BAB II

LANDASAN TEORI

2.1. Tinjauan Jurnal

Untuk memperkuat penelitian ini, penulis mengambil dua jurnal referensi yang dijadikan teori dasar untuk materi yang berhubungan dengan komponen SMD.

> Menurut hariyanto didik (2009:18) menjelaskan pengertian resistor sebagai berikut:
> Resistor adalah komponen dasar elektronika yang digunakan untuk membatasi jumlah arus yang mengalir dalam satu rangkaian. Sesuai dengan namanya resistor bersifat resistif dan umumnya terbuat dari bahan karbon. Dari hukum Ohms diketahui, resistansi berbanding terbalik dengan jumlah arus yang mengalir melaluinya. Satuan resistansi dari suatu resistor disebut Ohm atau dilambangkan dengan simbol W (Omega).

Sedangkan pengertian kapasitor sebagai berikut:

> Menurut Zain (2013:153) menjelaskan bahwa:
> Kapasitor merupakan komponen pasif elektronika yang sering dipakai didalam merancang suatu sistem yang berfungsi untuk mengeblok arus DC, Filter, dan penyimpan energi listrik. Didalamnya 2 buah pelat elektroda yang saling berhadapan dan dipisahkan oleh sebuah insulator. Sedangkan bahan yang digunakan sebagai insulator dinamakan dielektrik.

2.2. Konsep Dasar Program

Menurut Irawan (2009:137) ”program adalah perangkat lunak yang terdiri dari sekumpulan kode, bisa dieksekusi dan dijalankan melalui sistem operasi”.

Adapun tahapan-tahapan secara umum dalam pembangunan program antara lain :

1. Definisi Masalah

   Tahapan ini bertujuan untuk merumuskan dari permasalahan yang ada, sehingga akan didapatkan usulan-usulan yang dapat menyelesaikan permasalahan tersebut.

2. Analisa Kebutuhan

   Tahapan ini bertujuan untuk menentukan masukan-masukan dan keluarannya, mengetahui kebutuhan-kebutuhan penggunanya, menentukan spesifikasi minimum dan spesifikasi yang disarankan dari perangkat yang akan digunakan sebagai penerapan aplikasi tersebut.

3. Desain Algoritma

   Untuk menentukan urutan langkah-langkah komputasi yang tepat yang nantinya akan dieksekusi oleh komputer untuk memproses masukan-masukan mejadi keluaran yang diinginkan.

4. Pengkodean

   Tahapan penulisan pada teknologi pemrograman yang dipilih dengan mengikuti dari desain algortima yang sudah dibuat sebelumnya.

5. Testing

   Tahapan yang bertujuan untuk mengetahui kesalahan-kesalahan yang mungkin masih ada dalam aplikasi yang sedang dibangun, yang kemudian dilakukan koreksi sampai aplikasi dibebaskan dari kesalahan.

6. Dokumentasi

   Menyimpan semua informasi mengenai cara mengoperasikan aplikasi tersebut, prosedur-prosedur yang bekerja dan bagaimana aplikasi tersebut berjalan.

7. Pemeliharaan

Pemeliharaan digunakan untuk memantau aktivitas dari analisis sistem pada saat perangkat lunak telah dipergunakan oleh pemakai.

2.2.1. OOP *(Object Oriented Programming)*

Menurut Sukamto dan Shalahuddin (2013:100) mengatakan "Metodologi berorientasi objek adalah suatu strategi pembangunan perangkat lunak yang mengorganisasikan perangkat lunak sebagai kumpulan objek yang berisi data dan operasi yang diberlakukan terhadapnya".

2.2.2. Java

Menurut Supardi (2010:1) mengatakan "*Java* merupakan bahasa pemprograman yang dikembangkan dari bahasa pemprograman *C++*. Sehingga bahasa pemprograman ini seperti bahasa *C++* ".

Sekali anda menuliskan sebuah program dengan menggunakan *java,* anda dapat menjalankannya hampir di semua komputer dan perangkat lain yang support Java, dengan sedikit perubahan atau tanpa perubahan sama sekali dalam kodenya. Aplikasi dengan berbasis *java* ini dikompulasikan ke dalam *p-code* dan bisa dijalankan dengan *java virtual machine*. Fungsionalitas dari *java* ini dapat berjalan dengan platform system operasi yang berbeda karena sifatnya yang umum dan non-spesifik.

Slogan *java* adalah "Tulis sekali, jalankan di manapun". Sekarang ini *java* menjadi sebuah bahasa pemrograman yang populer dan dimanfaatkan secara luas untuk pengembangan perangkat lunak. Kebanyakan perangkat lunak yang menggunakan *java* adalah ponsel feature dan ponsel pintar atau smartphone. Kelebihan *java* salah satunya adalah multiplatform. *Java* dapat dijalankan dalam

beberapa platform komputer dan sistem operasi yang berbeda. Hal ini sesuai dengan slogannya yang sudah dibahas sebelumnya. Kelebihan lain dari *java* yaitu memiliki library yang lengkap. Library yang dimaksud disini adalah sebuah kumpulan dari program yang disertakan dalam *java*. Hal ini akan memudahkan pemrograman menjadi lebih mudah. Kelengkapan library semakin beragam jika ditambah dengan karya komunitas *java*.

2.2.3. Android

Menurut Safaat (2011:1) mengungkapkan bahwa "*Android* adalah sebuah sistem informasi untuk perangkat *mobile* berbasis linux yang mencakup sistem informasi, *middleware* dan aplikasi". Secara umum Android adalah platform yang terbuka (*Open Source*) bagi para pengembang untuk menciptakan aplikasi mereka sendiri untuk digunakan oleh berbagai piranti bergerak.

Perangkat berbasis *android* hanya mempunyai satu layar *foreground*. Saat menghidupkan *android*, yang pertama kali terlihat adalah *home*, kemudian bila memulai sebuah aplikasi, maka *User Interface* (UI) aplikasi tersebut akan menumpuk diatas layar sebelumnya (*home*). Semua proses tersebut direkam di *application stack* oleh sistem *activity manager*. Menekan tombol back akan membuat kita kembali ke halaman sebelumya.

Pada proses pengembangan aplikasi berbasis *android* ada beberapa komponen aplikasi yang sangat harus diperhatikan, antara lain :

1. Activity

   Fungsi *activity* adalah untuk mewakili satu buah *user interface*.

2. Service

   Service tidak memiliki *user interface*, namun berjalan di belakang layar.

3. Intens

   Intens adalah mekanisme untuk menggambarkan sebuah *action* secara detail seperti bagaimana cara mengambil sebuah photo.

4. Content Providers

   Menyediakan cara untuk mengakses data yang dibutuhkan oleh suatu *activity*, seperti aplikasi berbasis peta.

5. Resource

   Resource digunakan untuk menyimpan *file-file non-coding* yang diperlukan pada sebuah aplikasi.

2.2.4. Eclipse

Menurut Safaat (2011:16) menjelaskan bahwa " *Eclipse* adalah IDE untunk pengembangan *java* atau *android*", dapat dijalankan di semua *platform* (*platform-independent*). Berikut ini adalah sifat dari *Eclipse*:

1. *Multi-platform*: Target sistem operasi *Eclipse* adalah *Microsoft Windows*, *Linux,Solaris, AIX, HP-UX* dan *Mac OS X*.
2. *Mulit-language*: *Eclipse* dikembangkan dengan bahasa pemrograman *Java*, akan tetapi Eclipse mendukung pengembangan aplikasi berbasis bahasa pemrograman lainnya, seperti *C/C++ Cobol*, *Python*, *Perl PHP*, dan lain sebagainya.
3. *Multi-role*: Selain sebagai IDE untuk pengembangan aplikasi, *Eclipse* pun bisa digunakan untuk aktivitas dalam siklus pengembangan perangkat lunak, seperti dokumentasi, test perangkat lunak, pengembangan web, dan lain sebagainya.

Eclipse pada saat ini merupakan salah satu IDE favorit dikarenakan gratis dan *open source*, yang berarti setiap orang boleh melihat kode pemrograman perangkat lunak ini. Selain itu, kelebihan dari *eclipse* yang membuatnya populer adalah kemampuannya untuk dapat dikembangkan oleh pengguna dengan komponen yang dinamakan *plug*-in.

2.2.5. Android SDK

Menurut Safaat (2011:15) menjelaskan bahwa "*Android SDK* adalah tool *API (Applicaramming Interface)* yang diperlukan untuk memulai mengembangkan aplikasi pada platform android menggunakan bahasa pemprograman java". Untuk *source* SDK *android* ini dapat dilihat dan diunduh langsung dari situs resmi pengembang *android* di http//:www.developer.android.com.

2.3. Metode Algoritma

Dalam pembuatan aplikasi ini, sesuai dengan rumus untuk menghitung salah satu komponen SMD yaitu berupa resistor, berikut algoritmanya:

$$\Omega = D1+D2 * D3$$

Gambar II. 1. Algortima Menghitung Resistor SMD 3 digit

Keterangan :

Ω = Nilai Hambatan (Ω)
D1 = Nilai Pada Digit 1
D2 = Nilai Pada Digit 2
D3 = Nilai Pada Digit 3

2.4. Pengujian Sistem

Untuk melakukan pengujian pada program ,maka digunakan pengujian *black box* dan *white box* . Berikut pengertian *black box* dan *white box*.

Menurut Pressman (2010:597) "pengujian kotak hitam, juga disebut pengujian perilaku, berfokus pada persyaratan fungsional perangkat lunak". Dengan demikian, teknik pengujian kotak hitam memungkinkan perakayasa perangkat lunak untuk membuat beberapa kumpulan kondisi masukan yang sepenuhnya akan melakukan semua kebutuhan fungsional untuk suatu program. Pengujian kotak hitam bukan teknik alternatif untuk teknik kotak putih. Sebaliknya ini merupakan pendekatan pelengkap yang mungkin dilakukan untuk mengungkap kelas kesalahan yang berbeda dari metode kotak putih.

Menurut Pressman (2010:597) pengujian kotak hitam berupaya untuk menemukan kesalahan dalam kategori berikut:

1. Fungsi yang salah atau hilang
2. Kesalahan antarmuka
3. Kesalahan dalam struktur data atau akses basis data eksternal
4. Kesalahan perilaku atau kinerja
5. Kesalahan inisialisasi dan penghentian.

Menurut Efendi (2011:50) menjelaskan bahwa " Pengujian *White Box* merupakan pengujian yang dititik beratkan kepada prosedur cara program tertentu bekerja dengan memberikan suatu kondisi". White Box Testing adalah pengujian dengan melihat ke dalam modul untuk meneliti kode-kode program yang ada, dan menganalisis apakah ada kesalahan atau tidak. Jika ada modul yang menghasilkan output yang tidak sesuai dengan proses bisnis yang dilakukan, maka baris-baris

program, variabel, dan parameter yang terlibat pada unit tersebut akan dicek satu persatu dan diperbaiki, kemudian di-*compile* ulang.

2.5. Peralatan Pendukung

Dalam perancangan aplikasi *android* ini diperlukan *tools* yang bisa membantu dalam pengerjaan aplikasi komponen SMD. Adapun tools yang digunakan penulis dalam perancangan aplikasi ini adalah UML *(Unifield Modeling Language).*

Menurut Sukamto dan Shalahuddin (2013:133) menjelaskan bahwa "UML adalah suatu yang salah, tapi perlu ditelaah dimanakah UML digunakan dan hal apa yang ingin divisualkan",

Pada perkembangan teknik pemrograman berorientasi objek, munculah sebuah standarisasi bahasa pemodelan untuk pembangunan perangkat lunak yang dibangun dengan menggunakan teknik pemrograman yang berbasis objek yaitu *Unifield Modeling Language* atau lebih dikenal dengan singkatan UML. UML muncul karena adanya kebutuhan pemodelan visual untuk mengimplmentasikan, menggambarkan, membangun, dan dokumentasi dari sistem perangkat lunak. UML merupakan bahasa visual untuk pemodelan dan komunikasi mengenai sebuah sistem dengan menggunakan diagram dan teks-teks pendukung.

Secara analogi jika dengan bahasa yang kita gunakan sehari-hari belum tentu penyampaian bahasa dengan puisi adalah hal yang salah. Sistem informasi bukanlah ilmu pasti, maka jika ada banyak perbedaan dan implementasi didalam bidang sistem informasi merupakan hal yang sangat wajar.

1. Use Case

Menurut Sukamto dan Shalahuddin (2013:155) menjelaskan bahwa *"Use Case* atau diagram *use case* merupakan pemodelan untuk kelakuan sistem informasi yang akan dibuat". *Use case* mendeskripsikan sebuah intraksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat, *use case* digunakan untuk mengetahui fungsi apa saja ada didalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi itu. syarat penamaan pada *use case* adalah nama didefinisikan sesimpel mungkin dan dapat dipahami, ada dua hal utama pada *use case* yaitu pendefinisian atau apa saja yang disebut aktor dan *use case.*

a. Aktor merupakan orang, proses atau sistem lain yang berintraksi dengan sistem informasi ayang akan dibuat diluar sistem sistem informasi yang akan dibuat itu sendiri jadi walaupun jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang.
b. *Use case* merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor.

2. Class Diagram

Menurut Sukamto dan Shalahuddin (2013:141) menjelaskan bahwa "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.

a. Artibut merupakan variabel-variabel yang dimiliki oleh suatu kelas.
b. Operasi atau metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas.

3. Sequence Diagram

Menurut Sukamto dan Shalahuddin (2011:137) menjelaskan bahwa ”Diagram sekuen menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek”. Oleh karena itu untuk menggambarkan diagram sekuen makan harus diketahui objek-objek yang terlibat dalan *use case* beserta metode-metode yang dimiliki kelas yang diimplementasikan menjadi objek itu.

Banyaknya diagram sekuen yang harus digambar adalah sebanyak pendefinisian *use case* yang memiliki proses sendiri atau yang penting semua *use case* yang didefiniskan maka diagram sekuen yang harus dibuat juga semakin banyak.

4. Activity Diagram

Menurut Sukamto dan Shalahuddin (2013:161) menjelaskan bahwa “Diagram aktivitas atau *activity diagram* menggambarkan *work flow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis”,

Yang perlu diperhatikan disini adalah bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor jadi aktivitas yang dapat dilakukan sistem.