BAB 2
LANDASAN TEORI

2.1 Sistem pengendalian

Sistem pengendalian adalah susunan suatu komponen yang dihubungkan sedemikian rupa untuk mengatur suatu kondisi agar mencapai kondisi yang diharapkan. System pengendalian ini secara umum terdiri dari tiga elemen pokok, yaitu *input*, proses dan *output. Output* merupakan hal yang dihasilkan oleh kendalian, artinya yang dikendalikan. Sedangkan *input* adalah yang mempengaruhi kendalian, yang mengatur *output* [5]

2.2 Mikrokontroler

Teknologi mikrokontroler sebagai salah satu produk teknologi semikonduktor berkontribusi besar untuk menunjang aktivitas manusia. Aduino merupakan salah satu dari sekian produk edukasi mikro kontroller sebagai proyek rintisan berlisensi terbuka dan mampu difungsikan sebagai produk akhir. Dalam modul juga disetrai komponen elektronika [3].

Struktur serta antarmuka Arduino yang sederhana memberi kemudahan pengguna dalam memahami parameter (visualisasi maupun non-visual) seperti konsep sensor atau penerapan sensor elektronik yang tidak selalu bisa diamati langsung seperti sinar infra merah. Konsep bahasa Arduino dengan mentargetkan ke pin tertentu (secara fisik) menjadikannya lebih mudah dipahami oleh berbagai kalangan.

Modul *hardware* Arduino di ciptakan oleh Massimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino, David A. Mellis dan Nicholas Zambetti di Ivera Italia pada tahun 2005. Arduino memakai standar lisensi *open source*, mencakup *hardware* (skema rangkaian, desain PCB atau *Printed Circuit Board*), *firmware bootloader*, dokumen, serta perangkat lunak IDE (*Integrated Development Environment*) sebagai aplikasi programmer *board* Arduino. Agar mikrokontroler bisa berkomunikasi dengan IDE Arduino, pada mikrokontroler harus sudah terprogram *boot loader* pada blok memori *Flash*. Semua produk Arduino secara

default sudah terinstal *boot loader* dan dapat diprogram berulang kali. Modul Arduino yang sudah dirilis sejak tahun 2009 diantaranya Diecimila, Uno, Duemilanove, Nano, Mega dan LilyPad. Setiap seri modul menggunakan seri mikrokontroller yang berbeda.

Arduino adalah kit elektronik atau papan rangkaian elektronik *open source* yang di dalamnya terdapat komponen utama yaitu sebuah chip mikrokontroler dengan jenis AVR dari perusahaan Atmel.

Mikrokontroler itu sendiri adalah chip atau IC (*Integrated Circuit*) yang bisa diprogram menggunakan komputer. Tujuan menanamkan program pada mikrokontroler adalah agar rangkaian elektronik dapat membaca input, memproses input tersebut dan kemudian menghasilkan output sesuai yang diinginkan. Jadi mikrokontroler bertugas sebagai otak yang mengendalikan input, proses dan output sebuah rangkaian elektronik.

Mikrokontroler ada pada perangkat elektronik di sekeliling kita. Misalnya handphone, MP3 player, DVD, televise, AC, dll. Mikrokontroler juga dipakai untuk keperluan mengendalikan robot. Baik robot mainan, maupun robot industri.

Karena komponen utama Arduino adalah mikrokontroler, maka Arduino pun dapat deprogram menggunakan komputer sesuai kebutuhan kita. Kegunaan arduino tergantung pada kita yang membuat program. Arduino bisa digunakan untuk mengontrol LED, bisa juga digunakan untuk mengontrol helikopter.

Arduino mempunya kelebihan diantaranya:

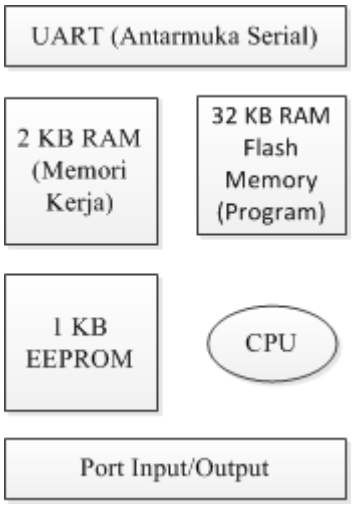
1. Tidak perlu perangkat chip programmer karena di dalamnya sudah ada bootloader yang akan menangani *upload* program dari computer.
2. Sudah memiliki sarana komunikasi USB, sehingga pengguna Laptop yang tidak memiliki port serial/RS323 bisa menggunaknnya.
3. Bahasa pemrograman relatif mudah karena software Arduino dilengkapi dengan kumpulan *library* yang cukup lengkap.
4. Memiliki modul siap pakai (*shield*) yang bisa ditancapkan pada board Arduino. Misalnya shield GPS, Ethernet, GSM/GPRS, SD Card, dll.

Bahasa pemrograman Arduino adalah bahasa C. tetapi bahasa ini sudah dipermudah menggunakan fungsi-fungsi yang sederhana sehingga pemula pun bisa mempelajarinya dengan cukup mudah.

Untuk membuat program arduino dan mengupload ke dalam board Arduino, dibutuhkan software Arduino IDE yang bisa di download secara gratis di http://arduino.cc/en/Main/Software.

Komponen utama di dalam papan Arduino adalah sebuah mikrokontroler 8 bit dengan merk ATmega yang dibuat oleh perusahaan Atmel Corporation. Berbagai papan Arduino menggunakan tipe ATmega yang berbeda-beda tergantung dari spesifikasinya, sebagai contoh Arduino UNO menggunakan ATmega328.

Untuk memberikan gambaran mengenai apa saja yang terdapat di dalam sebuah mikrokontroler, pada Gambar 2.1 diperlihatkan contoh diagram blok sederhana dari mikrokontroler ATmega328 Arduino Uno.



Gambar 2.1 Diagram Blok Mikrokontroler Arduino Uno [6]

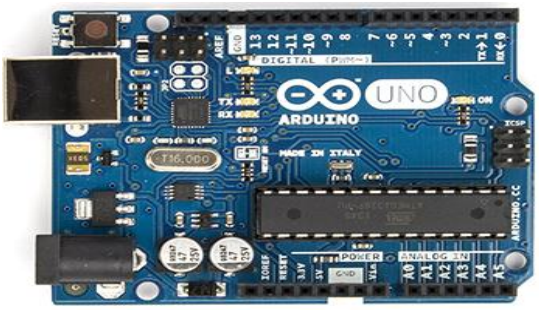
Penjelasan pada blok-blok diatas yaitu:

a. *Universal Asynchronous Receiver/Transmitter* (*UART*) adalah antar muka yang digunakan untuk komunikasi serial seperti pada RS-232, RS-422 dan RS-485.

b. 2 KB RAM pada *memory* kerja bersifat *volatile* (hilang saat daya dimatikan), digunakan oleh variable-variabel di dalam program.
c. 32 KB RAM *flash memory* bersifat *non-volatile*, digunakan untuk menyimpan program yang dimuat dari komputer. Selain program, *flash memory* juga menyimpan *bootloader* yang merupakan inisiasi yang ukurannya kecil, dijalanakn oleh CPU saat daya dihidupkan. Setelah *bootloader* selesai dijalankan, berikutnya program di dalam RAM akan dieksekusi.
d. 1 KB EEPROM bersifat *non-volatileI*, digunakan untuk menyimpan data yang tidak boleh hilang saat daya dimatikan. Tidak digunakan pada papan Arduino.
e. *Central Processing Unit* (CPU), bagian dari mikrokontroler untuk menjalankan setiap instruksi dari program.
f. Port input/output, pin-pin untuk menerima data (*input*) digital atau analog, dan mengeluarkan data (*output*) digital atau analog.

2.2.1 Arduino UNO

Arduino Uno adalah salah satu produk berlabel Arduino yang sebenarnya adalah suatu papan elektronik yang mengandung mikrokontroler ATmega328 (sebuah keeping yang secara fungsional bertindak seperti sebuah computer). Piranti ini dapat dimanfaatlkan untuk mewujudkan rangkaian elektronik dari yang sederhana hingga yang kompleks [7]**.** Pengendalian LED hingga pengontrolan robot dapat diimplementasikan dengan menggunakan papan yang berukuran relative kecil ini. Bahkan, dengan penambahan komponen tertentu, piranti ini bisa dipakai untuk pemantauan jarak jauh melalui internet, misalnya pemantauan kondisi pasien di rumah sakit dan pengendalian alat-alat di rumah.

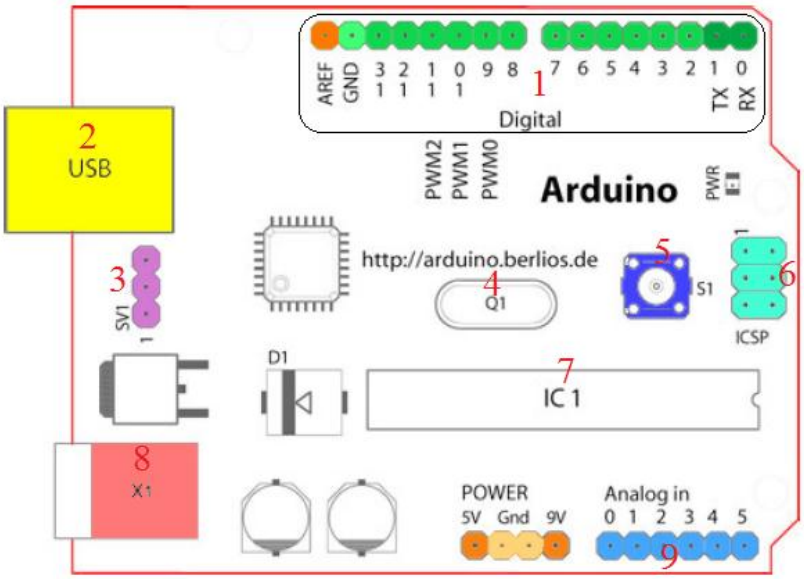


Gambar 2.2 Arduino Uno R3 [8]

Arduino uno mengandung mikroprosesor (berupa Atmel AVR) dan dilengkapi dengan *oscillator* 16 Mhz (yang memungkinkan operasi berbasis waktu dilaksanakan dengan tepat), dan regulator (pembangkit tegangan) 5 volt. Sejumlah pin tersedia di papan. Pin 0 hingga 13 digunakan untuk isyarat digital, yang hanya bernilai 0 atau 1. Pin A0-A5 digunakan untuk isyarat analog. Arduino uno dilengkapi dengan *static random-access memory* (SRAM) berukuran 2KB untuk memegang data, *flash memory* berukuran 32KB dan *erasable programmable read-only memory* (EEPROM) untuk menyimpan program. Dalam penelitian ini penulis menggunakan dua buah Arduino Uno R3 dikarenakn keterbatasan pin pada arduino uno R3.

2.2.2 Bagian-bagian Papan Arduino

Setelah mengenal bagian-bagian utama dari mikrokontroler ATmega sebagai komponen utama, selanjutnya kita akan mengenal bagian-bagian dari papan Arduino itu sendiri seperti pada Gambar 2.3



Gambar 2.3 Bagian-bagian Papan Arduino [6]

Berikut penjelasan bagian-bagian dari papan arduino:

1. 14 pin input/output digital (0-13)

   Berfungsi sebagai input atau output, dapat diatur oleh program.

   Khusus untuk 6 buah pin 3, 5, 6, 9, 10 dan 11, dapat juga berfungsi sebagai pin analog output dimana tegangan outputnya dapat diatur. Nilai sebuah pin output analog dapat deprogram antara 0-255, dimana hal itu mewakili nilai tegangan 0-5V.

2. USB

   Berfungsi untuk:

   a. Memuat program dari komputer ke dalam papan
   b. Komunikasi serial antara papan dan komputer
   c. Memberi daya listrik kepada papan

3. Sambungan SV1

   Sambungan atau *jumper* untuk memilih sumber daya papan, apakah dari sumber eksternal atau menggunakan USB. Sambungan ini tidak diperluakan lagi pada papan Arduino versi terakhir karena pemilihan sumber daya eksternal atau USB dilakukan secara otomatis.

4. Q1 – Kristal (*quartz crystal oscillator*)

   Jika mikrokontroler dianggap sebagai sebuah otak, maka Kristal adalah jantungnya karena komponen ini menghasilkan detak-detak yang dikirim kepada mikrokontroler agar melakukan sebuah operasi untuk setiap detaknya. Kristal ini dipilih yang berdetak 16 juta kali per detik (16Mhz).

5. Tombol Reset S1

   Untuk me-*reset* papan sehingga program akan mulai lagi dari awal. Perhatikan bahwa tombol reset ini bukan untuk menghapus program atau mengosongkan mikrokontroler.

6. *In-Circuit Serial Programming (ICSP)*

   Port ICSP memungkinkan pengguna untuk memprogram mikrokontroler secara langsung, tanpa melalui *bootloader*. Umumnya pengguna Arduino tidak melakukan ini sehingga ICSP tidak terlalu dipakai walupun disediakan.

7. IC1 – Mikrokontroler ATmega

   Komponen utama dari papan Arduino, di dalamnya terdapat CPU, ROM dan RAM.

8. X1 – Sumber daya eksternal

   Jika hendak disuplai dengan sumber daya eksternal, papan Arduino dapat diberikan tegangan DC antara 9-12V.

9. 6 Pin input analog

   pin ini sangat berguna untuk membaca tegangan yang dihasilkan oleh sensor analog, seperti sensor suhu. Program dapat membaca nilai sebuah pin input antara 0 – 1023, dimana hal itu mewakili nilai tegangan 0-5V.

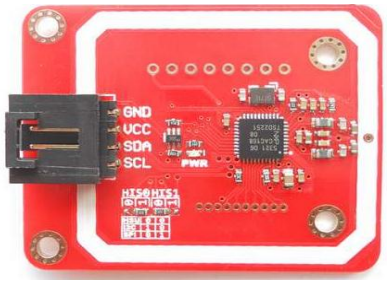
2.3 Modul-modul (*shield*) Mikrokontroler

Shield merupakan modul atau alat tambahan yang kompatibel dengan board Arduino, berikut adalah modul-modul yang digunakan pada pada sistem pengendali lampu dan pemanfaatan rfid pada kunci elektronik.

2.3.1 RFID *Reader*

RFID *reader* merupakan penghubung antara *software* aplikasi dengan antenna yang akan meradiasikangelombang radio ke RFID *tag*. Gelombang radio yang ditransmisikan oleh antenna berpropagasi pada ruangan di sekitarnya. Akibatnya dapat berpindah secara *wireless* ke *tag* RFID [9].

Pembaca yang digunakan adalah PN532, yang merupakan *reader* dengan frekuensi 13,56 MHz. Gambar 2.4 menunjukan RFID Reader PN532.



Gambar 2.4 RFID *Reader Module* PN532 [10]

Fitur RFID Module PN532

1. Support II2, SPI dan HSU (High Speed UART)
2. RFID reader/writer mendukung tipe:
   a. Mifare 1k, 4k, Ultralight dan Kartu/tag DesFire
   b. ISO/IEC 14443-4 kartu seperti CD97BX, CD light, DesFire, P5CN072 (SMX)
   c. Kartu Innovision Jewel seperti kartu IRT5001
   d. Kartu FeliCa seperti RCS_860 dan RCS_854
3. Kompatibel dengan Arduino tinggal *plug and play*.
4. Dibangun pada PCB Antena, dengan jarak komunikasi 5cm - 7cm
5. *On-board* tingkat *shifter*, Standard 5V TTL untuk I2C dan UART, 3.3V TTL SPI
6. Bekerja sebagai RFID *reader/writer*
7. Bekerja pada kartu 1443-A atau kartu virtual
8. Ukuran kecil: 43mm * 41mm * 4mm

2.3.2 Tag RFID

RFID *tag* terdiri dari *chip* rangkaian sirkuit yang terintegrasi dan sebuah antena. Rangkaian elektronik dari RFID *tag* umumnya memiliki memori yang memungkinkan RFID *tag* mempunyai kemampuan untuk menyimpan data. Memori pada *tag* dibagi menjadi sel-sel. Beberapa sel menyimpan data *Read Only*, seperti ID *number*. Semua RFID *tag* mendapatkan ID *number* pada saat *tag* tersebut diproduksi.

Berdasarkan catu daya, RFID *tag* digolongkan menjadi :

1. *Tag* Aktif

Tag yang catudayanya didapat dari baterai dan dapat dibaca (*Read*) dan ditulis (*Write*). Dengan adanya baterai internal *tag* aktif dapat mengirimkan informasi dalam jarak yang lebih jauh dan *reader* hanya membutuhkan daya yang kecil untuk membaca *tag* ini. Bentuk fisik dari *tag* aktif dapat dilihat pada Gambar 2.5

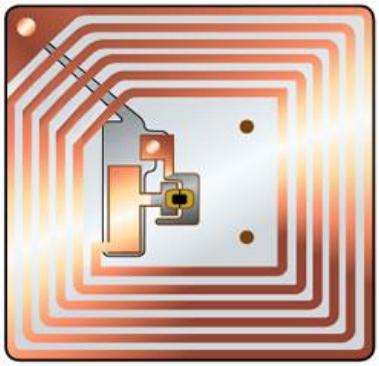


Gambar 2.5 *Tag* RFID Aktif [11]

2. *Tag* Pasif

Tag ini hanya dapat dibaca saja (*Read*) dan tidak memiliki internal baterai seperti halnya *tag* aktif. Sumber tenaga untuk mengaktifkan tag ini didapat dari RFID *reader*. Ketika medan gelombang radio dari *reader* didekati oleh *tag* pasif, koil antena yang terdapat pada *tag* pasif ini akan membentuk suatu medan magnet. Medan magnet ini akan menginduksi suatu tegangan listrik yang memberi tenaga pada *tag* pasif. Kelemahan *tag* pasif hanya dapat mengirimkan informasi dalam jarak yang dekat dan untuk membaca *tag* ini, RFID *reader* harus memancarkan

gelombang radio yang cukup besar sehingga menggunakan daya yang cukup besar. Bentuk fisik dari *tag* pasif dapat dilihat pada Gambar 2.6



Gambar 2.6 *Tag* RFID Pasif [12]

Tugas akhir ini menggunakan modul RFID *reader* yang khusus untuk medeteksi RFID *tag* pasif dengan frekuensi 13,56 MHz. RFID *tag* yang kompatibel dengan modul RFID reader ini adalah tipe mifare 1 S50, S70 Ultralight, Pro dan DESFire. Gambar 2.7 memperlihatkan *tag* RFID yang akan digunakan dan Tabel 2.1 memperlihatkan spesifikasi *tag* RFID tipe Mifare 1 S50.



Gambar 2.7 *Tag* RFID Mifare 1 S50 [13]

Tabel 2.1 Spesifikasi RFID *tag Mifare* 1 S50

Parameter	Spesifikasi
Frekuensi	13,56 MHz
Jangkauan baca	5 – 7 cm
Dimensi	85,6 x 54 x 0.86 (mm)

2.3.3 *GPRS Shield*

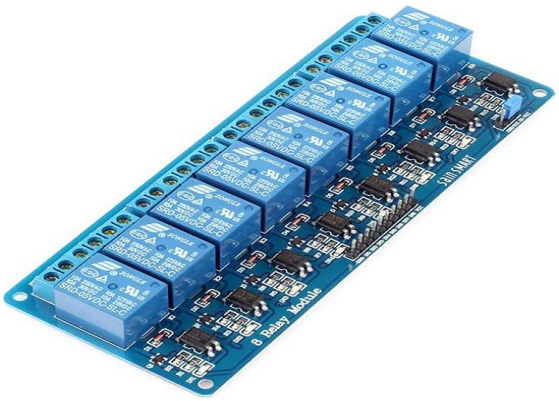
GPRS Shield digunakan sebagai sebuah ekspansi fitur dari Arduino untuk dapat menerima serta mengirimkan SMS dari/kembali ke *user* [9]. Jenis GPRS *Shield* yang digunakan dalam perancangan ini sendiri adalah GPRS *Shield* yang di dalamnya menggunakan modul SIM900 yang memiliki konsumsi daya yang cukup rendah. Gambar 2.8 memperlihatkan modul GPRS yang akan dipakai.



Gambar 2.8 GPRS *Shield* [14]

2.3.4 *Relay*

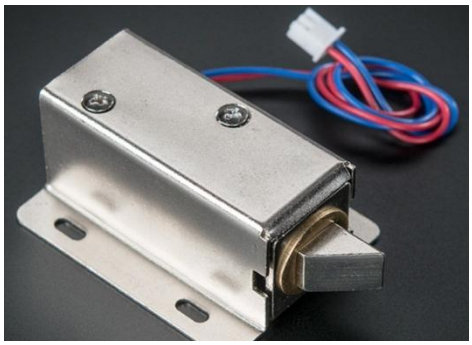
Prinsip Kerja *Relay* - *Relay* adalah Saklar (*Switch*) yang dioperasikan secara listrik dan merupakan komponen *Electromechanical* (Elektromekanikal) yang terdiri dari 2 bagian utama yakni Elektromagnet (*Coil*) dan Mekanikal (seperangkat Kontak Saklar/*Switch*). *Relay* menggunakan Prinsip Elektromagnetik untuk menggerakkan Kontak Saklar sehingga dengan arus listrik yang kecil (*low power*) dapat menghantarkan listrik yang bertegangan lebih tinggi. Sebagai contoh, dengan *Relay* yang menggunakan Elektromagnet 5V dan 50 mA mampu menggerakan *Armature Relay* (yang berfungsi sebagai saklarnya) untuk menghantarkan listrik 220V. Contoh relay dapat dilihat pada Gambar 2.9



Gambar 2.9 *Relay 8 Channel* [15]

2.3.5 *Solenoid*

Bagian ini berfungsi sebagai aktuator. Prinsip dari *solenoid* sendiri akan bekerja sebagai pengunci dan akan aktif ketika diberikan tegangan sebesar 12V. di dalam *solenoid* terdapat kawat yang melingkar pada inti besi. Ketika arus listrik mengalir melalui kawat ini, maka terjadi medan magnet untuk menghasilkan energi yang akan menarik inti besi ke dalam. Gambar 2.10 memperlihatkan *solenoid* sebagai pengganti kunci mekanik.



Gambar 2.10 *Solenoid Door Lock* [16]

2.3.6 Sensor PIR (*Passive Infrared*)

Sensor PIR (*Passive Infrared*) merupakan *sensor* penerima radiasi sinar infra merah, yaitu rangkaian modul sebagai detektor level flukturasi infra merah. Modul sensor akan memberikan sinyal luaran *HIGH* saat terjadi diferensiasi sinyal positif akibat pergerakan benda yang memancarkan radiasi infra merah

seperti pada manusia. Gambar 2.11 memperlihatkan Sensor PIR yang digunakan pada perancangan tugas akhir ini.



Gambar 2.11 Sensor PIR [17]

2.3.7 *Buzzer*

Buzzer adalah sebuah komponen elektronika yang berfungsi untuk mengubah getaran listrik menjadi getaran suara. Pada dasarnya prinsip kerja *buzzer* hampir sama dengan *loud speaker*, jadi *buzzer* juga terdiri dari kumparan yang terpasang pada diafragma dan kemudian kumparan tersebut dialiri arus sehingga menjadi elektromagnet, kumparan tadi akan tertarik ke dalam atau keluar, tergantung dari arah arus dan polaritas magnetnya, karena kumparan dipasang pada diafragma maka setiap gerakan kumparan akan menggerakkan diafragma secara bolak-balik sehingga membuat udara bergetar yang akan menghasilkan suara. *Buzzer* ini digunakan sebagai indikator *(alarm).* Gambar 2.12 memperlihatkan buzzer yang dipakasi sebagai alarm pada perancangan tugas akhir ini.



Gambar 2.12 *Active Buzzer* [18]

2.3.8 *LCD Keypad*

LCD Keypad dikembangkan untuk kompatibel dengan *board* Arduino, untuk menyediakan antarmuka yang user-friendly yang memungkinkan pengguna untuk memilih menu, membuat pilihan dll. Ini terdiri dari 1.602 karakter putih LCD *backlight* biru. Tombol terdiri dari 5 tombol - pilih, atas, kanan, bawah dan kiri. Untuk menyimpan pin IO digital, antarmuka *keypad* hanya menggunakan satu saluran ADC. Gambar 2.13 memperlihatkan *LCD Keypad* yang digunakan sebagai *user interface rfid*.



Gambar 2.13 *LCD Keypad* [19]

2.4 Teknologi Yang Digunakan

Teknologi yang digunakan pada sistem pengendali lampu dan pemanfaatan rfid pada kunci elektronik adalah sebgai berikut:

2.4.1 *RFID*

Radio Frequency Identification (RFID) merupakan sebuah teknologi yang menggunakan metoda *auto-ID* atau *Automatic Identification*. *Auto-ID* adalah metoda pengambilan data dengan identifikasi objek secara otomatis tanpa ada keterlibatan manusia. *Auto-ID* bekerja secara otomatis sehingga dapat meningkatkan efisiensi dalam mengurangi kesalahan dalam memasukkan data.

RFID adalah teknologi penangkapan data yang dapat digunakan secara elektronik untuk mengidentifikasi, melacak dan menyimpan informasi yang sebelumnya tersimpan dalam id *tag* dengan menggunakan gelombang radio. RFID adalah sebuah metode identifikasi secara otomatis dengan menggunakan suatu piranti yang disebut RFID *tag* atau *transporder*. Data yang ditransmisikan dapat berupa kode–kode yang bertujuan untuk mengidentifikasi suatu objek tertentu.

Pada RFID proses identifikasi dilakukan oleh RFID *reader dan* RFID *tag*. RFID *tag* diletakkan pada suatu benda atau objek yang akan diidentifikasi. Tiap-tiap RFID *tag* memiliki data angka identifikasi (ID *number*) yang unik, sehingga tidak ada RFID *tag* yang memiliki ID *number* yang sama.

2.4.2 *GSM (Global System for Mobile Communication)*

GSM merupakan teknologi seluler generasi kedua yang menggunakan modulasi digital. Menyediakan kapasitas lebih besar, kualitas suara serta sekuritas yang lebih baik jika dibandingkan dengan teknologi generasi pertama. GSM merupakan teknologi *Time Division Multiple Access* (TDMA) sebagai *interfacenya*. Pada teknologi ini, suatu pita dengan frekuensi tertentu yang lebih lebar dibagi-bagi kedalam beberapa *time slot*. Hal ini berarti bahwa beberapa panggilan dapat menggunakan kanal frekuensi yang sama, pada suatu slot waktu yang berbeda-beda [20].

2.4.3 *GPRS (General Packet Radio Service)*

General Packet Radio Service (GPRS) adalah pelayanan baru bukan suara (non voice) dengan nilai tambah yang mengijinkan informasi untuk dikirim dan diterima ke jaringan telepon mobile yang berbeda, kecepatan maksimumnya adalah 171,2 kilobits per detik (kbps), dengan menggunakan 8 timeslot pada

waktu yang sama [21]. Salah satu aplikasi yang didukung oleh GPRS adalah koneksi internet, yang merupakan pengembangan dari teknologi Wireless Aplication Protocol (WAP) yang hanya menampilkan data yang kecil, sebagai tuntutan kebutuhan komunikasi data yang tinggi maka koneksi internet GPRS merupakan salah satu aplikasi yang memberikan transfer data dalam jumlah besar dan dalam kecepatan tinggi dibanding dengan koneksi internet dengan menggunakan dial up Public Switch Telephony Network (PSTN).

2.4.4 *Internet*

Internet adalah suatu sistem jaringan komunikasi beberapa komputer yang terhubung tanpa batas waktu maupun tempat, sehingga dapat dikatakan sebagai suatu komunitas jaringan global [22].

2.4.5 *Google Cloud Messaging*

Google Cloud Messaging untuk Android (GCM) adalah layanan yang membantu pengembang mengirim data dari *server* untuk aplikasi Android mereka ke perangkat Android lainnya. Ini bisa menjadi pesan singkat untuk memberitahu aplikasi Android bahwa ada data baru yang akan diambil dari *server* (misalnya, film diunggah oleh seorang teman), atau bisa juga pesan yang berisi sampai dengan 4KB data *payload* (sehingga aplikasi seperti *instant messaging* dapat menerima pesan langsung).

Layanan GCM menangani semua aspek antrian pesan dan pengiriman ke aplikasi target, dan berjalan pada perangkat target. GCM memungkinkan aplikasi android untuk mngirimkan pesan kepada *server* untuk *membroadcast* sebuah notifikasi kepada seluruh *client* yang ada. Hanya membutuhkan sebuah *account* Gmail maka akan langsung dapat menggunakan fasilitas GCM ini.

2.5 Android

Android adalah sebuah sistem operasi untuk perangkat *mobile* berbasis *linux* yang mencakup sistem operasi, *middleware* dan aplikasi. Android menyediakan *platform* terbuka bagi para pengembang untuk menciptakan aplikasi mereka [23].

2.5.1 Sejarah Android

Awalnya, Google inc. membeli Android Inc, yang merupakan pendatang baru yang membuat peranti lunak ponsel/*smartphone*. Kemudian untuk mengembangkan Android, dibentuklah *Open Handset Alliance*, konsorsium dari 34 perusahaan peranti keras, peranti lunak dan telekomunikasi, termasuk Google, HTC, Intel, Motorola, Qualcomm, T-Mobile, dan Nvidia.

Pada saat perilisan perdana Android, 5 November 2007, Android bersama *Open Handset Alliance* menyatakan mendukung pengembangan *open source* pada perangkat *mobile*. Di lain pihak, Google merilis kode-kode Android dibawah lisensi Apache, sebuah lisensi perangkat lunak dan *open platform* perangkat seluler.

Di dunia ini terdapat dua jenis distributor sistem operasi Android. Pertama yang mendapat dukungan penuh dari Google atau *Google Mail Services* (GMS) dan kedua adalah yang benar-benar bebas distribusinya tanpa dukungan langsung Google atau dikenal dengan *Open Handset Distribution* (OHD).

Sekitar September 2007 Google mengenalkan *Nexus One*, salah satu jenis *smartphone* yang menggunakan Android sebagai sistem operasinya. Telepon seluler ini diproduksi oleh HTC *Corporation* dan tersedia di pasaran pada 5 Januari 2010. Pada 9 Desember 2008, diumumkan anggota baru yang bergabung dalam program kerja Android ARM Holdings, Atheros Softbank, Sony Ericsson, Toshiba Corp dan Vodafone Group Plc. Seiring pembentukan *Open Handset Alliance*, OHA mengumumkan produk pertama mereka, Android perangkat *mobile* yang merupakan modifikasi kernel *Linux* 2.6. sejak Android dirilis telah dilakukan berbagai pembaruan berupa perbaikan *bug* dan penambahan fitur baru.

Pada saat ini sebagian besar *vendor-vendor smartphone* sudah memproduksi *smartphone* berbasis Android, *vendor-vendor* itu antara lain HTC, Motorola, Samsung, LG, HKC, Huawei, Archos, Webstation Camangi, Dell, Nexus, SciPhone, WayteQ, Sony Ericsson, LG, Acer, Philips, T-Mobile, Nexian, IMO, Asus dan masih banyak lagi *vendor smartphone* di dunia yang memproduksi android. Hal ini, karena android itu adalah sistem operasi yang *open source* sehingga bebas didistribusikan dan dipakai oleh *vendor* manapun.

Tidak hanya menjadi sistem operasi di *smartphone*, saat ini Android menjadi pesaing utama dari *Apple* pada sistem operasi Tablet PC. Pesatnya pertumbuhan Android selain faktor yang disebutkan di atas adalah karena Android itu sendiri adalah *platform* yang sangat lengkap baik itu sistem operasinya, Aplikasi dan *Tool* pengembangan, market aplikasi android serta dukungan yang sangat tinggi dari komunitas *open source* di dunia, sehingga android terus berkembang pesat baik dari segi teknologi maupun dari segi jumlah *device* yang ada di dunia.

2.5.2 Arsitektur Android

Secara garis besar Arsitektur Android dapat dijelaskan dan digambarkan sebagai berikut:

a. *Application* dan *Widgets*

   Application dan *Widgets* ini adalah layer dimana kita berhubungan dengan aplikasi saja, dimana biasanya kita *download* aplikasi kemudian kita lakukan instalasi dan jalankan aplikasi tersbut. Di *layer* terdapat aplikasi inti termasuk klien email, program SMS, kalender, peta, browser, kontak, dan lain-lain. Semua aplikasi ditulis menggunkan bahasa pemrograman java.

b. *Application Frmeworks*

   Adroid adalah "*Open Development Platform*" yaitu android menawarkan kepada pengembang atau memberi kemampuan pada pengembang untuk membangun aplikasi yang bagus dan inovatif. Pengembang bebas untuk mengakses perangkat keras, akses informasi *reseource*, menjalankan *service background*, mengatur *alarm*, dan menambahkan status *notification*, dan sebagainya. Pengembang memiliki akses penuh menuju API *framework* seperti yang dilakukan oleh aplikasi yang kategori inti. Arsitektur aplikasi dirancang supaya kita dengan mudah dapat menggunakan kembali komponen yang sudah digunakan. Sehingga bisa kita simpulkan *Application Frameworks* ini adalah *layer* di mana para pembuat aplikasi melakukan pengembangan/pembuatan aplikasi yang akan dijalankan di sistem operasi Android, karena pada *layer* inilah aplikasi dapat dirancang dan dibuat, seperti

content providers yang berupa sms dan panggilan telepon. Komponen-komponen yang termasuk di dalam *Application Frameworks* adalah sebagai berikut:

a. *Views*
b. *Content Provider*
c. *Reseource Manager*
d. *Notification Manager*
e. *Activity Manager*
c. *Libraries*

Libraries ini adalah *layer* di mana fitur-fiturAndroid berada, biasanya para pembuat aplikasi mengakses *libraries* untuk menjalankan aplikasinya. Berjalan di atas kernel, *Layer* ini meliputi berbagai *library* C/C++ inti seperti Libc dan SSL, serta:

1. *Libraries* media untuk pemutaran media audio dan video.
2. *Libraries untuk manajemen tampilan.*
3. *Libraries Graphic* mencakup SGL dan OpenGL untuk grafis 2D dan 3D.
4. *Libraries* SQLite untuk dukungan database.
5. *Libraries* SSL dan *Webkit* terintegrasi dengan *web browser* dan *security.*
6. *Libraries LiveWebcore* mencakup modern *web browser* dengan *engine embedded web view*.

d. *Android Run Time*

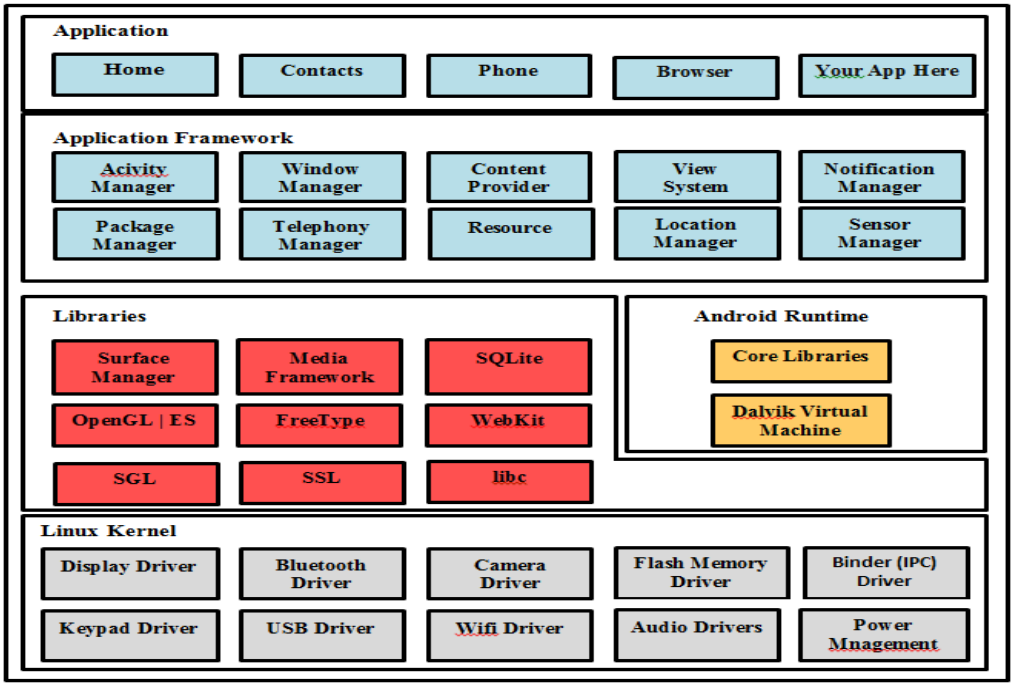
Layer yang membuat aplikasi Android dapat dijalankan daimana di dalam prosesnya menggunakan Implementasi *Linux*. *Dalvik Virtual Machine* (DVM) merupakan yang membentuk dasar kerangka aplikasi Android. Di dalam Android *Run Time* dibagi menjadi dua bagian yaitu:

1. *Core Libraries*: Aplikasi Adroid dibangun dalam bahasa Java, sementara *Dalvik* sebagai virtual mesinnya bukan *Virtual Machine* Java, sehingga diperlukan sebuah *libraries* yang berfungsi untuk menterjemahkan bahasa Java/ C yang ditangani oleh *Core Libraries*.

2. *Dalvik Virtual Machine*: Virtual mesin berbasis *register* yang dioptimalkan untuk menjalankan fungsi-fungsi secara efisien, di mana merupakan pengembangan yang mampu membuat linux kernel untuk melakukan *threading* dan management tingkat rendah.

e. *Linux Kernel*

Linux kernel adalah *layer* di mana inti dari *operating* sistem dari Android itu berada. Berisi *file-file system* yang mengatur sistem *processing, memory, reseource, drivers* dan sistem-sistem operasi android lainnya. Linux *kernel* yang digunakan Android adalah linux *kernel release* 2.6.



Gambar 2.14 Arsitektur Android [23]

2.5.3 Komponen Aplikasi

Fitur penting Android adalah bahwa satu aplikasi dapat menggunakan elemen dari aplikasi lain (untuk aplikasi yang memungkinkan). Sebagai contoh, sebuah aplikasi memerlukan fitur *scroller* dan aplikasi lain telah mengembangkan fitur scroller yang baik dan memungkinkan aplikasi lain menggunakannya. Maka pengembang tidak perlu lagi mengembangkan hal serupa untuk aplikasinya, cukup menggunakan scroller yang telah ada Agar fitur tersebut dapat bekerja,

sistem harus dapat menjalankan aplikasi ketika setiap bagian aplikasi itu dibutuhkan, dan pemanggilan objek java untuk bagian itu. Oleh karenanya Android berbeda dari sistem-sistem lain, Android tidak memiliki satu tampilan utama program seperti fungsi *main()* pada aplikasi lain. Sebaliknya, aplikasi memiliki komponen penting yang memungkinkan sistem untuk memanggil dan menjalankan ketika dibutuhkan.

2.5.3.1 *Activities*

Suatu *activity* akan menyajikan *user interface* (UI) kepada pengguna, sehingga pengguna dapat melakukan interaksi. Sebuah aplikasi android bisa jadi hanya memiliki satu *activity*, tetapi umumnya aplikasi memiliki banyak *activity* tergantung pada tujuan aplikasi dan desain dari aplikasi tersebut. Satu *activity* biasanya akan dipakai menampilkan aplikasi atau yang bertindak sebagai *user interface* (UI) saat aplikasi diperlihatkan kepada *user*. Untuk pindah dari satu *activity* ke *activity* lain kita dapat melakukannya dengan satu *even*, misalnya *click* tombol, memilih opsi atau menggunakan triggers tertentu. Secara hirarki sebuah *windows activity* dinyatakan dengan *method Activity.setContentView(). ContentView* adalah objek yang berada pada *root* hirarki.

2.5.3.2 *Service*

Service tidak memiliki *Graphic User Interface* (GUI), tetapi *service* berjalan secara *background*, sebagai contoh dalam mainkan musik, *service* mungkin memainkan musik atau mengambil data dari jaringan, tetapi setiap *service* harus berada dalam kelas induknya. Misalnya, *media player* sedang memutar lagu dari *list* yang ada, aplikasi ini akan memiliki dua atau lebih *activity* yang memungkinkan *user* untuk memilih lagu misalnya, atau menulis sms sambil *player* sedang jalan. Untuk menjaga musik tetap dijalankan, *activity player* dapat menjalankan *service*. *Service* dijalankan pada *thread* utama dari proses aplikasi.

2.5.3.3 *Broadcast Receiver*

Broadcast Receiver berfungsi menerima dan bereaksi untuk menyampaikan notifikasi. Contoh *broadcast* seperti notifikasi zona waktu

berubah, baterai low, gambar telah selesai di ambil oleh *camera*, atau pengubahan bahasa referensi yang digunakan. Aplikasi juga dapat menginisiasi *broadcast* misalnya memberikan informasi pada aplikasi lain bahwa ada data yang telah di unduh ke perangkat dan siap untuk digunakan.

Broadcast Receiver tidak memiliki *user interface* (UI), tetapi memiliki sebuah *activity* untuk merespon informasi yang mereka terima atau mungkin menggunakan *Notification Manager* untuk memberi tahu kepada pengguna, seperti lampu latar atau *vibrating* (getaran) perangkat dan lain sebagainya.

2.5.3.4 *Content Provider*

Content Provider membuat kumpulan aplikasi data secara spesifik sehingga bisa digunakan oleh aplikasi lain. Data disimpan dalam *file* sistem seperti *database* SQLite. *Content Provider* menyediakan cara untuk mengakses data yang dibutuhkan oleh suatu *activity*, misalnya ketika kita menggunakan aplikasi yang membutuhkan peta (*Map*), atau aplikasi yang membutuhkan untuk mengakses data kontak dan navigasi, maka di sinilah fungsi *content provider*.

2.5.4 Struktur Penulisan JSON

Tanda kurung kurawal ({}) menandakan JSON *Object* dan tanda kurung kotak ([]) menandakan JSON Array.

Berikut penjelasan lebih lanjut mengenai struktur JSON. JSON terbuat dari dua struktur:

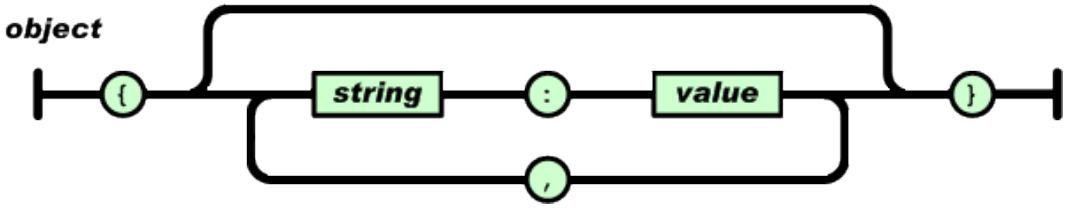
1. Kumpulan pasangan nama/nilai. Pada beberapa bahasa, hal ini dinyatakan sebagai objek (*object*), rekaman (*record*), struktur (*struct*), kamus (*dictionary*), tabel hash (*hash table*), daftar berkunci (*keyed list*) atau *associative array*.
2. Daftar nilai terurutkan (*an ordered list of values*). Pada kebanyakan bahasa, hal ini dinyatakan sebagai larik (*array*), vektor (*vector*), daftar (*list*) atau urutan (*sequence*).

Struktur-sturktur data tersebut dikenal sebagai struktur data universal. Pada dasarnya, semua bahasa pemrograman modern mendukung struktur data ini dalam bentuk yang sam maupun berlainan. Hal ini pantas disebut demikian, karena

format data mudah dipertukarkan dengan bahasa-bahasa pemrograman yang juga berdasarkan pada struktur data ini.

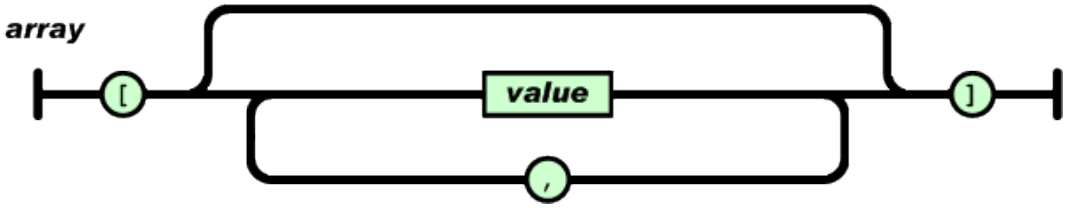
JSON menggunakan bentuk sebagai berikut:

1. *Object* adalah sepasang nama/nilai yang tidak terurutkan. Objek dimulai dengan { (kurung kurawal buka) dan diakhiri dengan } (kurung kurawal tutup). Setiap nama di ikuti dengan : (titik dua) dan setiap pasangan nama/nilai dipisahkan oleh , (koma). Ilustrasinya dapat dilihat pada Gambar 2.15.



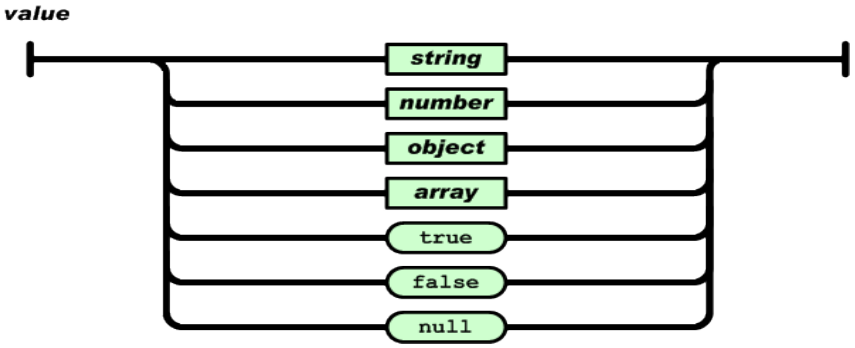
Gambar 2.15 Bentuk Penulisan *JSON Object* [24]

2. Array adalah kumpulan nilai yang terurutkan. Larik dimulai dengan [(kurung kotak buka) dan diakhiri dengan] (kurung kotak tutup). Setiap nilai dipisahkan oleh tanda koma (,). Ilustrasinya seperti pada Gambar 2.16



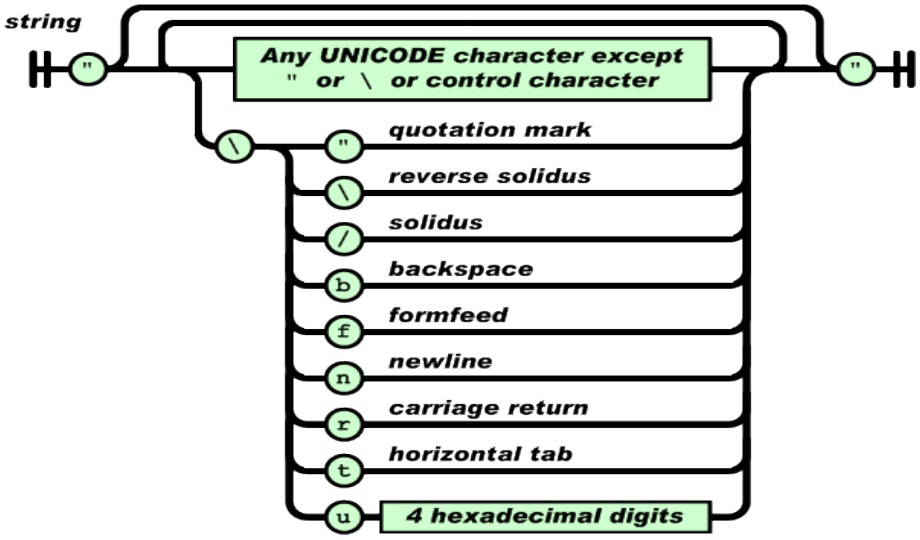
Gambar 2.16 Bentuk Penulisan *JSON Array* [24]

3. Nilau (*value*) dapat berupa sebuah string dalam tanda kutip ganda, atau angka, atau true atau false atau null, atau sebuah objek atau sebuah larik. Struktur-struktur tersebut dapat disusun bertingkat. Ilustrasinya bisa dilihat pada Gambar 2.17.



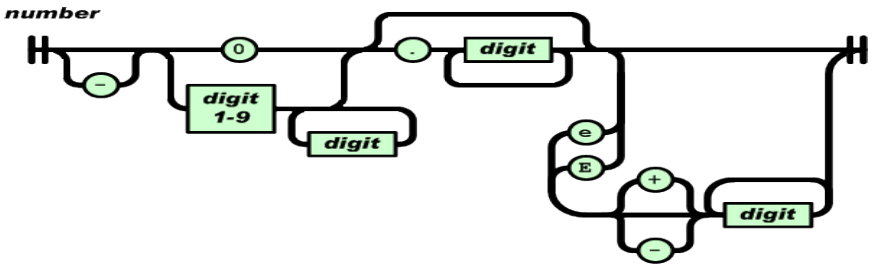
Gambar 2.17 Bentuk Penulisan *Value* [24]

4. *String* adalah kumpulan dari nol atau lebih karakter *Unicode*, yang dibungkus dengan tanda kutif ganda. Di dalam string dapat digunakan *backslash escapes* "\" untuk membentuk karakter khusus. Sebuah karakter mewakili karakter tunggal pada string. String sangat mirip dengan string C atau Java. Ilustrasinya dapat dilihat pada Gambar 2.18.



Gambar 2.18 Bentuk Penulisan *String* [24]

5. Angka sangat mirip dengan angka di C atau Java, kecuali format octal dan heksadesimal tidak digunakan. Ilustrasinya bisa dilihat pada Gambar 2.19.



Gambar 2.19 Bentuk Penulisan Angka [24]

2.6 JSON (*JavaScript Object Notation*)

JavaScript Object Notation merupakan format pertukaran data yang ringan, mudah dibaca dan ditulis oleh manusia, serta mudah diterjemahkan dan dibuat (*generate*) oleh komputer [24]. Format ini dibuat berdasarkan bagian dari bahasa pemrograman JavaScript, Standar ECMA-262 Edisi ke -3 – Desember 1999. JSON merupakan format teks yang tidak bergantung pada bahasa pemrograman apapun karena menggunakan gaya bahasa yang umum digunakan oleh programmer bahasa C termasuk C, C++, C#, Java, JavaScript, Perl, Python dan lain-lain. Oleh karena sifat-sifat tersebut, menjadikan JSON ideal sebagai bahasa pertukaran data.

Berikut ini adalah contoh sebuah konversi data xml ke format JSON:

```
{
        “contact”: [
        {
                “name”: “Kresna Abimanyu”,
                “company”: “PT. Aimee Butik Selaras”,
                “address”: “Jl. Serua Raya No. 12”,
                “city”: “ Depok”,
                “state”: “Indonesia”,
                “zip”: “120224”,
                “phone”: “08571411900”,
                “email”: abimanyu@aimeebutik.com
        }]}
```

2.7 Perangkat Lunak Pendukung

Pada pembuatan aplikasi ini penulis membutuhkan beberapa perangkat lunak untuk membantu membuat aplikasi ini. Berikut diantaranya:

2.7.1 Android Studio



Gambar 2.20 Logo Android Studio [25]

Android Studio merupakan lingkungan pengembangan Android baru yang berdasarkan intelliJ IDEA. Android Stuido yang sudah mencapai versi v0.3.2 pada bulan Oktober 2013. Android Studio ini mirip cara kerjanya dengan Eclipse + ADT plugin, Android Studio menyediakan alat pengembang Android yang terintegrasi dalam pengembangan dan debugging program. IntelliJ IDEA sendiri, merupakan sebuah komersial java IDE powerfull layaknya Eclipse dan netbeans yang dikembangkan oleh JetBrains. IDEA mendukung beberapa bahsa seperti: Java, JavaScript, HTML/ XHTML/ CSS, XML/ XSL, ActionScript/ MXML, Phyton, Ruby/ Jruby, SQL, PHP, dll.

Dengan kemampuan intelliJ IDEA yang dimilikinya, Android studio menawarkan pada anda:

1) Mendukung pengembangan berbasis Gradle.
2) Dapat menspesifikasikan pemfaktoran android dan melakukan perbaikan dengan cepat.
3) Memiliki lint tools untuk menangkap kinerja, kegunaan kompatibilitas versi dan masalah lainnya.
4) Memiliki kemampuan penandaan aplikasi dan ProGuard.
5) Memiliki kemampuan wizard berdasarkan template untuk membuat desain dan komponen android yang umum.

6) Sebuah layout editor yang kaya dan memungkinkan anda untuk melakukan drag-drop UI, pratinjau layout pada beberapa konfigurasi layar, dan banyak lagi.
7) Dibangun untuk membangun Google Cloud Platform sehingga mudah untuk mengintegrasikan Google Cloud Messaging dan App Engine sebagai komponen server-side.

2.7.2 IDE Arduino

IDE Arduino ini merupakan aplikasi yang mencakup *editor*, *compiler* dan *uploader* yang dapat digunakan oleh semua seri modul keluarga Arduino, seperti Arduino Deumilanove, Uno, Bluetooth, Mega. Kecuali ada beberapa tipe board produksi Arduino yang memakai mikrokontroler di luar seri AVR, seperti mikroprosesor ARM. *Editor sketch* pada IDE Arduino juga mendukung fungsi penomoran baris, *syntax highlighting* untuk pengecekan sintaksis kode *sketch* [3].

Proses kompilasi IDE Arduino diawali dengan proses pengecekan kesalahan *sintaksis sketch*, kemudian memanfaatkan pustaka *Processing* dan *avr-gcc sketch* dikompilasi menjadi berkas *object*, lalu berkas-berkas *object* digabungkan oleh pustaka Arduino menjadi berkas biner. Berkas biner ini diunggah ke cip mikrokontroler via kabel USB, *serial port* DB9, atau *Serial Bluetooth*.

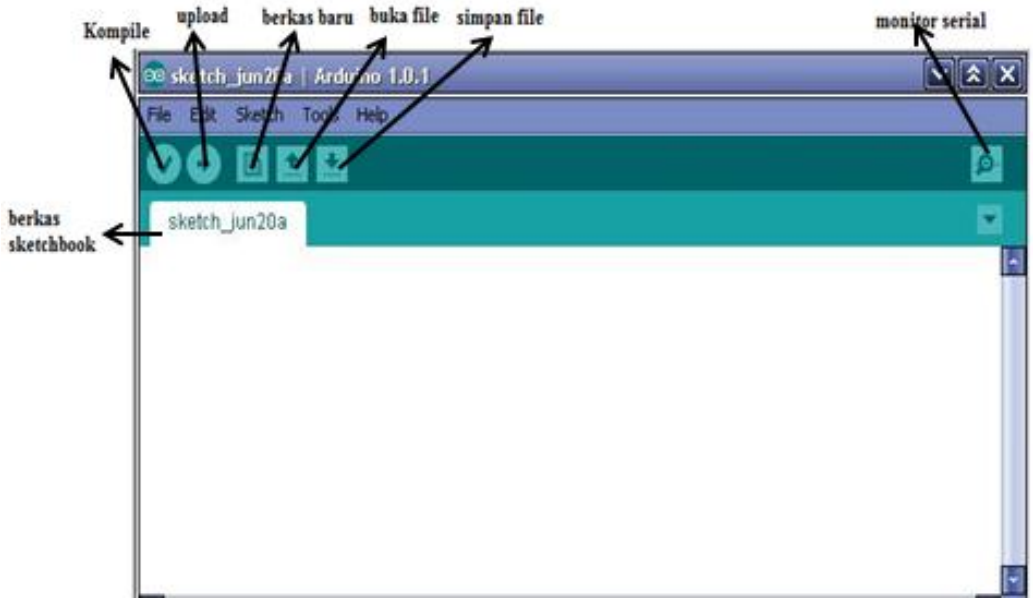
Compiler IDE Arduino juga memanfaatkan pustaka *open source* AVRLibc sebagai standar *de-facto* pustaka referensi dari fungsi register mikrokontroler AVR. Pustaka AVRLibc ini sudah disertakan dalam satu paket program IDE Arduino. Meskipun demikian, kita tidak perlu mendefinisikan directive #include dari pustaka AVRLibc pada *sketch* karena otomatis *compiler* me-*link* pustaka AVRLibc tersebut.

Ukuran berkas biner HEX hasil ompilasi akan semakin besar jika kode *sketch* semakin kompleks. Berkas biner memiliki ekstensi **.hex** beisi data intruksi program yang bisa dipahami oleh mikrokontroler target.

Selain itu, *port parallel* jiga bisa dipakai untuk mengunggah *bootloader* ke mikrokontroler. Meskipun demikian, cara ini sudah jarang digunakan karena kini

hampir tidak ada *mainboard* PC yang masih menyediakan *port paralel*, dan pada *notebook* juga sudah tidak menyertakan *port paralel*.

Dari Gambar 2.21 terlihat *button* (tombol) yang ada di IDE Arduino, button ***compile*** berfungsi untuk kompilasi *sketch* tanpa unggah ke *board* bisa dipakai untuk pengecekan kesalahan kode sintaks *sketch*. *Button* ***upload*** untuk mengunggah hasil kompilasi *sketch* ke *board* target. Pesan *error* akan terlihat jika *board* belum terpasang atau alamat *port* COM belum terkonfigurasi dengan benar.



Gambar 2.21 Menu ikon IDE Arduino versi 1.0.1

2.7.3 Adobe Photoshop CS6

Adobe Photoshop adalah *software* yang dibuat oleh perusahaan Adobe *System*, yang di khususkan untuk pengeditan foto atau gambar dan pembuatan *effect*. Perangkat lunak ini banyak digunakan oleh *Fotografer* Digital dan perusahaan iklan sehingga dianggap sebagai pemimpin pasar (*market leader*) untuk perangkat lunak pengolah gambar. Meskipun pada awalnya Photoshop dirancang untuk menyunting gambar untuk cetakan berbasis-kertas, Photoshop yang ada saat ini juga dapat digunakan untuk memproduksi gambar untuk *World Wide Web*. Beberapa versi terakhir juga menyertakan aplikasi tambahan, Adobe *Image Ready* untuk keperluan tersebut.

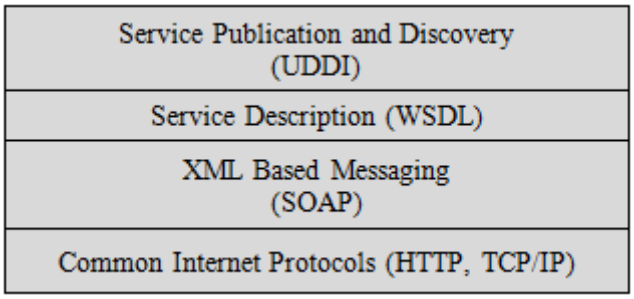
Photoshop juga memiliki hubungan erat dengan beberapa perangkat lunak penyunting media, animasi dan *authoring* buatan-*Adobe* lainnya. File format asli Photoshop, *PSD*, dapat di*export* ke dan dari Adobe *Image Ready*. Adobe Illustrator, Adobe Premiere Pro, After Effects dan Adobe Encore DVD untuk membuat DVD professional, menyediakan penyuntingan gambar *non-linear* dan layanan *special effect* seperti *background*, tekstur, dan lain-lain untuk keperluan televisi, film, dan situs web.

2.7.4 *Web Service*

Web service merupakan suatu komponen *software* yang merupakan *self-caontaining*, aplikasi modular *self-discribing* yang dapat dipublikasikan, dialokasikan, dan dilaksanakan pada web. *Web service* adalah teknologi yang mengubah kemampuan *internet* dengan menambahkan kemampuan *transactional web*, yaitu kemampuan *web* untuk saling berkomunikasi dengan pola *program-to-program* (P2P). *fokus web* selama ini didominasi oleh komunikasi *program-to-user* dengan interaksi *business-to-consumer* (B2C), sedangkan *transactional web* akan di dominasi oleh *program-to-program* dengan interaksi *business-to-business* [**26**].

Di bawah ini merupakan gambar blok bangunan *web service* yang mana menyediakan fasilitas komunikasi jarak jauh antara dua aplikasi yang merupakan layer arsitektur *web service*.

a. Layer 1: protokol *internet* standar yang digunakan sebagai sarana transportasi adalah HTTP dan TCP/IP.
b. Layer 2: *Simple Object Access Protocol* (SOAP) berbasiskan XML dan digunakan untuk pertukaran informasi antar sekelompok layanan.
c. Layer 3: *Web Service Definition Language* (WDSL) digunakan untuk mendeskripsikan attribut layanan.
d. Layer 4: *Universal Description, Discovery and Integration*, yang mana merupakan direktori pusat untuk deskripsi layanan.



Gambar 2.22 Blok *Web Service* [26]

2.7.4.1 XML

XML merupakan dasar terbentuknya *web service* yang digunakan untuk mendeskripsikan data. Pada level paling detail *web service* secara keseluruhan dibentuk diatas XML. Fungsi utama dari XML adalah komunikasi antar aplikasi, integrasi data, dan komunikasi aplikasi eksternal dengan partner luaran. Dengan standarisasi XML, aplikasi-aplikasi yang berbeda dapat dengan mudah berkomunikasi antar satu dengan yang lain.

2.7.4.2 SOAP

SOAP merupakan protokol untuk pertukaran informasi dengan desentralisasi dan terdistribusi. SOAP merupakan gabungan antara HTTP dengan XML karena SOAP umumnya menggunakan protocol HTTP sebagai sarana transport datanya dan data akan dipertukarkan ditulis dalam format XML. Karena SOAP menggunakan HTTP dan XML maka SOAP memungkinkan pihak-pihak yang mempunyai *platform*, sistem operasi dan perangkat lunak yang berbeda dapat saling mempertukarkan datanya. SOAP mengatur bagaimana *request* dan respon dari suatu *web service* bekerja.

2.7.4.3 WDSL

WDSL merupakan sebuah bahasa berbasis XML yang digunakan untuk mendefinisikan *web service* dan menggambarkan bagaimana cara untuk mengakses *web service* tersebut. Fungsi utama WSDL dalam *web service* adalah untuk mengotomasi mekanisme komunikasi *business-to-business* dalam *web service* melalui protokol *internet*.

WSDL merupakan representasi kontrak antara *requestor dan providernya*. Secara teknis merupakan representasi kontrak antara kode klien dan kode di server. Dengan menggunakan WDSL klien dapat memanfaatkan fungsi-fungsi publik yang disediakan oleh server.

2.7.4.4 NuSOAP

NuSOAP adalah sebuah kumpulan *clas-clas* PHP yang memungkinkan *user* untuk mengirim dan menerima pesan SOAP melalui protokol HTTP, NuSOAP didistribusikan oleh NuSphere Corporation (http://www.nusphere.com) sebagai *open source toolkit* dibawah lisensi GNU LGPL.

NuSOAP merupakan toolkit *web service* berbasis komponen. NuSOAP memiliki sebuah *class* dasar yang menyediakan *method* seperti serialisasi variabel dan pemaketan SOAP-*Envelope*. Interaksi *web service* dilakukan dengan *class client* yang disebut dengan *class "soapclient"* dan *class server* yang disebut dengan *class "soap_server"*. *Class-class* ini mengizinkan *user* untuk melakukan proses pengiriman dan penerimaan pesan-pesan SOAP dengan bantuan beberapa *class-class* pendukung lainnya untuk meengkapi proses tersebut.

2.7.5 MySQL

MySQL merupakan salah satu *database server* yang berkembang di lingkungan *open source* dan didistribusikan *secara free* (gratis) dibawah lisensi GPL.

MySQL merupakan RDBMS (*Relational Database Management System*) server. RDBMS adalah program yang memungkinkan pengguna *database* untuk membuat, mengelola dan menggunakan data pada suatu model *relational*. Dengan demikian tabel-tabel yang ada pada *database* memiliki relasi antara satu tabel dengan tabel lainnya.

2.7.5.1 Keunggulan MySQL

Beberapa keunggulan MySQL yaitu:

1. Cepat, handal dan mudah dalam penggunaanya

   MySQL lebih cepat tiga sampai empat kali dari pada *database server* komersial yang beredar saat ini, mudah diatur dan tidak memerlukan seseorang yang ahli untuk mengatur pemasangan MySQL.

2. Didukung oleh berbagai bahasa

   Database server MySQL dapat memberikan pesan error dalam berbagai bahasa seperti Belanda, Portugis, Spanyol, Inggris, Perancis, Jerman dan Italia.

3. Mampu membuat tabel berukuran sangat besar

   Ukuran maksimal dari setiap tabel yang dapat dibuat dengan MySQL adalah 4 GB sampai dengan ukuran *file* yang dapat ditangani oleh sistem operasi yang dipakai.

4. Lebih Murah

   MySQL bersifat *open source* dan didistribusikan dengan gratis tanpa biaya untuk UNIX *platform*, OS/2 dan *Windows Platform*.

2.8 Analisis Berbasis Objek

Analisis berorientasi objek atau Object Oriented Analys (OOA) adalah tahapan untuk menganalisis spesifikasi atau kebutuhan akan sistem yang akan dibangun dengan konsep berorientasi objek, apakah benar kebutuhan yang ada dapat diimplementasikan menjadi sebuah sistem beroriantasi objek.

OOA biasanya menggunakan kartu CRC (*Component, Responsibility, Collaborator*) untuk membangun kelas-kelas yang akan digunakan atau menggunakan UML (*Unified Modeling Language*) pada bagian diagram usecase, diagram kelas dan diagram objek.

2.9 Pemrograman Berorientasi Objek

Metodologi berorientasi objek adalah suatu strategi pembangunan peralngkan lunak yang mengorganisasikan perangkat lunak sebagai kumpulan objek yang berisi data dan operasi yang diberlakukan terhadapnya [**27**].

Metodologi berorientasi objek merupakan suatu cara bagaimana sistem perangkat lunak dibangun melalui pendekatan objek secara sistematis. Metode berorientas objek didasarkan pada penerapan prinsip-prinsip pengelolaan kompleksitas. Metode berorientasi objek meliputi rangkaian aktivitas analisis berorientasi objek, perancangan berorientasi objek, pemrograman berorientasi objek, dan pengujian berorientasi objek.

Pada saat ini, metode berorientasi objek banyak dipilih karena metodologi lama banyak menimbulkan masalah seperti adanya kesulitan pada saat mentransformasi hasil dari satu tahap pengembangan ke tahap berikutnya, misalnya pada metode pendekata terstruktur, jenis aplikasi yang dikembangkan saat ini berbeda dengan masa lalu. Aplikasi yang dikembangkan pada saat ini beragam dengan *platform* yang berbeda-beda, sehingga menimbulkan tuntutan kebutuhan metodologi pengembangan yang dapat mengakomodasi ke semua jenis aplikasi tersebut.

Keuntungan menggunakan metodologi berorientasi objek adalah sebagai berikut:

1. Meningkatkan produktivitas

   Karena kelas dan objek yang ditemukan dalam suatu masalah masih dapat dipakai ulang untuk masalah lainnya yang melibatkan objek tersebut (*reusable*).
2. Kecepatan pengembangan

   Karena sistem yang dibangun dengan baik dan benar pada saat analisis dan perancangan akan menyebabkan berkurangnya kesalahan pada saat pengodean.
3. Kemudahan pemeliharaan

   Karena dengan model objek, pola-pola yang cenderung tetap dan stabil dapat dipisahkan dan pola-pola yang mungkin sering berubah-ubah.
4. Adanya konsistensi

   Karena sifat pewarisan dan penggunaan notasi yang sama pada saat analisis, perancangan maupun pengkodean.

5. Meningkatkan kualitas perangkat lunak

   Karena pendekatan pengembangan lebih dekat dengan dunai nyata dan adanya konsistensi pada saat pengembangannya, perangkat lunak yang dihasilkan akan mampu memenuhi kebutuhan pemakai serta mempunyai sedikit kesalahan.

2.9.1 Konsep Dasar Berorientasi Objek

Pendekatan berorientasi objek merupakan suatu teknik atau cara pendekatan dalam melihat permasalahan dan sistem. Pendekatan berorientasi objek akan memandang sistem yang akan dikembangkan sebagai suatu kumpulan objek yang berkorespondensi dengan objek-objek dunia nyata.

Ada banyak cara untuk mengabstraksikan dan memodelkan objek-objek tersebut, mulai dan abstraksi objek, kelas, hubungan antar kelas sampai abstraksi sistem. Saat mengabstraksikan dan memodelkan objek, data dan proses-proses yang dipunyai oleh objek akan dikemas menjadi suatu kesatuan.

Dalam rekayasa perangkat lunak, konsep pendekatan berorientasi objek dapat diterapkan pada tahap analisis, perancanggan, pemrograman, dan pengujian perangkat lunak. Ada berbagai teknik yang dapat digunakan pada masing-masing tahap tersebut, dengan aturan dan alat bantu pemodelan tertentu.

Sistem berorientasi objek merupakan sebuah sistem yang dibangun dengan berdasarkan metode berorientasi objek adalah sebuah sistem yang komponennya dikemas menjadi kelompok data dan fungsi. Setiap komponen dalam sistem tersebut dapat mewarisi atribut dan sifat dan komponen lainnya, dan dapat berinteraksi satu sama lain.

Berikut ini adalah beberapa konsep dasar yang harus dipahami tentang metodologi berorientasi objek:

1. Kelas (*class*)

   Kelas adalah sekumpulan objek-objek dengan karakteristik yang sama. Kelas merupakan definisi statis dan himpunan objek yang sama yang mungkin lahir atau diciptakan dan kelas tersebut. Sebuah kelas akan mempunyai sifat (atribut), kelakuan (metode/operasi), hubungan

(*relationship*) dan arti. Suatu kelas dapat diturunkan dan kelas yang lain, dimana atribut dan kelas semula dapat diwariskan ke kelas yang baru. Secara tekniks kelas adalah sebuah struktur dalam pembuatan perangkat lunak. Kelas merupakan bentuk struktur pada kode program yang menggunakan metodologi berorientasi objek.

2. Objek (*object*)

   Objek adalah abstraksi dari sesuatu yang mewakili dunia nyata benda, manusia, satua organisasi, tempat, kejadian, struktur, status, atau hal-hal lain yang bersifat abstra. Objek merupakan suatu entitas yang mampu menyimpan informasi dan mempunyai operasi (kelakuan) yang dapat diterapkan atau dapat berpengaruh pada status objeknya. Objek mempunyai siklus hidup yaitu diciptakan, dimanipulasi, dan dihancurkan.

   Secara teknis, sebuah kelas saat program dieksekusi makan akan dibuat sebuah objek. Objek dilihat darisegi teknis adalah elemen pada saat program dieksekusi yang akan diciptakan, dimanipulasi, dan dihancurkan saat eksekusi sehinga sebuah objek hanya ada saat sebuah program dieksekusi. Jika masih dalam bentuk kode, disebut sebagai kelas jadi pada saat program dieksekusi, yang kita punya adalah objek, di dalam teks program yang kita lihat hanyalah kelas.

3. Metode (*method*)

   Operasi atau metode pada sebuah kelas hampir sama dengan fungsi atau prosedur pada metodologi struktural. Sebuah kelas boleh memiliki lebih dari satu metode atau operasi. Metode atau operasi yang berfungsi untuk memanipulasi objek itu sendiri. Operasi atau metode merupakan fungsi atau transformasi yang dapat dilakukan terhadap objek atau dilakukan oleh objek. Metode atau operasi dapat berasal dari *event*, aktifitas atau aksi keadaan, fungsi, atau kelakuan dunnia nyata. Contoh metode atau operasi misalnya *Read, Write, Move, Copy,* dan sebagainya.

4. Atribut (*attribute*)

   Atribut dari sebuah kelas adalah variabel global yang dimiliki sebuah kelas. Atribut dapat beruoa nilai atau elemen-elemen data yang dimiliki

oleh objek dalam kelas objek. Atribut dipunyai secara individual oleh sebuah objek, misalnya berat, jenis, nama, dan sebagainya.

5. Abstraksi (*abstraction*)

   Prinsip untuk merepresentasikan dunia nyata yang kompleks menjadi satu bentuk model yang sederhana dengan mengabaikan aspek-aspek lain yang tidak sesuai dengan permasalahan.

6. Enkapsulasi (*encapsulation*)

   Pembungkusa atribut data dan layanan (operasi-operasi) yang dipunyai objek untuk menyembunyikan implementasi dan objek sehingga objek lain tidak mengetahui cara kerja-nya.

7. Pewarisan (*inheritance*)

   Mekanisme yang memungkinkan satu objek mewarisi sebagian atau seluruh definisi dan objek lain sebagai bagian dan dirinya.

8. Antarmuka (*interface*)

   Antarmuka sangat mirip dengan kelas, tapi tanpa atribut kelas dan memiliki metode yang dideklarasikan tanpa isi. Deklarasi metode pada sebuah *interface* dapat diimplementasikan oleh kelas lain.

9. *Reusability*

   Pemanfaatan kembali objek yang sudah didefinisikan untuk suatu permasalahan pada permasalahan lainnya yang melibatkan objek tersebut.

10. Generalisasi dan Spesialisasi

    Menunjukkan hubungan antara kelas dan objek yang umum dengan kelas dan objek yang khusus. Misalnya kelas yang lebih umum (generalisasi) adalah kendaraan darat dan kelas khususnya (spesialisasi) adalah mobil, motor, dan kereta.

11. Komunikasi Antarobjek

    Komunikasi antarobjek dilakukan lewat pesan yang dikirim dari satu objek ke objek lainnya.

12. Polimorfisme (*polymorphism*)

    Kemampuan suatu objek digunakan di banyak tujuan yang berbeda dengan nama yang sehingga menghemat baris program.

13. *Package*

Package adalah sebuah kontainer atau kemasan yang dapat digunakan untuk mengelompokkan kelas-kelas sehingga memungkinkan beberapa kelas yang bernama sama disimpan dalam *package* yang berbeda.

2.9.1.1 Pengenalan UML

Pada perkembangan teknologi perangkat lunak, diperlukan adanya bahasa yang digunakan untuk memodelkan perangkat lunak yang akan dibuat dan perlu adanya standarisasi agar orang di berbagai negara dapat mengerti pemodelan perangkat lunak. Seperti yang kita ketahui bahwa menyatukan banyak kepala untuk menceritakan sebuah ide dengan tujuan untuk memahami hal yang tidaklah mudah, oleh karena itu diperlukan sebuah bahasa pemodelan perangkat lunak yang dapat dimengerti oleh banyak orang.

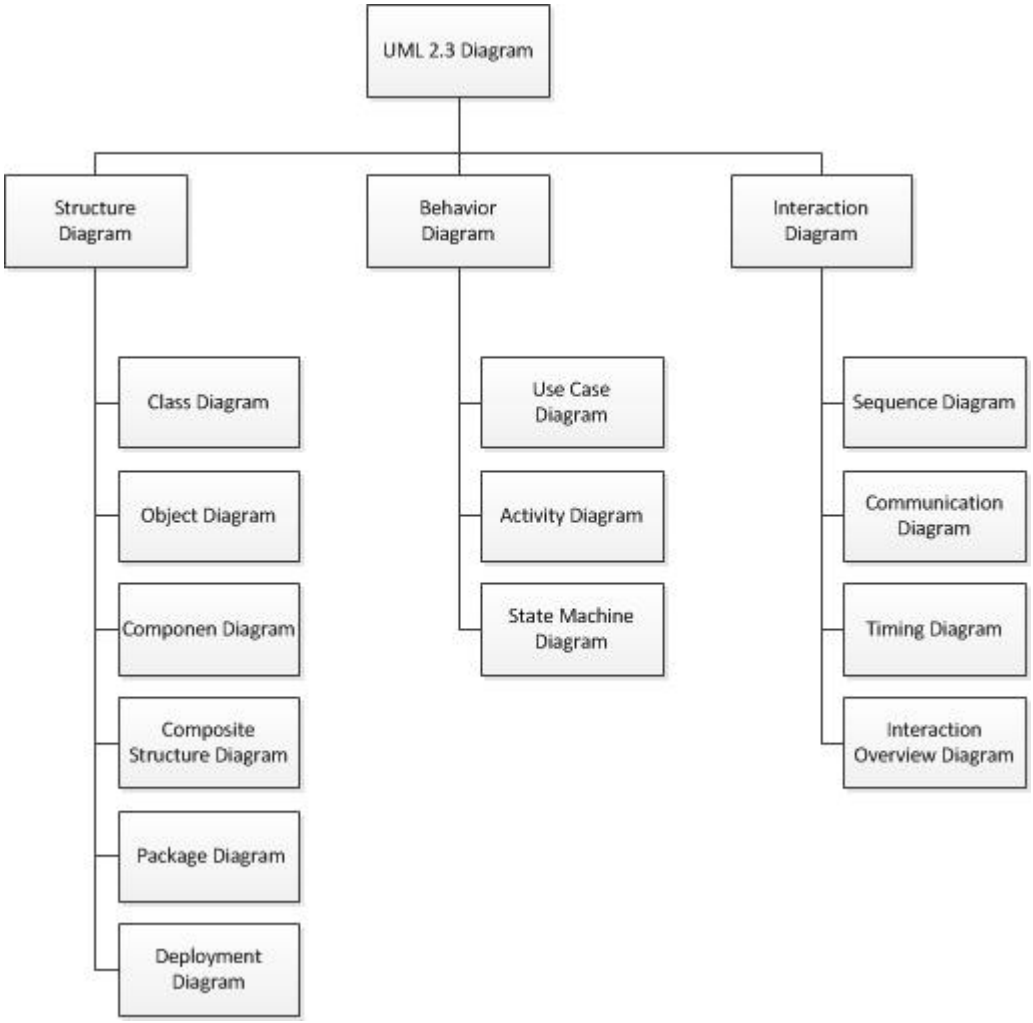
Banyak orang yang telah membuat bahasa pemodelan pembangunan perangkat lunak sesuai dengan teknologi pemrograman yang berkembang pada saat itu, misalnya yang sempat berkembang dan digunakan banyak pihak adalah *Data Flow Diagram* (DFD) untuk memodelkan perangkat

lunak yang menggunakan pemrograman prosedural atau struktural, kemudian juga ada *State Transition Diagram* (STD) yang digunakan untuk memodelkan sistem *real time* (waktu nyata). Pada perkembangan teknil pemrograman berorientasi objek, munculah sebuah standarisasi bahasa pemodelan untuk pembangunan perangkat lunak yang dibangun dengan menggunakan teknik pemrograman berorientasi objek, yaitu *Unified Modeling Language* (UML). UML muncul karena adanya kebutuhan pemodelan voisual untu menspesifikasikan, menggambarkan, membangun, dan dokumentasi dari sistem perangkat lunak. UML merupakan bahasa visual untuk pemodelan dan komunikasi mengenai sebuah sistem dengan menggunakan digaram dan teks-teks pendukung.UML hanya berfungsi untuk melakukan pemodelan. Jadi penggunaan UML tidak terbatas pada metodologi tertentu, meskipun pada kenyataannya UML paling banyak digunakan pada metodologi berorientasi objek. Seperti yang kita ketahui di dunia sistem informasi yang tidak dapat dibakukan, semua tergantung

kebutuhan, lingkunan dan konteksnya. Begitu juga dengan perkembangan penggunaan UML bergantung pada level abstraksi penggunaannya. Jadi, belum tentu pandangan yang perbeda dalam penggunaan UML adalah suatu yang salah, tapi perlu ditelaah dimanakah UML digunakan dan hal apa yang ingin digambarkan. Secara analogi jika dengan bahasa yang digunakan sehari-hari, belum tentu penyampaian bahasa dengan puisi adalah hal yang salah. Sistem informasi bukanlah ilmu pasti, maka jika ada banyak perbedaan dan interpretasi di dalam bidang sistem informasi merupakan hal yang sangat wajar.

2.9.1.2 Diagram UML

Pada UML 2.3 terdiri dari 13 macam diagram yang dikelompokkan dalam 3 kategori. Pembagian kategori dan macam-macam diagram tersebut dapat dilihat pada gambar berikut:



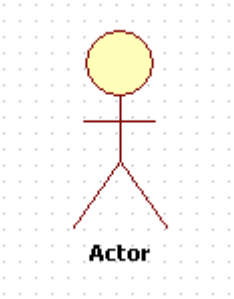
Gambar 2.23 Diagram UML [27]

2.9.2 *Use Case Diagram*

Diagram *use case* merupakan pemodelan untuk kelakuan sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih actor dengan sistem informasi yang akan dibuat. Secara kasar *use case* igunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem informasi dan siapa saja yang berhak dalam menggunakan fungsi-fungsi itu.

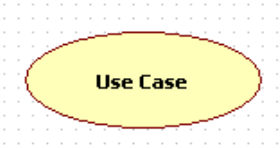
Syarat penamaan pada *use case* adalah nama didefinisikan sesimple mungkin dan dapat dipahami. Ada dua hal utama pada *use cae* yaitu pendefinisian apa yang disebut actor dan *use case*.

1. Aktor merupakan orang, proses atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi walupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang.



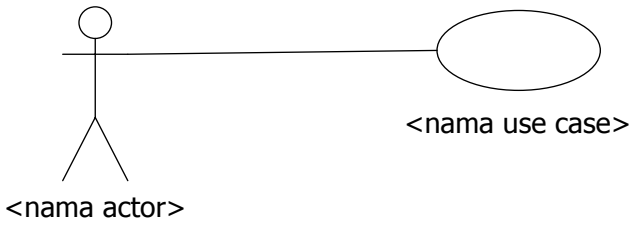
Gambar 2.24 Aktor [27]

2. *Use case* merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor.



Gambar 2.25 *Use Case* [27]

3. Relationship atau relasi digambarkan sebagai bentuk garis antara dua simbol dalam *use case* diagram. Relasi antara actor dan *use case* disebut juga dengan asosiasi (*association*). Asosiasi ini digunakan untuk menggambarkan bagaimana hubungan antara keduanya. Relasi-relasi yang terjadi pada *use case* diagram bisa antara actor dengan *use case* atau *use case* dengan *use case*.



Gambar 2.26 *Relationship*

Relasi antara aktor dengan *use case* :

a. *Include*, pemanggilan *use case* oleh *use case* lain atau untuk menggambarkan suatu use case termasuk di dalam *use case* lain (diharuskan). Contohnya adalah pemanggilan sebuah fungsi program. Digambarkan dengan garis lurus berpanah dengan tulisan <<*include*>>.
b. *Extend*, digunakan ketika hendak menggambarkan variasi pada kondisi perilaku normal dan menggunakan lebih banyak kontrol form dan mendeklarasikan ekstension pada *use case* utama. Atau dengan kata lain adalah perluasan dari *use case* lain jika syarat atau kondisi terpenuhi. Digambarkan dengan garis berpanah dengan tulisan <<*extend*>>.
c. *Generalization/Inheritance,* dibuat ketika ada sebuah kejadian yang lain sendiri atau perlakuan khusus dan merupakan pola berhubungan base-parent *use case*. Digambarkan dengan garis berpanah tertutup dari base *use case* ke *parent use case*.

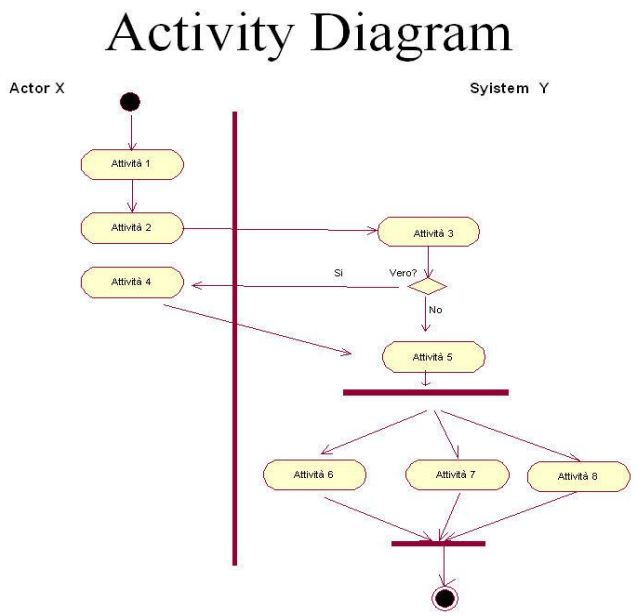
2.9.3 *Activity Diagram*

Activity Diagram menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak. Yang perlu diperhatikan disini adalah bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan sistem.

Diagram aktivitas juga banyak digunakan untuk mendefinisikan hal-hal berikut:

1. Rancangan proses bisnis dimana setiap urutan aktivitas yang digambarkan merupakan proses bisnis sistem yang didefinisikan.
2. Urutan atau pengelompokan tampilan dari sistem/*user interface* dimana setiap aktivitas dianggap memiliki sebuah rancangan antarmuka tampilan.
3. Rancangan pengujian dimana setiap aktivitas dianggap memerlukan sebuah pengujian yang perlu didefinisikan kasus ujinya.

4. Rancangan menu yang ditampilkan pada perangkat lunak.

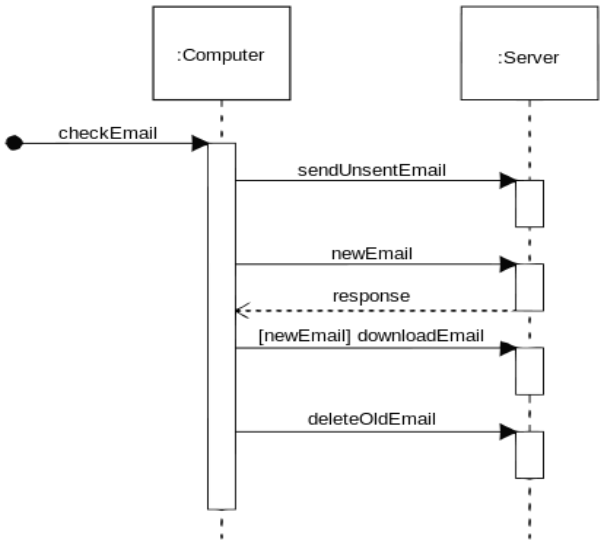


Gambar 2.27 ***Activity Diagram***

2.9.4 *Sequence Diagram*

Diagram sekuen menggambarkan keakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dengan *message* yang dikirimkan dan diterima antar objek. Oleh karena itu untuk menggambar diagram sekuen maka harus diketahui objek-objek yang terlibat dalam sebuah *use ccase* beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu. Membuat diagram sekuen juga dibutuhkan untuk melihat sekenario yang ada pada *use case*.

Banyaknya diagram sekuen yang harus digambar adalah minimal sebanyak pendefinisian *use case* yang memiliki proses sendiri atau yang penting semua *use case* telah didefinisikan interaksi jalannya pesan sudah dicakup pada diagram sekuen sehingga semakin banyak *use case* yang didefinisikan maka diagram sekuen yang harus dibuat juga semakin banyak.



Gambar 2.28 *Sequence Diagram*

2.9.5 *Class Diagram*

Diagram kelas atau *class diagram* menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi. Atribut merupakan variabel-variabel yang dimiliki oleh suatu kelas. Operasi atau metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas.

Kelas-kelas yang ada pada struktur sistem harus dapat melakukan fungsi-fungsi sesuai dengan kebutuhan sistem. Susunan struktur kelas yang baik pada diagram kelas ebaiknya memiliki jenis-jenis kelas berikut :

1. Kelas *Main*

   Kelas yang memiliki fungsi awal dieksekusi ketika sistem dijalankan.
2. Kelas yang menangani tampilan sistem

   Kelas yang mendefinisikan dan mengatur tampilan ke pemakai.
3. Kelas yang diambil dari pendefinisian *use case*

   Kelas yang menangani fungsi-fungsi yang harus ada diambil dari pendefinisian *use case.*

4. Kelas yang diambil dari pendefinisian data

   Kelas yang digunakan untuk memegang atau membungkus data menjadi sebuah kesatuan yang diambil maupun akan disimpan ke basis data.

Jenis-jenis kelas diatas juga dapat digabungkan satu sama lain sesuai dengan pertimbangan yang dianggap baik asalkan fungsi-fungsi yang sebaiknya ada pada struktur kelas tetap ada. Susunan kelas juga dapat ditambahkan kelas utilitas seperti koneksi ke basis data, membaca *file teks*, dan lain sebagainya sesuai kebutuhan.

Dalam mendefinisikan metode yang ada di dalam kelas perlu memperhatikan apa yang disebut dengan *cohesion* dan *coupling*. *Cohesion* adalah ukuran seberapa dekat keterkaitan instruksi di dalam sebuah metode terkait satu sama lain sedangkan *coupling* adalah ukuran seberapa dekat keterkaitan instruksi antara metode yang satu dengan metode yang lain dalam sebuah kelas. Sebagai aturan secara umum maka sebuah metode yang dibuat harus memiliki kadar *choesion* yang kuat dan kadar *coupling* yang lemah.

Dalam class diagram terdapat beberapa relasi (hubungan antar class) yaitu:

1. *Generalization dan Inheritence*

   Diperlukan untuk memperlihatkan hubungan pewarisan (inheritance) antar unsur dalam diagram kelas. Pewarisan memungkinkan suatu kelas mewarisi semua atribut, operasi, relasi, dari kelas yang berada dalam hirarki pewarisannya.

2. *Associations*

   Hubungan statis antar *class*. Umumnya menggambarkan class yang memiliki atribut berupa *class* lain, atau class yang harus mengetahui ekstensi *class* lain. Dalam notasi UML kita mengenal asosiasi 2 arah (*bidirectional*) dan 1 arah (*undirectional*).

3. *Aggregation*

   Hubungan antar-*class* dimana *class* yang satu (*part class*) adalah bagian dari *class* lainnya (*whole class*).

4. *Composition*

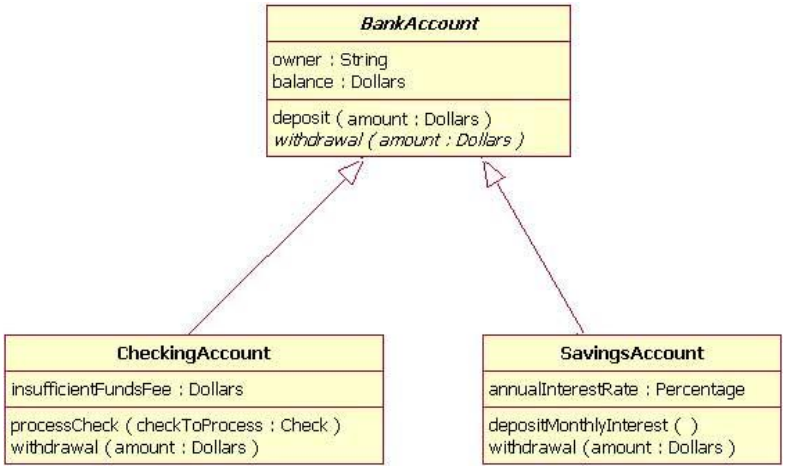
   Aggregation dengan ikatan yang lebih kuat. Di dalam *composite aggregation*, siklus hidup *part class* sangat bergantung pada *whole class* sehingga bila objek instance dari *whole class* dihapus maka objek *instance* dari *part calss* juga akan terhapus.

5. *Depedency*

   Hubungan antar-*class* dimana sebuah *class* memiliki ketergantungan pada *class* lainnya tetapi tidak sebaliknya.

6. *Realization*

   Hubungan antar-*class* dimana sebuah *class* memiliki keharusan untuk mengikuti aturan yang ditetapkan *class* lainnya. Biasanya *realization* digunakan untk menspesifikasikan hubungan antara sebuah *interface* dengan *class* yang mengimplementasikan *interface* tersebut.



Gambar 2.29 *Class Diagram*