok Sistem Pengenalan Suara Secara Umum

Proses pengenalan ucapan dengan pendekatan ini memiliki berapa langkah yaitu :

1) Pengukuran parameter, yang dapat dilakukan dengan menggunakan berbagai metode seperti analisis *filter bank*, pengkodean terprediksi linear (*linear predictive coding*), dan transformasi *fourier* diskrit (*discrete fourier transform*).
2) Pembandingan pola, yang dilakukan dengan mengukur perbedaan antara parameter pola yang diuji (*test pattern*) dengan pola referensi (*reference pattern*).
3) Pengambilan keputusan, didasarkan pada hasil perbandingan antara pola yang diuji dengan pola referensi.

Dalam metode ini ucapan yang ingin dikenali sebelumnya harus sudah ada dalam kumpulan data pola referensi yang dikumpulkan melalui proses latihan sebelumnya.

II.3.4. Sinyal Suara Ucapan

Sinyal suara ucapan manusia dapat dipandang sebagai sinyal yang berubah lambat terhadap waktu (*slowly time varying signal*), jika diamati pada selang waktu yang singkat yaitu 5-100 ms. Pada selang waktu tersebut, katakteristik sinyal suara ucapan dapat dianggap stasioner. Untuk selang waktu yang lebih panjang (dengan orde 0.2 detik atau lebih), karakteristik sinyal berubah untuk merefleksikan suara berbeda yang diucapkan.

II.3.5. Klasifikasi Berdasarkan Sinyal Eksitasi

Berdasarkan sinyal eksitasi yang dihasilkan pada proses produksi suara, sinyal suara ucapan dapat dibagi menjadi tiga bagian yaitu *silence*, *unvoiced*, dan *voiced*:

1) Sinyal *silence* : sinyal pada saat tidak terjadi proses produksi suara ucapan, dan sinyal yang diterima oleh pendengar dianggap sebagai bising latar belakang.
2) Sinyal *unvoiced* : terjadi pada saat pita suara tidak bergetar, dimana sinyal eksitasi berupa sinyal *random*.
3) Sinyal *voiced* : terjadi jika pita suara bergetar, yaitu pada saat sinyal eksitasi berupa sinyal pulsa kuasi-periodik. Selama terjadinya sinyal *voiced* ini, pita suara bergetar pada frekuensi fundamental – inilah yang dikenal sebagai *pitch* dari suara tersebut.

II.3.6. Analisis Sinyal Ucapan

Informasi yang terdapat di dalam sebuah sinyal ucapan dapat dianalisis dengan berbagai cara. Beberapa peneliti telah membagi beberapa level pendekatan

untuk menggambarkan informasi tersebut, yaitu level akustik, fonetik, fonologi, morfologi, sintatik, dan semantik.

1. Level Akustik

Sinyal ucapan merupakan variasi tekanan udara yang dihasilkan oleh sistem artikulasi. Untuk menganalisa aspek-aspek akustik dari sebuah sinyal ucapan, dapat dilakukan dengan transformasi dari bentuk sinyal ucapan menjadi sinyal listrik dengan menggunakan tranduser seperti *microphone*, telepon, dan sebagainya. Setelah melalui berbagai pengolahan sinyal digital, maka akan di peroleh informasi yang menunjukkan sifat-sifat akustik dari sinyal ucapan tersebut yang meliputi frekuensi fundamental, intensitas, dan distribusi energi spektral.

2. Level Fonetik

Level ini menggambarkan bagaimana suatu sinyal suara diproduksi oleh organ-organ di dalam tubuh manusia.

3. Level Fonologi

Di dalam level ini, dikenal istilah fonem yang merupakan unit terkecil yang membentuk sebuah kalimat atau ucapan. Deskripsi ini memuat informasi durasi, intensitas, dan *pitch* dari fonem-fonem yang membangun kalimat tersebut.

II.4. Masalah-masalah yang Penting Pada Sistem Pengenalan Ucapan

Pada kenyataannya, Sistem Pengenalan Ucapan sampai saat ini masih perlu meningkatkan bebrapa hal guna memperbaiki kinerjanya, terutama masalh akurasi. Dalam pengenalan ucapan disebutkan beberapa masalah mengapa sulit meningkatkan akurasi. Masalah tersebut dijelaskan sebagai berikut :

1) Semakin besar ukuran kosakata, semakin tinggi kesalahan dalam pengenalan ucapan. Sebagai contoh, 10 digit dari “nol” sampai “sembilan” dapat dengan mudah dikenali. Tetapi dengan meningkatkannya ukuran *vocabulary* menjadi 200, 5000, atau 100000 maka bisa terjadi rata-rata kealahan 3%, 7%, atau 45% sementara itu, setiap bahasa mempunyai huruf ketika diucapkan sulit untuk dibedakan. Sebagai contoh huruf C, D. J dalam kasus bahasa Indonesia.
2) Tingkat ketergantungan kepada pembicara. Sistem pengenalan ucapan *Speaker Dependence* hanya ditujukan unutk pembicara tunggal. Sedangkan *Speaker Indenpence* ditujukan untuk siapapun. Jenis *Speaker Independence* sulit untuk dilakukan karena parameter suatu system pengenalan ucapan biasanya dilatih menggunakan *Speaker* tertentu, yang sangat spesifik ucapannya. Hasil penelitian menunjukan bahwa *Speaker Independence* mempunyai kesalahan rata-rata 3 sampai 5 kali lebih besar dibanding *Speaker Dependence.*
3) Penggalan ucapan, *Isolated Speech* berarti ucapan kata tunggal. *Discontinuous Speech* berarti satu kalimat penuh tetapi kata dipilah berdasarkan keadaan diam *(silence)*. *Isolated* dan *Continous Speech* relatif lebih mudah karena pengglan kata dapat dikenali dan kata per kata dapat diucapkan lebih “bersih”. Sedangkan *Contionous Speech* lebih sulit dilakukan karena tidak diketahui pengaglan kata dan juga dipengaruhi oleh artikulasi gabungan kata.
4) Persyaratan bahasa. Berbagai pekerjaan sulit untuk dilakukan bilamana tidak memenuhi syarat-syarat penggunaan bahasa. Termasuk didalamnya adalah Sintaks, Simantik dan Grammarnya. Sebagai contoh, *query* terhadapa jasa penerbangan dengan pernyataan ‘Pesawat itu berwarna merah’, diterima oleh

Sintaks, Simantik maupun Grammar. Sedangkan pernyataan ‘Pesawat itu sedang marah’, diterima Sintaks dan Grammar tetapi ditolak oleh Simantik. Sehingga, semua struktur bahasa perlu ditinjau dalam membangun sistem pengenalan ucapan.

5) Ucapan Spontanitas. Sistem pengenalan ucapan dapat ditinjau dari pembicara yang sedang membaca skrip atau berbicara secara spontan. Pembicara spontan seperti emm, uh, e dan lain-lain, lebih sulit dikenali.
6) Kondisi lingkungan. Kinerja system pengenalan ucapan dapat juga di uji melalui lingkungan alamiah yang terdapat banyak interferensi, derau dan lain-lain. Sehingga usaha untuk dapat mengenali ucapan dari pembicara yang dimaksudkan dengan suara bersih telah banyak diteliti dengan berbagai metode. Diantaranya, penghilangan derau, penggandaan mikrofon, pembatas lebar pita frekuensi, dan perubahan gaya ucapan.

Bahkan ada kelemahan sistem pengenalan ucapan yang mendasar, yaitu bahasa pembicaraan akan lebih efektif bila diterapkan pada interaksi antara manusia dengan komputer. Hal ini diakibatkan oleh lambatnya komputer merepresentasikan informasi dan sulitnya melakukan *review* dan *edit*. Sehingga perancangan Sistem Pengenalan Ucapan harus mengerti tentang *acoustic memory* dan *prosody*.

II.5. Konversi Sinyal Analog ke Digital

Sinyal yang terdapat pada kehidupan sehari-hari disebut dengan sinyal analog. Untuk dapat memproses sinyal analog tersebut, maka harus diubah terlebih dahulu menjadi sinyal digital atau sinyal diskrit. Sinyal analog memiliki

sifat kontinu pada domain waktu dan amplitudo, sedangkan sinyal digital memiliki sifat diskrit pada domain waktu dan amplitudo. Untuk mengubah sinyal analog menjadi sinyal digital maka diperluas proses "digitalisasi". Proses ini terdiri dari 3 tahap, yaitu :

1) Sampling

Sampling mengambil bagian sinyal analog yang kontinu dan membagi menjadi beberapa bagian diskrit dalam domain waktu yang dilakukan dengan cara periodik menganalisis sinyal analog dengan menggunakan frekuensi *sampling* tertentu. Untuk mengetahui frekuensi *sampling* yang digunakan, sesuai dengan teori *Nyquist*, frekuensi *sampling* sebaiknya 2 kali dari frekuensi tertinggi dari sinyal yang akan di analisis. Perusahaan telepon sebagai contoh menggunakan frekuensi *sampling* sebesar 8kHz sebab suara manusia berkisar antara 3-4kHz. Dengan adanya frekuensi *sampling* ini maka sinyal diatas 4kHz akan dibuang.

2) *Quantization*

Quantization akan mengubah sinyal analog menjadi bentuk digital. Tahap ini akan mengambil suatu nilai tertentu untuk mengubah bentuk digital. Sebagai contoh pada perusahaan telepon membagi sinyal analog menjadi 256 bagian. Hal ini diperlukan sebab sinyal analog memiliki kemungkinan nilai yang tidak terbatas.

3) *Coding*

Coding mengubah sinyal digital menjadi angka biner. Setelah sinyal di *sampling,* misal menggunakan frekuensi *sampling* 8kHz, maka angka

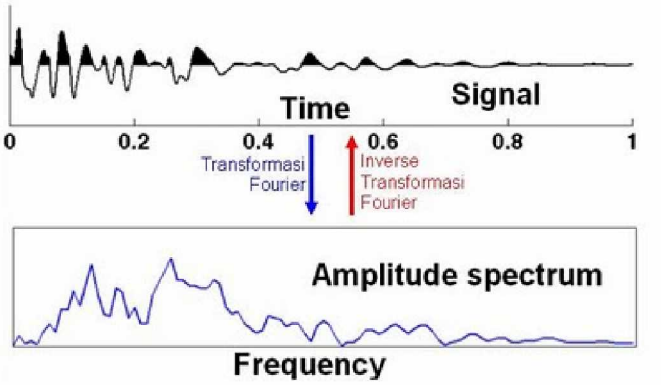
biner yang dihasilkan memiliki jangkauan antara 1 sampai dengan 256 (2^8 = 256). Dengan menggunakan frekuensi *sampling* yang semakin besar maka akan didapatkan tingkat akurasi atau ketelitian yang lebih tinggi.

Apabila frekuensi *sampling* yang dipilih lebih rendah dari frekuensi tertinggi yang terdapat pada sinyal maka terjadi suatu fenomena yang disebut dengan *aliasing* dimana fenomena tersebut muncul ketika sinyal digital diubah kembali ke sinyal analog. *Aliasing* merupakan munculnya komponen yang tidak diinginkan pada saat mengubah sinyal digital ke sinyal analog. Komponen-komponen tersebut tidak terdapat pada sinyal analog awal (*original*). Selain itu, *aliasing* juga menyebabkan beberapa bagian dari sinyal analog awal hilang pada saat diubah kembali dari sinyal digital menjadi sinyal analog.

Selain itu, frekuensi tertinggi dari sinyal terkadang hanya berupa *noise* saja. Oleh karena itu untuk mencegah agar *noise* tidak ter-*sampling* pula dan mencegah *aliasing*, maka dapat digunakan *filter* terlebih dahulu. *Filter* yang digunakan adalah *filter low-pass* karena *filter* ini akan melakukan penyaringan terhadap frekuensi atas dan membiarkan frekuensi bawah.

II.6. *Fast Fourier Transform* (FFT)

Transformasi *Fourier* adalah suatu model transformasi yang memindahkan domain spasial atau domain waktu menjadi domain frekuensi.



Gambar II.4. Transfrormasi Fourier

Transformasi *Fourier* merupakan suatu proses yang banyak digunakan untuk memindahkan domain dari suatu fungsi atau obyek ke dalam domain frekuensi. Di dalam pengolahan citra digital atau simyal suara, transformasi *fourier* digunakan untuk mengubah domain spasial pada citra/sinyal menjadi domain frekunsi. Analisa-analisa dalam domain frekuensi banyak digunakan seperti filtering. Dengan menggunakan transformasi *fourier*, sinyal suara atau citra dapat dilihat sebagai suatu obyek dalam domain frekuensi.

FFT (*Fast Fourier Transform*) adalah teknik perhitungan cepat dari DFT (*Discrete Fourier Transform*). Transformasi fourier diskrit atau disebut dengan *Discrete Fourier Transform* (DFT) adalah model transformasi fourier yang dikenakan pada fungsi diskrit, dan hasilnya juga diskrit.

FFT adalah DFT dengan teknik perhitungan yang cepat dengan memanfaatkan sifat periodikal (konjugasi) dari transformasi *fourier*. Perhatikan rumus dari DFT :

$$F(u) = \frac{1}{N}\sum_{x=0}^{x=N-1} f(x) exp\,[-2j\pi ux/N] \qquad (2.1)$$

Atau dapat ditulis dengan :

$$= \frac{1}{N} \sum_{x=0}^{x=N-1} f(x) \left(\cos\left(\frac{2\pi ux}{N}\right) - j \, sin \, \left(\frac{2\pi ux}{N}\right)\right) \qquad (2.2)$$

II.7. Algoritma *Divide and Conquer*

Algoritma *Divide and Conquer* merupakan algoritma yang sangat populer di dunia ilmu komputer. *Divide and Con*quer merupakan algoritma yang berprinsip memecah-mecah permasalahan yang terlalu besar menjadi beberapa bagian kecil sehingga mudah untuk diselesaikan. Langkah-langkah umum algoritma *Divide and Conquer* :

- *Divide* : membagi masalah menjadi beberapa sub-masalah yang memiliki kemiripan dengan masalah semula namun berukuran lebih kecil (idealnya berukuran hampir sama).
- *Conquer* : memecahkan atau menyelesaikan masing-masing sub-masalah secara rekursif.
- *Combine* : menggabungkan solusi masing-masing sub-masalah sehingga membentuk solusi masalah semula.

Objek masalah yang dibagi adalah masukan (*input*) atau *instances* yang berukuran n: tabel (*array*), matriks, dan sebagainya, bergantung pada masalahnya. Tiap-tiap sub-masalah mempunyai karakteristik yang sama (*the same type*) dengan karakteristik masalah asal, sehingga metode *Divide and Conquer* lebih natural diungkapkan dalam skema rekursif.

Sesuai dengan karakteristik pembagian dan pemecahan masalah tersebut, maka algoritma ini dapat berjalan baik pada persoalan yang bertipe rekursif

(perulangan dengan memanggil dirinya sendiri). Dengan demikian, algoritma ini dapat diimplementasikan dengan cara iteratif (perulangan biasa), karena pada prinsipnya iteratif hampir sama dengan rekursif.

Salah satu penggunaan algoritma ini yang paling populer adalah dalam hal pengolahan data yang bertipe *array* (elemen larik). Karena pada umumnya pengolahan *array* pada umumnya selalu menggunakan prinsip rekursif atau iteratif. Penggunaan secara spesifik adalah untuk mencari nilai minimal dan maksimal serta untuk mengurutkan elemen *array*. Dalam hal pengurutan ini ada empat macam algoritma pengurutan yang berdasar pada algoritma *Divide and Conquer*, yaitu *merge sort*, *insert sort*, *quick sort*, dan *selection sor*t. *Merge sort* dan *quick sort* mempunyai kompleksitas algoritma $O(n^2 log\ n)$. Hal ini lebih baik jika dibandingkan dengan pengurutan biasa dengan menggunakan algoritma *Brute Force*. Berikut skema umum algoritma *Divide and Conquer* :

```
procedure DIVIDE_n_CONQUER(input n : integer)
{Masukan: masukan yang berukuran n
 Keluaran: solusi dari maslah semula
}
Deklarasi
   r, k : integer
Algoritma
   if n ≤ n0 then {masalah sudah cukup kecil}
        SOLVE sub-masalah yang berukuran n ini
   else
        Bagi menjadi r sub-masalah, masing-
        masing berukuran n/k
        for masing-masing dari r sub-masalah
        do
             DIVIDE_n_CONQUER(n/k)
```

```
        endfor
        COMBINE solusi dari r sub-masalah
        menjadi solusi masalah semula
    endif
```

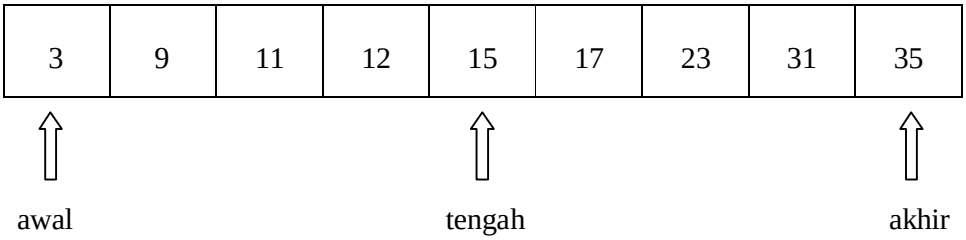
II.7.1. *Binary Search* Algoritma *Divide and Conquer*

Binary Search (Pencarian Biner) dapat dilakukan jika data sudah dalam keadaan urut. Dengan kata lain, apabila data belum dalam keadaan urut, pencarian biner tidak dapat dilakukan. Dalam kehidupan sehari-hari, sebenarnya kita juga sering menggunakan pencarian biner. Misalnya saat ingin mencari suatu kata dalam kamus.

Prinsip dari pencarian biner dapat dijelaskan sebagai berikut :

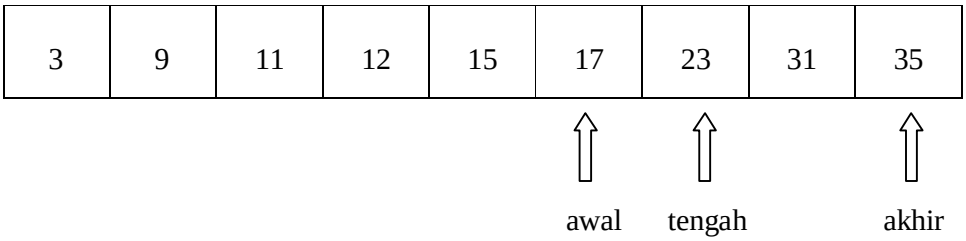
1. Mula-mula diambil posisi awal = 1 dan posisi akhir = N
2. Cari posisi data tengah dengan rumus (posisi awal + posisi akhir) / 2
3. Data yang dicari dibandingkan dengan data tengah.
4. Jika lebih kecil, proses dilakukan kembali tetapi posisi akhir dianggap sama dengan posisi tengah – 1.
5. Jika lebih besar, proses dilakukan kembali tetapi posisi awal dianggap sama dengan posisi tengah + 1.
6. Demikian seterusnya sampai data tengah sama dengan yang dicari.

Untuk lebih jelasnya, perhatikan contoh berikut. Misalkan kita ingin mencari 17 pada sekumpulan data berikut :

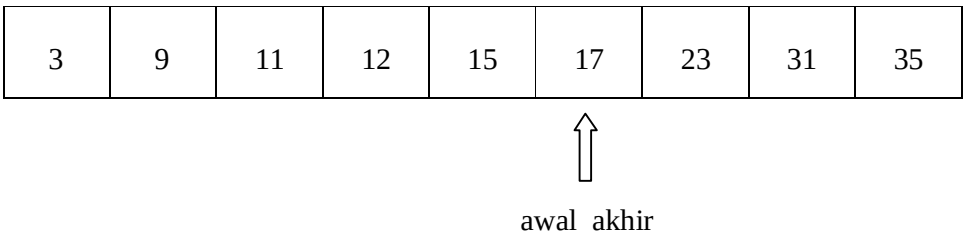


1. Mula–mula dicari data tengah, dengan rumus (1+ 9) / 2 = 5.
2. Berarti data tengah adalah data ke-5, yaitu 15.
3. Data yang dicari, yaitu 17, dibandingkan dengan data tengah ini.

4. Karena 17 > 15, berarti proses dilanjutkan tetapi kali ini posisi awal dianggap sama dengan posisi tengah + 1 atau 6.



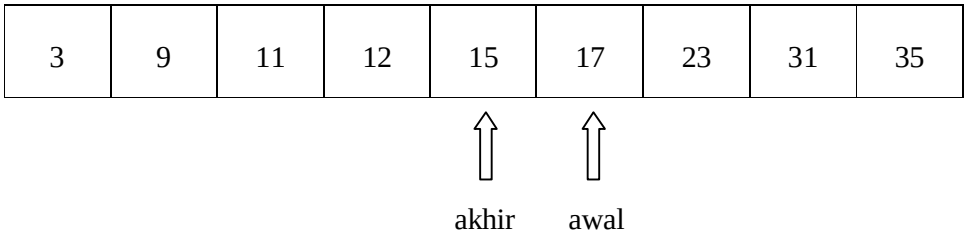
1. Data tengah yang baru didapat dengan rumus (6 + 9) / 2 = 7. Berarti data tengah yang baru adalah data ke-7, yaitu 23.
2. Data yang dicari, yaitu 17 dibandingkan dengan data tengah ini.
3. Karena 17 < 23, berarti proses dilanjutkan tetapi kali ini posisi akhir dianggap sama dengan posisi tengah – 1 atau 6.



1. Data tengah yang baru didapat dengan rumus (6 + 6) / 2 = 6. Berarti data tengah yang baru adalah data ke-6, yaitu 17.

2. Data yang dicari dibandingkan dengan data tengah ini dan ternyata sama. Jadi data ditemukan pada indeks ke-6.
3. Bagaimana jika data yang dicari tidak ada, misalnya 16?
4. Pencarian biner ini akan berakhir jika data ditemukan atau posisi awal lebih besar dari posisi akhir.
5. Jika posisi awal sudah lebih besar daripada posisi akhir berarti data tidak ditemukan.

Untuk lebih jelasnya perhatikan proses pencarian 16 pada data di atas. Prosesnya hampir sama dengan pencarian 17. Tetapi setelah posisi awal = posisi akhir = 6, proses masih dilanjutkan lagi dengan posisi awal = 6 dan posisi akhir=5.



Disini dapat dilihat bahwa posisi awal lebih besar daripada posisi akhir, yang artinya data tidak ditemukan.

Secara umum, algoritma pencarian biner dapat dituliskan sebagai berikut :

1. l ← 1.
2. r ← N.
3. ketemu ← false.
4. selama (l < = r) dan (not ketemu) kerjakan baris 5 sampai dengan 8.
5. m ← (l + r) / 2

6. Jika (Data [m] = x) maka ketemu ← true.
7. Jika (x < Data [m]) maka r ← m – 1.
8. Jika (x > Data [m]) maka l ← m + 1.
9. If (ketemu) maka m adalah indeks dari data yang dicari, jika tidak data tidak ditemukan.

II.7.2. Algoritma *Divide and Conquer* Dalam Pencocokan Suara

Algoritma *Divide and Conquer* dapat diiplementasikan untuk masalah pencocokan pola suara. Proses awalnya ialah mengkonversi data spektrum suara ke dalam bentuk diskrit. Dari bentuk diskrit itulah yang kemudian diambil nilai fiturnya lalu mengaplikasikan algoritma *Divide and Conquer* untuk mengoptimalkan waktu pencocokan (kompleksitas berkurang). Ketika mengkonversi gelombang suara ke bentuk diskrit, gelombang diperlebar dengan cara memperinci berdasarkan frekuensi. Hal ini dilakukan agar proses algoritma selanjutnya lebih mudah dilakukan. Namun, efek buruknya ialah *array of array* data yag terbentuk akan lebih banyak. Dari tiap elemen array data tersebut, diambil ciri fiturnya. Ciri fitur tersebut yang nantinya akan dibandingkan dengan *template* data suara.

Proses *Divide and Conquer* :

1) Bagi dua proses pecocokan menjadi beberapa sub-masalah.
2) Urutkan data fitur suara, bagi dua data fitur suara, ambil bagian yang terkecil atau terbesar dari data yang masuk.
3) Untuk setiap elemen data pada *template*, dicocokan dengan data baru menggunakan *binary search Divide and Conquer*.

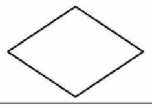
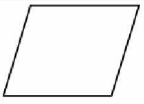
4) Setiap sub-masalah disatukan kembali dan dianalisis, kecocokan dari segi data fitur dan apakah data yang diucapkan sesuai dengan kata yang tersedia pada *template* data.
5) Hitung nilai peramalan *error* terkecil, yang terkecil adalah solusi atau *output* yang dihasilkan.
6) Verifikasi data, jika sesuai maka akan masuk ke proses selanjutnya, sesuai dengan aplikasi yang mengimplementasikan algoritma ini.

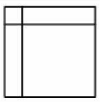
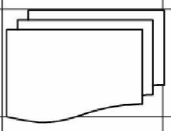
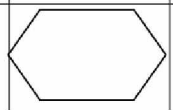
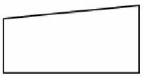
II.8. Pemodelan Sistem

II.8.1. Bagan Alir Sistem *(Flow Chart)*

Bagan alir sistem (*flowchart*) merupakan bagan yang menunjukkan arus pekerjaan dari sistem secara keseluruhan, menjelaskan urutan dari prosedur-prosedur yang ada di dalam sistem serta menunjukkan apa yang dikerjakan di dalam sistem.

Tabel II.1. Simbol - simbol Flowchart

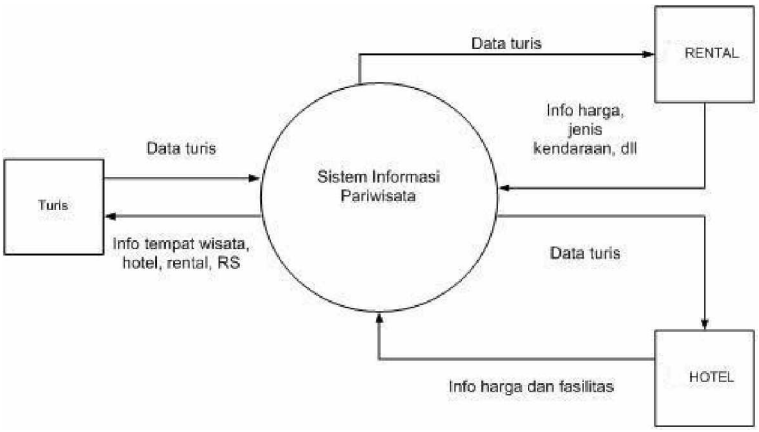
No	Simbol	Nama	Deskripsi
		- Process - Proses	- An operation or action step - Langkah operasi atau tindakan -
		- Alternate process - Proses alternatif	- An alternate to the normal process step -
		- Decision	- A question or branch in the process - Proses pertanyaan atau cabang - kondisi yang akan menghasilkan beberapa kemungkinan jawaban/aksi
		- Data I/O	- Indicates data inputs and outputs to and from process - Proses memasukkan atau mengeluarkan data
		- Predifined process	- A formally defined sub-process - Sub proses yang ditetapkan

		- Internal storage	- Data storage in memory - Penyimpanan data
		- Document	- A document or report - Sebuah dokumen atau laporan - symbol yang menyatakan input berasal dari dokumen dalam bentuk kertas atau output dicetak ke kertas
		- Multi-Document	- Same as document, except, well, multiple documents - Dokumen yang sama, dokumen rangkap
		- Terminator	- A strart or stop point in process - Permulaan atau akhir proses
		- Preparation	- A preparation or set-up process step - Langkah persiapan atau proses set-up
		- Manual Input	- Manually input into a system - Memasukkan data secara manual

II.8.2. Diagram Konteks

Yaitu diagram tingkat atas, merupakan diagram dari sebuah sistem yang menggambarkan aliran-aliran data yang masuk dan keluar dari sistem dan dari entitas luar. Hal yang harus diperhatikan adalah :

1) Memberikan gambaran tentang seluruh sistem.
2) Terminal yang memberikan masukan ke sistem disebut *source*.
3) Terminal yang telah menerima keluaran disebut *sink*.
4) Hanya ada satu proses.
5) Tidak boleh ada data *store*.



Gambar II.5. Contoh Diagram Konteks

II.8.3. Data Flow Diagram (DFD)

DFD adalah suatu model logika data atau proses yang dibuat untuk menggambarkan darimana asal data dan kemana tujuan data yang keluar sistem, dimana data disimpan, proses apa yang menghasilkan data tersebut dan interaksi antara data yang tersimpan dan proses yang akan dikenakan pada data tersebut.

DFD sering digunakan untuk menggambarkan suatu sistem yang telah ada atau sistem baru yang akan dikembangkan sacara logika tanpa mempertimbangkan lingkungan fisik dimana data tersebut mengalir (misalnya lewat telepon, surat dan sebagainya). Atau lingkungan fisik dimana data tersebut akan disimpan (misalnya file kartu, *hard disk*, *tape*, disket dan sebagainya). DFD merupakan alat yang cukup populer saat ini, karena dapat menggambarkan arus data didalam sistem dengan terstruktur dan jelas. Lebih lanjut DFD merupakan dokumentasi dari sistem yang baik. Beberapa simbol yang akan digunakan di dalam DFD antara lain adalah sebagai berikut :

1. Kesatuan luar (*External Entity*)

   Setiap sistem pasti mempunyai batas sistem yang memisahkan suatu sistem dengan lingkungan luarnya. Sistem akan menerima *input* dan menghasilkan *output* kepada lingkungan luarnya. Kesatuan luar (*external entity*) merupakan kesatuan dilingkungan luar sistem dapat berupa orang, organisasi atau sistem lainnya yang berada dilingkungan luarnya yang akan memberikan *input* atau menerima *output* dari sistem. Kesatuan luar ini kebanyakan adalah salah satu dari berikut ini:

   a. Suatu kantor, departemen atau divisi dalam perusahaan tetapi di luar sistem yang sedang dikembangkan.
   b. Orang atau sekelompok orang di organisasi tetapi di luar sistem yang sedang dikembangkan.
   c. Suatu organisasi atau orang di luar organisasi.
   d. Sistem informasi yang lain di luar sistem yang sedang dikembangkan.
   e. Sumber asli dari suatu transaksi.
   f. Penerimaan akhir dari suatu laporan yang dihasilkan oleh sistem.

2. Aliran Data *(Data Flow)*

   Aliran data di DFD diberi simbol suatu panah. Aliran data ini mengalir diantara proses (*process*), simpan data (*data store*) dan kesatuan luar (*external entity*). Aliran data ini menunjukkan aliran dari data yang dapat berupa masukkan untuk sistem atau hasil dari proses sistem.

3. Proses (*Process*)

   Proses adalah kegiatan atau kerja yang dilakukan oleh orang, mesin atau komputer dari hasil suatu arus data yang masuk ke dalam proses untuk dihasilkan arus data yang akan keluar dari proses yang digambarkan secara umum. Suatu proses dapat ditunjukkan dengan simbol lingkaran atau dengan simbol empat persegi panjang tegak dengan sudut – sudutnya tumpul.

4. Berkas atau Simpanan Data (*Data Store*)

   Berkas atau simpanan data merupakan simpanan dari data yang dapat berupa :

   a) Suatu file atau database di sistem komputer.

   b) Suatu arsip atau catatan manual.

II.9. Metode Pengembangan Perangkat Lunak

Merupakan sebuah model *chaos* yang menggambarkan "perkembangan P/L sebagai sebuah kesatuan dari pemakai ke pengembang dan ke teknologi" disebut dengan "*prescriptive*" karena Menentukan sekumpulan elemen proses (aktivitas, aksi, tugas, produk kerja, jaminan kualitas, dll untuk setiap proyek), setiap model proses juga menentukan alur kerjanya.

Pada saat kerja bergerak maju menuju sebuah sistem yang lengkap, keadaan yang digambarkan secara rekursif diaplikasikan kepada kebutuhan pemakai dan spesifikasi teknis P/L pengembang Saat ini, *prescriptive* memberikan

jawaban secara *definitive* untuk masalah pengembangan P/L dalam setiap perubahan lingkungan komputasi

II.9.1. Model Sekuensial Linear/Terstruktur

Pada Tugas akhir ini menggunakan metode penggunaan perangkat lunak siklus kehidupan klasik atau model air terjun (*waterfall*). Model ini mengusulkan sebuah pendekatan kepada perkembangan *software* yang sistematik dan sekuensial yang mulai pada tingkat dan kemajuan sistem pada seluruh analisis, desain, kode, pengujian, dan pemeliharaan. Dimodelkan setelah siklus rekayasa konvensional, model sekuensial linier melingkupi aktivitas – aktivitas sebagai berikut :

1. Rekayasa dan pemodelan sistem/informasi

Karena sistem merupakan bagian dari sebuah sistem yang lebih besar, kerja dimulai dengan membangun syarat dari semua elemen sistem dan mengalokasikan beberapa subset dari kebutuhan ke *software* tersebut. Pandangan sistem ini penting ketika *software* harus berhubungan dengan elemen-elemen yang lain seperti *software*, manusia, dan database. Rekayasa dan anasisis system menyangkut pengumpulan kebutuhan pada tingkat sistem dengan sejumlah kecil analisis serta disain tingkat puncak. Rekayasa informasi mancakup juga pengumpulan kebutuhan pada tingkat bisnis strategis dan tingkat area bisnis.

2. Analisis kebutuhan *Software*

Proses pengumpulan kebutuhan diintensifkan dan difokuskan, khusunya pada *software*. Untuk memahami sifat program yang dibangun, analis harus memahami domain informasi, tingkah laku, unjuk kerja, dan *interface* yang

diperlukan. Kebutuhan baik untuk sistem maupun software didokumentasikan dan dilihat lagi dengan pelanggan.

3. Desain

Desain *software* sebenarnya adalah proses multi langkah yang berfokus pada empat atribut sebuah program yang berbeda; struktur data, arsitektur *software*, representasi *interface*, dan detail (algoritma) prosedural. Proses desain menterjemahkan syarat/kebutuhan ke dalam sebuah representasi *software* yang dapat diperkirakan demi kualitas sebelum dimulai pemunculan kode. Sebagaimana persyaratan, desain didokumentasikan dan menjadi bagian dari konfigurasi *softwar*e.

4. Generasi Kode

Desain harus diterjemahkan kedalam bentuk mesin yang bias dibaca. Langkah pembuatan kode melakukan tugas ini. Jika desain dilakukan dengan cara yang lengkap, pembuatan kode dapat diselesaikan secara mekanis.

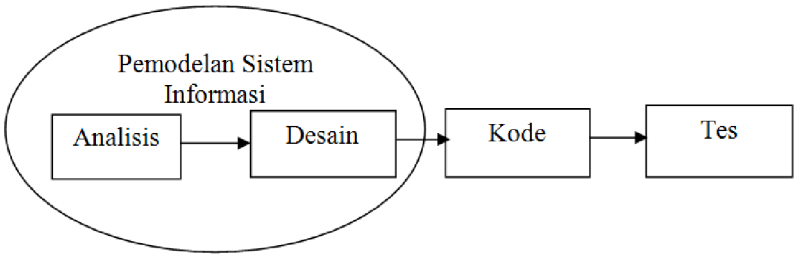
5. Pengujian

Sekali program dibuat, pengujian program dimulai. Proses pengujian berfokus pada logika internal *software*, memastikan bahwa semua pernyataan sudah diuji, dan pada eksternal fungsional, yaitu mengarahkan pengujian untuk menemukan kesalahan – kesalahan dan memastikan bahwa input yang dibatasi akan memberikan hasil aktual yang sesuai dengan hasil yang dibutuhkan.

6. Pemeliharaan

Software akan mengalami perubahan setelah disampaikan kepada pelanggan (perkecualian yang mungkin adalah software yang dilekatkan).

Perubahan akan terjadi karena kesalahan – kesalahan ditentukan, karena *software* harus disesuaikan untuk mengakomodasi perubahan – perubahan di dalam lingkungan eksternalnya (contohnya perubahan yang dibutuhkan sebagai akibat dari perangkat peripheral atau sistem operasi yang baru), atau karena pelanggan membutuhkan perkembangan fungsional atau unjuk kerja. Pemeliharaan *software* mengaplikasikan lagi setiap *fase* program sebelumnya dan tidak membuat yang baru lagi.



Gambar II.6. Model Sekuensial Linear

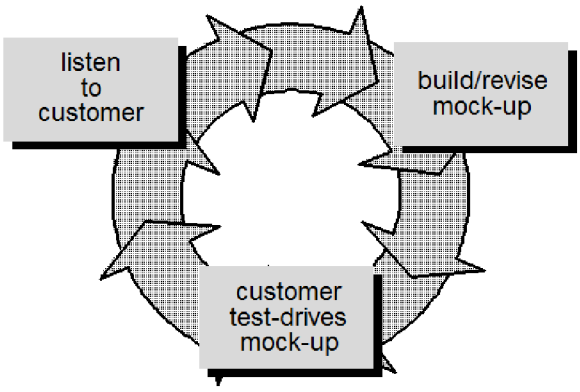
Masalah yang kadang terjadi ketika model sekuensial linier diaplikasikan adalah :

1. Jarang sekali proyek nyata mengikuti aliran sekuensial yang dianjurkan oleh model. Meskipun model linier bisa mengakomodasi iterasi, model ini melakukannya dengan cara tidak langsung. Sebagai hasilnya, perubahan – perubahan dapat menyebabkan keraguan pada saat tim proyek berjalan.
2. Kadang – kadang sulit bagi pelanggan untuk menyatakan semua kebutuhannya secara eksplisit. Model linier sekuensial memerlukan hal ini dan mengalami kesulitan untuk mengakomodasi ketidakpastian natural yang ada pada bagian awal beberapa proyek.

3. Pelanggan harus bersifat sabar. Sebuah versi kerja dari program – program kerja itu tidak akan diperoleh sampai akhir waktu proyek dilalui. Sebuah kesalahan besar, jika tidak terdeteksi sampai program yang bekerja tersebut dikaji ulang, bisa menjadi petaka.
4. Pengembang sering melakukan penundan yang tidak perlu. Sifat alami dari siklus kehidupan klasik membawa kepada *blocking state* di mana banyak anggota tim proyek harus menunggu tim yang lain untuk melengkapi tugas yang saling memiliki ketergantungan. *Blocking state* cenderung menjadi lebih lazim pada awal dan akhir sebuah proses sekuensial linier.

II.9.2. Model Prototype

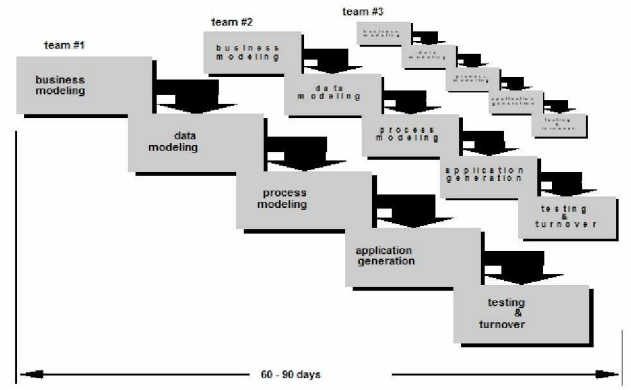
Model *prototype* dibangun dari mengumpulkan berbagai kebutuhan, kemudian tim pengembang akan nertemu dengan pelanggan untuk menentukan tujuan dari perangkat lunak, dan mengidentifikasi kebutuhan-kebutuhan yang telah diketahui oleh pelanggan, dan batasan-batasan apa saja yang dapat dikategorikan sebagai tugas utama. Hasilnya akan dibangun rancangan sementara yang mewakili berbagai aspek dari perangkat lunak yang kelak akan digunakan oleh pelanggan/pengguna (seperti bentuk pendekatan input yang digunakan dan bentuk *output*). Idealnya model *prototype* melayani sebuah mekanisme untuk mengidentifikasi kebutuhan perangkat lunak. Dimana jika nantinya sebuah model *prototype* berhasil dibuat, seorang *developer* harus berusaha mendayagunakan *tools* yang ada (semisal, *report generator, windows manager*) dapat bekerja dengan baik (cepat).



Gambar II.7. Model Prototype

II.9.3. Model RAD

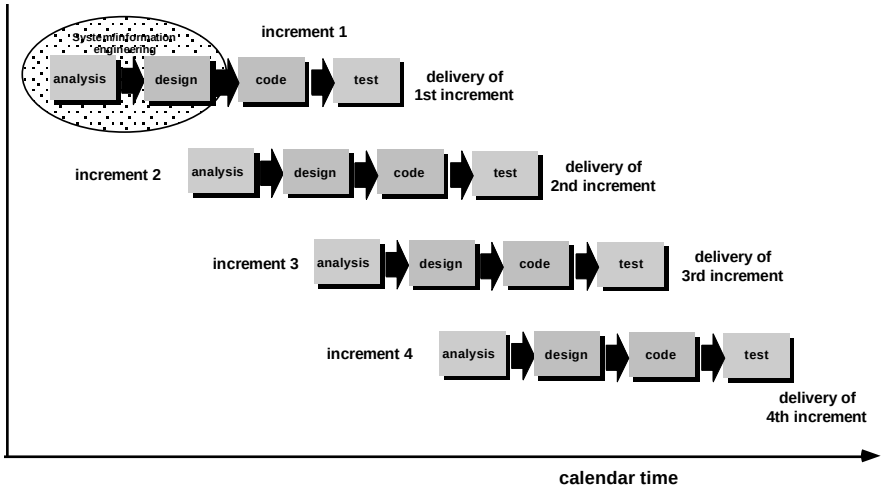
Rapid Aplication Development (RAD) adalah sebuah model proses perkembangan perangkat lunak sekuensial linier yang menekankan siklus perkembangan yang sangat pendek. Model RAD ini merupakan sebuah adaptasi "kecepatan tinggi" dari model sequensial linier dimana perkembangan cepat dicapai dengan menggunakan pendekatan konstruksi berbasis komponen.



Gambar II.8. Model RAD

II.9.4. Model Evolusioner

Model evolusioner adalah model iterative. Model itu ditandai dengan tingkah laku yang memungkinkan perekayasa perangkat lunak mengembangkan versi perangkat yang lebih lengkap sedikit demi sedikit.



Gambar II.9. Model Evolusioner

II.10. MATLAB *(Matrix Laboratory)*

Bahasa pemograman sebagai media untuk berinteraksi antara manusia dan komputer saat dibuat semakin mudah dan cepat. Sebagai contoh, dapat dilihat dari perkembangan bahasa pemograman *Pascal* yang terus memunculkan varian baru sehingga akhirnya menjadi *Delphi*, demikian pula dengan *Visual Basic* serta *C* dengan *C++ Builder*. Pada akhirnya semua bahasa pemograman akan semakin memberikan kemudahan pemakainya *(programmer)* dengan penambahan fungsi-fungsi baru yang sangat mudah digunakan bahkan oleh pemakai tingkat pemula.

Matlab muncul di dunia bahasa pemrograman yang cenderung dikuasai oleh bahasa yang telah mapan. Tentu saja sebagai bahasa pemrograman yang baru

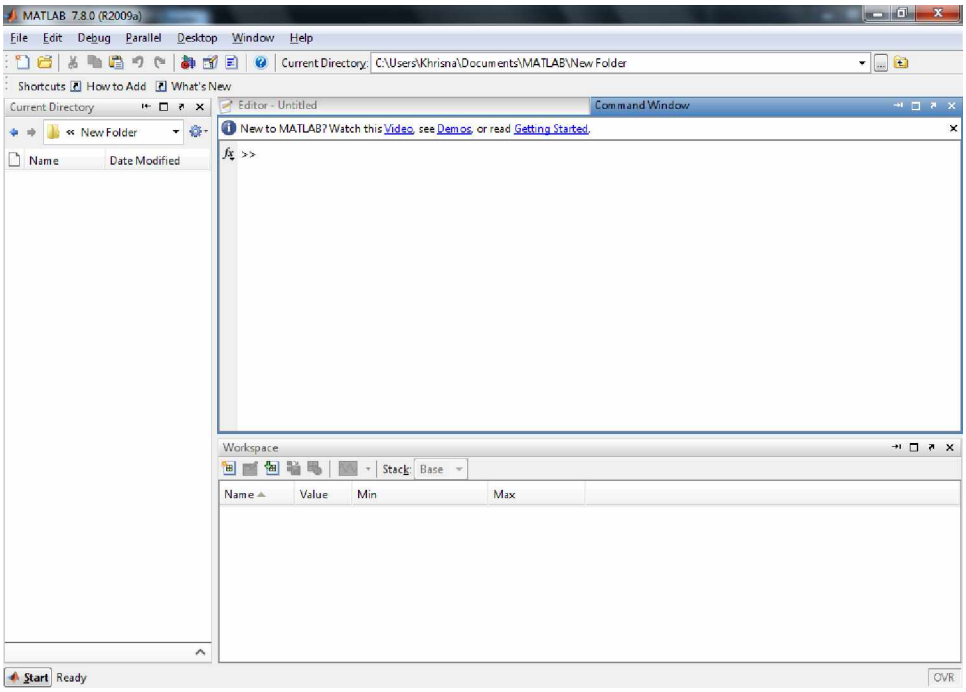
Matlab akan sukar mendapat hati dari pemakai. Namun *Matlab* hadir tidak dengan fungsi dan karakteristik yang umumnya ditawarkan bahasa pemrograman lain yang biasanya hampir seragam. *Matlab* dikembangkan sebagai bahasa pemrograman sekaligus alat visualisasi, yang menawarkan banyak kemampuan untuk menyelesaikan berbagai kasus yang berhubungan langsung dengan disiplin keilmuan matematika. *Matlab* memiliki kemampuan mengintegrasikan komputasi, visualisasi, dan pemrograman dalam sebuah lingkungan yang tunggal dan mudah digunakan. *Matlab* menyediakan beberapa pilihan untuk dipelajari, mempelajari metode visualisasi saja, pemograman saja, atau kedua-duanya.

Matlab adalah bahasa pemograman level tinggi yang dikhususkan untuk komputasi teknis. Bahasa ini mengintegrasikan kemampuan komputasi, visualisasi dan pemrograman dalam sebuah lingkungan yang tunggal dan mudah digunakan. *Matlab* memberikan sistem interaktif yang menggunakan konsep *array* tanpa membutuhkan pendeklarasian *array* seperti pada bahasa pemrograman yang lain. Selain itu, *Matlab* juga memberikan kemudahan bagi *programmer/developer* program yaitu untuk menjadi pembanding yang sangat handal, hal tersebut dapat dilakukan karena kekayaannya akan fungsi matematika, fisika, statistika, dan visualisasi

II.10.1. Lingkungan Kerja MATLAB

Sebagaimana bahasa pemrograman lainnya, *Matlab* juga menyediakan lingkungan kerja terpadu yang sangat mendukung dalam membangun sebuah aplikasi. Pada setiap versi *Matlab* terbaru, lingkungan terpadu akan semakin dilengkapi. Lingkungan terpadu ini terdiri dari beberapa *form* yang memiliki

kegunaan masing-masing. Setiap pertama kali membuka aplikasi *Matlab*, maka akan menyimpan *mode/setting* terakhir lingkungan kerja yang digunakan sebagai mode/setting lingkungan kerja pada saat membuka palikasi *Matlab* di waktu berikutnya.



Gambar II.10. Tampilan Utama MATLAB

II.10.2. GUIDE MATLAB

GUIDE atau *GUI Builder* merupakan sebuah *Graphical User Interface* (GUI) yang dibangun dengan objek grafis seperti tombol (*pushbutton*), *edit, slider, text, combo*, sumbu (*axes*), maupun menu dan lain-lain utnuk kita gunakan. Sebagai contoh, ketika menggerakan *slider*, maka akan bisa melihat perubahan sebuah nilai. Kemudian ketika menekan tombol OK, maka aplikasi akan dijalankan. Aplikasi yang menggunakan GUI umumnya lebih mudah dipelajari

dan digunakan karena orang yang menjalankannya tidak perlu mengetahui perintah yang ada dan bagaimana perintah bekerja.

Tidak seperti bahasa pemrograman lainnya, *GuideMatlab* memiliki banyak keunggulan tersendiri, antara lain :

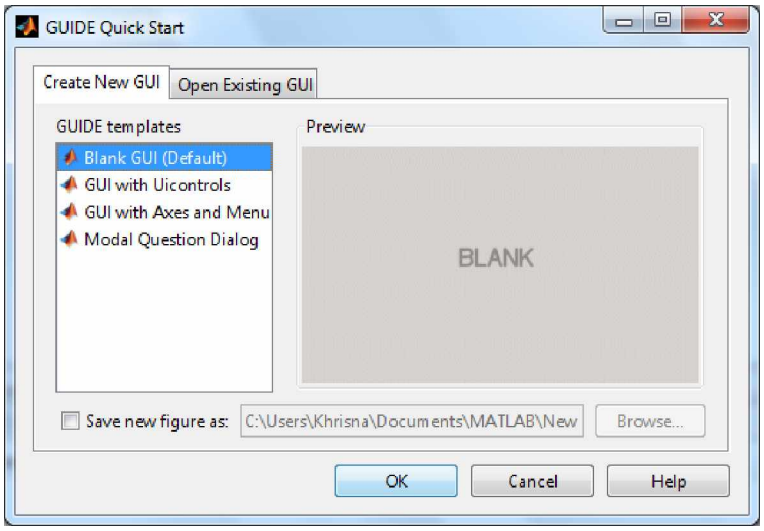
1. *GuideMatlab* cocok untuk aplikasi-aplikasi berorientasi sains.
2. *Matlab* memiliki banyak fungsi *built-in* yang siap digunakan dan pemakai tidak perlu repot membuatnya sendiri.
3. Ukuran file, baik Fig-file maupun M-file yang dihasilkan relatif kecil.
4. Kemampuan grafisnya cukup handal dan tidak kalah dengan bahasa pemrograman yang lainnya.

II.10.2.1. Memulai GUIDE

Untuk memulai penggunaan GUI *Matlab*, dapat dilakukan dengan dua cara yaitu :

1. Melalui command *Matlab* kita ketikan >>guide, atau
2. Klik tombol Start pada window utama dan pilih *Matlab*, lalu pilih GUIDE (*GUI Builder*).

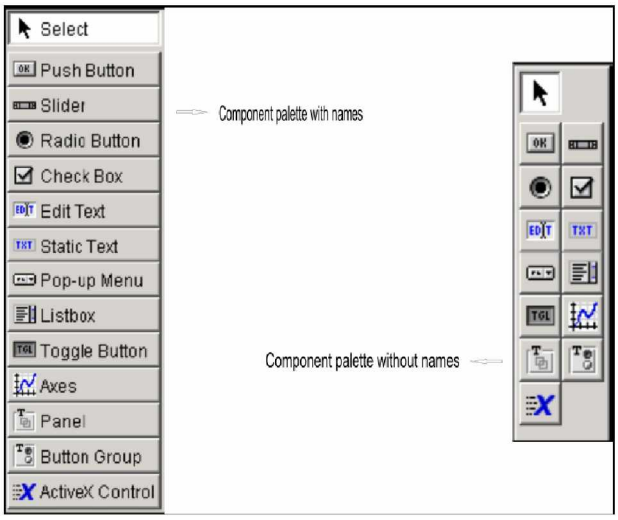
Selanjutnya, akan dibawa ke sebuah kotak dialog pilihan GUIDE *Quick Start*.



Gambar II.11. GUIDE Quick Start MATLAB

II.10.2.2. Komponen GUIDE

Untuk membuat sebuah *Graphical User Interface Matlab* dengan fasilitas GUIDE, dimulai dengan membuat desain sebuah *figure*. Untuk membuat sebuah desain *figure*, dapat memanfaatkan *Uicontrol* (*control user interface*) yang telah tersedia pada editor *figure*. Banyak sekali *control user interface* (komponen) yang ada pada GUIDE *Matlab*.

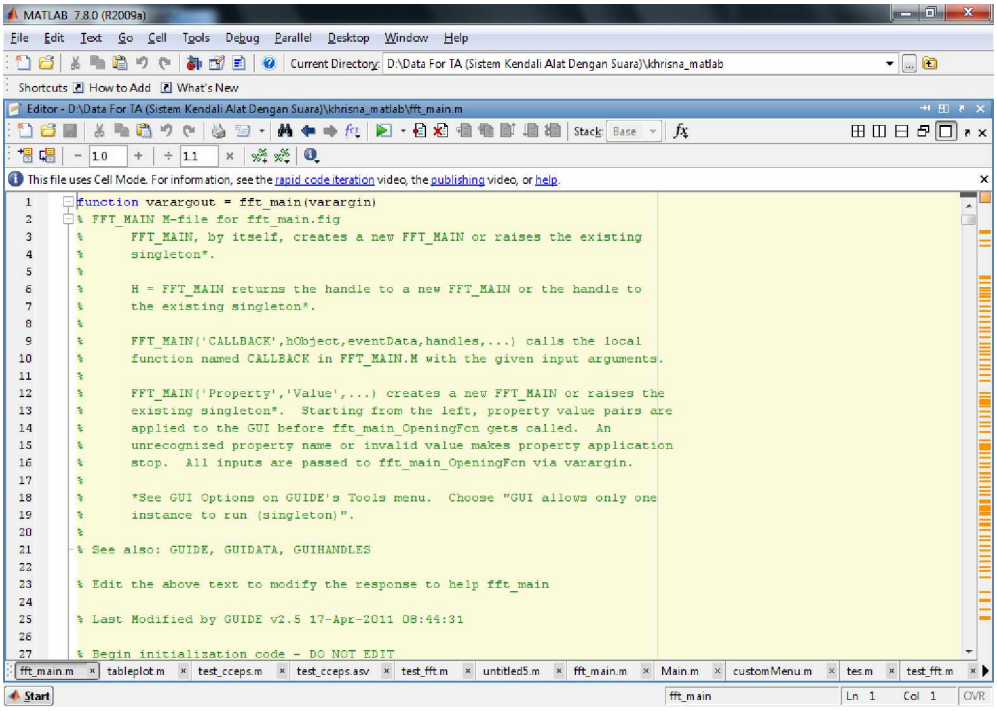


Gambar II.12. Komponen GUIDE Control MATLAB

II.10.3. Aplikasi *M-File*

Dalam membuat program GUI, matlab membuat program aplikasi berupa *M-file* yang menyediakan kerangka untuk mengontrol GUI. Kerangka dapat membantu membuat program menjadi lebih efisien dan sempurna. Semua kode termasuk kode *callback* dimasukan dalam sebuah subfungsi dalam *M-file*. Pendekatan ini memungkinkan *M-file* memiliki titik masukan tunggal (*single entry point*) yang dapat memanggil *callback* yang sesuai, atau sebuah subfungsi penolong yang ingin digunakan dalam GUI.

Fungsi M-file mirip dengan *script* M-file dimana keduanya merupakan file teks dengan ekstensi (.m). Sebagaimana *script M-file*, fungsi *M-file* tidak dimasukan ke dalam jendela *command*, tetapi merupakan suatu file tersendiri yang dibuat dengan editor teks. Fungsi *M-file* menyediakan cara sederhana untuk menambah kemampuan *Matlab*, bahkan banyak fungsi standar *Matlab* yang merupakan fungsi *M-file*.



Gambar II.13. Contoh Bagian M-file Matlab

Aplikasi *M-file* mengerjakan sesuatu yang berbeda tergantung pada argumen yang dilewatkan ketika dipanggil. Subfungsi yang ditambahkan ke dalam GUIDE memerlukan argumen tertentu, tetapi memiliki daftar panjang argumen variabel. Karena argumen terakhir adalah varargin, maka akan dapat menambahkan argumen apapun yang diinginkan ke dalam subfungsi. Perlu diperhatikan, jika menggunakan aplikasi *M-file* yang dibuat oleh GUIDE, maka Fig-file dan *M-file* harus disimpan dengan nama yang sama.

II.11. Mikrokontroler ATmega8535

Mikrokontroler adalah IC yang dapat diprogram berulang kali, baik ditulis atau dihapus. Biasanya digunakan untuk pengontrolan otomatis dan manual pada perangkat elektronika.

Beberapa tahun terakhir, mikrokontroler sangat banyak digunakan terutama dalam pengontrolan robot. Seiring perkembangan elektronika, mikrokontroler dibuat semakin kompak dengan bahasa pemograman yang juga ikut berubah. Salah satunya adalah mikrokontroler AVR *(Alf and Vegard's Risc processor)* ATmega8535 yang menggunakan teknologi RISC *(Reduce Instruction Set Computing)* dimana program berjalan lebih cepat karena hanya membutuhkan *siklus clock* untuk mengeksekusi satu instruksi program.

Secara umum, AVR dapat dikelompokan menjadi 4 kelas yaitu kelas ATtiny, keluarga AT90Sxx, keluarga ATmega, dan AT86RFxx. Pada dasarnya yang membedakan masing-masing kelas adalah memori, peripheral, dan fungsinya. Dari segi arsitektur dan instruksi yang digunakan, mereka bisa dikatakan hampir sama.

Mikrokontroler AVR ATmega8535 memiliki fitur yang cukup lengkap. Mikrokontroler AVR ATmega8535 telah dilengkapi dengan ADC *internal*, EEPROM *internal*, *Timer/Counter*, PWM, *analog comparator*, dll. Sehingga dengan fasilitas yang lengkap ini memungkinkan kita belajar mikrokontroler keluarga AVR dengan lebih mudah dan efisien, serta dapat mengembangkan kreativitas penggunaan mikrokontroler ATmega8535.

Fitur-fitur yang dimiliki oleh mikrokontroler ATmega8535 adalah sebagai berikut:

1) Saluran I/O sebanyak 32 buah, yaitu port A, port B, port C, dan port D.
2) ADC internal sebanyak 8 saluran.
3) Tiga buah *Timer/Counter* dengan kemampuan pembandingan.

4) CPU yang terdiri atas 32 buah register.
5) SRAM sebesar 512 byte.
6) Memori Flash sebesar 8 kb dengan kemampuan *Read While Write.*
7) Port antarmuka SPI.
8) EEPROM sebesar 512 byte yang dapat diprogram saat operasi.
9) Antarmuka komparator analog.
10) Port USART untuk komunikasi serial.
11) Sistem mikroprosesor 8 bit berbasis RISC dengan kecepatan maksimal 16 MHz.
12) Dan lain-lainnya.

II.11.1. Konstruksi ATmega8535

Mikrokontroler ATmega8535 memiliki 3 jenis memori, yaitu memori program, memori data dan memori EEPROM. Ketiganya memiliki ruang sendiri dan terpisah.

a) Memori program

ATmega8535 memiliki kapasitas memori progam sebesar 8 Kbyte yang terpetakan dari alamat 0000h – 0FFFh dimana masing-masing alamat memiliki lebar data 16 bit. Memori program ini terbagi menjadi 2 bagian yaitu bagian program *boot* dan bagian program aplikasi.

b) Memori data

ATmega8535 memiliki kapasitas memori data sebesar 608 byte yang terbagi menjadi 3 bagian yaitu register serba guna, *register I/O* dan SRAM. ATmega8535 memiliki 32 byte register serba guna, 64 byte *register I/O* yang

dapat diakses sebagai bagian dari memori RAM (menggunakan instuksi LD atau ST) atau dapat juga diakses sebagai *I/O* (menggunakan instruksi *IN* atau *OUT*), dan 512 byte digunakan untuk memori data SRAM.

c) Memori EEPROM

ATmega8535 memiliki memori EEPROM sebesar 512 byte yang terpisah dari memori program maupun memori data. Memori EEPROM ini hanya dapat diakses dengan menggunakan register-register *I/O* yaitu register EEPROM *Address*, *register* EEPROM *Data*, dan register EEPROM *Control*. Untuk mengakses memori EEPROM ini diperlakukan seperti mengakses data eksternal, sehingga waktu eksekusinya relatif lebih lama bila dibandingkan dengan mengakses data dari SRAM.

ATmega8535 merupakan tipe AVR yang telah dilengkapi dengan 8 saluran ADC internal dengan fidelitas 10 bit. Dalam mode operasinya, ADC ATmega8535 dapat dikonfigurasi, baik secara *single ended input* maupun *differential input*. Selain itu, ADC ATmega8535 memiliki konfigurasi pewaktuan, tegangan referensi, mode operasi, dan kemampuan filter derau yang amat fleksibel, sehingga dengan mudah disesuaikan dengan kebutuhan ADC itu sendiri.

ATmega8535 memiliki 3 modul *timer* yang terdiri dari 2 buah *timer/counter* 8 bit dan 1 buah *timer/counter* 16 bit. Ketiga modul *timer/counter* ini dapat diatur dalam mode yang berbeda secara individu dan tidak saling mempengaruhi satu sama lain. Selain itu, semua *timer/counter* juga dapat difungsikan sebagai sumber interupsi. Masing-masing *timer/counter* ini memiliki register tertentu yang digunakan untuk mengatur mode dan cara kerjanya.

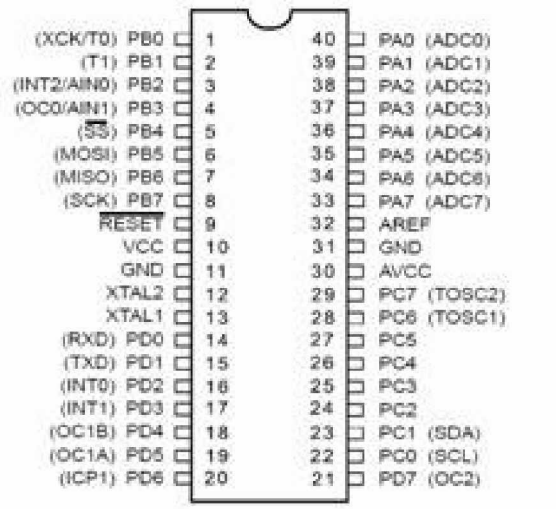
Serial Peripheral Interface (SPI) merupakan salah satu mode komunikasi serial *syncrhronous* kecepatan tinggi yang dimiliki oleh ATmega8535. *Universal Syncrhronous and Asyncrhronous Serial Receiver and Transmitter* (USART) juga merupakan salah satu mode komunikasi serial yang dimiliki oleh ATmega8535. USART merupakan komunikasi yang memiliki fleksibilitas tinggi, yang dapat digunakan untuk melakukan transfer data baik antar mikrokontroler maupun dengan modul-modul eksternal termasuk PC yang memiliki fitur UART.

USART memungkinkan transmisi data baik secara *syncrhronous* maupun *asyncrhronous*, sehingga dengan memiliki USART pasti kompatibel dengan UART. Pada ATmega8535, secara umum pengaturan mode *syncrhronous* maupun *asyncrhronous* adalah sama. Perbedaannya hanyalah terletak pada sumber *clock* saja.

Jika pada mode *asyncrhronous* masing-masing peripheral memiliki sumber *clock* sendiri, maka pada mode *syncrhronous* hanya ada satu sumber *clock* yang digunakan secara bersama-sama. Dengan demikian, secara hardware untuk mode *asyncrhronous* hanya membutuhkan 2 pin yaitu TXD dan RXD, sedangkan untuk mode *syncrhronous* harus 3 pin yaitu TXD, RXD dan XCK.

II.11.2. Arsitektur Pin Mikrokontroler ATmega8535

Agar dapat berkomunikasi antara komputer dengan mikrokontroler, perlu mengenal arsitektur dari mikrokontroler dan teknik komunikasi secara serial di komputer.



Gambar II.14. Arsitektur Pin Mikrokontroler ATmega8535

Tabel II.2 berikut adalah keterangan setiap fungsi Pin dari gambar arsitektur pin mikrokontroler Atmega8535 :

Tabel II.2. Fungsi Tiap-tiap Pin Mikrokontroler ATmega8536

Pin	Fungsi
VCC	Input sumber tegangan (+)
GND	*Ground* (-)
Port A (PA7 … PA0)	sebagai input analog dari ADC (*Analog to Digital Converter*). Port ini juga berfungsi sebagai port I/O dua arah, jika ADC tidak digunakan.
Port B (PB7 … PB0)	sebagai port I/O dua arah. Port PB5, PB6 dan PB7 juga berfungsi sebagai MOSI, MISO dan SCK yang dipergunakan pada proses *downloading*.
Port B (PC7 … PC0)	sebagai port I/O dua arah
Port B (PD7 … PD0)	sebagai port I/O dua arah. Port PD0 dan PD1 juga berfungsi sebagai RXD dan TXD, yang dipergunakan untuk komunikasi serial
RESET	*Input reset*
XTAL1	*Input* ke *amplifier inverting osilator* dan input ke *sirkuit clock internal*
XTAL2	*Output* dari *amplifier inverting osilator*
AVCC	Input tegangan untuk Port A dan ADC
AREF	Tegangan referensi untuk ADC

II.12. Komunikasi Data Serial

Komunikasi data pada umumnya dapat dilakukan dengan dua cara, yaitu secara serial dan paralel. Komunikasi data serial dilakukan dengan mengirimkan dan menerima data 8 *bit* secara satu per satu sehingga akan lebih lambat dibandingkan komunikasi data paralel yang dapat melakukan pengiriman dan menerima data hingga 8 *bit* sekaligus. RS232 (*Recomended Standard number* 232) merupakan seperangkat alat yang diciptakan oleh *Electrical Industry Association* yang berfungsi sebagai antarmuka dalam melakukan transfer data dengan komputer yang mana pengiriman data dilakukan dengan mengirimkan kode biner. Pada seperangkat komputer biasanya tersedia dua bah *Communication Port* yang sering disebut COM, yaitu COM1 dan COM2.

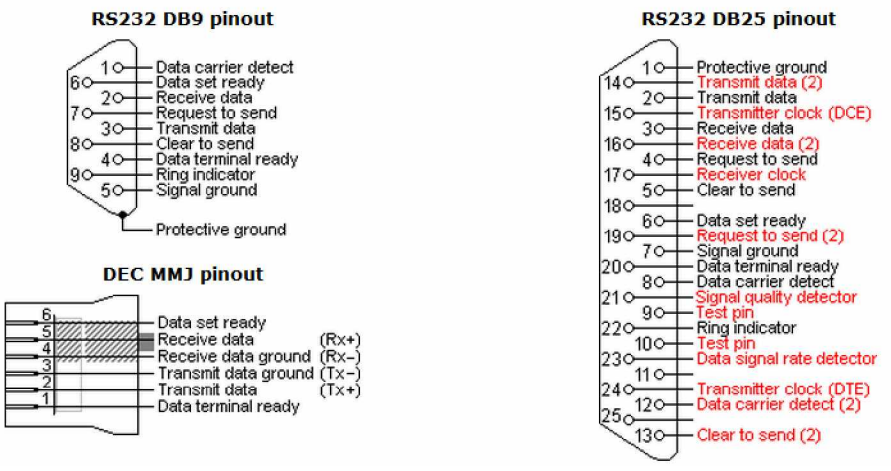
Pada dasarnya ada dua jenis komunikasi serial, yaitu komunikasi data serial sinkron dimana pengiriman *clock* dilakukan secara bersamaan dengan data serial dan komunikasi data serial asinkrin dimana pengiriman *clock* dilakukan secara dua tahap, yaitu saat data dikirimkan dan saat data diterima. RS232 pada komputer memiliki konektor, yaitu konektor dengan 25 pin (DB-25 *connector*) dan konektor dengan 9 pin (DB-9 *connector*). Pada dasarnya hanya 3 pin yang terpakai, yaitu pin pengirim, pin penerima dan pin untuk *ground*. Pada pengiriman data serial, semakin jauh jarak kirim maka kemungkinan terjadinya derau (noise) semakin besar.

Dalam setiap proses transfer data serial, RS232 memerlukan sebuah *Data Terminal Equipment* (DTE) dan *Data Communication Equipment* (DCE) pada masing-masing terminal. Pengiriman data dilakukan secara *bit* per *bit* dan

kecepatan transfer data juga harus sama antara penerima dan pengirim. Jika kecepatannya tidak sama maka akan terjadi *overflow*. Kecepatan transmisi tersebut biasanya disebut dengan *baud rate*. *Baud rate* yang biasa dipakai antara lain adalah 110, 300, 1200, 2400, 4800, 19200, 38400, 57600, 115200, 230400, 460800, atau 921600 bps. Panjang data *bit* yang sering digunakan di antaranya adalah 4, 5, 6, 7, dan 8 *bit*.

Pada komunikasi data serial pada dasarnya yang dikirimkan adalah tegangan dan kemudian dibaca dalam data *bit*. Besar level tegangannya adalah antara -25 sampai +25 Volt. Untuk *bit* dengan logika 1 maka besar level tegangannya adalah antara +3 sampai -25 Volt, sedangkan untuk logika0 maka besar level tegangannya adalah antara +3 sampai dengan +25 Volt.

Konektor *port* serial terdiri dari dua jenis, yaitu konektor 25 pin (DB25) dan 9 pin (DB9) yang saling berpasangan (*male* dan *female*). Bentuk dari konektor DB25 sama persis dengan *port* paralel. Konektor DB-9 dan DB-25 diperlihatkan pada gambar II.16 :



Gambar II.15. Konektor Serial DB-9, DB-25, dan Data Equipment Communication

Tabel II.3 berikut adalah penggunaan pin pada konektor serial DB-9 dan konektor serial DB-25.

Tabel II.3. Tabel Penggunaan Pin Pada DB-9 dan DB-25

Pin DB-9	Pin DB-25	Nama Sinyal	Jenis
1	8	*Data Carrier Detect*	*Input*
2	3	*Received Data (RxD)*	*Input*
3	2	*Transmitted Data (TxD)*	*Output*
4	20	*Data Terminal Ready (DTR)*	*Output*
5	7	*Ground*	-
6	6	*Data Set Ready (DSR)*	*Input*
7	4	*Request To Send (RTS)*	*Output*
8	5	*Clear To Send (CTS)*	*Input*
9	22	*Ring Indicator*	*Input*

Keterangan :

1) Pin 1 (*Data Carrier Detect*) berfungsi untuk mendeteksi boleh atau tidaknya DTE menerima data.
2) Pin 2 (*Received Data*) berfungsi sebagai jalur penerimaan data dari DCE ke DTE.
3) Pin 3 (*Transmitted Data*) berfungsi sebagai jalur pengiriman data dari DTE ke DCE.
4) Pin 4 (*Data Terminal Ready*) berfungsi untuk memberitahu kesiapan terminal DTE.
5) Pin 5 (*Ground*) berfungsi sebagai saluran *ground*.
6) Pin 6 (*Data Set Ready*) berfungsi untuk menyatakan bahwa status data tersambung pada DCE.
7) Pin 7 (*Request To Send*) berfungsi untuk mengirim sinyal informasi dari DTE ke DCE bahwa akan ada data yang akan dikirim.

8) Pin 8 (*Clear To Send*) berfungsi untuk memberitahu pada DTE bahwa DCE siap untuk menerima data.
9) Pin 9 (*Ring Indicator*) berfungsi untuk memberitahu pada DTE ada terminal yang menginginkan komunikasi dengan DCE.