BAB 2
TINJAUAN PUSTAKA

2.1 Profil Tempat Kerja Praktek

JSO atau Jaya Sukma Organik merupakan usaha dibidang pertanian organik yang termasuk kedalam kelompok tani desa sukajaya. Jaya Sukma Organik dibangun pada tanggal 22 April 2017 di Subang Jawa Barat sebagai usaha pertanian dan perkebunan. Saat ini Jaya Sukma Organik beroperasi di dua tempat yakni di desa sukajaya sebagai tempat pengolahan dan hasil dari pengolahan tersebut akan dikirim ke subang untuk pengemasan dan sebagainya.

2.1.1 Logo Jaya Sukma Organik

Logo dari Jaya Sukma Organik (JSO) dapat di,lihat pada Gambar 2.1 Logo Jaya Sukma Organik.

Gambar 2.1 Logo Jaya Sukma Organik

2.1.2 Visi dan Misi Perusahaan

2.1.2.1 Visi

Menjadi bidang usaha pertanian organik terbaik yang dapat menghasilkan bibit-bibit tanaman dan pupuk kimia maupun organik yang bermutu dan bermanfaat.

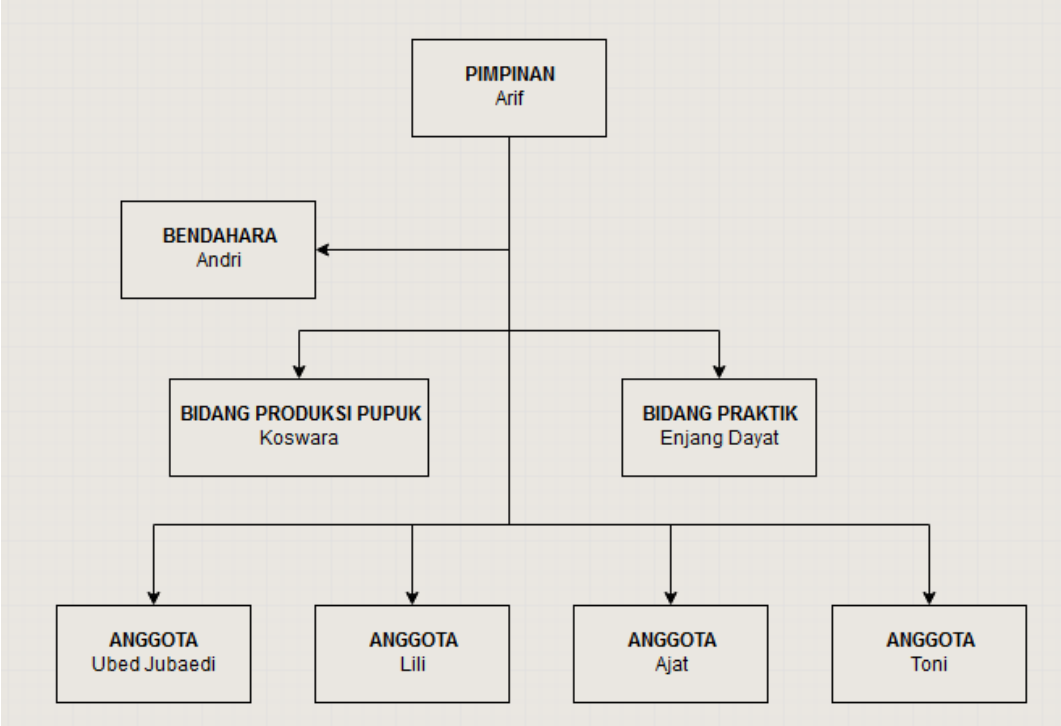
2.1.2.2 Misi

Misi dari Jaya Sukma Organik adalah sebagai berikut :

1. Memberikan pelayanan, mutu, dan kepuasan yang terbaik kepada pelanggan.
2. Menjaga kepercayaan dan loyalitas pelanggan.
3. Membangun perkebunan organik yang bermutu.
4. Mengembangkan bibit tanaman dan pembibitan buah yang berkualitas.

2.2 Struktur Organisasi

Struktur Organisasi dari Jaya Sukma Organik adalah sebagai berikut.



Gambar 2.2 Struktur Organisasi JSO

2.3 Landasan Teori

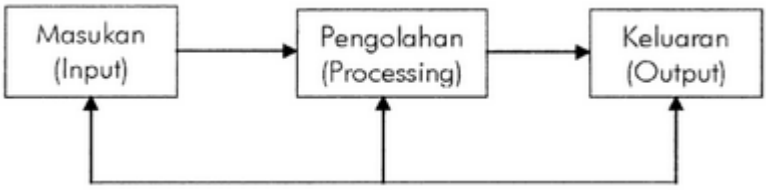
Landasan teori merupakan penjelasan berbagai konsep dasar dan teori-teori yang berkaitan dalam perancangan dan penerapan *Internet of Things* (IoT) untuk kontrol alat pengubah sampah organik menjadi pupuk kering menggunakan arduino uno.

2.3.1 Definisi Sistem

Secara sederhana sistem dapat diartikan sebagai suatu kumpulan atau himpunan dari unsur atau variabel-variabel yang saling terorganisasi, saling berinteraksi, dan saling bergantung satu sama lain.

Murdick dan Ross (1993) mendefinisikan sistem sebagai perangkat elemen yang digabungkan satu dengan lainya untuk suatu tujuan bersama. Sementara, definisi sistem dalam kamus *Webster's Unbriged* adalah elemen-elemen yang saling berhubungan dan membentuj satu kesatuan atau organisasi.

Menurut *Scott* (1996), sistem terdiri dari unsur-unsur seperti masukan *(input)*, pengolahan *(processing)* , serta keluaran *(output)*. Ciri pokok sistem menurut Gapspert ada empat, yaitu sistem itu beroprasi dalam suatu lingkungan, terdiri atas unsur-unsur, ditandai dengan saling berhubungan, dan memunyai satu fungsi atau tujuan utama.



Gambar 2.3 Model Sistem

Gambar diatas menunjukan bahwa sistem atau pendekatan sistem minimal harus mempunyai empat komponen, yakni masukan, pengolahan, keluaran, dan balikan atau kontrol.

2.3.2 Jaringan Komputer

Jaringan Komputer adalah himpunan "interkoneksi" antara 2 komputer *autonomous* atau lebih yang terhubung dengan media transmisi kabel atau tanpa kabel(wireless). Bila sebuah komputer dapat membuat komputer lainnya restar, shutdown, atau melakukan kontrol lainnya, maka komputer-komputer tersebut bukan *autonomous* (tidak melakukan kontrol terhadap komputer lain dengan akses penuh)[8].

2.3.3 Internet of Things

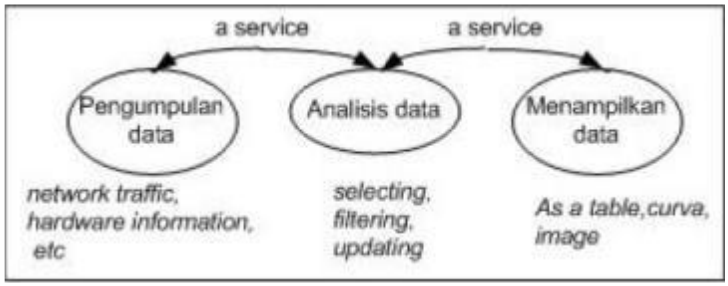
Menurut Wikipedia, *Internet of Things* atau dikenal juga dengan singkatan IoT, merupakan sebuah konsep yang bertujuan untuk memperluas manfaat dari konektivitas internet yang tersambung secara terus-menerus. Adapun kemampuan seperti berbagi data, remote control, dan sebagainya, termasuk juga pada benda di dunia nyata. Contohnya bahan pangan, elektronik, koleksi, peralatan apa saja, termasuk benda hidup yang semuanya tersambung ke jaringan lokal dan global melalui sensor yang tertanam dan selalu aktif.

Makna serupa yang lain *Internet of Things* (IoT) adalah sebuah konsep/skenario dimana suatu objek yang memiliki kemampuan untuk mentransfer data melalui jaringan tanpa memerlukan interaksi manusia ke manusia atau manusia ke komputer.

"A Things"pada *Internet of Things* dapat didefinisikan sebagai subjek misalkan orang dengan monitor implant jantung, hewan peternakan dengan transponder biochip, sebuah mobil yang telah dilengkapi built-in sensor untuk memperingatkan pengemudi ketika tekanan banrendah. Sejauh ini, IoT paling erat hubungannya dengan komunikasi *machine-to-machine* (M2M) dibidang manufaktur dan listrik, perminyakan, dan gas. Produk dibangun dengan kemampuan komunikasi M2M yang sering disebut dengan sistemcerdas atau "smart"[2].

2.3.4 Sistem Monitoring

Sistem adalah suatu jaringan kerja dari prosedur-prosedur yang saling berhubungan, berkumpul bersama-sama untuk melakukan suatu kegiatan atau untuk menyelesaikan suatu sasaran yang tertentu (Jogiyanto, 2005). Sistem *monitoring* merupakan suatu proses untuk mengumpulkan data dari berbagai sumber daya. Biasanya data yang dikumpulkan merupakan
data yang *realtime* (Gheyb Jhuana Ohara, 2012). Secara garis besar tahapan dalam sebuah sistem monitoring terbagi ke dalam tiga proses besar seperti yang terlihat pada Gambar 2.3 Proses dalam sistem monitoring.



Gambar 2.4 Proses dalam sistem monitoring

Proses-proses yang terjadi pada suatu sistem *monitoring* dimulai dari pengumpulan data seperti data dari *network traffic*, *hardware information*, dan lain-lain yang kemudian data tersebut dianalisis pada proses analisis data dan pada akhirnya data tersebut akan ditampilkan.

2.3.5 Sistem Kendali

Sistem kendali telah memegang peranan yang sangat penting dalam perkembangan ilmu dan teknologi. Peranan sistem kendali meliputi semua bidang kehidupan. Dalam peralatan, misalnya proses pada industry pesawat terbang, peluru kendali, pesawat ruang angkasa, dan lain-lain. Sedangkan dalam bidang non teknis meliputi bidang biologi, ekonomi, social, kedokteran, dan lainlain. Sistem kendali yang semakin berkembang dapat meningkatkan kinerja sistem, kualitas produksi, dan menekan biaya produksi. Sistem kendali dapat dikatakan sebagai hubungan antara komponen yang membentuk sebuah konfigurasi sistem, yang akan menghasilkan tanggapan sistem yang diharapkan Jadi harus ada yang dikendalikan, yang merupakan suatu sistem fisis, yang biasa disebut dengan kendalian.

2.3.6 Mikrokontroller

Menurut Chamim (2012) Mikrokontroler adalah sebuah sistem komputer yang seluruh atau sebagian besar elemennya dikemas dalam satu chip IC, sehingga sering disebut single chip microcomputer. Mikrokontroler merupakan sistem komputer yang mempunyai salah satu atau beberapa tugas yang sangat spesifik.

2.3.7 Firebase

Firebase adalah suatu layanan dari Google yang digunakan untuk mempermudah para pengembang aplikasi dalam mengembangkan aplikasi. Dengan adanya Firebase, pengembang aplikasi bisa fokus mengembangkan aplikasi tanpa harus memberikan usaha yang besar. Dua fitur yang menarik dari *Firebase* yaitu *Firebase Remote Config* dan *Firebase Realtime Database*. Selain itu terdapat fitur pendukung untuk aplikasi yang membutuhkan pemberitahuan yaitu *Firebase Notification*.

Firebase menyediakan realtime database dan backend sebagai layanan. Layanan ini menyediakan pengembang antarmuka pemrograman aplikasi yang memungkinkan data aplikasi disinkronkan diseluruh klien dan disimpan di Firebase cloud. Perusahaan menyediakan pustaka klien yang memungkinkan integrasi dengan *Android*, iOS, *JavaScript*, Java, *Swift* dan *Node*.js aplikasi. *Database* juga dapat diakses melalui *REST* API dan mengikat untuk beberapa *JavaScript Frameworks* seperti *AngularJS, React, Ember*.js dan *Backbone*.js. *Developer* yang menggunakan realtime database dapat mengamankan data dengan menggunakan aturan keamanan yang diberlakukan oleh server perusahaan. *Cloud Firestore* yang merupakan generasi mendatang dari *Firebase Realtime Database* dirilis untuk penggunaan beta. *Firebase Realtime Database* merupakan *database* yang di-host pada *cloud*. Data disimpan sebagai JSON dan disinkronkan secara *realtime* ke setiap klien yang terhubung.

2.3.8 MIT App Inventor

App Inventor adalah aplikasi web *open* source yang awalnya dikembangkan oleh Google, dan saat ini dikelola oleh *Massachusetts Institute of Technology* (MIT).

App Inventor memungkinkan pengguna baru untuk memprogram komputer untuk menciptakan aplikasi perangkat lunak bagi sistem operasi Android. App Inventor menggunakan antarmuka grafis, serupa dengan antarmuka pengguna pada Scratch dan StarLogo TNG, yang memungkinkan pengguna untuk men-drag-and-drop objek visual untuk menciptakan aplikasi yang bisa dijalankan pada perangkat Android. Dalam menciptakan App Inventor, Google telah melakukan riset yang

berhubungan dengan komputasi edukasional dan menyelesaikan lingkungan pengembangan online Google.

2.3.9 Pemrograman Berorientasi Objek

Pemrograman berorientasi objek (*object-oriented programming* disingkat OOP) merupakan pemograman yang berorientasikan kepada objek. Semua data dan fungsi di dalam paradigma ini dibungkus dalam kelas-kelas atau objek-objek. Setiap objek dapat menerima pesan, memproses data, dan mengirim pesan ke objek lainnya. Rancangan berorientasi objek merupakan suatu teknik yang memusatkan rancangan pada data (objek) dan interface. Fasilitas pemrograman berorientasi objek pada Java pada dasarnya adalah sama dengan C++. Feature pemrograman berorientasi objek pada Java benarbenar sebanding dengan C++, perbedaan utama antara Java dengan C++ terletak pada penurunanberganda (*multiple inheritance*), untuk ini Java memiliki cara penyelesaian yang lebih baik.

Model data berorientasi objek dikatakan dapat memberi fleksibilitas yang lebih, kemudahan mengubah program, dan digunakan luas dalam teknik piranti lunak skala besar. Lebih jauh lagi, pendukung OOP mengklaim bahwa OOP lebih mudah dipelajari bagi pemula dibanding dengan pendekatan sebelumnya, dan pendekatan OOP lebih mudah dikembangkan dan dirawat[9].

2.3.9.1 Konsep Dasar Berorientasi Objek

Berikut ini adalah konsep-konsep dasar yang harus diperhatikandan di pahami dalam metodologi berorientasi objek:

1. Class

   Merupakan kumpulan atas definisi data dan fungsi-fungsi dalam suatu unit untuk suatu tujuan tertentu. Sebagai contoh '*class of dog*' adalah suatu unit yang terdiri atas definisi-definisi data dan fungsi-fungsi yang menunjuk pada berbagai macam perilaku/ turunan dari anjing. Sebuah class adalah dasar dari modularitas dan struktur dalam pemrograman berorientasi *object*. Sebuah *class* secara tipikal sebaiknya dapat dikenali oleh seorang *non-programmer* sekalipun terkait dengan domain permasalahan yang ada, dan kode yang terdapat dalam sebuah *class*

sebaiknya (relatif) bersifat mandiri dan independen (sebagaimana kode tersebut digunakan jika tidak menggunakan OOP). Dengan modularitas, struktur dari sebuah program akan terkait dengan aspek-aspek dalam masalah yang akan diselesaikan melalui program tersebut. Cara seperti ini akan menyederhanakan pemetaan dari masalah ke sebuah program ataupun sebaliknya[9].

2. Objek

   Membungkus data dan fungsi bersama menjadi suatu unit dalam sebuah program komputer, objek merupakan dasar dari modularitas dan struktur dalam sebuah program komputer berorientasi objek.

3. Abstraksi

   Kemampuan sebuah program untuk melewati aspek informasi yang diproses olehnya, yaitu kemampuan untuk memfokus pada inti. Setiap objek dalam sistem melayani sebagai model dari "pelaku" abstrak yang dapat melakukan kerja, laporan dan perubahan keadaannya, dan berkomunikasi dengan objek lainnya dalam sistem, tanpa mengungkapkan bagaimana kelebihan ini diterapkan. Proses, fungsi atau metode dapat juga dibuat abstrak, dan beberapa teknik digunakan untuk mengembangkan sebuah pengabstrakan.

4. Enkapsulasi

   Memastikan pengguna sebuah objek tidak dapat mengganti keadaan dalam dari sebuah objek dengan cara yang tidak layak; hanya metode dalam objek tersebut yang diberi izin untuk mengakses keadaannya. Setiap objek mengakses *interface* yang menyebutkan bagaimana objek lainnya dapat berinteraksi dengannya. Objek lainnya tidak akan mengetahui dan tergantung kepada representasi dalam objek tersebut.

5. Polimorfisme melalui pengiriman pesan

   Tidak bergantung kepada pemanggilan subrutin, bahasa orientasi objek dapat mengirim pesan; metode tertentu yang berhubungan dengan sebuah pengiriman pesan tergantung kepada objek tertentu di mana pesa tersebut dikirim. Contohnya, bila sebuah burung menerima pesan "gerak cepat", dia akan

menggerakan sayapnya dan terbang. Bila seekor singa menerima pesan yang sama, dia akan menggerakkan kakinya dan berlari. Keduanya menjawab sebuah pesan yang sama, namun yang sesuai dengan kemampuan hewan tersebut. Ini disebut polimorfisme karena sebuah variabel tungal dalam program dapat memegang berbagai jenis objek yang berbeda selagi program berjalan, dan teks program yang sama dapat memanggil beberapa metode yang berbeda di saat yang berbeda dalam pemanggilan yang sama. Hal ini berlawanan dengan bahasa fungsional yang mencapai polimorfisme melalui penggunaan fungsi kelas-pertama.

2.3.9.2 UML

UML *(Unified Modelling Language)* adalah salah satu alat bantu yang sangat handal di dunia pengembangan sistem yang berorientasi objek. Hal ini disebabkan karena UML menyediakan bahasa pemodelan visual yang memungkinkan bagi pengembang sistem untuk membuat cetak biru atas visi mereka dalam bentuk yang baku, mudah dimengerti serta dilengkapi dengan mekanisme yang efektif untuk berbagi dan mengkomunikasikan rancangan mereka dengan yang lain.

UML merupakan kesatuan dari bahsa pemodelan yang dikembangkan oleh Booch, *Object modelling technique (OMT)* dan *Object Oriented Software Enginering (OOSE).* Metode Booch dari Gredy Booch sangat terkenal dengan nama metode *Design Object Oriente.* Keunggulan dari metode Booch adalah pada detile dan kayanya dengan notasi dan elemen[11].

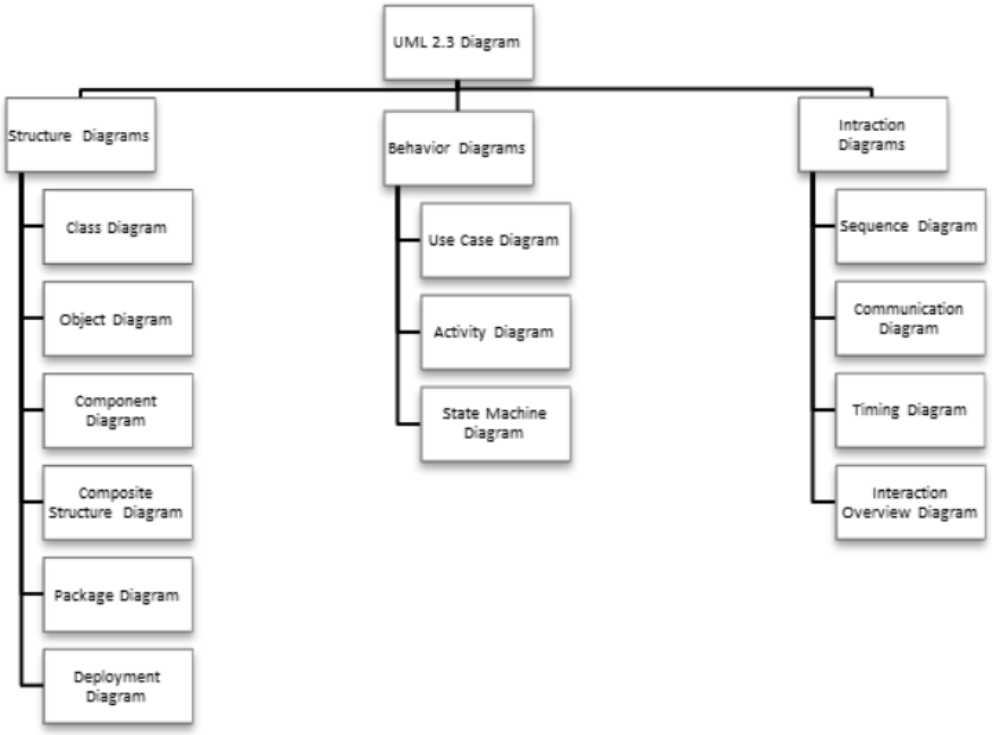
Dengan menggunakan UML kita dapat membuat model untuk semua jenis aplikasi piranti lunak, dimana aplikasi tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. Tetapi karena UML juga menggunakan *class* dan *operation* dalam konsep dasarnya, maka ia lebih cocok untuk penulisan piranti lunak dalam bahasabahasa berorientasi objek seperti C++, Java, C# atau VB.NET. Walaupun demikian, UML tetap dapat digunakan untuk modeling aplikasi prosedural dalam VB atau C. Seperti bahasa-bahasa lainnya, UML mendefinisikan notasi dan *syntax*/semantik. Notasi

UML merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram piranti lunak. Setiap bentuk memiliki makna tertentu, dan UML *syntax* mendefinisikan bagaimana bentuk-bentuk tersebut dapat dikombinasikan. Notasi UML terutama diturunkan dari 3 notasi yang telah ada sebelumnya: Grady Booch OOD (Object-Oriented Design), Jim Rumbaugh OMT (Object Modeling Technique), dan Ivar Jacobson OOSE (Object-Oriented Software Engineering).

Sejarah UML sendiri cukup panjang. Sampai era tahun 1990 seperti kita ketahui puluhan metodologi pemodelan berorientasi objek telah bermunculan di dunia. Diantaranya adalah: metodologi booch , metodologi coad , metodologi OOSE , metodologi OMT , metodologi shlaer-mellor, metodologi wirfs-brock, dsb. Masa itu terkenal dengan masa perang metodologi (*method war*) dalam pendesainan berorientasi objek. Masing-masing metodologi membawa notasi sendiri-sendiri, yang mengakibatkan timbul masalah baru apabila kita bekerjasama dengan group/perusahaan lain yang menggunakan metodologi yang berlainan[9].

2.3.9.3 Diagram UML

Pada UML terdapat 13 macam diagram yang dikelompokan dalam 3 kategori diagram. Kategori dan macam-macam diagram tersebut dapat dilihat pada Gambar 2.4 Diagram UML[9].



Gambar 2.5 Diagram UML

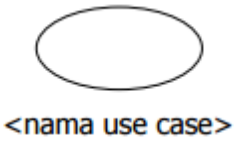
Berikut ini penjelasan singkat dari pembagian kategori tersebut :

1. Structure diagrams yaitu kumpulan diagram yang digunakan untuk menggambarkan suatu struktur statis dari sistem yang dimodelkan.
2. Behavior diagrams yaitu kumpulan diagram yang digunakan untuk menggambarkan kelakuan sistem atau rangkaian perubahan yang terjadi pada sebuah sistem.
3. Interaction diagrams yaitu kumpulan diagram yang digunakan untuk menggambarkan interaksi sistem dengan sistem lain maupun interaksi antar subsistem pada suatu system.

2.3.9.4 Use Case Diagram

Use case adalah deskripsi fungsi dari sebuah sistem dari perspektif pengguna. Use case bekerja dengan cara mendeskripsikan tipikal interaksi antara user sebuah sistem dengan sistemya itu sendiri melalui sebuah cerita bagaimana sebuah sistem dipakai. Urutan langkah-langkah yang menerangkan antara pengguna dan sistem disebut scenario. Setiap scenario mendeskripsikan urutan

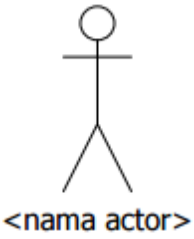
kejadian. Setiap urutan diinisialisasi oleh orang, sistem yang lain, perangkat keras atau urutan waktu. Dengan demikian use case secara singkat bisa dikatakan sebagai serangkaian scenario yang digabungkan bersama-sama oleh tujuan umum pengguna[11].



Gambar 2.6 Bentuk *Use Case*

Actor adalah sesuatu (entitas) yang berhubungan dengan sistem dan berpartisipasi dalam *use case*. *Actor* menggambarkan orang, sistem atau entitas *Eksternal* yang secara khusus membangkitkan sistem dengan input atau masukan kejadian-kejadian, atau menerima sesuatu dari sistem. *Actor* dilukiskan dengan peran yang mereka mainkan dalam *use case*, seperti *Staff*, Kurir dan lain-lain.

Dalam *use case* diagram terdapat satu aktor pemulai atau *initiator actor* yang membangkitkan rangsangan awal terhadap sistem, dan mungkin sejumlah aktor lain yang berpartisipasi atau participating *actor*. Akan sangat berguna untuk mengetahui siapa aktor pemulai tersebut[11].



Gambar 2.7 Bentuk Aktor pada UML

Relasi (*relationship*) 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*[11].

Gambar 2.8 Bentuk Relasi pada UML

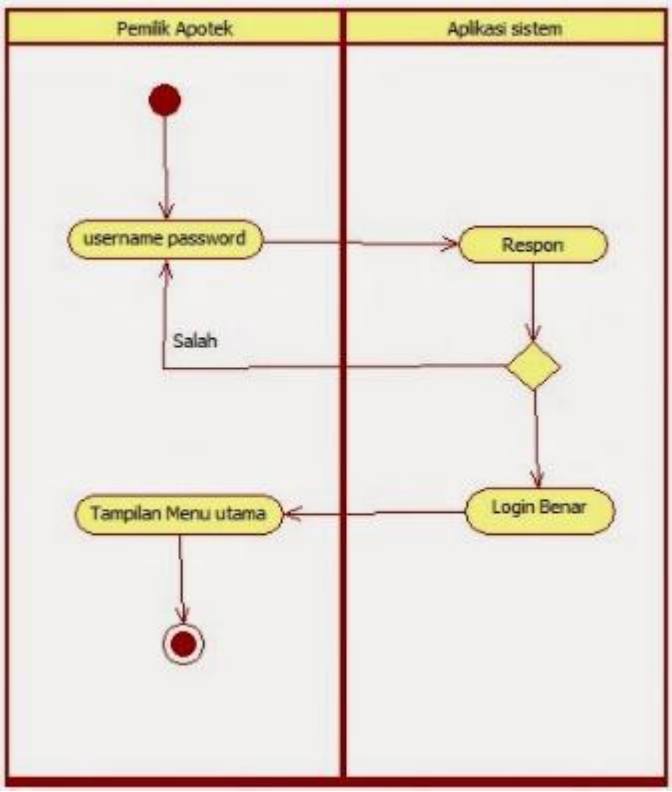
Relasi antara *use case* dengan *use case* :

1. *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*>>.
2. *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*>>.
3. *Generalization/Inheritance*, dibuat ketika ada sebuah kejadian yang lain sendiri atau perlakuan khusus dan merupakan pola berhubungan baseparent *use case*. Digambarkan dengan garis berpanah tertutup dari *base use case* ke parent *use case*.

2.3.9.5 *Activity Diagram*

Activity Diagram adalah bagian penting dalam UML yang menggambarkan aspek dinamis dari sistem. Logika prosedural, proses bisnis dan alir kerja suatu bisnis bisa dengan mudah dideskripsikan dalam *activity diagram*. *Activity Diagram* mempunyai peran seperti *flowchart*, akan tetapi perbedaanya dengan *flowchart* adalah *activity diagram* bisa mendukung perilaku paralel sedangkan *flowchart* tidak bisa. Diagram aktivitas juga banyak digunakan untuk mendefinisikan hal-hal berikut [11]:

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

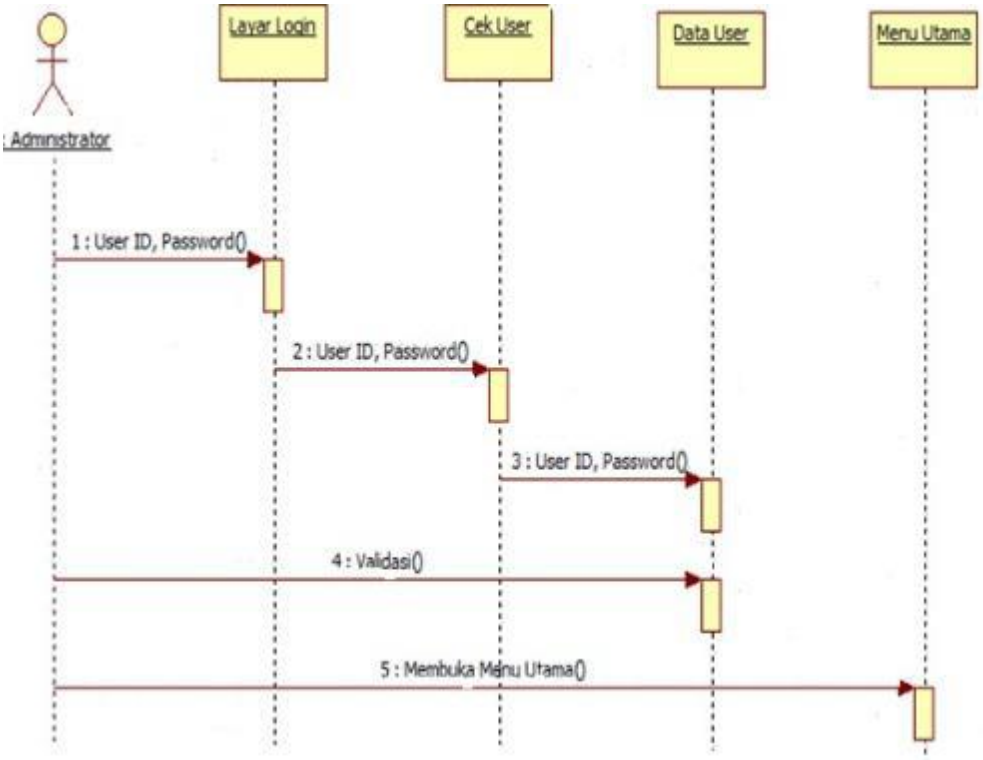


Gambar 2.9 Contoh *Activity* Diagram

2.3.9.6 *Squence* Diagram

Squence Diagram digunakan untuk menggambarakan perilaku pada sebuah scenario. Diagram ini menunjukan sejumlah contoh obyek dan *message* yang diletakan diantara obyek-obyek ini didalam *use case*.

Komponen utama *squence* diagram terdiri atas obyek yang dituliskan dengan kotak segiempat bernama. *Message* diwakili oleh garis dengan tanda panah dan waktu yang ditunjukan dengan progres vertikal[11].



Gambar 2.10 Contoh *Squence* Diagram

2.3.9.7 *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 fungsifungsi sesuai dengan kebutuhan sistem. Susunan struktur kelas yang baik pada diagram kelas ebaiknya memiliki jenis-jenis kelas berikut[11] :

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 usecase 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 class 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.

2.3.10 Perangkat Keras

Perangkat keras adalah suatu perangkat yang dapat membantu mempermudah dalam membuat alat. Perangkat keras berupa perangkat berbentuk fisik. Perangkat keras yang digunakan antara lain adalah:

2.3.10.1 Arduino Uno

Arduino UNO adalah papan mikrokontroler *open-source* berbasis mikrokontroler *Microchip* ATmega328P dan dikembangkan oleh Arduino.cc. Papan ini dilengkapi dengan set pin input / output digital dan analog yang dapat dihubungkan ke berbagai papan ekspansi dan sirkuit lainnya.

Gambar 2.11 Arduino Uno

Spesifikasi arduino uno adalah sebagai berikut:

Tabel 2.1 Spesifikasi Arduino Uno

Microcontroller	ATMega328P
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 *provide* PWM *output*)
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATMega328P)
SRAM	2 KB
EEPROM	1 KB
Clock Speed	16 MHz
Length	68.6 mm

2.3.10.2 Relay

Relay adalah sebuah Saklar (*Switch*) yang dioperasikan secara listrik dan merupakan komponen *Electromechanical* 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 2A. Modul relay digunakan sebagai kontrol penyambung sekaligus pemutus arus listrik yang tehubung pada pompa air, motor DC.

Gambar 2.12 Relay 1 Channel

Spesifikasi relay 2 channel adalah sebagai berikut:

Tabel 2.2 Spesifikasi Relay 2 Channel

Relay	1 Channel
Signal Control	TTL Level
Max Switch Voltage	250 VAC 30 VDC
Indikator	LED
Operating Voltage	5V
Contact Action	10ms / 5ms

2.3.10.3 Motor DC

Motor DC adalah motor listrik yang memerlukan suplai tegangan arus searah pada kumparan medan untuk diubah menjadi energi gerak mekanik. Kumparan medan pada motor dc disebut stator (bagian yang tidak berputar) dan kumparan jangkar disebut rotor (bagian yang berputar). Motor arus searah, sebagaimana namanya, menggunakan arus langsung yang tidak langsung/direct-unidirectional. Motor DC memiliki 3 bagian atau komponen utama untuk dapat berputar sebagai berikut. Keceptan motor DC dapat di atur melalui pengolahan dari PWM (*pulse width modulation*) dari Arduino itu sendiri. Seperti yang telah disebutkan di atas bahwa salah satu hal yang dapat mempengaruhi kecepatan motor adalah tegangan.

Gambar 2.13 Motor DC

Berikut adalah spesifikasi dari Motor DC:

Tabel 2.3 Spesifikasi Motor DC

Vsuplai	DC 24V
Arus	14 A
Daya	250 W
Speed	2750 rpm
Dimensi motor	Diameter 10 cm x tebal 8 cm
Berat	2 kg

2.3.10.4 Sensor Ultrasonic

Sensor ultrasonik adalah sebuah sensor yang berfungsi untuk mengubah besaran fisis (bunyi) menjadi besaran listrik dan sebaliknya. Cara kerja sensor ini didasarkan pada prinsip dari pantulan suatu gelombang suara sehingga dapat dipakai untuk menafsirkan eksistensi (jarak) suatu benda dengan frekuensi tertentu. Disebut sebagai sensor ultrasonik karena sensor ini menggunakan gelombang ultrasonik (bunyi ultrasonik). Gelombang ultrasonik adalah gelombang bunyi yang mempunyai frekuensi sangat tinggi yaitu 20.000 Hz. Bunyi ultrasonik tidak dapat di dengar oleh telinga manusia. Bunyi ultrasonik bisa merambat melalui zat padat, cair dan gas. Reflektivitas bunyi ultrasonik di permukaan zat padat hampir sama dengan reflektivitas bunyi ultrasonik di permukaan zat cair.

Gambar 2.14 Sensor Ultrasonik

Spesifikasi sensor ultrasonik adalah sebagai berikut:

Tabel 2.4 Spesifikasi Sensor Ultrasonik

Tegangan	5V
Arus	15 mA
Frequensi	40KHz
Minimum jarak	2cm
Maksimum jarak	4m
Sudut pantul	15 derajat
Minimum Waktu penyulutan	10 mikrodetik
Dimensi	45 x 20 x 15 mm

2.3.10.5 IR Obstacle

IR *Obstrale Sensor* infrared merupakan sebuah modul yang berfungsi sebagai pendeteksi halangan atau object didepannya. Sensor inframerah ini menggunakan prinsip pantulan cahaya infrared sebagai penentu nilai nya. Ketika modul sensor mendeteksi sebuah halangan atau object di depan sensor maka akan diperoleh pantulan cahaya dengan intensitas yang diatur sesnitivitas nya dengan sebuah potensiometer.

Gambar 2.15 IR *Obstacle*

Spesifikasi sensor IR Obstacle adalah sebagai berikut:

Tabel 2.5 Spesifikasi IR *Obstacle*

Tegangan	3V-5V
Comparator	LM393
Jarak Deteksi	2cm - 30cm
Sudut Deteksi	35 Derajat
Ukuran board	3.1 cm x 1.5 cm

2.3.10.6 Modul Wifi NodeMCU ESP8266

Modul Wifi NodeMCU ESP8266 adalah sebuah modul *wifi* yang akhir-akhir ini semakin digemari para hardware developer. Modul *wifi* serbaguna ini sudah bersifat SoC (System on Chip), sehingga kita bisa melakukan programming langsung ke ESP8266 tanpa memerlukan mikrokontroller tambahan. Kelebihan lainnya, *ESP8266* ini dapat menjalankan peran sebagai adhoc akses poin maupun klien sekaligus. *Modul Wifi ESP8266* ini digunakan sebagai media komunikasi antara perangkat *mobile/smartphone* yang digunakan pengguna sebagai kontrol langsung terhadap sistem.

Gambar 2.16 NodeMCU ESP8266

Gambar 2.17 ESP8266

Spesifikasi NodeMCU ESP8266 adalah sebagai berikut:

Tabel 2.6 Spesifikasi NodeMCU ESP8266

Protocols	802.11 b/g/n/e/i
Frequency Range	2.4G ～ 2.5G (2400M ～ 2483.5M)
Antena	PCB Trace, External, IPEX Connector, Ceramic Chip
Operating Voltage	2.5V ~ 3.6V
Current	80mA
Security	WPA/WPA2
Encryption	WEP/TKIP/AES
Network Protocols	IPv4, TCP/UDP/HTTP/FTP
Peripheral Interface	UART/SDIO/SPI/I2C/I2S/IR GPIO/ADC/PWM/LED
User Configuration	AT Instruction Set, Cloud Server, *Android*/iOS App

2.3.10.7 Power Suplay

Power supplay adalah sebagai alat atau perangkat keras yang mampu menyuplai tenaga atau tegangan listrik secara langsung dari sumber tegangan listrik ke tegangan listrik yang lainnya. *Power supply* biasanya digunakan untuk komputer sebagai penghantar tegangan listrik secara langsung kepada komponen-komponen atau perangkat keras lainnya yang ada di komputer tersebut, seperti hardisk, kipas, motherboard dan lain sebagainya. *Power supply* memiliki *input* dari tegangan yang

berarus *alternating* current (AC) dan mengubahnya menjadi arus *direct current* (DC) lalu menyalurkannya ke berbagai perangkat keras yang ada dikomputer kita. Karena memang arus *direct current* (DC)-lah yang dibutuhkan untuk perangkat keras agar dapat beroperasi, *direct current* biasa disebut juga sebagai arus yang searah sedangkan *alternating current* merupakan arus yang berlawanan.

2.3.11 Perangkat Lunak

Berikut ini beberapa perangkat lunak pendukung dalam penyusunan penelitian Perancangan dan implementasi iot untuk kontrol alat pengubah sampah organik menjadi pupuk kering ini adalah

2.3.11.1 Arduino IDE

Arduino IDE adalah *software* yang digunakan untuk memprogram mikrokontroller hususnya Arduino yang ditulis dengan menggunakan Java. Bagian-bagian dari Arduino IDE terdiri dari:

1. Editor program, sebuah window yang memungkinkan pengguna menulis dan mengeditprogram dalam bahasa *Processing*.
2. *Compiler*, sebuah modul yang mengubah kode program (bahasa Processing) menjadi kode biner. Bagaimanapun sebuah mikrokontroler tidak akan bisa memahami bahasa*Processing*. Yang bisa dipahami oleh mikrokontroler adalah kode biner. Itulah sebabnya *compiler* diperlukan dalam hal ini.
3. *Uploader*, sebuah modul yang memuat kode biner dari komputer ke dalam memory didalam papan Arduino.

Sebuah kode program Arduino umumnya disebut dengan istilah *sketch.* Kata "*sketch*" digunakan secara bergantian dengan "kode program" dimana keduanya memiliki arti yang sama.

Gambar 2.18 Tampilan Pertama Arduino IDE build 1.8.10

Lampiran A **Bahasa Pemrograman Arduino**

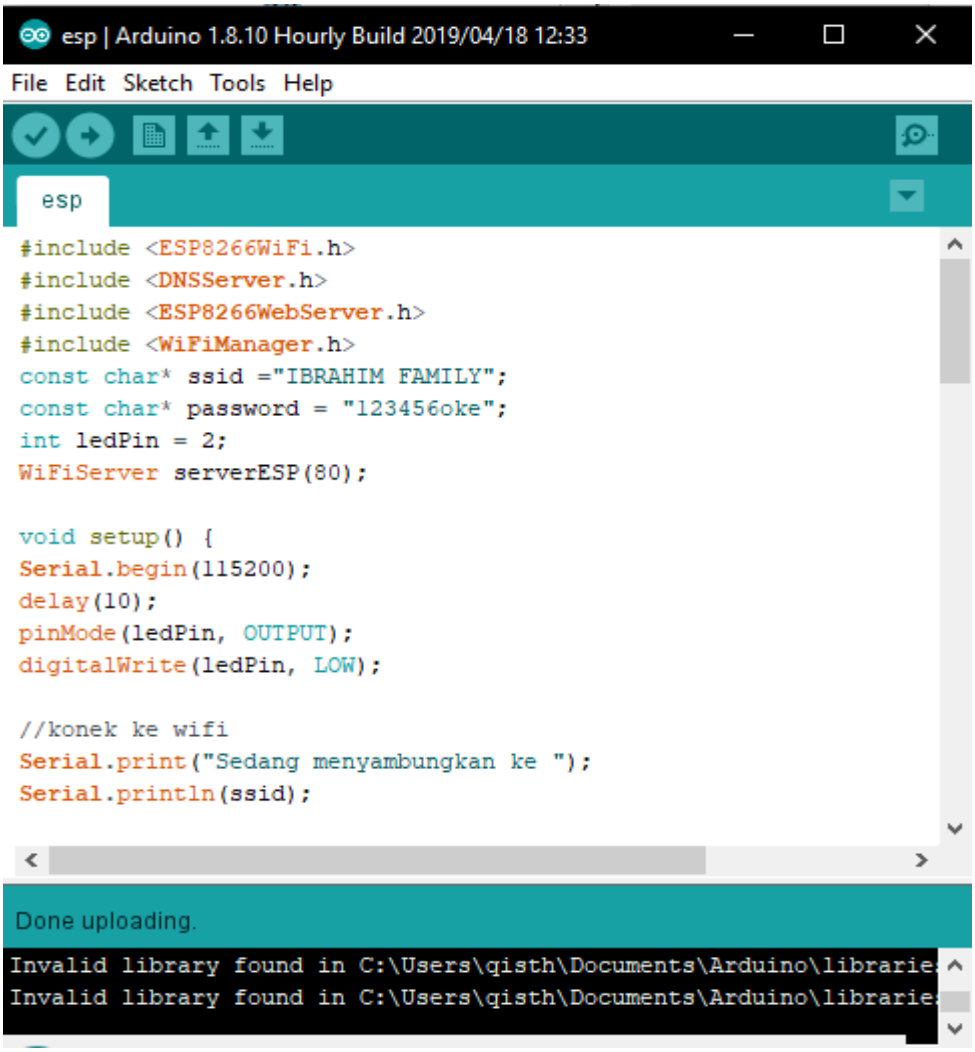
Seperti yang telah dijelaskan diatas program Arduino sendiri menggunakan bahasa C. walaupun banyak sekali terdapat bahasa pemrograman tingkat tinggi (*high level language*) seperti pascal, basic, cobol, dan lainnya. Walaupun demikian, sebagian besar dari paraprogramer profesional masih tetap memilih bahasa C sebagai bahasa yang lebih unggul, berikut alasan-alasannya:

1. Bahasa C merupakan bahasa yang *powerful* dan *fleksibel* yang telah terbukti dapat menyelesaikan program-program besar seperti pembuatan sistem operasi, pengolah gambar (seperti pembuatan game) dan juga pembuatan kompilator bahasa pemrograman baru.
2. Bahasa C merupakan bahasa yang *portabel* sehingga dapat dijalankan di beberapa sistem operasi yang berbeda. Sebagai contoh program yang kita tulis dalam sistem operasi windows dapat kita kompilasi didalam sistem operasi linux dengan sedikit ataupun tanpa perubahan sama sekali.
3. Bahasa C merupakan bahasa yang sangat populer dan banyak digunakan oleh programer berpengalaman sehingga kemungkinan besar *library* pemrograman telah banyak disediakan oelh pihak luar / lain dan dapat diperoleh dengan mudah.

4. Bahasa C merupakan bahasa yang bersifat modular, yaitu tersusun atas rutin-rutin tertentu yang dinamakan dengan fungsi *(function*) dan fungsi-fungsi tersebut dapat digunakan kembali untuk pembuatan program-program lainnya tanpa harus menulis ulang implementasinya.

5. Bahasa C merupakan bahasa tingkat menengah (*middle level language*) sehingga mudah untuk melakukan interface (pembuatan program antar muka) ke perangkat keras.

6. Struktur penulisan program dalam bahasa C harus memiliki fungsi utama, yang bernama main(). Fungsi inilah yang akan dipanggil pertama kali pada saat proses eksekusi program. Artinya apabila kita mempunyai fungsi lain selain fungsi utama, maka fungsi lain tersebut baru akan dipanggil pada saat digunakan.



Gambar 2.19 Tampilan Arduino IDE