Bermain dengan Internet of Things & BigData

Sitrusta Sukaridhoto, ST. Ph.D

dhoto@pens.ac.id

Politeknik Elektronika Negeri Surabaya

2016

Daftar Isi

Bab 1

Pengenalan IoT dan Big Data

Teknologi Internet of Things dan Big Data semakin berkembang dari waktu ke waktu. Semakin banyak aplikasi-aplikasi yang berhasil dibuat dengan penggabungan kedua teknologi tersebut, diantaranya *Smart Environment* dan *Robotic Application*.

Akan dijelaskan dalam tiga sub bab yakni sub bab pertama mengenai IoT, sub bab kedua menjelaskan mengenai Big Data dan sub bab ketiga menjelaskan mengenai contoh aplikasi yang dapat dibuat.

1.1 Internet of Things (IoT)

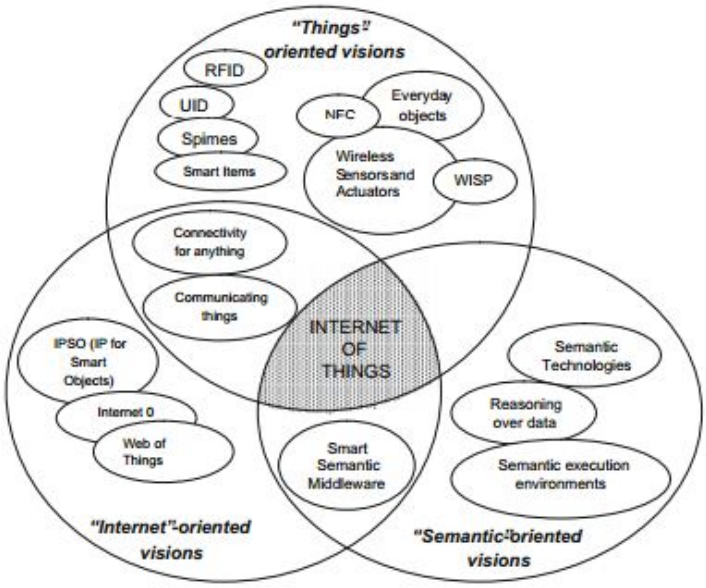
Internet of Things (IoT) menurut Rekomendasi ITU-T Y.2060 [1] didefinisikan sebagai sebuah penemuan yang mampu menyelesaikan permasalahan yang ada melalui penggabungan teknologi dan dampak sosial, sementara itu jika ditinjau dari standarisasi secara teknik, IoT dapat digambarkan sebagai infrastruktur global untuk memenuhi kebutuhan informasi masyarakat, memungkinkan layanan canggih dengan interkoneksi baik secara fisik dan virtual berdasarkan pada yang telah ada dan perkembangan informasi serta teknologi komunikasi(ICT).

1.1.1 Definisi dari IoT

Untuk memahami definisi dari Internet of Things dapat dilihat dari gabungan dari 2 kata yakni "Internet" dan "Things". Dimana "Internet" sendiri didefinisikan sebagai sebuah jaringan komputer yang menggunakan protokol-

protokol internet (TCP/IP) yang digunakan untuk berkomunikasi dan berbagi informasi dalam lingkup tertentu. Sementara "Things" dapat diartikan sebagai objek-objek dari dunia fisik yang diambil melalui sensor-sensor yang kemudian dikirim melalui Internet. Namun, dari hasil objek yang telah dikirimkan masih memerlukan penyajian ulang yang diharapkan dapat lebih mudah dimengerti oleh *stack holder*. Untuk mempermudah model penyimpanan dan pertukaran informasi diperlukan adanya Teknologi Semantic. Oleh karena itu untuk mewujudkan Internet of Things diperlukan 3 komponen pendukung yakni Internet, Things dan Semantic [2].

Gambar 1.1 menggambarkan mengenai konsep utama, teknologi dan standarisasi dari paradigma Internet of Things.

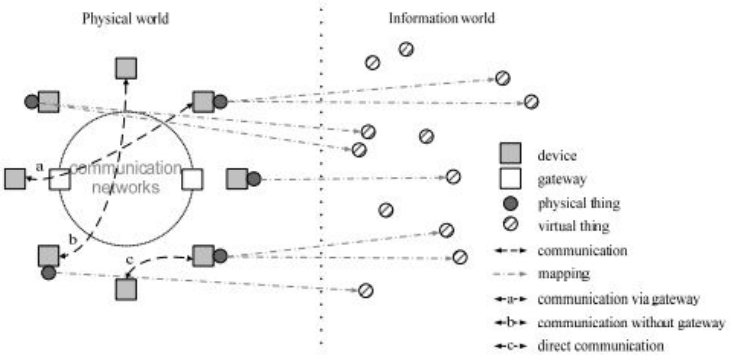


Gambar 1.1: Paradigma dari "Internet of Things"

1.1.2 Sudut Pandang Teknik IoT

Gambar 1.2 menggambarkan mengenai sudut pandang teknik dari Internet of Things.

Objek fisik *(physical things)* dapat direpresentasikan dalam dunia informasi *(information world)* melalui satu atau lebih objek virtual *(virtual*



Gambar 1.2: Sudut Pandang Teknik dari "Internet of Things"

things). Tetapi objek virtual dapat berdiri sendiri tanpa adanya pemetaan *(mapping)* dengan objek fisik.

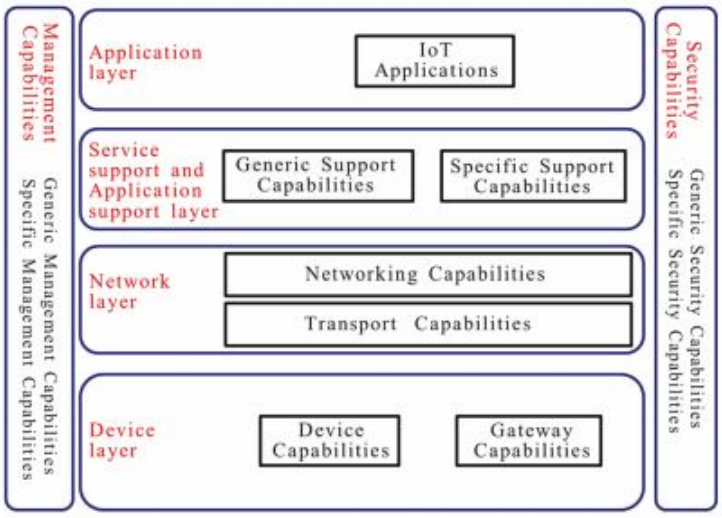
Sebuah perangkat *(device)* adalah sebuah peralatan yang memiliki kemampuan untuk berkomunikasi dan beberapa kemampuan tambahan *(sensing, actuation, data capture, data storage and data processing).* Dimana sebuah perangkat nantinya akan mengambil informasi-informasi yang dibutuhkan dan menyajikannya sebagai sebuah informasi serta mengirimkannya untuk pengolahan selanjutnya. namun demikian ada juga perangkat yang langsung dapat mengolahnya berdasarkan informasi dan komunikasi yang diterima.

Komunikasi antar perangkat *(communication between devices)*: perangkat mampu berkomunikasi melalui sebuah jaringan komunikasi melewati sebuah *gateway* (kasus A), berkomunikasi tanpa melewati *gateway* (kasus B) atau secara langsung *(direct)* atau *ad-hoc* (kasus C) atau komninasi antar keduanya.

1.1.3 Model Referensi dari IoT

Gambar 1.3 menggambarkan mengenai model referensi dari IoT yang terdiri dari 4 layer dan juga kemampuan management serta kemampuan keamanan *(security).* Keempat layer tersebut adalah

- Layer Aplikasi *(Application Layer)*
- Layer Layanan dan Pendukung Aplikasi *(Service Support and Application Support Layer)*



Gambar 1.3: Model Referensi dari "Internet of Things"

- Layer Jaringan *(Network Layer)*
- Layer Perangkat *(Device Layer)*

1.1.3.1 Layer Aplikasi

Pada layer aplikasi, seluruh aplikasi dari IoT ada dalam layer aplikasi.

1.1.3.2 Layer Layanan dan Pendukung Aplikasi

Pada layer layanan dan pendukung aplikasi, dikelompokkan menjadi dua kemampuan diantaranya :

- Kemampuan Dukungan Generic : Kemampuan dukungan generik merupakan kemampuan umum yang dapat digunakan oleh aplikasi IOT yang berbeda, seperti pengolahan data atau penyimpanan data. Kemampuan ini dapat juga dipanggil oleh kemampuan dukungan khusus, misalnya untuk membangun kemampuan dukungan khusus lainnya.
- Kemampuan Dukungan Khusus : Kemampuan dukungan khusus merupakan kemampuan tertentu yang memenuhi persyaratan aplikasi diversifikasi. Bahkan kemampuan ini dapat terdiri dari berbagai kelompok

kemampuan yang lebih khusus dalam rangka memberikan fungsi pendukung yang berbeda untukaplikasi IOT yang berbeda.

1.1.3.3 Layer Jaringan

Pada layer jaringan, dapat dibagi menjadi dua bagian yakni sebagai berikut:

- Kemampuan Membangun Jaringan : Dimana kemampuan ini menyediakan fungsi kontrol yang relevan dari konektivitas jaringan , seperti akses dan kontrol kemampuan transportasi, manajemen mobilitas atau otentikasi, otorisasi dan akuntansi (AAA).
- Kemampuan Transportasi : Fokus pada penyediaan konektivitas untuk pengangkutan layanan IOT dan aplikasi informasi data khusus, serta transportasi kontrol IOT - kontrol terkait dan manajemen informasi.

1.1.3.4 Layer Perangkat

Pada layer perangkat, dapat diklasifikasikan menjadi dua kelompok diantaranya:

- **Kemampuan Perangkat** : Kemampuan perangkat termasuk namun tidak terbatas pada :
  - Interaksi langsung dengan jaringan komunikasi : Perangkat yang mampu mengumpulkan dan mengunggah informasi langsung (tanpa menggunakan kemampuan gateway)untuk komunikasi jaringan dan dapat langsung menerima informasi (misalnya: perintah) dari komunikasi jaringan.
  - Interaksi tidak langsung dengan jaringan komunikasi : Perangkat yang tidak mampu secara langsung mengumpulkan dan mengunggah informasi ke jaringan komunikasi, yaitu dapat melalui kemampuan gateway. Disisi lain, perangkat tidak langsung dapat menerima informasi (misalnya: perintah) dari jaringan komunikasi.
  - Jaringan Ad-Hoc: Perangkat dapat membangun jaringan secara ad- hoc dibeberapa skenario yang memerlukan peningkatan skalabilitas dan penyebaran dengan cepat.
  - Tidur dan Bangun: Kemampuan Perangkat yang dapat mendukung kondisi ”tidur” dan ”bangun” sebagai mekanisme untuk menghemat energi.

- **Kemampuan Gateway** : Kemampuan gateway termasuk namun tidak terbatas pada:

  - Dukungan beberapa Interface : Pada lapisan perangkat, kemampuan gerbang *(gateway)* mendukung perangkat untuk terhubung melalui berbagai jenis teknologi kabel atau nirkabel, seperti kontroler area network *(Controller Area Network)*(CAN), ZigBee, Bluetooth atau Wi-Fi. Pada layer jaringan, gateway memiliki kemampuan untuk dapat berkomunikasi dengan berbagai teknologi, seperti PSTN *(Public Switched Telephone Network)*, generasi kedua atau generasi ketiga (2G atau 3G), jaringan LTE *(Long-Term Evolution)*, Ethernet atau DSL *(Digital Subscriber Lines)*.
  - Konversi Protokol: Ada dua situasi dimana kemampuan *gateway* diperlukan. Situasi pertama adalah ketika komunikasi pada layer perangkat menggunakan berbagai protokol layer perangkat, misalnya, protokol teknologi ZigBee dan protokol teknologi Bluetooth. Situasi kedua adalah ketika komunikasi melibatkan layer perangkat dan layer jaringan dengan protokol yang berbeda. Seperti misalnya sebuah protokol teknologi ZigBee pada lapisan perangkat dan teknologi 3G protokol pada lapisan jaringan.

1.1.3.5 Kemampuan Management

Seperti halnya jaringan komunikasi sebelumnya, kemampuan management IoT juga meliputi kesalahan tradisional *(traditional fault)*, konfigurasi, accounting, performansi dan keamanan *(performance and security (FCAPS) classes)*, contohnya seperti kesalahan manajemen, konfigurasi manajemen, akuntansi manajemen, performansi manajemen dan keamanan manajemen.

Kemampuan manajemen IOT dapat dikategorikan ke dalam kemampuan manajemen generik dan kemampuan manajemen tertentu .Kemampuan manajemen generik esensial dalam IOT meliputi :

- Manajemen Perangkat : seperti diantaranya aktivasi perangkat remot dan de-aktivasi, diagnostik, firmware atau memperbarui perangkat lunak, manajemen status dari perangkat kerja ;

- Manajemen Topologi Jaringan Lokal

- Manajemen *Traffic* dan *Congestion* : seperti mengatasi jaringan yang sedang *overload*. Serta kemampuan manajemen tertentu yang digabungkan erat dengan kebutuhan aplikasi spesifik , seperti persyaratan pemantauan daya saluran transmisi dari *smart grid*.

1.1.3.6 Kemampuan Keamanan

Ada dua jenis kemampuan keamanan yakni kemampuan keamanan generik dan kemampuan keamanan tertentu. Dimana kemampuan keamanan generik dari aplikasi termasuk kedalam pejelasan berikut:

- Pada Layer Aplikasi: otorisasi, otentikasi, aplikasi kerahasiaan data dan perlindungan integritas, perlindungan privasi, audit keamanan dan anti-virus.
- Pada Layer Jaringan: otorisasi, otentikasi, penggunaan data dan data yang menandakan kerahasiaan, serta sinyal perlindungan integritas *(signalling integrity protection)*.
- Pada Layer Perangkat: otentikasi, otorisasi, perangkat validasi integritas, kontrol akses, kerahasiaan data dan perlindungan integritas.

Kemampuan keamanan juga dapat digabungkan dengan kebutuhan aplikasi-aplikasi tertentu, seperti *mobile payment* dan persyaratan keamanan.

1.2 Big Data

Dalam era digital saat ini, terjadi adanya ledakan data akibat penggunaan beragam aplikasi dalam akses data. Google memproses lebih dari 24 petabyte data per hari. Byte data tersebut dihasilkan dari berbagai proses akses data baik data teks, gambar, suara maupun streaming. Sehingga diperlukan sebuah revolusi dan transformasi baru untuk menanggulangi kebutuhan akses data tersebut. Big data merupakan sebuah revolusi dan transformasi baru untuk meningkatan kecepatan akses data, skala penggunaan data serta kapasitas penyimpanan data.

1.2.1 Definisi dari Big Data

Big data adalah sekumpulan beberapa set data besar dan complex yang akan menjadi susah untuk diproses sehingga membutuhkan database serta

perangkat tertentu untuk memprosesnya. (wikipedia, 2014). Sekumpulan data tersebut akan dianalisa menjadi sekumpulan data yang kecil akan tetapi nampak menjadi data yang besar. Big Data telah digunakan untuk menyampaikan segala macam konsep termasuk jumlah data yang sangat besar, analisis media sosial, penerapan next generation dalam hal manajemen data, data real-time, dan lain-lain. Big data memiliki dua tipe data yaitu data struktural dan data Unstruktural.

- Data Struktural adalah sejumlah data yang dapat dengan mudah untuk dikategorikan dan dianalisis. Data-data ini biasanya dihasilkan oleh perangkat jaringan sensor yang tertanam pada peragkat elektronik, smartphone dan GPS. Data struktural juga mencakup hal-hal seperti angka penjualan, saldo rekening dn data transaksi.

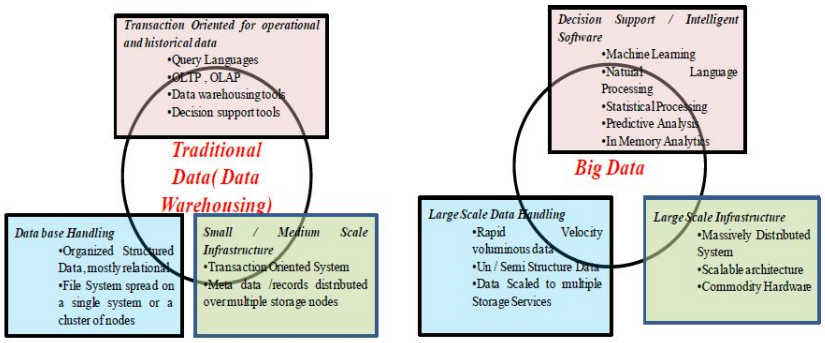
- Data Unstruktural biasanya data informasi yang bersifat lebih kompleks seperti halnya ulasan pelanggan pada situs komersial, foto, dan multimedia lainnya serta jejaring sosial. Data ini tidak dapat dengan mudah untuk dipisahkan kedalam kategori atau dianalisis secara numerik.

1.2.2 Data Vs Tradisional Data

Big Data mengacu pada arsitektur sebuah data yang berskala untuk memfasilitasi suatu perangkat dalam hal penanganan volume data, kecepatan dalam mengakses data serta variabilitas data. Sedangkan tradisional data mengasumsikan sebuah data dengan volume yang lebih kecil, proses pembaharuan data yang konsisten berdasarkan struktur data yang ada serta beroperasi pada server tunggal. Namun pada Big Data beragam format (data geospasial, data 3D, teks, audio, video, data struktural, data unstruktural) akan diolah menjadi satu kesatuan sistem tanpa berpedoman dalam suatu struktur yang ada. Berikut ini perbedaan antara Big Data dan Traditional Data dari segi ukuran data, infrastruktur serta pengolahan data disajikan pada Gambar 1.4.

Pada pengelompokkan abstraksi layer yang digunakan pada traditional data dan big data memiiki beberapa perbedaan. Pada 1.5 disajikan perbedaan abstraksi layer antara big data dan traditional data.

Secara umum abstraksi layer traditional data diklasifikasikan menjadi tiga layer yaitu:



Gambar 1.4: Big Data Vs Traditional Data

- Physical Layer
  Merupakan level terbawah pada abstraksi layer dimana menggunakan tingkat kompleksitas struktur data yang kecil untuk penyimpanan datanya.

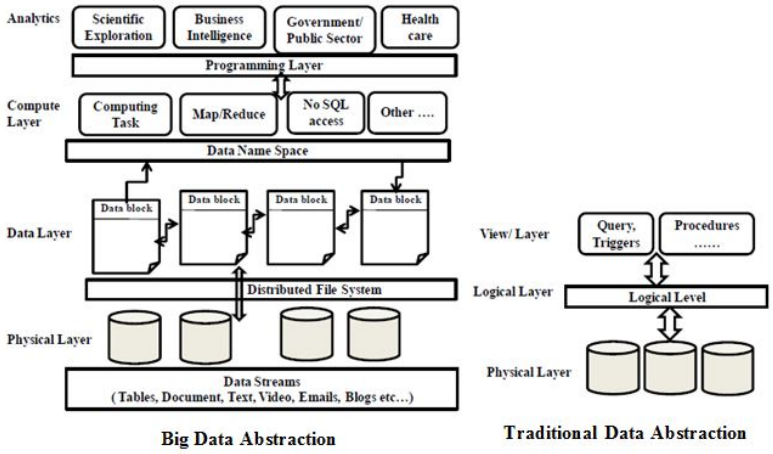
- Logical Layer
  Pada layer ini menjelaskan tipe data yang tersimpan pada database dimana beberapa aktifitas yang dilakukan pada layer ini adalah proses desain data base serta tools yang digunakan untuk proses back up data base.

- View Layer
  Merupakan layer tertinggi dimana di dalamnya terdapat proses pengolahan data yang digunakan.

Berbeda halnya pada Big Data mengadopsi empat model abstraksi layer di dalamnya diantaranya:

- Physical layer
  Pada layer ini menjelaskan struktur data yang digunakan pada big data. Pada Big data mengalamatkan beberapa bentuk dan tipe data dengan performansi data transfer yang cepat dan efisien.

- Data Layer
  Pada layer ini menjelaskan proses organisasi data, akses serta pengambilan data. Pada proses organisasi data ini akan disertai dengan proses pemberian indeks pada suatu data yang nantinya akan didistribusikan pada suatu perangkat.



Gambar 1.5: Abstraksi Layer Big Data Vs Traditional Data

- Computing Layer
  Pada layer ini terdiri penggunaan software dalam hal proses pemodelan data, pemrograman serta pemrosesan data.

- Analytics layer
  Pada layer ini terdapat sebuah standarisasi dan teknik yang digunakan untuk mengembangkan software yang digunakan untuk mendapatkan hasil yang lebih spesifik.

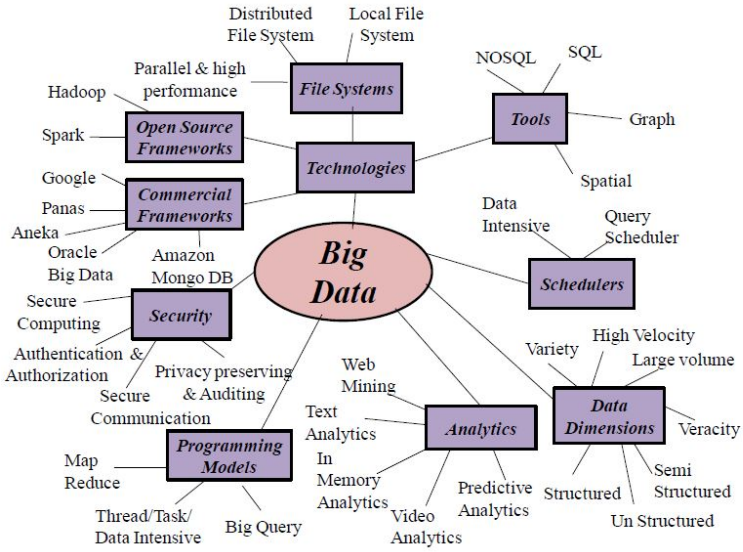
Sehingga apabila antara big data dan traditional data dibandingkan big data memiliki tingkat komplektisitas yang lebih tinggi guna mendapatkan hasil yang lebih spesifik, akses data cepat dan efisien.

1.2.3 Sistem Taksonomi Big Data

Untuk menyusun sebuah Big Data maka diperlukan beberapa komponen penyusun yang saling berhubungan sehingga dapat terbentuk sebuah sistem Big Data. Komponen penyusun Big data dikelompokkan menjadi sebuah Sistem taksonomi yang digambarkan pada 1.6.

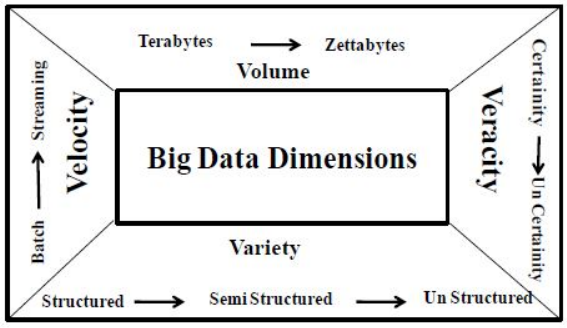
1.2.3.1 Dimensi Data

Karakteristik Big Data dikelompokkan kedalam 4 dimensi yang biasa disebut dengan 4V: Volume, Velocity, Variety, dan Veracity yang ditampilkan pada



Gambar 1.6: Sistem taksonomi pada Big Data

1.7.



Gambar 1.7: Dimensi Big Data

- Volume, Berhubungan dengan skala ukuran data yang digunakan. Volume data berkembang pesat karena adanya beberapa aplikasi bisnis, sosial, web dan eksplorasi ilmiah.
- Velocity, Berhubungan dengan kecepatan akses data yang berkaitan dengan kebutuhan data streaming yang bersifat real time.

- Variety, Berhubungan dengan beberapa bentuk data yang digunakan dalam proses analisis data.

- Veracity, Berhubungan dengan ketidakpastian dan keakuratan suatu data. Pada beberapa kondisi tingkat akurasi akan didapatkan apabila dilakukan adanya proses filtering dan selecting data.

1.2.3.2 Teknik Analisis Big Data

Analisis adalah proses menganalisis data menggunakan model statistik, data mining serta beberapa teknologi komputasi data. Big Data analisis mengacu pada serangkaian prosedur dan model statistik untuk mengestrak informasi dari beberapa set data yang berukuran besar. Berikut ini beberapa aplikasi yang digunakan untuk analisis pada Big Data:

- Teks Analisis
  Proses ini berasal darisumber informasi berupa teks. Teknologi dalam analisis teks berasal dari beberapa bidang mendasar termasuk linguistik, statistik dan machine learning. Secara umum, analisis teks yang modern menggunakan model statistik, ditambah dengan teori linguistik, untuk menangkap pola dalam bahasa manusia sehingga mesin dapat memahami arti dari teksdan melakukan berbagai tugas analisis teks.

- Memori Analisis
  Dalam analisis memori adalah proses yang mencerna data dalam jumlah besar dari berbagai sumber yang langsung menuju ke memori sistem untuk mendapatkan perhitungan kinerja yang efisien. Pada memori analisis ini terjadi sebuah pendekatan untuk pengalamatan data saat berada pada RAM guna mendapatkan respon waktu yang cepat dalam proses pengalamatan

- Prediktif Analisis
  Suatu proses yang digunakan untuk memprediksi kejadian yang akan datang dengan beberapa bantuan seperti data statistik, pemodelan, machine learning dan data mining dengan menganalisa kejadian yang pernah ada.

- Grafik Analisis
  Grafik analisis ini digunakan untuk mempelajari perilaku segala sesuatu yang terhubung pada sistem. Perilaku yang diamati guna untuk mencari komponen yang bersifat lemah atau kuat pada suatu sistem.

1.2.3.3 Teknologi Big Data

Teknologi yang digunakan pada Big Data secara umum diklasifikasikan menjadi 3 bagian yaitu file sistem cara efektif untuk mengatur data, Framework dalam proses komputasi, serta tools atau perangkat yang digunakan untuk analisis, berikut penjelasan dari masing-masing komponen:

1. File Sistem
   Pada file sistem ini bertanggung jawab dalam hal organisasi, penyimpanan, penamaan, sharing dan perlindungan file. Manajemen file pada Big Data mirip dengan sistem file terditribusi, akan tetapi dalam hal membaca kinerja, serta akses datanya dilakukan secara simutan. Pada penciptaan file sistem Big Data, proses sinkronisasi menjadi tantangan utama dalam hal desain dan implementasi. File sistem pada big data harus mengandung beberapa syarat berikut:

   - Mode akses terdistribusi dan transparansi lokasi
     File didistribusikan pada suatu lokasi dimana semua klien dapat mengakses dengan cara yang sama dimanapun lokasinya.
   - Kegagalan penanganan
     Program aplikasi dan klien tetap harus beroperasi walaupun terdapat beberapa kegagalan komponen dalam sistem. Hal ini dapat ditanggulangi dengan beberapa tingkat replikasi dan redudansi pada sistem.
   - Heterogenitas
     Layanan Berkas harus disediakan di seluruh hardware dan sistem operasi yang memiliki platform berbeda.
   - Toleransi untuk partisi jaringan
     Kondisi pemberian toleransi pada saat terjadinya operasi yang terputus antara klien terhadap jaringan yang ada. File sistem harus cukup toleran untuk menangani situasi inidan menerapkan mekanisme sinkronisasi yang tepat.

2. Framework Komputasi
   Frameworks yang digunakan dalam proses komputasi pada Big data yang bersifat open source diantaranya:

   - **Apache Hadoop**
     Framework open source yang handal, terukur serta bersifat terdistribusi. Framework ini memungkinkan untuk didistribusikan beberapa pengolahan beberapa set data berukuran besar pada beberapa komputer meggunakan model pemrograman sederhana.
   - **Spark**
     Apache Spark adalah mesin yang cepat untuk pengolahan data skala besar. Hal ini mencakup Shark SQL, Spark Streaming, MLIB machine learning dan Graphx alat analisis grafik. Spark dapat berjalan pada Hadoop YARN cluster manager, dan dapat membaca data Hadoop yang ada.

   Selain itu terdapat beberapa framework komputasi big data yang besifat komersial diantaranya: Google menawarkan Big Query untuk beroperasi di Google Big Tables. Amazon mendukung Big Data melalui Hadoop cluster dan juga NoSQL melalui columnar database menggunakan Amazon DynamoDB. Amazon Elastic MapReduce (ESDM) adalah kerangka Hadoop yang berhasil untuk memproses data dalam jumlah besar dengan cepat, mudah dan hemat biaya di seluruh intansi EC2 Amazon. Windows menawarkan HDInsight layanan implementasi dari Hadoop yang berjalan pada platform Microsoft Azure. RackSpace menawarkan Horton Hadoop pada platform OpenStack, Aneka menawarkan NET berbasis platform MapReduce desktop, dan framework pada perusahaan lainnya berbasis open source Hadoop adalah Horton dan Cloudera.

3. Tools
   Perangkat yang digunakan pada beberapa sistem yang ada ada Big Data adalah sebagai berikut:

   - **Key-Value Store**
     Key-value pair (KVP) table digunakan untuk menyediakan sistem manajemen data pada beberapa teknologi NoSQL. Konsep KVP tabel ini memiliki 2 kolom dengan 1 kunci dan yang lain berfungsi sebagai valuenya. Data value dapat berisikan data single atau

data block yang berisikan banyak value didalamnya dengan format yang mengandung beberapa kode program didalamnya. KVP tabel menggunakan sistem pengindeksan yang disertai pencarian cepat suatu data serta penyisipan maupun penyimpanan data secara cepat. Contoh dari Key-Value Store adalah Amazons Dynamo dan Oracles Berkeley DB.

- **Document Oriented Database**
  Merupakan suatu database yang berorientasi pada suatu dokumen dan berfungsi untuk menyimpan, mengambil dan mengelola data yang bersifat semi-struktural data. Konsep utama dari database ini adalah mengkodekan dokumen data kedalam beberapa format standar seperti JavaScript Object Notation (JSON), Binary JavaScript Object Notation (JSON) atau XML. Contoh database ini adalah Apache CouchDB dan 10 gen MongoDB .

- **Big Table Database**
  Pada sistem database ini berkebalikan dengan sistem database pada Key-value store. Big table database ini membuat beberapa data secara berkelompok, dimana setiap datanya terdiri dari data sebagai kunci dan data sebagai value pada setiap barisnya. Jenis penyimpanan ini berguna untuk streaming data pada web log, data time series yang berasal dari beberapa perangkat sensor, dll. Contoh penggunaan data base ini adalah Hbase dan Google Big Table.

- **Graph Database**
  Merupakan sebuah database yang menggunakan struktur data berbasis grafik yang berfungsi untuk penyimpanan data semantik. Database ini setiap entitas data mengandung pointer langsung menuju data yang dituju tanpa adanya proses pencarian. Graph database ini biasanya digunakan untuk memproses beberapa data complex yang saling berhubungan dari berbagai sumber seperti jejaring sosial. Data ini biasanya didapatkan dari data table yang terhubung langsung dengan Big Table dan Cassandra. Contoh dari penggunaan grafik database meliputi data grafik yang tak terbatas adalah Neo4j yang bersifat open source database.

1.2.3.4 Model Pemrograman Big Data

Berbagai macam model pemrograman yang digunakan pada Big Data meliputi penggunaan data intensif, komputasi data streaming, pemrosesan banyak data, tingkat performansi yang tinggi (throughput), dan pemrosesan data yang berorientasi pada kolom data. Berikut ini beberapa proses pemrograman pada Big data

- **MapReduce**
  Merupakan sebuah model pemrograman tingkat tinggi yang membangun sebuah peta dan mengurangi fungsi pengelompokan data yang didistribusikan dari beberapa node yang ada. Peta yang dibangun berfungsi untuk memfillter dan mengurutkan data sedangkan fungsi Reduce adalah menggabungkan beberapa hasi outputan peta untuk menjadi suatu hasil akhir. Beberapa contoh aplikasi yang menggunakan model pemrograman MapReduce adalah Hadoop MapReduce, Apache Spark, Aneka MapReduce.

- **Thread/Task Data Intensive Mode**
  Model pemrograman yang digunakan pada aplikasi tingkat tinggi dengan sebuah logika komputasi yang didasarkan pada batas waktu tenggat sebuah aplikasi. Model pemrograman ini biasanya digunakan untuk pemrograman suatu alur kerja, misalnya Aneka.

- **Machine Learning Tools**
  Merupakan sebuah pembelajaran pada suatu mesin yang digunakan untuk pengambian sebuah keputusan. Contoh beberapa perangkat yang menggunakan pemrograman ini adalah Hadoop Mahout.

- **Big Query Language**
  Merupakan sebuah generasi baru dalam hal pengenalan suatu bahasa. Model pemrograman ini biasanya melibatkan pencarian data teks. Salah satu contoh aplikasi dari pemrograman ini adalah pencarian suatu kata berdasarkan frekuensi kemunculan menggunakan Google Big Query Data Analisis.

Selain model pemrograman, sistem komputasi pada Big Data juga membutuhkan mekanisme penjadwalan. Pada sistem komputasi big data terdapat dua jenis penjadwaan yang digunakan yaitu Query Schedulling dan Intensif

Data Scheduling. Query Scheduling merupakan sebuah penjadwaan yang didasarkan dengan adanya permintaan dari beberapa data baru yang masuk. Sedangkan intensif data merupakan penjadwalan yang didasarkan pada besaran data yang ada pada database guna mendapatkan waktu yg lebih efektif dan efisien

1.2.3.5 Sistem Security pada Big Data

Sistem Database pada big data dapat mengungkap data-data yang ada pada suatu instansi perusahaan dengan cara memantau beberapa peristia yang ada secara real time. Data-data tersebut perlu adanya suatu perllindungan untuk memastikan bahwa hanya orang tertentu yang memiliki hak akses terhadap data tersebut. Sistem keamanan pada big data dirancang dengan beberapa mekanisme sistem keamanan untuk melindungi data berskala besar. Berikut ini beberapa elemen yang dibutuhkan untuk mekanisme sistem keamanan pada big data:

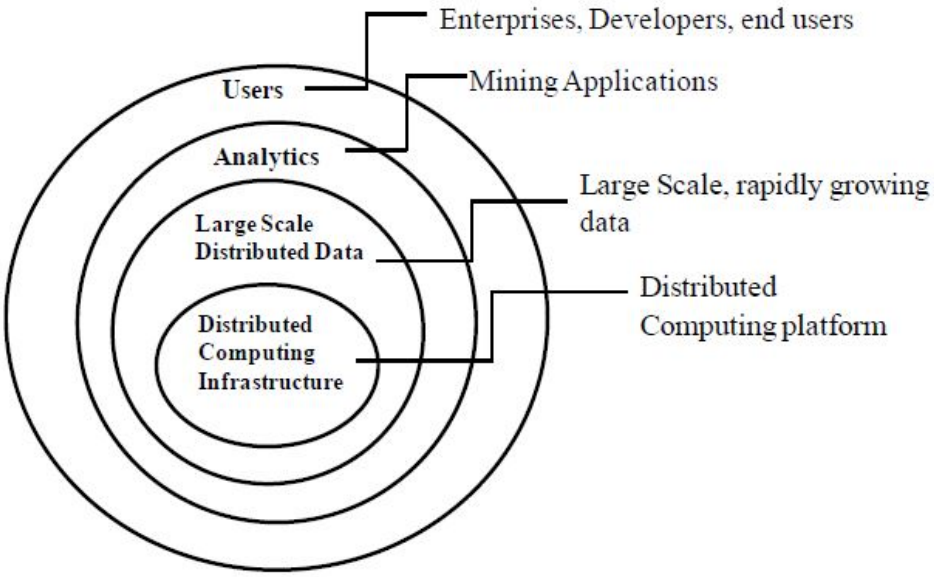
- Infrastruksur sistem Komputasi yang terdistribusi
  Merupakan sebuah mekanisme yang menyediakan sistem keamanan ketika data dianalisis pada sistem yang terdistribusi.

- Skala Besar Pendistribusian Data
  Merupakan mekanisme sistem keamanan yang menenrapkan adanya penggunaan teknik enkripsi data yang telah tersimpan pada database big data.

- Security Analisis
  Adalah mekanisme yang digunakan untuk mengembangkan framework yang telah diberi sistem security. Salah satu bentuk pengembangannya adalah adanya proses autentikasi mekanisme dalam mempublikasikan data. Beberapa jenis mekanisme autentikasi yang digunakan seperti one time passwords (OTP), Multilevel autentikasi dan autentikasi berdasarkan peranan akses data.

- Privasi dan Security User
  Merupakan mekanisme keamanan berupa privasi, integritas dan autentikasi secara keseluruhan dalam hal validasi user.

Berikut ini 1.8 menunjukkan elemen mekanisme sistem keamanan pada Big Data.



Gambar 1.8: Elemen Mekanisme Sistem Security pada Big Data

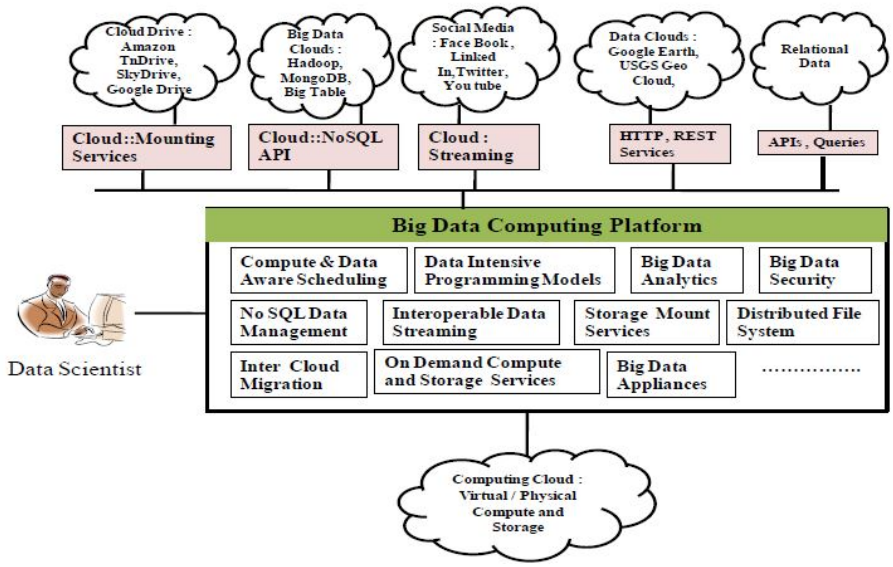
1.2.4 Big Data pada Sistem Cloud

Big Data pada sistem cloud merupakan sebuah platform generasi baru untuk membangun sistem analisis baru pada sebuah data melalui sebuah infrastruktur dengan skala yang bersifat elastis atau secara otomatis akan mengikuti ukuran data yang ada. Big data pada sistem Cloud memiliki beberapa keunggulan fitur diantaranya:

- Skala besar dapat didisitribusikan dengan cepat, dan memiliki akses penyimpanan serta fasilitas komputasi tanpa batas
- Informasi data didefinisikan berdasarkan sistem akses datanya bukan berdasarkan jalur atau nama filenya.
- File sistem dapat secara dinamis membuat dan memetakan data yang ada
- Sistem akses data berskala besar bersifat transparansi terhadap peran akses yang dimiliki
- Sistem kinerja yang tinggi memungkinkan untuk proses komputasi lebih cepat

- Dapat memproses segala bentuk dan dimensi data yang ada
- Mampu mengembangkan dan menyebarkan penggunaan sistem analisis terhadap lingkungan yang berbeda
- Mekanisme replikasi terhadap data dan sistem komputasi yang digunakan
- Penggunaan mode intensif data pada segala platform yang digunakan.

Sehingga Big data pada sistem cloud merupakan hasil integrasi antara proses komputasi Big Data menggunakan berbagai platform dengan sistem cloud yang berfungsi sebagai database sistem yang digunakan.



Gambar 1.9: Hasil Integrasi Sistem Cloud dengan Proses Komputasi pada Big Data

1.2.4.1 Arsitektur layer Big Data Cloud

Pada Big Data memiliki tiga layer utama yaitu layer infrastruktur, layer Big data Platform serta layer aplikasi. Pada masing-masing layer tersebut akan dibagi lagi menjadi beberapa sub layer yang saling berhubungan. Berikut ini penjelasan masing-masing layer yang dimiliki oleh big data cloud:

1. Layer Infrastruktur
   Pada Layer ini menyediakan layanan untuk manajemen data, pengiriman data, sistem komputasi, penyimpanan dan infrastruktur jaringan. Pada layer ini diklasifikasikan menjadi dua sub layer diantaranya layer sumber jaringan (resource layer) dan layer perangkat (interface layer).

   - Resource Layer
     Pada resource layer ini mengatur segala sesuatu sumber yang berasal dari local data yang dapat diakses melalui protokol standard dan suatu perangkat jaringan, resource ini merupakan resource yang bersifat physical. Selain itu resource layer juga dapat mengatur resource yang bersifat virtual atau cloud resource. Cloud resource ini merupakan resource yang dikirimkan oleh beberapa cloud provider dari sisi komputasi, penyimpanan serta aplikasi cloud yang disediakan.
   - Interface Layer
     Pada Interface layer ini memfasilitasi penggunaan standard protokol yang bersifat terbuka, sehingga protokol tersebut dapat mengakses berdasarkan segala jenis layanan yang dimilikinya. Standard yang dimiliki oleh interface layer ini dapat digunakan untuk mengakses sumber komputasi, sumber penyimpanan serta sumber aplikasi yang digunakan. Pada interface layer ini dapat diklasifikasikan lagi menjadi empat komponen yaitu, protokol perangkat jaringan, CCMI (Cloud Compute Management Interface), CS/DMI (Cloud Storage/Data Management Interface) dan CASI (Cloud Application Services Interface). Pada masing-masing komponen tersebut berkerja dalam hal jaringan, managemen data, penyimpanan data dan pengembangan aplikasi.

2. Layer Big Data Platform
   Layer ini merupakan layer tengah yang dapat diklasifikasika lagi menjadi beberapa sub layer didalamnya berdasarkan fungsionalitasnya. Sub layer tersebut diantaranya Foundation Layer, Runtime Layer, Model Pemrograman serta SDK (Software Development Kit) Layer.

   - Foundation Layer
     Pada layer ini menawarkan mekanisme untuk memanajemen resourse yang ada, penyimpanan data yang digunakan, proses mana-

jemen data, sistem security yang digunakan serta penerapan penggunaan sistem virtual/cloud.

- Runtime Layer
  Pada Layer ini memfokuskan pada mekanisme penjadwalan serta mekanisme manajemen perkerjaan yang dilakukan oleh masing-masing elemen yang ada. Beberapa contoh aplikasi dari layer ini adalah Thread, Task, Map Reduce, Data Aware Scheduler.

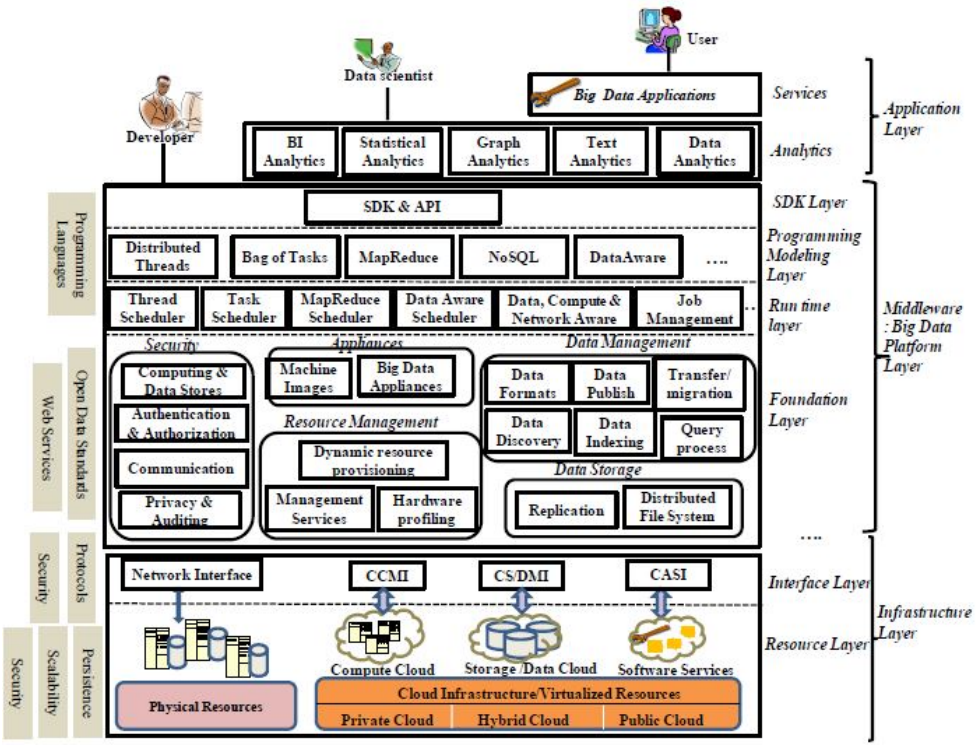
3. Layer Aplikasi Pada layer aplikasi ini big data cloud menyediakan beberapa metode statistik, probabilitas, dan teknik machine learning untuk mengembangkan perangkat yang digunakan untuk proses analisa. Layer ini menawarkan SDK, API dan perangkat untuk pengembangan analisa, kemampuan memanajemen data, serta memungkinkan untuk memonitor lingkungan dari Big data.

Kemudian masing-masing elemen arsitektur layer dari big data cloud akan dijalankan oleh beberapa user yang terikat didalamnya. Seperti halnya Developers yang berfungsi untuk desain aplikasi. Data Scientist yang berfungsi untuk membuat aplikasi yang bersifat analitik. Sedangkan End User sendiri merupakan pengguna yang berfungsi untuk menganalisa kesuluruhan dari sistem yang ada. Sehingga keseluruhan sistem yang ada pada arsitektur layer big data cloud disajikan pada 1.10

1.3 Contoh Aplikasi IoT dan Big Data

Saat ini terdapat beberapa perangkat ataupun aplikasi yang menggunakan sistem dari Internet Of Things maupun sistem dari Big data. Salah satu aplikasi yang menggunakan teknologi dari Internet of Things adalah Ubiquitous Network Robot Platform Development For Tracked Swarm Disaster Robot in Human Detection. Aplikasi ini merupakan salah satu aplikasi yang memanfaatkan teknologi robot dalam hal untuk menanggulangi sebuah bencana.

Robot memiliki keunggulan dalam mampu menjangkau daerah yang sulit dan berbahaya bagi manusia dan dapat dipersiapkan secara cepat serta mampu melakukan kegiatan penyelamatan seperti pencarian, pengujian, inspeksi (dalam skala terbatas.Pengembangan robotika untuk operasi penyelamatan bertumpu pada dua kegiatan, yaitu pengembangan platform hardware seperti robot beroda, robot ular, robot berkaki atau yang sejenisnya dan perangkat

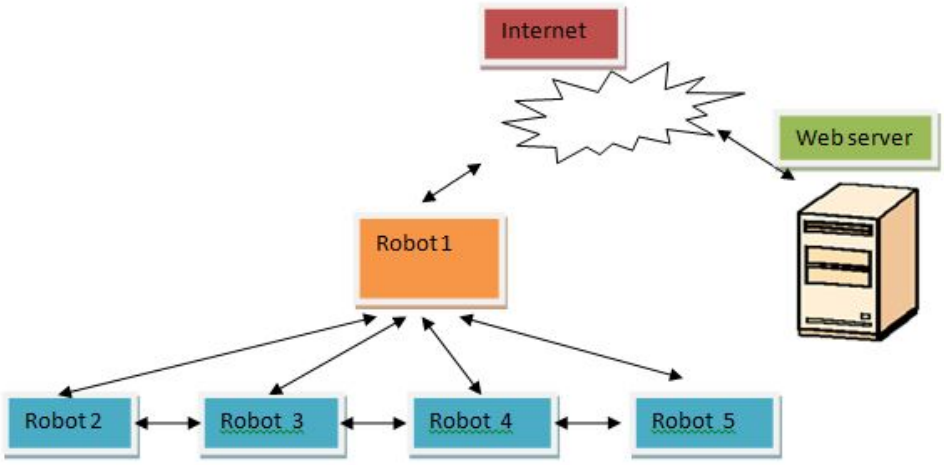


Gambar 1.10: Arsitektur Big Data Cloud

lunak yang mendukung kinerja robot tersebut. Selain itu, konsep robot dengan menggunakan konsep cloud pun sudah memasuki zamannya untuk membebaskan sumber internal agar dapat digunakan untuk fungsi yang lain.

Swarm robot merupakan sekumpulan robot yang dapat dimanfaatkan untuk mengidentifikasi medan bencana. Komunikasi diantara anggota robot dapat mewujudkan pendelegasian tugas sehingga target dapat tercapai. Penggunaan UNRPF pada swarm robot dapat memanfaatkan teknologi internet of things (IOT) pada sebuah model area bencana.

Dengan adanya penerapan teknologi IoT maka dapat digunakan untuk mengontrol swarm robot yang memiliki kemampuan navigasi dan berjalan dengan baik serta dapat mendeteksi korban dalam sebuah model medan bencana dimana user dapat menggunakannya secara mobile berbasis web serta penampilan data hasilpendeteksiankorbanoleh robot-robot tersebutsecara real time. Berikut ini penggambaran jaringan komunikasi antar robot dan internet yang memanfaatkan teknologi IoT 1.11.



Gambar 1.11: Desain Komunikasi dan Jaringan Swarm Robot

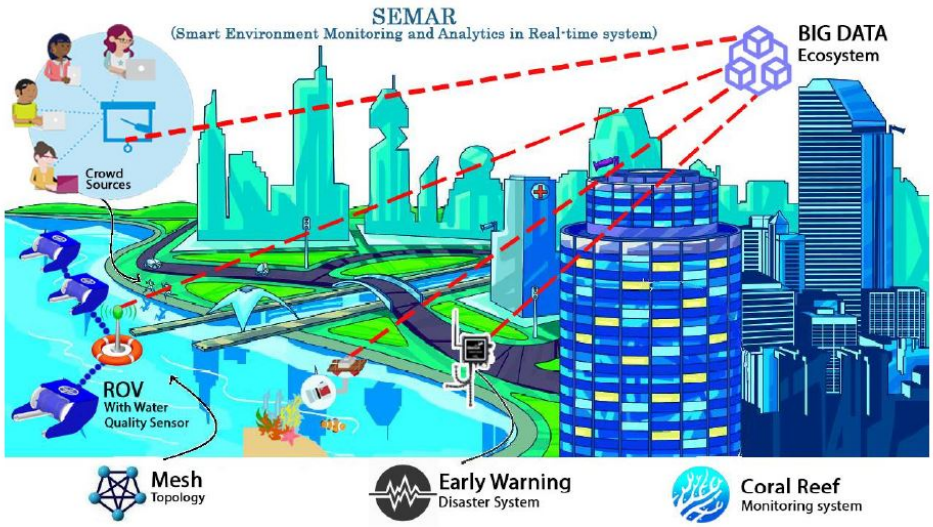
Kemudian contoh aplikasi lain yang menggunakan penerapan teknologi IoT dan Big Data adalah SEMAR (Smart Environment Monitoring and Analytics in Real-time system). SEMAR System merupakan sebuah aplikasi monitoring kualitas air berdasarkan lingkungan yang ada secara real-time.Pada SEMAR sistem ini terdiri beberapa aplikasi sistem yang saling terkoneksi diantaranya:

1. ROV yang dilengkapi dengan sensor kondisi air untuk mengetahui kondisi air pada sungai yang mampu dikendalikan secara jarak jauh dan mampu mengirim data sensor ke pusat data melalui komunikasi kabel dan atau nirkabel.

2. Sensor kondisi air dan banjir yang diletakkan di daerah tertentu dengan tujuan mampu mengetahui kondisi air pada sungai, dan juga mampu mengukur arus air dan ketinggian air sungai sebagai pendeteksi awal bencana banjir.

3. Pusat data yang menggunakan teknologi Big Data.

4. Sistem pendeteksi terumbu karang dengan menggunakan teknologi Rekonstruksi 3D bawah laut.



Gambar 1.12: SEMAR System

Teknologi Big Data pada sistem SEMAR ini berfungsi sebagai pusat data yang berasal dari sensor sensor air tersebut. Semakin lama volume data dari sensor akan semakin membesar dan akan semakin sulit untuk mendapatkan informasi lebih lanjut. Tanpa adanya analisa dari data-data yang banyak tersebut, maka data-data tersebut menjadi tidak berguna dan hanya membuang resource. Dengan adanya penerapan Big Data Analytic melalui beberapa parameter sensor yang digunakan dapat dibuat klasifikasi kualitas air dan memberikan early warning akan perubahan kondisi air sungai di suatu wilayah sehingga menghasilkan output informasi yang berguna bagi masyarakat, pemerintah dan pihak terkait.Sistem komunikasi yang digunakan pada tiap sensor agar menjadi satu kesatuan sistem SEMAR, menggunakan teknologi IoT (Internet of Things). Sehingga dengan adanya penerapan IoT pada sistem SEMAR ini user dapat memonitor kondisi kualitas air secara real time berdasarkan satu kesatuan komunikasi yang dihasilkan dari semua perangkat yang digunakan.

Bab 2

Sistem Benam

Sistem Benam adalah sistem komputer tujuan-khusus, yang seluruhnya dimasukkan ke dalam alat yang dikontrol. Kata benam (embedded) menunjukkan bahwa sistem ini merupakan bagian yang tidak dapat berdiri sendiri. Sebuah sistem benam memiliki kebutuhan tertentu dan melakukan tugas yang telah diset sebelumnya, tidak seperti komputer pribadi serba guna. Contoh sistem atau aplikasinya antara lain adalah instrumentasi medik, process control, automated vehicles control, dan perangkat komunikasi. Sistem Benam terdiri dari Single Board Computer (SBC) dan Single Board Micro Computer (SBMC)

2.1 Single Board Computer (SBC)

Single Board Computer (SBC) adalah komputer yang lengkap yang dibangun pada papan sirkuit tunggal, dengan mikroprosesor, memori, masukan/luaran (I/O), dan fitur lain yang dibutuhkan pada sebuah komputer fungsional. SBC dibuat sebagai alat demonstrasi atau pengembangan sistem, untuk sistem pendidikan, atau untuk digunakan sebagai pengendali komputer tertanam (embedded). Banyak jenis komputer portabel yang mengintegrasikan semua fungsi mereka ke sebuah papan sirkuit tunggal. Berbeda dengan desktop dan komputer pribadi, SBC sering tidak bergantung pada slot ekspansi untuk fungsi perifer atau ekspansi dan sebagai pengganti acapkali menyediakan pin GPIO (General-purpose input/output).

Perusahaan elektronika mengeluarkan banyak produk-produk SBC seperti Raspberry Pi, Intel Galileo, BeagleBoard, BeagleBone Black, Cubie Board,

dan lain-lain. SBC saat ini memiliki memori yang besar (128 MB hingga 2 GB, bahkan sebagian sudah lebih), memiliki eksternal storage (SD Card/USB Disk), dan memiliki prosesor dengan kecepatan Megahertz hingga Gigahertz, sebagian bahkan sudah Quad Core.

Sebuah SBC biasanya memiliki sebuah sistem operasi seperti Linux, FreeBSD, atau OS Open Source lainnya. Untuk pemrograman dapat dibuat program dengan bahasa pemrograman apapun seperti C, Python, bahkan Lisp atau Prolog. SBC juga memiliki kemampuan komputasi yang sangat besar seperti pengolahan/pemrosesan sinyal, citra, suara dan video.

Dalam buku ini akan digunakan Raspberry Pi 2 sebagai perangkat Single Board Computer (SBC) dalam membangun Internet of Things (IoT).

Raspberry Pi merupakan sebuah SBC yang dikembangkan oleh Raspberry Pi Foundation di Inggris dengan tujuan untuk digunakan sebagai media pembelajaran dan pengenalan dasar-dasar dari ilmu komputer. Raspberry Pi berdasarkan pada Sistem Broadcom BCM2835 pada chip (SoC), yang termasuk di dalamnya terdapat ARM1176JZF-S 700 MHz, VideoCore IV GPU dan dilengkapi dengan RAM 256 MB yang kemudian diupgrade menjadi 512 MB pada model B dan B+.

February 2015, Raspberry Pi 2 dirilis di pasaran. Board komputer baru tersedia hanya satu dalam sekali konfigurasi (model B) dan menyediakan fitur baru diantaranya Broadcom BMC2836 SoC dengan CPU ARM Cortex-A7 quad-core dan dual-core GPU Video IV. RAM sebesar 1 GB dengan spesifikasi yang mirip dengan generasi B+ sebelumnya.

Pada februari 2016, Raspberry Pi 3 dirilis, dengan membawa spesifikasi BCM 2837 dengan CPU 1.2 GHz Quad Core 64 bit prosesor ARMv8 Cortex A53, dengan Wi-Fi built-in BCM4338 802.11n 2.4 GHz dan Bluetooth 4.1 Low Energy(BLE).

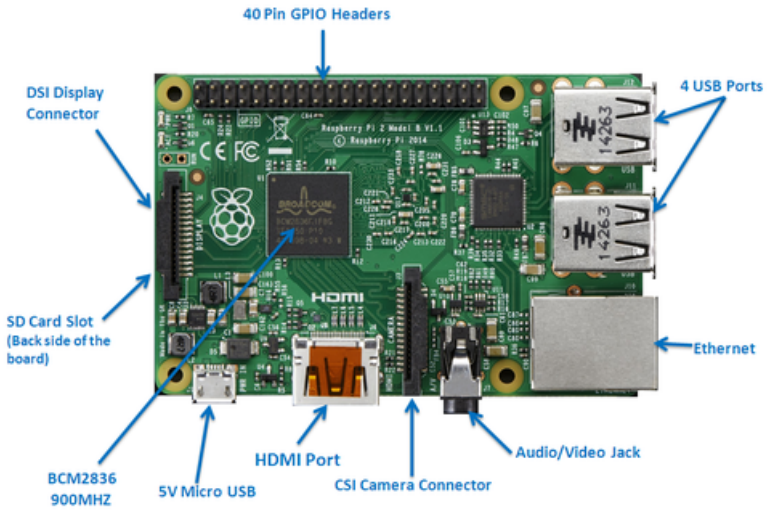
2.1.1 Spesifikasi Perangkat Keras

Perangkat Raspberry Pi 2 dapat dilihat pada gambar 1.1. Adapun spesifikasi teknis dari Raspberry Pi 2 yaitu [38]:

1. Prosesor Quad Core ARM Cortex-A7 900MHz

2. RAM 1GB

3. 40 pin extended GPIO



Gambar 2.1: Spesifikasi Perangkat Keras Raspberry Pi 2

4. 4 port USB
5. 3.5mm jack audio dan Composite video
6. Port HDMI
7. Port interface kamera CSI untuk koneksi ke kamera
8. Port interface display DSI untuk koneksi ke layar sentuh
9. Slot Micro SD port untuk instalasi sistem operasi dan media penyimpanan
10. Grafis dengan VideoCore IV 3D Graphics Core
11. Sumber Power dengan Micro USB

2.1.2 Instalasi Sistem Operasi

Beberapa sistem operasi yang dapat digunakan pada Raspberry Pi yaitu [4]:

1. Raspbian
2. Ubuntu Mate

3. Snappy Ubuntu Core
4. Windows 10 IoT Core
5. OSMC
6. OpenElec
7. Pinet
8. Risc OS

Dalam buku ini akan dijelaskan langkah-langkah menginstall Sistem Operasi Raspbian untuk Sistem Benam. Hal-hal yang perlu dipersiapkan adalah:

1. Komputer Personal

   Komputer Personal dapat menggunakan Sistem Operasi WIndows (Windows 7 atau di atasnya), Sistem Operasi Linux (Ubuntu atau Debian, atau distro lainnya), Mac OS (Versi X atau di atasnya).

2. Single Board Computer dan aksesorisnya

   Dalam buku ini digunakan Raspberry Pi 2 dengan kartu SD/MMC kapasitas 4 GB atau lebih class 10. Power Supply 5V 3A dengan konektor DC. Monitor dengan koneksi HDMI. Mouse dan Keyboard USB. Kabel HDMI.

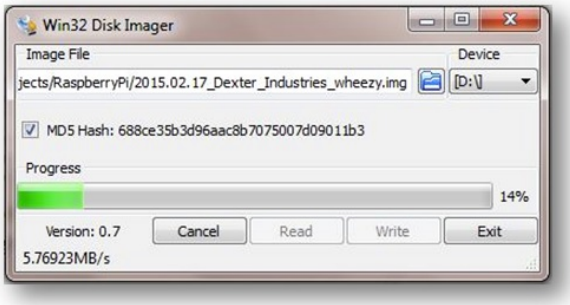
3. Image Sistem Operasi

   Sistem operasi yang didukung oleh perangkat Raspberry Pi 2 yang dapat didownload di laman https://www.raspberrypi.org/downloads/. Dalam buku ini Sistem Operasi yang digunakan adalah Sistem Operasi Raspbian. Untuk memudahkan melakukan burning ke SD Card dapat menggunakan software tambahan, yaitu Win32DiskImager untuk Sistem Operasi berbasis Windows yang dapat didownload di laman http://sourceforge.net/projects/win32diskimager/ atau ApplePi-Baker untuk Sistem Operasi berbasis Macintosh yang dapat di download di laman http://www.tweaking4all.com/hardware/raspberry-pi/macosx-apple-pi-baker/).

4. Koneksi Internet

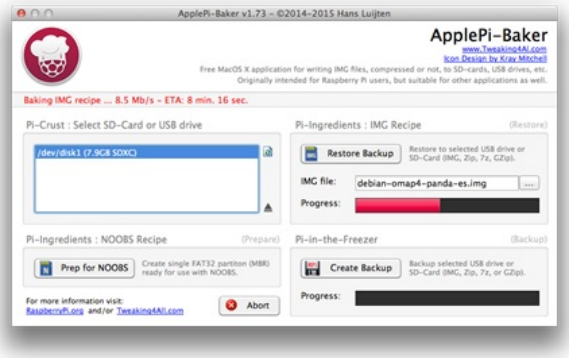
Langkah-langkah instalasi:

1. Mempersiapkan image sistem operasi
2. Koneksikan SD Card ke Komputer Personal
3. Burn image sistem operasi ke SD Card Untuk Sistem Operasi Linux menggunakan command di Linux : $sudo\ dd\ if =< image_file > of = /dev/sdb$. Untuk sistem operasi WIndows dengan Win32 Disk Imager:



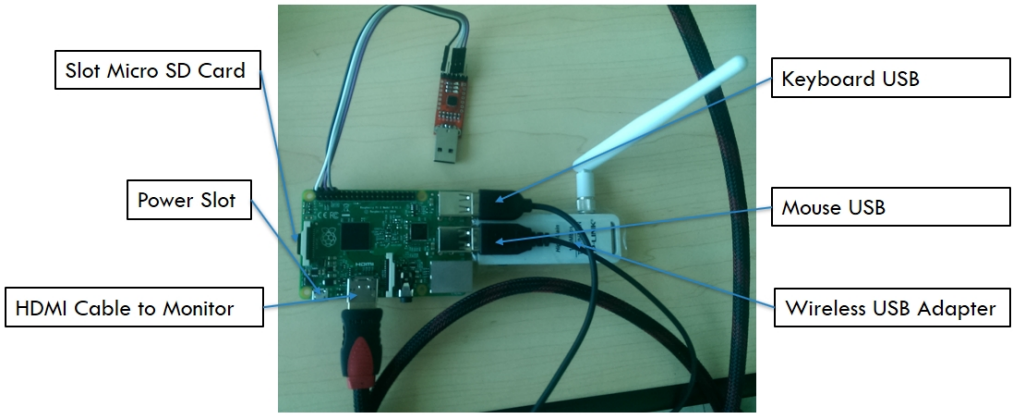
Gambar 2.2: Win 32 Disk Imager

Untuk sistem operasi Macintosh dengan ApplePi-Baker:



Gambar 2.3: ApplePi Baker

4. Persiapkan Raspberry Pi 2 dan pasang SD Card dari Komputer Personal ke Raspberry Pi 2

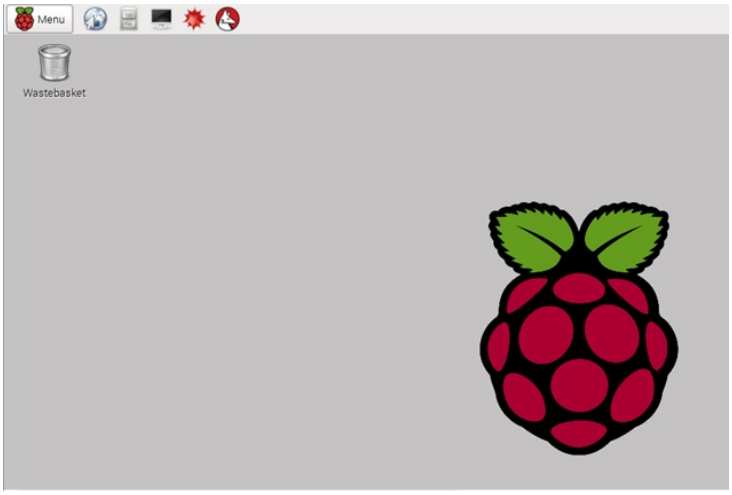


Gambar 2.4: Raspberry Pi dan perangkat-perangkat tambahannya

5. Nyalakan Raspberry Pi 2

6. Proses instalasi dimulai

7. Tunggu proses instalasi hingga selesai Jika proses instalan selesai tampilan Desktop Raspbian adalah seperti berikut



Gambar 2.5: Raspbian Desktop

2.1.3 Pengembangan Aplikasi dengan Python

Single Board Computer yang telah terinstall sistem operasi dapat digunakan untuk melakukan pengembangan aplikasi. Terdapat banyak bahasa pemrograman yang dapat digunakan dalam mengembangkan aplikasi pada Single Board Computer diantara bahasa pemrograman tersebut yang akan dibahas dalam buku ini adalah bahasa pemrograman Python. Secara default Raspbian OS sudah menyertakan python sebagai bahasa pemrograman yang dapat langsung digunakan.

Menggunakan python pada Raspbian OS atau Linux pada umumnya dapat menggunakan cara interaktif pada terminal, dengan mengetik python, lalu mengetikkan code python seperti contoh berikut:

```
>>> 1 + 2
3
>>> name = "Sarah"
>>> "Hello " + name
'Hello Sarah'
```

Gambar 2.6: Interaktif Code Python

Dapat juga dengan mengetikkan kode pada sebuah file yang disimpan dalam ekstensi .py dan dieksekusi dengan mengetik $python < spasi > nama_file.py$:

```
print("Hello world")
```

Gambar 2.7: Pemrograman Python

2.2 Single Board Micro Computer (SBMC)

Single Board Micro Computer (SBMC) atau mikrokontroler adalah komputer kecil di dalam sebuah rangkaian terintegrasi berisi sebuah prosesor core, memory, dan programmable input/output peripherals. Dalam diskusi sehari-hari dan forum internet mikrokontroller sering dikenal dengan simbol nC, uC, atau MCU. Terjemahan bebas dari pengertian tersebut, bisa dikatakan bahwa mikrokontroller adalah komputer yang berukuran mikro dalam 1 chip IC (integrated circuit) yang terdiri dari prosessor, memory, dan

I/O yang bisa kita kontrol dengan memprogramnya. I/O juga sering disebut dengan GPIOP (General Purpose Input Output Pins) yang berarti : pin yang bisa kita program sebagai input atau output sesuai kebutuhan [7].

Sebagian perangkat elektronik yang ada sekarang ini memiliki mikrokontroler pada bagian intinya. Mikrokontroler yang dioptimalkan untuk mengendalikan input saja atau output saja. Mereka pada umumnya memiliki kemampuan komputasi yang rendah jika dibandingkan dengan prosesor yang digunakan pada komputer multimedia atau komputer server. Mikrokontroler membutuhkan daya yang lebih rendah dibanding prosesor lainnya dan lebih mudah untuk berinteraksi dengan dunia fisik melalui sirkuit input yang disebut sensor dan sirkuit output yang disebut aktuator. Mikrokontroler juga dapat berkomunikasi dengan prosesor lain melalui berbagai antarmuka komunikasi (communication interface)[8].

Pada buku ini digunakan board Arduino sebagai single board micro computer (SBMC) dikarenakan kelebihannya adalah kita tidak direpotkan dengan rangkaian minimum system dan downloader atau programmer karena sudah built in dalam satu board. Oleh sebab itu kita bisa fokus dalam pengembangan sistem. Jika dilihat dari sejarah pembuatan arduino berawal dari sebuah thesis yang dibuat oleh Hernando Barragan, di institute Ivrea, Italia pada tahun 2005, dikembangkan oleh Massimo Banzi dan David Cuartielles dan diberi nama Arduin of Ivrea. Lalu diganti nama menjadi Arduino yang dalam bahasa Italia berarti teman yang berani.

Tujuan awal dibuat Arduino adalah untuk membuat perangkat mudah dan murah, dari perangkat yang ada saat itu. Dan perangkat tersebut ditujukan untuk para siswa yang akan membuat perangkat desain dan interaksi[9].

Adapun untuk hal platform memiliki kemudahan dalam penggunaan dan penulisan kode. Arduino IDE menggunakan bahasa pemrograman C++ dengan versi yang telah disederhanakan.

2.2.1 Arduino Mega 2560

Arduino Mega 2560 adalah board Arduino yang merupakan perbaikan dari board Arduino Mega sebelumnya. Arduino Mega awalnya memakai chip ATmega1280 dan kemudian diganti dengan chip ATmega2560, oleh karena itu namanya diganti menjadi Arduino Mega 2560. Pada saat tulisan ini dibuat, Arduino Mega 2560 sudah sampai pada revisinya yang ke 3 (R3). Berikut spesifikasi Arduino Mega 2560 R3[8].

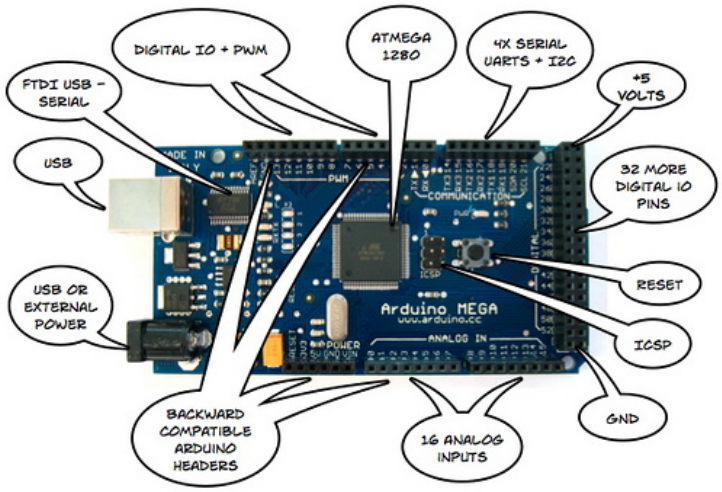
Table 2.1: Spesifikasi Arduino Mega 2560

Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage(recommended)	7-12V
Input Voltage(limit)	6-20V
Digital I/O Pins	54 (of which 15 provide PWM output)
Analaog Input Pins	16
DC Current per I/O Pin	20 mA
DC Current for 3.3 V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz
Length	101.52 mm
Width	53.3 mm
Weight	37 g

2.2.2 Spesifikasi Perangkat Keras

Perangkat Arduino Mega 2560 dapat dilihat pada gambar 1.3. Adapun spesifikasi teknis dari Arduino Mega 2560 yaitu [10]:

1. USB atau eksternal Power
2. Chip Atmega 16U2
3. 54 pin digital input/output (15 dapat digunakan sebagai output PWM)
4. 16 pin analog input
5. Circuit Reset
6. Header ICSP
7. 16 MHZ crsytal oscillator



Gambar 2.8: Hardware Define Arduino Mega 2560
(http://techno-robotics.com/wp-content/uploads/2015/10/55.jpg)

2.2.3 Instalasi Sistem Operasi

IDE Arduino Untuk memprogram board Arduino, kita butuh aplikasi IDE (Integrated Development Environment) bawaan dari Arduino. Aplikasi ini berguna untuk membuat, membuka, dan mengedit source code Arduino (Sketches, para programmer menyebut source code arduino dengan istilah "sketches"). Sketch merupakan source code yang berisi logika dan algoritma yang akan diupload ke dalam IC mikrokontroller (Arduino). Cara Instalasi IDE Arduino [11]:

1. Mengunduh file installer IDE Arduino di https://www.arduino.cc/en/Main/Software. Di halaman tersebut terdapat file installer untuk berbagai macam operating system yaitu Windows, MacOS dan Linux.

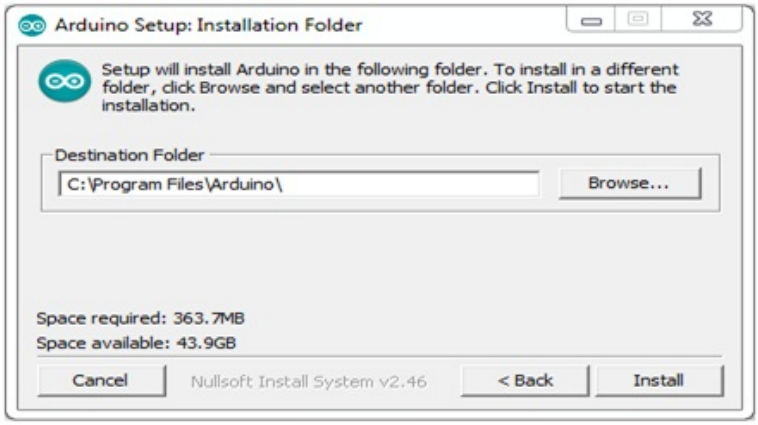
2. Setelah berhasil mengunduhnya maka double click file tersebut untuk memulai proses instalasi.

3. Setelah file installer dijalankan, akan muncul jendela Licence Agreement. Klik saja tombol Agree

4. Masukan folder instalasi untuk arduino atau biarkan default di C:\

Gambar 2.9: Gambar Proses Instalasi

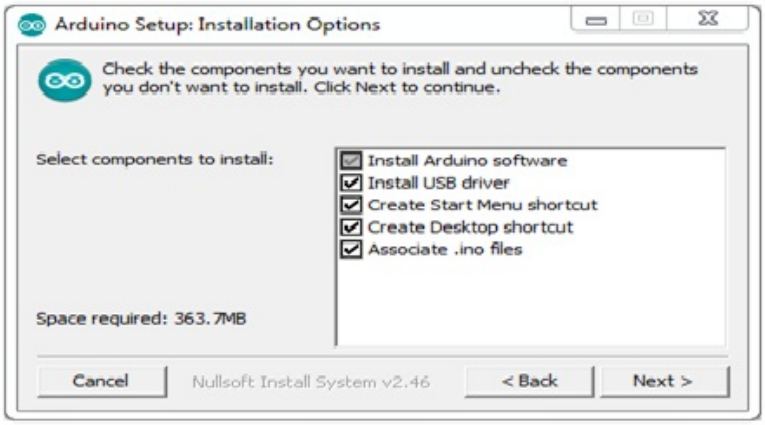
Program Files\ Arduino



Gambar 2.10: Gambar Proses Instalasi

5. Setelah itu akan muncul jendela Setup Installation Options.Sebaiknya dicentang semua opsinya

6. Selanjutnya proses instalasi akan dimulai sampai selesai

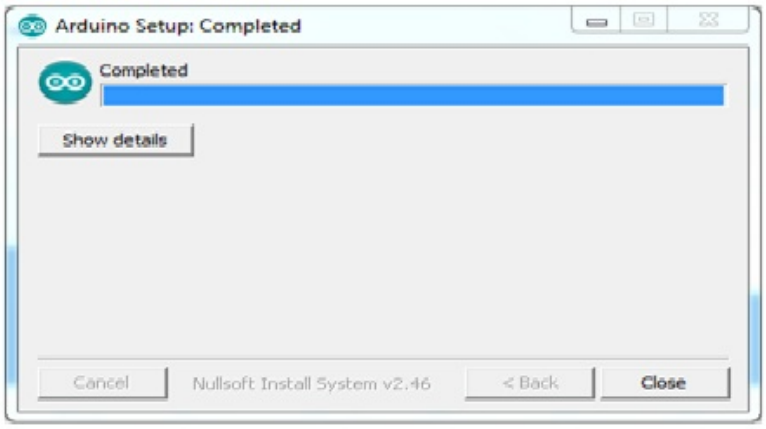


Gambar 2.11: Gambar Proses Instalasi

7. Software IDE Arduino jika dijalankan akan muncul jendela sebagai berikut[7].

Interface Arduino IDE tampak seperti gambar di atas. Dari kiri ke kanan dan atas ke bawah, bagian-bagian IDE Arduino terdiri dari:

- Verify : pada versi sebelumnya dikenal dengan istilah Compile. Sebelum aplikasi diupload ke board Arduino, biasakan untuk memverifikasi terlebih dahulu sketch yang dibuat. Jika ada kesalahan pada sketch, nanti akan muncul error. Proses Verify / Compile mengubah sketch ke binary code untuk diupload ke mikrokontroller.
- Upload : tombol ini berfungsi untuk mengupload sketch ke board Arduino. Walaupun kita tidak mengklik tombol verify, maka sketch akan di-compile, kemudian langsung diupload ke board. Berbeda dengan tombol verify yang hanya berfungsi untuk memverifikasi source code saja.
- New Sketch : Membuka window dan membuat sketch baru
- Open Sketch : Membuka sketch yang sudah pernah dibuat. Sketch yang dibuat dengan IDE Arduino akan disimpan dengan ekstensi file .ino

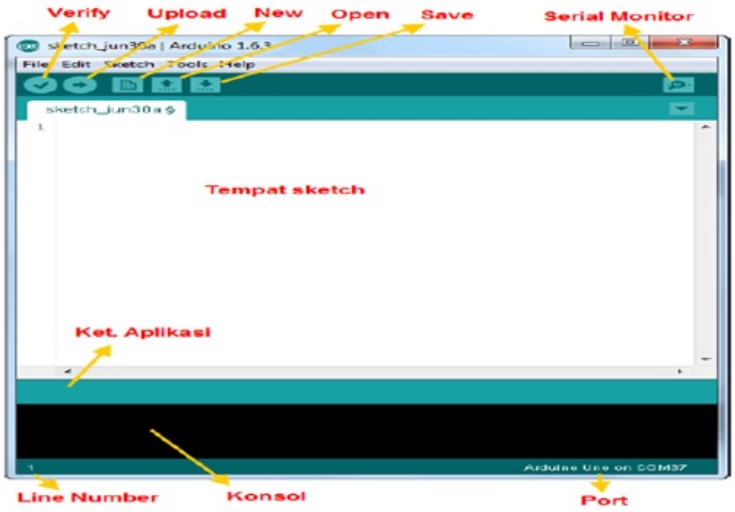


Gambar 2.12: Gambar Proses Instalasi

- Save Sketch : menyimpan sketch, tapi tidak disertai mengcompile.
- Serial Monitor : Membuka interface untuk komunikasi serial
- Keterangan Aplikasi : pesan-pesan yang dilakukan aplikasi akan muncul di sini, misal "Compiling" dan "Done Uploading" ketika kita mengcompile dan mengupload sketch ke board Arduino
- Konsol : Pesan-pesan yang dikerjakan aplikasi dan pesan-pesan tentang sketch akan muncul pada bagian ini. Misal, ketika aplikasi mengcompile atau ketika ada kesalahan pada sketch yang kita buat, maka informasi error dan baris akan diinformasikan di bagian ini.
- Baris Sketch : bagian ini akan menunjukkan posisi baris kursor yang sedang aktif pada sketch.
- Informasi Port : bagian ini menginformasikan port yang dipakai oleh board Arduino.

2.2.4 Pengembangan Aplikasi dengan arduino IDE

Pada bagian ini akan diberikan sebuah contoh aplikasi arduino IDE untuk menampilkan "Hello Word" pada sebuah LCD menggunakan arduino.Pada arduino IDE menggunakan libary LiquidCrystal yang bekerja pada semua LCD dengan driver Hitachi HD44780. Sebelum memasukan program ke



Gambar 2.13: Gambar Proses Instalasi

dalam arduino hal yang perlu dilakukan adalah menyusun rangkaian sebagai berikut;

- LCD RS pin ke pin digital 12
- LCD Enable pin to digital pin 11
- LCD D4 pin to digital pin 5
- LCD D5 pin to digital pin 4
- LCD D6 pin to digital pin 3
- LCD D7 pin to digital pin 2
- LCD R/W pin to ground
- LCD VSS pin to ground
- LCD VCC pin to 5V
- Sambungkan potensiometer 10 KOhm ke +5v dan GND , dan Pin LCD 3 ke potensiometer

Skecthnya sebagai berikut;

```
// memasukan kode untuk library:
#include <LiquidCrystal.h>

// menginisialisasi library dengan nomor-nomor pin
   interface
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

void setup() {
  // mengatur jumlah kolom dan baris LCD:
  lcd.begin(16, 2);
  // mencetak pesan ke LCD.
  lcd.print("hello, world!");
}

void loop() {
  // mengatur kursor ke kolom 0, deret 1
  // (catatan: deret 1 adalah baris kedua, karena
     mulai perhitungan dari 0):
  lcd.setCursor(0, 1);
  // mencetak jumlah detik dihitung setelah reset:
  lcd.print(millis() / 1000);
}
```

Bab 3

Akuisisi Data

3.1 Tipe Akuisisi Data

Pada Internet of Thing akusisi data merupakan hal yang sangat penting untuk dilakukan. Pada akusisi data akan dibahas bagaimana cara untuk mengambil sebuah data dari sebuah sensor. Tipe akusisi data yang akan dibahas disini yaitu GPIO, USART dan I2C.

3.1.1 GPIO

General Purpose Input/Output(GPIO) merupakan pin generik pada sirkuit terpadu (chip) yang perilakunya (termasuk apakah pin itu input atau output) dapat dikontrol (diprogram) oleh pengguna saat berjalan.

Pin GPIO tidak memiliki penetapan awal sebelumnya, dan diseting tidak terpakai secara default. Idenya adalah bahwa kadang-kadang sistem integrator yang sedang membangun sistem lengkap mungkin perlu beberapa baris kontrol digital dan ini tersedia dari sebuah chip sehingga dapat mengatur sirkuit tambahan untuk digunakan.

Pada hakekatnya hampir semua SBC (single-board computer) menyediakan GPIO untuk ekspansi disambungkan ke modul atau komponen lainnya. Papan sirkuit embedded seperti Arduino, BeagleBone, Raspberry Pi dan lainnya, sering kali memanfaatkan GPIO untuk membaca data atau sinyal dari berbagai sensor lingkungan seperti IR , video, suhu, orientasi 3 dimensi, percepatan dan sebagainya, disamping untuk menulis atau mengirim data melalui output ke motor DC (melalui modul PWM), audio, display LCD, atau lampu LED.

3.1.2 USART

A Universal Synchronous/Asynchronous Receiver/Transmitter(USART) merupakan jenis perangkat antarmuka serial yang dapat diprogram untuk berkomunikasi secara asinkron atau secara sinkron.

Kemampuan sinkron USART ini yang terutama ditujukan untuk mendukung protokol sinkron seperti IBM's Synchronous transmit-receive (STR), Binary Synchronous Communications (BSC), Synchronous Data Link Control (SDLC), dan protokol sinkron ISO-standard High-Level Data Link Control (HDLC) link-layer, yang digunakan dengan modem sinkron frekuensi suara. Protokol ini dirancang untuk membuat penggunaan bandwidth terbaik ketika modem merupakan perangkat analog. Pada waktu itu, modem asinkron pita suara tercepat bisa mencapai kecepatan 300 bps menggunakan modulasi Frequency Shift Keying, sedangkan modem sinkron bisa dijalankan pada kecepatan hingga 9600 bps menggunakan Phase Shift Keying. Transmisi sinkron hanya menggunakan sedikit lebih dari 80 persen dari bandwidth transmisi asinkron, saat start dan stop bit yang tidak digunakan. Teknologi sebelumnya telah digantikan oleh modem dengan yang mengkonversi data asinkron ke bentuk sinkron. USART kadang-kadang masih terintegrasi dengan MCU. USART masih digunakan dalam router yang terhubung ke perangkat CSU / DSU eksternal.

3.1.3 I2C

I2C (Circuit Inter-Integrated) adalah multi-master, multi-slave, single-ended, serial computer bus yang diciptakan oleh Philips Semiconductor (sekarang NXP Semikonduktor). I2C biasanya digunakan untuk menghubungkan IC peripheral yang kecepatannya lebih rendah dengan prosesor dan mikrokontroler pada jarak pendek dan komunikasi intra-board.

I2C hanya menggunakan baris dua arah open-drain, Serial Data Line (SDA) dan Serial Clock Line (SCL), pulled up resistor. Tegangan yang digunakan adalah 5 V atau 3,3 V meskipun sistem dengan tegangan lainnya diperbolehkan.

Desain referensi I2C yaitu 7-bit atau 10-bit (tergantung pada perangkat yang digunakan) alamat ruang. Kecepatan bus I2C umum adalah 100 kbit/s mode standar dan 10 kbit/s mode kecepatan rendah, tetapi frekuensi clock rendah juga diperbolehkan. Revisi terbaru dari I2C dapat menghandel lebih banyak node dan berjalan pada kecepatan yang lebih cepat (400 kbit/s mode

cepat, 1 Mbit/s mode cepat plus atau Fm +, dan 3,4 Mbit/s mode kecepatan tinggi).

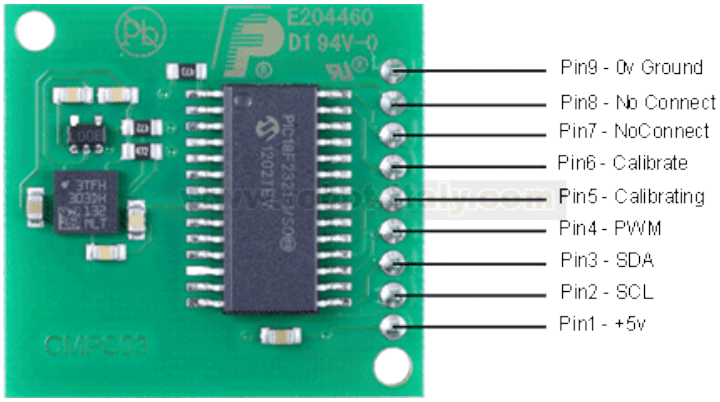
3.2 Jenis Sensor

Konsep Internet of Thing tidak akan terlepas dari dunia sensor. Sensor-sensor adalah bagian yang akan berperan sebagai reseptor atau indra pada lingkungan yang ada kemudian dihubungkan pada sebuah jaringan nirkabel. Data-data yang dihasilkan oleh sensor-sensor tersebut dapat ditampilkan secara real time dengan mempertimbangkan sifatnya yang bersifat pasif. Sebelum melangkah lebih jauh akan dijelaskan secara singkat karakteristik beberapa sensor-sensor yang akan digunakan dalam eksperimen. Sensor-sensor yang akan dibahas pada buku ini yaitu Sensor Magnet (Kompas), Sensor Air, Sensor Udara, Global Positioning System (GPS), IMU dan Gyro, Sensor Temperatur dan Kelembaban, Sensor Suara, Sensor Hall, Sensor Laser, Sensor PIR(Passive Infra Red), Sensor Ultrasonic (SRF02), Sensor Warna(Color), Sensor kemiringan (Tilt Sensor), Sensor Rotasi (Rotation Sensor), Sensor Reflektif Infrared, Sensor Moisture, Sensor Api (Flame Sensor), Sensor Ultraviolet, dan juga dilengkapi dengan Penggerak motor.

3.2.1 Sensor Magnet (Kompas)

Sensor magnet yang akan digunakan adalah berupa modul sensor CMPS03. Modul kompas ini telah dirancang khusus untuk digunakan dalam robot sebagai bantuan untuk sistem navigasi. Tujuannya adalah untuk menghasilkan nomor unik untuk mewakili arah robot. Modul kompas membutuhkan power supply 5V pada 15mA. Pada CMPS03, arah mata angin dibagi dalam bentuk derajat yaitu : Utara (0 derajat), Timur (90 derajat), Selatan (180 derajat) dan Barat (270 drajat).

Ada dua cara untuk mendapatkan data dari modul yaitu dengan sinyal PWM yang tersedia pada pin 4, atau dengan menggunakan antarmuka I2C pada pin 2,3. Alamat default I2C yang digunakan adalah 0x60. Pin 7 adalah pin masukan dengan memilih salah satu mode operasi yaitu 50Hz (rendah) atau 60Hz (tinggi). Pin 6 digunakan untuk mengkalibrasi kompas. Input (pin 6) sudah memiliki on-board pull-up resistor dan dapat dibiarkan tidak terhubung setelah kalibrasi. Pin 5 dan 8 dibiarkan tidak terhubung. Sebenarnya pin 8 adalah garis reset prosesor dan memiliki on-board pull-up resistor[12].



Gambar 3.1: Hardware sensor CMPS03[12]

Modul digital compass harus tetap dalam posisi horizontal terhadap permukaan bumi dengan sisi komponen berada dibagian atas. Jauhkan modul dari metal, terlebih lagi dari objek yang mengandung magnet.

3.2.2 Sensor Air

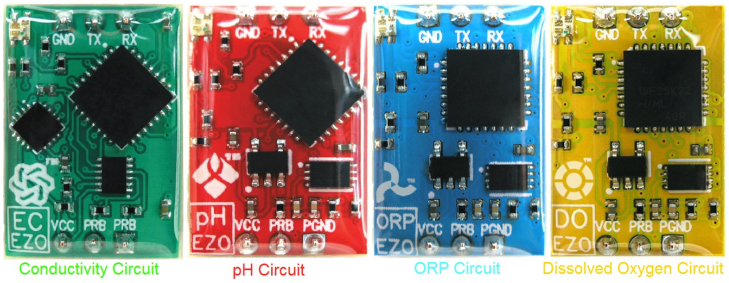
Sensor air dirancang untuk deteksi air, yang dapat secara luas digunakan dalam penginderaan curah hujan, ketinggian air, bahkan kebocoran cairan. Salah satu sensor air yang digunakan adalah Liquid Level Sensor. Modul ini dapat diterapkan untuk sistem alarm ketinggian cairan.

Modul ini merupakan aplikasi dari amplifikasi arus dengan transistor. Ketika tingkat cairan cukup tinggi untuk menghantarkan arus antara base dan catu daya positif, sejumlah arus dihasilkan antara basis dan emitor. Yang berarti bahwa arus listrik dihasilkan dalam faktor amplifikasi tertentu antara kolektor dan emitor, dan diterapkan pada tahan di emitor untuk menghasilkan tegangan. Kemudian, tegangan ini akan dikumpulkan oleh konverter AD[13]. Untuk menggunakan sensor Liquid Level yaitu dengan menghubungkan:

- VCC dengan 2.0V - 5.0V
- GND dengan power supply ground
- AOUT dengan MCU.IO (analog output)

Gambar 3.2: Hardware sensor Liquid Level

Selain itu pada sensor air juga terdapat sensor air yang dikeluarkan oleh ATLAS, yaitu Sensor ORP, Sensor Dissolved Oxygen, Sensor PH dan Sensor Conductivity[14].



Gambar 3.3: Board sensor air ATLAS[14]

3.2.2.1 Sensor ORP

Sirkuit Ezo kelas ORP, adalah sebuah sistem komputer footprint kecil yang secara khususn dirancang untuk digunakan dalam aplikasi robotika di mana insinyur embedded system membutuhkan pengukuran yang akurat dan tepat dari potensi penurunan oksidasi (Oxidation Reduction Potential). Sirkuit ORP akan menampilkan bacaan dari -1019.9mV ke + 1019.9mV. The Ezo sirkuit kelas ORP, mampu membaca ORP sampai tempat persepuluh.

Fiturnya yaitu lebar pembacaan berbagai ORP dari -1019.9mV ke + 1019.9mV, pembacaan keakurat ORP sampai ke persepuluh (+/- 1mV), protokol kalibrasi fleksibel mendukung titik kalibrasi untuk nilai apapun, kalibrasi diperlukan hanya sekali per tahun dengan probe Atlas Sciencetific ORP, mode pembacaan tunggal atau terus menerus, format data ASCII.

Dua protokol data:

- konektivitas UART asynchronous serial

- (RX / TX tegangan berkisar 0-VCC)
- I2C (default alamat I2C 0x62)
- Kompatibel dengan mikroprosesor yang mendukung UART, atau protokol I2C
- Operasi tegangan: 3.3V ke 5V
- Bekerja dengan berbagai macam off-the-shelf probe ORP

3.2.2.2 Sensor Dissolved Oxygen

Sirkuit kelas Atlas Scientific Ezo dissolved oxygen, adalah komputer footprint kecil sistem yang dirancang khusus untuk digunakan dalam aplikasi robotika di mana insinyur embedded system membutuhkan pengukuran yang akurat dan tepat dari oksigen terlarut dalam air. Sirkuit Ezo kelas D.O. adalah perangkat yang sangat kompleks yang memperhitungkan suhu, salinitas dan tekanan untuk menurunkan konsentrasi oksigen terlarut di air. Sinyal yang datang dari sebuah HDPE penyelidikan galvanik cukup lurus ke depan; 0mV berarti tidak ada oksigen dan tegangan dari 36mV ke 54 mV berarti 100 persen oksigen.

Fiturnya yaitu pembacaan penuh sensor dissolved oxygen dari 0,01-35,99 mg / L, akurasi pembacaan dissolved oxygen sampai ke tempat seratus (+/- 0,2), kompensasi suhu, kompensasi salinitas, kompensasi tekanan, protokol kalibrasi fleksibel mendukung satu atau dua kalibrasi, kalibrasi diperlukan hanya sekali per tahun dengan sebuah probe D.O. Atlas Scientific, mode pembacaan tunggal atau terus menerus, format data ASCII.

Dua protokol data:

- Konektivitas UART asynchronous serial
- RX / TX tegangan berkisar 0-VCC
- I2C (default alamat I2C 0x61)
- Kompatibel dengan mikroprosesor yang mendukung UART, atau protokol I2C
- Operasi tegangan: 3.3V ke 5V
- Bekerja dengan galvanik off-the-shelf Probe HDPE dissolved oxygen

3.2.2.3 Sensor PH

Sirkuit Ezo kelas pH, adalah sebuah sistem komputer footprint kecil yang dirancang khusus untuk digunakan dalam aplikasi robotika di mana insinyur sistem tertanam membutuhkan pengukuran yang akurat dan tepat dari pH. Sirkuit Ezo kelas pH, mampu pH membaca, sampai menjangkau ke tempat seperseribu.

Fiturnya yaitu range pembacaan pH 0,001-14,000, akurasi pembacaan pH sampai tempat seperseribu (+/- 0,02), pembacaan temperatur dependen atau temperatur independen, protokol kalibrasi fleksibel mendukung 1 titik, 2 titik, atau 3 kalibrasi titik, kalibrasi diperlukan hanya sekali per tahun dengan probe Atlas Scientific pH, mode pembacaan tunggal atau terus menerus, format data ASCII.

Dua protokol data

- Konektivitas UART asynchronous serial
- RX / TX tegangan berkisar 0-VCC
- I2C (default alamat I2C 0x63)
- Kompatibel dengan mikroprosesor yang mendukung UART, atau protokol I2C
- Operasi tegangan: 3.3V ke 5V
- Bekerja dengan off-the-shelf Probe PH manapun

3.2.2.4 Sensor Conductivity

Sirkuit Ezo kelas konduktivitas adalah sistem komputer footprint kecil yang secara khusus dirancang untuk digunakan dalam aplikasi robotika di mana insinyur sistem tertanam membutuhkan pengukuran yang akurat dan tepat Listrik Konduktivitas (EC), Total Dissolved Padat (TDS), salinitas dan Berat Jenis (SG) air laut. Hal ini penting untuk diingat bahwa sirkuit Ezo kelas konduktivitas hanya dapat digunakan untuk melakukan pengukuran dalam cairan dimana pelarut air.

Sirkuit Ezo kelas konduktivitas adalah perangkat yang sangat kompleks yang terdiri dari beberapa lapisan. Lapisan pertama perangkat driver probe konduktivitas. Sebuah probe konduktivitas adalah perangkat pasif yang

output tidak ada sinyal listrik. Sirkuit Ezo kelas konduktivitas transmit sebuah gelombang kotak persegi saat ini pada frekuensi yang berbeda-beda. Frekuensi bervariasi (23,81 Hz untuk 41,27 KHz) dari gelombang kotak persegi saat ini benar-benar penting secara akurat membaca konduktivitas. Fiturnya yaitu konduktivitas, total padatan terlarut, unit salinitas praktis, berat jenis air laut, akurasi +/- 2 persen, range penuh E.C. berkisar dari 0,07 mikrodetik / cm sampai 500.000 mikrodetik / cm, temperatur dependen atau temperatur bacaan independen, protokol kalibrasi fleksibel mendukung satu titik atau dua titik kalibrasi, kalibrasi untuk setiap nilai E.C., kalibrasi diperlukan hanya sekali per tahun, format data ASCII.

Dua protokol data

- UART konektivitas serial asynchronous
- (RX / TX tegangan berkisar 0-VCC)
- I2C (default alamat I2C 0x64)
- Operasi tegangan: 3.3V - 5V
- Bekerja dengan off the self dua Probe konduktivitas konduktor
- Bekerja dengan nilai K dari K 0,1 ke K 10

3.2.3 Sensor Udara

Sensor gas dengan kode MQ yang terdiri dari 2 bagian, yaitu sensor elektrokimia dan sebuah pemanas (internal heater). Sensor ini dapat mendeteksi berbagai tipe gas, dan akan lebih sensitif untuk jenis gas tertentu, tergantung jenis sensor yang terpasang. Semua sensor gas tipe ini harus dikalibrasi dengan mengukurnya pada udara / gas yang telah diketahui konsentrasinya.

Keluaran sensor ini berupa data analog yang dapat dibaca oleh pin-pin Analog Arduino. Sensor menggunakan tegangan 5V untuk supply internal heaternya, biasanya sangat cepat untuk mencapai 50-60 derajat celcius. Setelah kondisi "burn-in-time", internal heater pada umumnya memerlukan waktu 3 menit untuk mencapai pembacaan yang stabil (contohnya pada MQ2). Beberapa datasheet menggunakan istilah "preheat", atau sebuah prasyarat untuk membuat pembacaan sensor lebih stabil. Kondisi "burn-in-time" atau "preheat" biasanya dilakukan dalam waktu 12 hingga 48 jam

dengan melihat kembali datasheet untuk sensor yang bersangkutan[15]. Untuk menggunakan sensor Gas yaitu dengan menghubungkan:

- VCC dengan 2.5V - 5.0V
- GND dengan power supply ground
- AOUT dengan MCU.IO (analog output)
- DOUT dengan MCU.IO (digital output)

Gambar 3.4: Hardware sensor MQ-5

3.2.4 Global Positioning System (GPS)

Global Positioning System (GPS) adalah sistem navigasi berbasis satelit yang terdiri dari jaringan 24 satelit ditempatkan ke orbit oleh Departemen Pertahanan AS. GPS pada awalnya ditujukan untuk aplikasi militer, namun pada 1980-an, pemerintah membuat sistem yang tersedia untuk penggunaan sipil. GPS bekerja di semua kondisi cuaca, di mana saja di dunia, 24 jam sehari. Tidak ada biaya langganan atau biaya setup untuk menggunakan GPS.

Satelit GPS mengelilingi bumi dua kali sehari dalam orbit yang sangat tepat dan mengirimkan sinyal informasi ke bumi. Penerima GPS mengambil informasi ini dan menggunakan Trilaterasi untuk menghitung lokasi yang tepat. Pada dasarnya, penerima GPS membandingkan waktu sinyal yang ditransmisikan oleh satelit dengan waktu yang telah diterima. Perbedaan waktu memberitahu GPS penerima seberapa jauh satelit tersebut. Dengan pengukuran jarak dari beberapa satelit, receiver dapat menentukan posisi

pengguna dan menampilkannya pada unit peta elektronik. Sebuah penerima GPS harus terhubung ke sinyal dari minimal 3 satelit untuk menghitung posisi 2-D (lintang dan bujur) dan melacak gerakan. Dengan empat atau lebih satelit, receiver dapat menentukan posisi 3-D pengguna (lintang, bujur dan ketinggian). Setelah posisi pengguna telah ditentukan, unit GPS dapat menghitung informasi lain, seperti kecepatan, bearing, track, jarak perjalanan, jarak ke tujuan, waktu matahari terbit dan terbenam dan lain sebagainya[16].

3.2.5 IMU dan Gyro

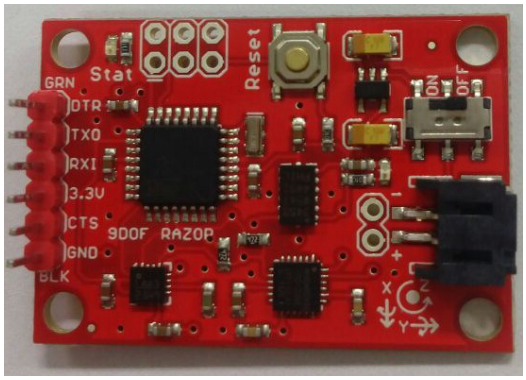
9DOF Razor IMU menggabungkan tiga buah sensor yaitu sebuah ITG-3200 (MEMS triple-axis gyro), ADXL345 (triple-axis accelerometer), dan HMC5883L (triple-axis magnetometer) untuk memberikan sembilan derajat pengukuran inersia. Output dari semua sensor diproses oleh ATmega328 on-board dan output melalui interface serial. Hal ini memungkinkan 9DOF Razor untuk digunakan sebagai mekanisme kontrol yang sangat kuat untuk UAV, kendaraan otomatis dan sistem stabilisasi gambar.

Board diprogram dengan bootloader 8MHz Arduino (stk500v1) dan beberapa contoh firmware untuk demo output dari semua sensor. Dengan menggunakan FTDI Basic Breakout yang terhubung secara serial ke pin TX dan RX dan supply 3.3V, akan dapat dilakukan pengujian sensor dengan membuka program terminal untuk 57600bps. Dapat juga menggunakan Arduino IDE untuk memprogram 9DOF Razor IMU, dengan memilih 'Arduino Pro atau Pro Mini (3.3V, 8mhz) w / ATmega328' sebagai pilihan.

9DOF Razor IMU beroperasi pada 3.3 VDC. Setiap daya yang diberikan ke konektor JST akan diatur sebagai tegangan operasi. Baterai Lipo adalah pilihan catu daya yang sangat baik. Header output dirancang untuk dapat dipasangkan dengan board 3.3V FTDI Basic Breakout, sehingga dapat dengan mudah menghubungkan board ke port USB komputer atau untuk koneksi nirkabel, dapat dihubungkan ke Bluetooth Mate atau XBee Explorer. Untuk menggunakan IMU dapat dipakai fitur USART baik itu pada mikrokontroler maupun raspiberry pi[17]. Untuk menggunakan sensor IMU yaitu dengan menghubungkan:

- 3.3V dengan 3.3V
- GND dengan power supply ground

- TXO dengan MCU.RX
- RXI dengan MCU.TX



Gambar 3.5: Hardware sensor Razor IMU

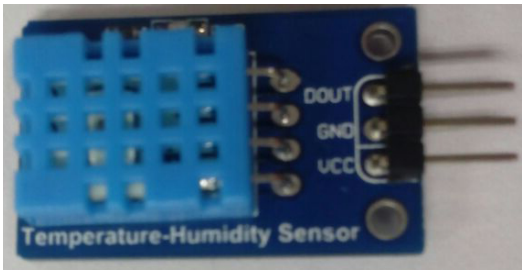
3.2.6 Sensor Temperatur dan Kelembaban

Sensor DHT11 merupakan sensor suhu dan kelembaban yang memiliki output sinyal digital dan dapat dikalibrasi dengan suhu dan kelembaban yang nyata, sensor ini memiliki keandalan yang tinggi dan stabilitas jangka panjang yang sangat baik. Sensor ini mencakup elemen resistif dan perangkat pensensoran yang mengukur suhu dengan NTC. Sensor ini memiliki kualitas yang sangat baik, respon cepat, kemampuan anti-gangguan, dan keuntungan kinerja yang tinggi[18]. Untuk menggunakan sensor DHT11 yaitu dengan menghubungkan:

- VCC dengan 3.3V - 5.5V
- GND dengan power supply ground
- DOUT dengan MCU.IO (digital output)

3.2.7 Sensor Suara

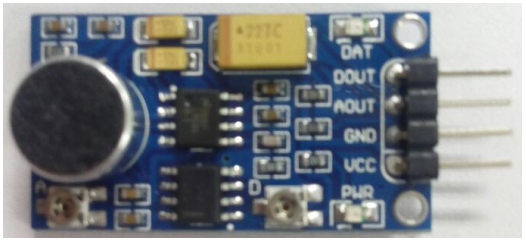
Modul sensor suara menyediakan fitur yang mudah untuk mendeteksi suara dan umumnya digunakan untuk mendeteksi intensitas suara. Modul



Gambar 3.6: Hardware sensor DHT11

ini dapat digunakan untuk keamanan, switch, dan aplikasi pemantauan. Akurasinya dapat dengan mudah disesuaikan untuk kenyamanan pemakaian. Menggunakan mikrofon sebagai input ke amplifier, peak detektor dan buffer. Ketika sensor mendeteksi suara, tegangan sinyal output dikirim ke mikrokontroler kemudian dilakukan pengolahan data yang diperlukan. Spesifikasinya yaitu: Tegangan Operasi 3.3V-5V, model output digital output (0 dan 1) dan analog output (0 hingga 255 untuk pembacaan 8 bit)[19]. Untuk menggunakan sensor suara yaitu dengan menghubungkan:

- VCC dengan 3.3V - 5.3V
- GND dengan power supply ground
- AOUT dengan MCU.IO (analog output)
- DOUT dengan MCU.IO (digital output)

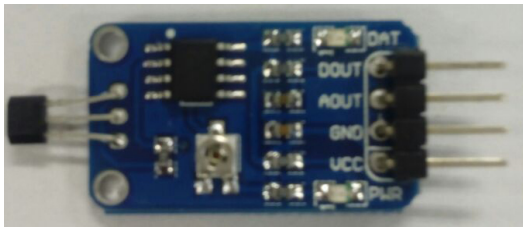


Gambar 3.7: Hardware sensor Suara

3.2.8 Sensor Hall

Sensor Hall 49E adalah sensor Hall linear yang universal dan berukuran kecil. Tingkat sinyal output 49E sebanding dengan intensitas medan magnet diterapkan untuk head yang sensitif. Untuk medan magnet nol, tegangan output dari 49E adalah setengah dari tegangan suplai. Modul ini dapat diterapkan untuk pengukuran kecepatan motor, deteksi posisi obyek, mobil pintar, blok bangunan elektronik, dll[20]. Untuk menggunakan sensor hall yaitu dengan menghubungkan:

- VCC dengan 2.3V - 5.3V
- GND dengan power supply ground
- AOUT dengan MCU.IO (analog output)
- DOUT dengan MCU.IO (digital output)

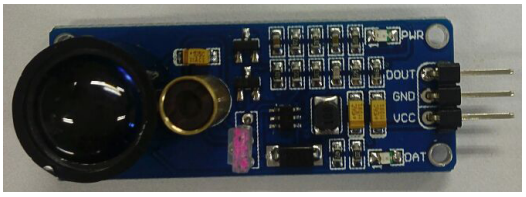


Gambar 3.8: Hardware sensor Hall

3.2.9 Sensor Laser

Sebuah sensor laser berisi pemancar dan penerima. Di pemancar terdapat tabung berosilasi dapat menghasilkan gelombang kejut di frekuensi 180KHz. Setelah diperkuat oleh transistor, gelombang kejut ini diterapkan pada tabung laser untuk pembangkitan. Pada penerima terdapat tabung penerima yang disesuaikan dengan tabung berosilasi dan dapat menerima cahaya yang dipantulkan. Sejak sensor laser mengadopsi teknologi pengolahan modulasi, tabung penerima hanya dapat menerima cahaya yang dipantulkan dalam frekuensi yang sama, sehingga efisien untuk mencegah cahaya tampak. Modul ini dapat diterapkan untuk deteksi halangan, perangkat penghitung di robot pipa intelektual, mobil ynag dapat menghindari rintangan, dll[21]. Untuk menggunakan sensor laser yaitu dengan menghubungkan:

- VCC dengan 2.5V - 5.0V
- GND dengan power supply ground
- DOUT dengan MCU.IO (digital output)

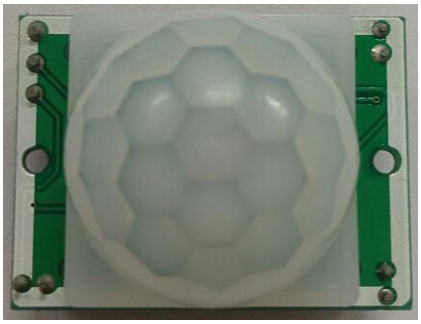


Gambar 3.9: Hardware sensor Laser

3.2.10 Sensor PIR(Passive Infra Red)

Sensor gerak PIR (Passive Infra Red) adalah sensor yang berfungsi untuk pendeteksi gerakan yang bekerja dengan cara mendeteksi adanya perbedaan/perubahan suhu sekarang dan sebelumnya. Sensor gerak menggunakan modul pir sangat simpel dan mudah diaplikasikan karena modul PIR hanya membutuhkan tegangan input DC 5V cukup efektif untuk mendeteksi gerakan hingga jarak 5 meter. Ketika tidak mendeteksi gerakan, keluaran modul adalah LOW. Dan ketika mendeteksi adanya gerakan, maka keluaran akan berubah menjadi HIGH. Adapun lebar pulsa HIGH adalah sekitar 0,5 detik. Sensitifitas Modul PIR yang mampu mendeteksi adanya gerakan pada jarak 5 meter memungkinkan kita membuat suatu alat pendeteksi gerak dengan keberhasilan lebih besar. Untuk pembacaan PIR dapat dengan menggunakan fitur GPIO. Untuk menggunakan sensor PIR yaitu dengan menghubungkan:

- VCC dengan 3.0V - 5.0V
- GND dengan power supply ground
- OUT dengan MCU.IO (digital output)



Gambar 3.10: Hardware sensor PIR

3.2.11 Sensor Ultrasonic (SRF02)

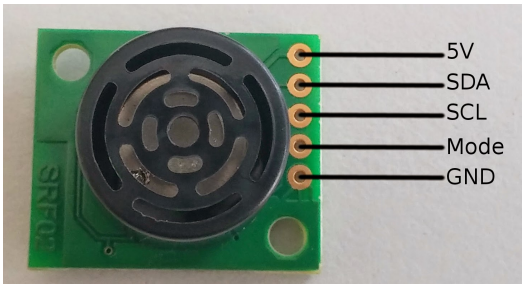
Sensor ultrasonik merupakan sensor yang bekerja dengan cara memancarkan suatu gelombang dan kemudian menghitung waktu pantulan gelombang tersebut. Gelombang ultrasonik bekerja pada frekuensi mulai 20 kHz hingga sekitar 20 MHz. Frekuensi kerja yang digunakan dalam gelombang ultrasonik bervariasi tergantung pada medium yang dilalui, mulai dari kerapatan rendah pada fasa gas, cair hingga padat[58].Secara matematis gelombang ultrasonik dapat dirumuskan sebagai:

$s = v.t/2$

Gelombang yang diserap akan dihitung oleh komparator dan diteruskan menjadi bilangan binary. Secara umum sensor ultrasonik digunakan untuk menghitung jarak dari suatu objek yang berada didepan sensor tersebut. Sehingga dengan fungsinya tersebut, sensor ultrasonik biasa digunakan pada perangkat yang membutuhkan perhitungan jarak. Contoh : smart robot, Kapal laut, kapal selam. SRF02 adalah sensor jarak dengan single tranducer. Dengan desain pcb yang lebih kecil sensor ultrasonik ini akan lebih detail dalam mendeteksi jarak[59]. Untuk mendapatkan data dari modul yaitu dengan menggunakan antarmuka I2C pada pin 2,3. Alamat default I2C yang digunakan adalah 0xE0.

3.2.12 Sensor Warna(Color)

Sensor Warna (Color Sensor) adalah sensor untuk mendetesi warna RGB. Modul sensor warna yang digunakan memiliki chip TCS3200. Terdapat



Gambar 3.11: Hardware sensor SRF02[25]

komponen utama di dalamnya yaitu photodiode dan pengkonversi arus ke frekuensi.

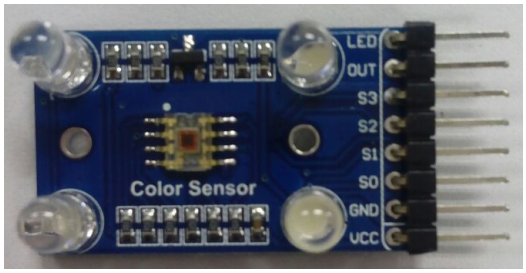
Photodiode pada IC TC3200 disusun secara array 8x8 dengan konfigurasi: 16 photodiode untuk menfilter warna merah, 16 photodiode untuk memfilter warna hijau, 16 photodiode untuk memfilter warna biru, dan 16 photodiode tanpa filter. Kelompok photodiode mana yang akan dipakai bias diatur melalui kaki selektor S2 dan S3.Photodiode akan mengeluarkan arus yang besarnya sebanding dengan kadar warna dasar cahaya yang menimpanya. Arus ini kemudian dikonversikan menjadi sinyal kotak dengan frekuensi sebanding dengan besarnya arus. Frekuensi Output ini bisa diskala dengan mengatur kaki selektor S0 dan S1. Dengan demikian, program yang kita perlukan untuk mendapatkan komposisi RGB adalah program penghitung frekuensi.

TCS3200D berisi empat jenis filter: filter merah, filter hijau, filter biru dan jelas tanpa filter. Ketika sensor diterangi oleh sinar cahaya, jenis filter (biru, hijau, merah, atau yang jelas) yang digunakan oleh perangkat dapat dipilih oleh dua input logika, S2 dan S3.

Output TCS3200D gelombang persegi (50% duty cycle) dengan frekuensi yang sesuai dengan intensitas cahaya dan warna, dan frekuensi berbanding lurus dengan intensitas cahaya. Frekuensi output khas TCS3200D adalah dalam range 2Hz sampai 500KHz. Pengguna dapat mengontrol nilai frekuensi 100%, 20%, dan 2% dengan dua output yang diprogram, S0 dan S1, seperti ditunjukan tabel sebagai berikut.

Aplikasi sensor warna ini dapat digunakan untuk penyortiran warna, induksi cahaya lingkungan dan kalibrasi, pembacaaan srip test, tes pencocokan warna, dll. Spesifikasi interfacenya ditunjukan oleh tabel sebagai berikut[26]. Untuk menggunakan sensor warna yaitu dengan menghubungkan:

- VCC dengan 2.7V - 5.5V
- GND dengan power supply ground
- LED dengan MCU.IO (controlling the 4 white LEDs)
- OUT dengan MCU.IO (RGB color output frequency)
- S0/S1 dengan MCU.IO (Output frequency scaling selection inputs)
- S2/S3 dengan MCU.IO (Photodiode type selection inputs)



Gambar 3.12: Hardware sensor Warna

3.2.13 Sensor kemiringan (Tilt Sensor)

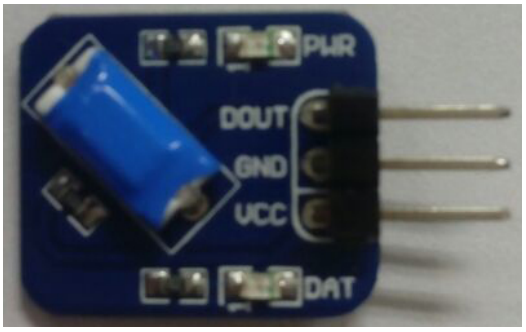
Sensor kemiringan (tilt sensor) atau vibrating sensor Waveshare yang digunakan ini pada hakekatnya adalah sebuah aplikasi switch mercury. Dalam sebuah kondisi tilting atau vibrating, sensor dapat membuat switch membuka atau menutup. Modul ini dapat diaplikasikan untuk mendeteksi getaran(vibrating), alarm anti pencuri, smart car, electronic building block, dsb. Spesifikasinya adalah sebagai berikut:

1. Tegangan Operasi: 3-5.5 V
2. Type Output: TTL level output
3. Dimension: 2.10 mm x 13.0 mm

Adapun spesifikasi untuk interfacenya sebagai berikut:

Indikator sinyal akan menyala, ketika sensor sedang bergetar atau dalam keadaan miring. Dan akan mati ketika sensor diletakkan datar[60]. Untuk menggunakan sensor tilt yaitu dengan menghubungkan:

- VCC dengan 3.0V - 5.5V
- GND dengan power supply ground
- DOUT dengan MCU.IO (digital output)



Gambar 3.13: Hardware sensor Tilt

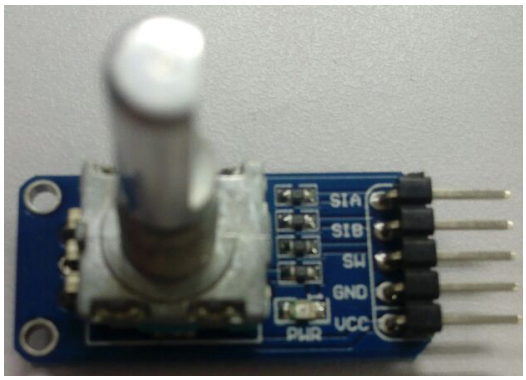
3.2.14 Sensor Rotasi (Rotation Sensor)

Sensor rotasi Waveshare yang digunakan ini memliki prinsip kerja menerjemahkan perpindahan rotary menjadi sinyal pulsa digital yang berurutan. Perpindahan dapat diperoleh dengan menghitung jumlah pulsa yang dikeluarkan dalam perjalanan searah jarum jam atau rotasi berlawanan arah jarum jam. Membandingkan dengan potensiometer, meja rotary tidak memiliki batas untuk menghitung. Jadi, ada tombol reset yang tersedia untuk me-reset counter untuk keadaan awal, restart pada 0. Modul ini dapat diaplikasikan untuk solusi lokasi dalam kontrol industri. Fitur yang dimiliki adalah sebagai berikut:

1. Voltage operasi: 3-5.3 V
2. Banyak pulsa per putaran: 15 Pulsa
3. Dimensi: 32 x 15 mm

Sedangkan spesifikasi untuk interfacenya sebagai berikut:
Ada tiga tindakan rotary encoder dalam sensor: rotasi searah jarum jam, rotasi berlawanan arah jarum jam, dan reset tekan tombol. Serial output dari modul bervariasi sesuai dengan tindakan yang berbeda[28]. Untuk menggunakan sensor rotasi yaitu dengan menghubungkan:

- VCC dengan 3.3V - 5.3V
- GND dengan power supply ground
- SW dengan MCU.IO (encoder button status)
- SIB dengan MCU.IO (output status)
- SIA dengan MCU.IO (output status)



Gambar 3.14: Hardware sensor Rotasi

3.2.15 Sensor Reflektif Infrared

Sensor inframerah Waveshare berisi dua bagian: pemancar inframerah dan penerima inframerah. Pemancar inframerah terdiri dari sebuah array LED inframerah sebagai luminophor, dan sebuah hubungan PN yang terbuat dari bahan khusus dengan efisiensi radiasi inframerah tinggi, yang biasanya adalah GaAs. Ketika arus diberikan ke persimpangan PN oleh tegangan bias maju, dapat membangkitkan sumber cahaya inframerah dengan berbagai pusat panjang gelombang 830nm-950nm. Kekuatan cahaya inframerah yang dibangkitkan adalah sebanding dengan arus disuntikkan. Namun, dalam kasus ini bahwa kelebihan arus lebih dari nilai maksimum, kekuatan cahaya inframerah bisa menurun dengan meningkatnya arus. Penerima inframerah adalah perangkat semikonduktor untuk menerjemahkan sinyal cahaya inframerah menjadi sinyal listrik. Dan komponen inti darinya adalah sebuah persimpangan PN yang terbuat dari bahan khusus.Persimpangan PN

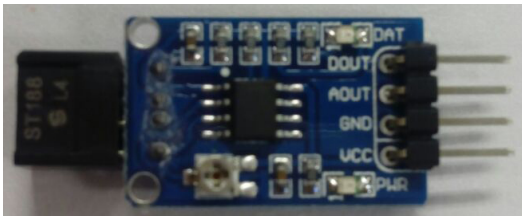
tersebut memiliki struktur yang berbeda dari dioda umum, memungkinkan lebih banyak cahaya inframerah yang akan diterima. Semakin meningkat intensitas cahaya inframerah, lebih banyak arus dapat dihasilkan. Fiturnya adalah sebagai berikut:

1. Jenis sensor: ST188
2. Chip komparator tegangan:LM393
3. Tegangan Operasi:3-5.3 V
4. Dimensi:30.2 x 11.9 mm

Modul ini dapat digunakan untuk robot cerdas, mobil penghindar bencana, perangkat penghitung dalam saluran pipa, perangkat pengikut jejak garis hitam dan putih, dsb. Adapun interfacennya sebagai berikut:

Hasil yang terdeteksi dapat diperiksa oleh indikator sinyal pada modul. Indikator sinyal akan menyala, ketika sensor dekat dengan penghalang. Dan akan mati, ketika sensor jauh dari penghalang. Juga, Anda dapat menemukan bahwa output serial berubah seiring dengan jarak dari sensor ke penghalang[61]. Untuk menggunakan sensor reflektif infrared yaitu dengan menghubungkan:

- VCC dengan 3.0V - 5.3V
- GND dengan power supply ground
- AOUT dengan MCU.IO (analog output)
- DOUT dengan MCU.IO (digital output)



Gambar 3.15: Hardware sensor Reflektif Infrared

3.2.16 Sensor Moisture

Modul ini merupakan aplikasi dari amplifikasi arus oleh sebuah transistor. Ketika air di dalam tanah cukup memadai untuk mengalirkan arus antara ground dan catu daya positif, sejumlah arus dihasilkan antara basis dan emitor. Sementara itu , arus listrik yang dihasilkan dalam faktor amplifikasi tertentu antara kolektor dan emitor, dan diterapkan pada tahanan di emitor untuk menghasilkan tegangan. Kemudian, tegangan ini akan dikumpulkan oleh AD converter. Modul ini dapat digunakan untuk Sistem air otomatis, pendeteksi kelembaban tanah, sistem irigasi otomatis, dsb. Fitur yang terdapat di dalamnya sebagai berikut:

1. Tegangan Operasi: 2-5 V
2. Type Output: Analog Output
3. Kedalaman yang dapat dideteksi:38mm
4. Dimensi:20 x 51 mm

Cara menggunakannya adalah masukan sensor ke dalam tanah lalu diberikan air sedikit demi sedikit, dan akan didapatkan perubahan output dari saya haus!ke cukup![30]. Untuk menggunakan sensor Moisture yaitu dengan menghubungkan:

- VCC dengan 2.0V - 5.0V
- GND dengan power supply ground
- AOUT dengan MCU.IO (analog output)

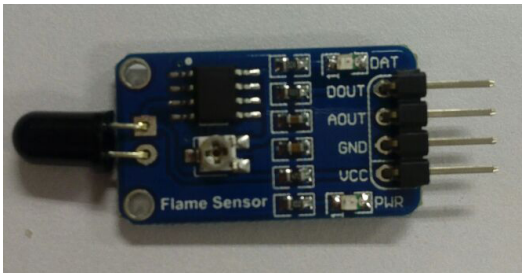
Gambar 3.16: Hardware sensor Moisture

3.2.17 Sensor Api (Flame Sensor)

Sensor Api adalah sensor yang dapat mendeteksi adanya titik api. Sensor Api Waveshare memiliki kesensitifan terhadap spektrum api, memiliki komparator tegangan LM 393, sensitifitas yang dapat diatur. Spesifikasinya adalah sebagai berikut:

Aplikasinya dapat digunakan untuk deteksi kebakaran, pemadam kebakaran robot dan alarm kebakaran. Interfacenya ditampilkan pada tabel sebagai berikut[62]. Untuk menggunakan sensor Api yaitu dengan menghubungkan:

- VCC dengan 3.3V - 5.3V
- GND dengan power supply ground
- AOUT dengan MCU.IO (analog output)
- DOUT dengan MCU.IO (digital output)



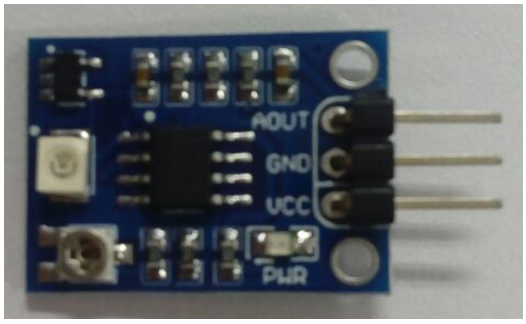
Gambar 3.17: Hardware sensor API

3.2.18 Sensor Ultraviolet

Sensor Ultraviolet adalah sensor untuk mendeteksi sinar ultraviolet. Modul sensor Waveshare yang akan digunakan terdapat sebuah sensor ultraviolet, GUVA, yang merupakan perangkat ideal untuk mendeteksi jumlah sinar UV tanpa filter panjang gelombang, karena hanya sinar UV sensitif. Dengan kata lain, panjang gelombang 365nm (UV-A) dan 320nm (UV-B) adalah batas cut-off dari GUVA. Fitur dari modul sensor adalah sebagai berikut.

Aplikasi dari modul sensor ini adalah sistem pendeteksi sinar Ultraviolet, perangkat monitor UV luar, lampu sterilisasi, dsb. Adapun interfacenya sebagai berikut[63]. Untuk menggunakan sensor UV yaitu dengan menghubungkan:

- VCC dengan 3.3V - 5.0V
- GND dengan power supply ground
- AOUT dengan MCU.IO (analog output)



Gambar 3.18: Hardware sensor Ultraviolet

3.2.19 Penggerak Motor

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. Motor DC sederhana memiliki dua kutub medan: kutub utara dan kutub selatan. Garis magnetik energi membesar melintasi ruang terbuka diantara kutub-kutub dari utara ke selatan. Untuk motor yang lebih besar atau lebih komplek terdapat satu atau lebih elektromagnet. Current Elektromagnet atau Dinamo. Dinamo yang berbentuk silinder, dihubungkan ke as penggerak untuk menggerakan beban. Untuk kasus motor DC yang kecil, dinamo berputar dalam medan magnet yang dibentuk oleh kutub-kutub, sampai kutub utara dan selatan magnet berganti lokasi. Commutator. Komponen ini terutama ditemukan dalam motor DC. Kegunaannya adalah untuk transmisi arus antara dinamo dan sumber daya.

3.3 Implementasi dan Pengambilan Data dengan Pemrograman Phython dan C

3.3.1 Membaca Input Analog dengan Arduino

```
int sensorPin = A0; // mendeklarasikan input pin analog
int sensorValue = 0; // variable awal sensor

void setup()
{
Serial.begin(9600); // seting komunikasi serial
}

void loop()
{
// membaca data analog dari sensor
sensorValue = analogRead(sensorPin);
Serial.println(sensorValue); // menampilkan nilai sensor secara serial
delay(1); // delay
}
```

3.3.2 Membaca Input Digital dengan Arduino

```
int sensorPin = 2; // mendeklarasikan input pin digital
int sensorValue; // variable sensor

void setup()
{
Serial.begin(9600); // seting komunikasi serial
// medeklarasikan sensorPin sebagai input
pinMode(sensorPin, INPUT);
}

void loop()
{
// membaca data digital dari sensor
sensorValue = digitalRead(sensorPin);
Serial.println(sensorValue); // menampilkan nilai sensor secara serial
```

```
delay(1); // delay
}
```

3.3.3 Membaca Input secara USART dengan Arduino

```
void setup()
{
// menginisialisasi port serial yang digunakan
Serial.begin(9600);
Serial1.begin(9600);
}

void loop()
{
// membaca data dari port 1, dan mengirim ke port 0
if (Serial1.available()) // apabila ada data pada port 1
{
int inByte = Serial1.read();
Serial.write(inByte); // mengirim data ke port 0
}
}
```

3.3.4 Membaca Input Color Sensor dengan Arduino

```
#include <TimerOne.h>
#define S0     6   // Please notice the Pin's define
#define S1     5
#define S2     4
#define S3     3
#define OUT    2

int   g_count = 0;     // count the frequecy
int   g_array[3];      // store the RGB value
int   g_flag = 0;      // filter of RGB queue
float g_SF[3];         // save the RGB Scale factor
```

```
// Init TSC230 and setting Frequency.
void TSC_Init()
{
  pinMode(S0, OUTPUT);
  pinMode(S1, OUTPUT);
  pinMode(S2, OUTPUT);
  pinMode(S3, OUTPUT);
  pinMode(OUT, INPUT);
  digitalWrite(S0, LOW);// OUTPUT FREQUENCY SCALING
     2%
  digitalWrite(S1, HIGH);
}

// Select the filter color
void TSC_FilterColor(int Level01, int Level02)
{
  if(Level01 != 0)
    Level01 = HIGH;

  if(Level02 != 0)
    Level02 = HIGH;

  digitalWrite(S2, Level01);
  digitalWrite(S3, Level02);
}
void TSC_Count()
{
  g_count ++ ;
}
void TSC_Callback()
{
  switch(g_flag)
  {
    case 0:
      Serial.println("->WB Start");
      TSC_WB(LOW, LOW);              //Filter
         without Red
      break;
```

```
    case 1:
      Serial.print("->Frequency R=");
      Serial.println(g_count);
      g_array[0] = g_count;
      TSC_WB(HIGH, HIGH);            //Filter
         without Green
      break;
    case 2:
      Serial.print("->Frequency G=");
      Serial.println(g_count);
      g_array[1] = g_count;
      TSC_WB(LOW, HIGH);             //Filter
         without Blue
      break;
    case 3:
      Serial.print("->Frequency B=");
      Serial.println(g_count);
      Serial.println("->WB End");
      g_array[2] = g_count;
      TSC_WB(HIGH, LOW);             //Clear(no
         filter)
      break;
    default:
      g_count = 0;
      break;
  }
}

 void TSC_WB(int Level0, int Level1)  //White
    Balance
 {
   g_count = 0;
   g_flag ++;
   TSC_FilterColor(Level0, Level1);
   Timer1.setPeriod(1000000);
 }
 void setup()
 {
```

```
  TSC_Init();
  Serial.begin(9600);
  Timer1.initialize();             // defaulte is 1
     s
  Timer1.attachInterrupt(TSC_Callback);
  attachInterrupt(0, TSC_Count, RISING);
  delay(4000);
  for(int i=0; i<3; i++)
  Serial.println(g_array[i]);
  g_SF[0] = 255.0/ g_array[0];     //R Scale factor
  g_SF[1] = 255.0/ g_array[1] ;    //G Scale factor
  g_SF[2] = 255.0/ g_array[2] ;    //B Scale factor
  Serial.println(g_SF[0]);
  Serial.println(g_SF[1]);
  Serial.println(g_SF[2]);
}
void loop()
{
  g_flag = 0;
  for(int i=0; i<3; i++)
  Serial.println(int(g_array[i] * g_SF[i]));
  delay(4000);
}
```

3.3.5 Membaca Input Rotation Sensor dengan Arduino

```
#define PinA 2
#define PinB 3

unsigned long time = 0;
long count = 0;
long num = 0;


void setup()
{
  Serial.begin(9600);
```

```
  pinMode(PinA,INPUT);
  pinMode(PinB,INPUT);

  attachInterrupt(0, blinkA, LOW);
  attachInterrupt(1, blinkB, LOW);

  time = millis();
}

void loop()
{
  while (num != count)
  {
    num = count;
    Serial.println(num);
  }
}

void blinkA()
{
  if ((millis() - time) > 3)
        count ++;
  time = millis();
}

void blinkB()
{
  if ((millis() - time) > 3)
        count --;
  time = millis();
}
```

3.3.6 Membaca Input secara I2C dengan Arduino

```
#include <Wire.h>
#define ADDRESS 0x60 // mendefinisikan alamat yang digunakan

void setup()
```

```
{
Wire.begin(); // menyambungkan ke I2C
Serial.begin(9600); // seting komunikasi serial
}

void loop()
{
byte highByte;
byte lowByte;

Wire.beginTransmission(ADDRESS); // memulai komunikasi sesuai alamat
Wire.write(2); // Mengirim register yang akan dibaca
Wire.endTransmission();

Wire.requestFrom(ADDRESS, 2); // meminta high byte
while(Wire.available() < 2); // Apabila terdapat byte untuk diterima
//membaca byte sebagai integer
highByte = Wire.read();
lowByte = Wire.read();
int bearing = ((highByte<<8)+lowByte)/10;

Serial.println(bearing); // menampilkan nilai sensor secara serial
delay(100);
}
```

3.3.7 Membaca Input Sensor Air Atlas dengan Arduino

```
//Include the software serial library
#include <SoftwareSerial.h>

//Arduino pin 7 to control pin pinX
int pinX = 4;
//Arduino pin 6 to control pin pinY
int pinY = 5;
```

```
//A 20 byte character array to hold incoming data
   from a pc/mac/other
char computerdata[20];
//A 30 byte character array to hold incoming data
   from the sensors
char sensordata[30];
//We need to know how many characters bytes have
   been received
//We need to know how many characters bytes have
   been received
byte computer_bytes_received = 0;
//We need to know how many characters bytes have
   been received
byte sensor_bytes_received = 0;

unsigned long previousMillis = 0;
const long interval = 2000;

float temp;
int ubah;
String dataString = "";

String ORP, EC, PH, DO, suhu;
char kirimsensor[100];

void setup() {
  //Set the digital pin as output.
  pinMode(pinY, OUTPUT);
  //Set the digital pin as output.
  pinMode(pinX, OUTPUT);
  pinMode(A0,INPUT);
  Serial.begin(9600);
  Serial1.begin(9600);
  Serial2.begin(9600);
  ubah = 0;
}

void serialEvent() {
```

```
  computer_bytes_received = Serial.readBytesUntil
     (13, computerdata, 20);
  computerdata[computer_bytes_received] = 0;
}

void loop()
{
  unsigned long currentMillis = millis();
  temp = read_temp();

  open_channel();
  if (Serial1.available() > 0)
  {
    sensor_bytes_received = Serial1.readBytesUntil
       (13, sensordata, 30);
    sensordata[sensor_bytes_received] = 0;

    if (ubah == 0 )
    {
      ORP = "";
      ORP = String(sensordata);
    }
    else if (ubah == 1)
    {
      DO = "";
      DO = String(sensordata);
    }
    else if (ubah == 2)
    {
      EC = "";
      EC = String(sensordata);
    }
    else if (ubah == 3)
    {
      suhu = "";
      suhu = String(temp);
      PH = "";
      PH = String(sensordata);
```

```
    }
    ubah++;
    if (ubah > 3)
      ubah = 0;

  }

  if (currentMillis - previousMillis >= interval) {
    // save the last time you blinked the LED
    previousMillis = currentMillis;
    dataString = "#" + ORP + "," + PH + "," + EC +
       "," + DO + "," + suhu + "*";
    Serial2.println(dataString);
  }
}


void open_channel()
{
  switch (ubah)
  {   //Looking to see what channel to open
    //If channel==0 then we open channel 0
    case 0:
      digitalWrite(pinX, LOW);
      digitalWrite(pinY, LOW);
      break;
    case 1:
      digitalWrite(pinX, HIGH);
      digitalWrite(pinY, LOW);
      break;
    case 2:
      digitalWrite(pinX, LOW);
      digitalWrite(pinY, HIGH);
      break;
    case 3:
      digitalWrite(pinX, HIGH);
      digitalWrite(pinY, HIGH);
      break;
```

```
  }
}

float read_temp(void)
{
  float v_out;
  float temp;
  v_out = analogRead(0);
  v_out *= 0.0048;
  v_out *= 1000;
  temp = (0.0512 * v_out) - 20.5128;
  return temp;
}
```

3.3.8 Membaca Input secara GPIO dengan Raspiberry Pi

```
import RPi.GPIO as GPIO;
import time;

GPIO.setmode(GPIO.BCM);
GPIO.setwarnings(False);
GPIO.setup(4,GPIO.IN);

previous = 0;
try:
print "Sensor Sedang bekerja!!";
print "Mulai merasakan..";
while True:
if GPIO.input(4)==1 and previous==0:
print "Pergerakan Terdeteksi";
previous=1;
elif GPIO.input(4)==0 and previous==1:
print "Siap";
previous=0;
time.sleep(0.01);
except KeyboardInterrupt:
print "Selesai";
```

```
GPIO.cleanup();
```

3.3.9 Membaca Input secara I2C dengan Raspiberry Pi

Langkah-langkah mengaktifkan I2C pada Raspberry pi

1. Mendapatkan I2C tools yakni dengan mendownload dengan perintah sebagai berikut pada kernel `sudo apt-get install i2c-tools`
2. Mengaktifkan komponen-komponen pendukung I2C dalam kernel menggunakan utilitas raspi-config `sudo raspi-config`
3. Dalam raspi-config, melangkah ke pilihan lebih lanjut dengan mengaktifkan I2C. Hal ini akan membangun I2C pada Raspberry pi secara otomatis kemudian reboot Raspberry Pi `sudo reboot`
4. Menguji apakah I2C telah bekerja dengan perintah berikut `sudo i2cdetect -y 1` (atau sudo i2cdetect y 0 pada model Raspberry pi yang lama)

Perintah ini akan menunjukkan perangkat mana yang telah dialamatkan pada bus I2C Jika langkah-langkah di atas tidak berhasil mengaktifkan I2C, maka dilakukan pengecekan apakah segalanya telah diatur dengan benar melalui langkah-langkah berikut:

- Mengedit file-file modul

  ```
  sudo nano/etc/modules
  ```

- Menambah baris-baris berikut pada bagian akhir jika tidak ada

  ```
  i2c-bcm2708
  I2c-dev
  ```

- Simpan file-file tersebut
- Mengedit file blacklist

  ```
  sudo nano /etc/modprobe.d/raspi-blacklist.conf
  ```

- Menghilangkan I2C dari blacklist dan menjadikannya berupa perintah komentar dengan menaruh simbol# di depan baris

```
#blacklist i2c-bc,2708
```

Kernal lama 3.18 membutuhkan untuk mengaktifkan I2C dalam pohon perangkat

```
sudo nano /boot/config.tx
```

- Menambahkan baris-baris pada akhir file jika tidak ditemukan

```
dtparam=i2c1=on (or dtparam=i2c0=on on older
   models)
dtparam=i2c_arm=on
```

[33] Berikut ini adalah contoh program menggunakan komunikasi I2C Raspberry Pi dengan menggunakan sensor Compass 03

```
import smbus
import time
bus = smbus.SMBus(0)
address = 0x60

def bearing255():
        bear = bus.read_byte_data(address, 1)
        return bear

def bearing3599():
        bear1 = bus.read_byte_data(address, 2)
        bear2 = bus.read_byte_data(address, 3)
        bear = (bear1 << glasses emotikon + bear2
        bear = bear/10.0
        return bear

while True:
        #this returns the value to 1 decimal place
           in degrees.
        bearing = bearing3599()
        #this returns the value as a byte between 0
           and 255.
        bear255 = bearing255()
        print bearing
```

```
        print bear255
        time.sleep(1)
```

3.3.10 Membaca Input secara USART dengan Raspiberry Pi

```
import serial
import sys
import string

# Baudrate dapat diganti sesuai input
ser = serial.Serial('/dev/S0', 57600)
while True :
    try:
        # Read data incoming on the serial line
        data=ser.readline()
        print data
    except:
        print "Unexpected error:", sys.exc_info()
        sys.exit()
```

Bab 4

Komunikasi Data

sectionInternet Protokol IP adalah standard protokol dengan nomer STD 5. Standar ini juga termasuk untuk ICMP, dan IGMP. Spesifikasi untuk IP dapat dilihat di RFC 791, 950, 919, dan 992 dengan update pada RFC 2474. IP juga termasuk dalam protokol internetworking.

Adanya IP Address merupakan konsekuensi dari penerapan Internet Protocol untuk mengintegrasikan jaringan komputer Internet di dunia. Seluruh host (komputer) yang terhubung ke Internet dan ingin berkomunikasi memakai TCP/IP harus memiliki IP Address sebagai alat pengenal host pada network. Secara logika, Internet merupakan suatu network besar yang terdiri dari berbagai sub network yang terintegrasi. Oleh karena itu, suatu IP Address harus bersifat unik untuk seluruh dunia. Tidak boleh ada satu IP Address yang sama dipakai oleh dua host yang berbeda. Untuk itu, penggunaan IP Address di seluruh dunia dikoordinasi oleh lembaga sentral Internet yang di kenal dengan IANA - salah satunya adalah Network Information Center (NIC).

4.1 Perangkat Jaringan

Dalam Komunikasi Jaringan banyak sekali perangkat yang dapat digunakan. Dalam buku ini akan dijelaskan beberapa perangkat yang digunakan untuk komunikasi data, antara lain :

1. Switch.

   Switch adalah komponen jaringan yang digunakan untuk menghubungkan beberapa HUB untuk membentuk jaringan yang lebih besar atau menghubungkan

komputer2 yang mempunyai kebutuhan bandwidth yang besar. Switch memberikan unjuk kerja yang jauh lebih baik dari pada HUB dengan harga yang sama atau sedikit lebih mahal.

Switch terbagi dalam 2 tipe utama: switch layer 2 dan layer 3. Switch layer 2 beroperasi pada layer data-link model OSI dan berdsarkan terknologi bridging. Switch tipe ini membangun koneksi logika antar port berdasarkan pada alamat MAC. Switch layer 2 dapat digunakan untuk memecah jaringan yang sedang berjalan ke dalam collision domain yang lebih kecil untuk meningkatkan unjuk kerja.

Switch layer 3 beroperasi pada layer 3 dari model OSI dasar teknologi routing. Switch tipe ini membangun koneksi logika antar port berdasarkan alamat jaringan. Switch ini dapat digunakan untuk menghubungkan jaringan jaringan yang berbeda di dalam suatu internetwork. switch layer 3 Kadang-kadang di sebut Switch routing atau switch multilayer.

2. Router

Router merupakan sebuah perangkat jaringan yang bekerja pada OSI Layer 3, Network Layer atau perangkat komputer yang tugasnya menyampaikan paket data melewati jaringan internet hingga sampai ketujuannya. Router adalah sebuah alat untuk mengirimkan paket data melalui jaringan atau internet untuk dapat menuju tujuannya, proses tersebut dinamakan routing.

Router memiliki fungsi utama untuk membagi atau mendistribusikan IP address, baik itu secara statis ataupun DHCP atau Dynamic Host Configuration Procotol kepada semua komputer yang terhubung ke router tersebut. Dengan adanya IP address yang unik yang dibagikan router tersebut kepada setiap komputer dapat memungkinan setiap komputer untuk saling terhubung serta melakukan komunikasi, baik itu pada LAN atau internet.

Gambar 1. Contoh Bentuk sebuah router

3. Gateway

   Gateway adalah sebuah perangkat yang dipakai untuk menghubungkan satu jaringan komputer dengan satu ataupun lebih jaringan komputer yang memakai protokol komunikasi yang berbeda sehingga informasi dari satu jaringan komputer bisa diberikan kepada jaringan komputer lain yang protokolnya tidak sama atau berbeda.

4. GSM

   GSM adalah sebuah teknologi komunikasi selular yang bersifat digital. jaringan komunikasi ini bekerja dengan mengirimkan data berdasarkan slot waktu yang membentuk jalur pada setiap sambungan dengan rentang waktu yang sangat cepat. Metode pengiriman data pada GSM disebut TDMA (Time Division Multiple Access), yang mana menggunakan waktu sebagai perantara akses.

5. Wifi

   Wifi adalah sebuah teknologi terkenal yang memanfaatkan peralatan elektronik untuk bertukar data secara nirkabel (menggunakan gelombang radio) melalui sebuah jaringan komputer, termasuk koneksi Internet berkecepatan tinggi

4.1.0.1 Format IP

Format dalam IP Address yang kita gunakan disini adalah IPV4. Dimana dalam sebuah Alamat IP versi 4 (sering disebut dengan Alamat IPv4) adalah sebuah jenis pengalamatan jaringan yang digunakan di dalam protokol jaringan TCP/IP yang menggunakan protokol IP versi 4. Panjang totalnya adalah 32-bit, dan secara teoritis dapat mengalamati hingga 4 miliar host komputer atau lebih tepatnya 4.294.967.296 host. Jumlah host tersebut didapatkan dari 256 (didapatkan dari 8 bit) dipangkat 4(karena terdapat 4 oktet) sehingga nilai maksimal dari alamt IP versi 4 tersebut adalah 255.255.255.255 dimana nilai dihitung dari nol sehingga nilai host yang dapat ditampung adalah 256x256x256x256 = 4.294.967.296 host. Contoh dari sebuah IP Versi 4 adalah 192.168.2.1.

Alamat IP yang dimiliki oleh sebuah host dibagi dengan menggunakan subnet mask jaringan kedalam dua bagian, yaitu :

1. Network Identifier (Net Id) adalah alamat jaringan yang digunakan untuk mengidentifikasi alamat jaringan di mana host berada

2. Host Identifier (HostID) alamat host yang digunakan khusus untuk mengidentifikasikan alamat host (dapat berupa workstation, server atau sistem lainnya yang berbasis teknologi TCP/IP) di dalam jaringan

4.1.0.2 Kelas IP

alamat IP versi 4 dibagi ke dalam beberapa kelas, dilihat dari oktet pertamanya, seperti terlihat pada tabel. Sebenarnya yang menjadi pembeda kelas IP versi 4 adalah pola biner yang terdapat dalam oktet pertama (utamanya adalah bit-bit awal/high-order bit), tapi untuk lebih mudah mengingatnya, akan lebih cepat diingat dengan menggunakan representasi desimal.

Kelas IP	Oktet Pertama Desimal	Oktet Pertama Biner
Kelas A	1 - 127	0xxx.xxxx
Kelas B	128 - 191	10xx.xxxx
Kelas C	192 - 223	110x.xxxx
Kelas D	224 - 239	1110.xxxx
Kelas E	240 - 255	1111.xxxx

Gambar 2. Tabel Oktet Kelas IP

4.1.1 Setting IP di Raspberry Pi

Untuk Menghubungkan Raspberry ke perangkat jaringan maka dibutuhkan pengaturan IP Address, adapun langkah pengaturan IP Address di raspberry bisa kita lakukan konfigurasi secara static atau dinamic, tergantung dari projek yang dibuat. Adapun langkah mengkonfigurasi IP Address di Raspberry adalah sebagai berikut: kita ingin konfigurasi jaringan secara static maka ubah file dan isi dengan konfigurasi seperti dibawah ini.

1. Buka file dengan perintah sudo nano /etc/network/interfaces.

```
Auto lo
iface lo inet loopback
Auto eth0
iface eth0 inet dhcp
```

2. Apabila ingin Konfigurasi dengan DHCP maka tidak perlu diubah isi dari file diatas.

3. Apabila konfigurasi jaringan secara static maka ubah isi file seperti berikut.

```
Auto eth0
iface eth0 inet statis
        Alamat 192.168.2.30
        netmask 255.255.255.0
        Jaringan 192.168.2.0
        siaran 192.168.2.255
        Gerbang 192.168.2.1
```

4.2 Pemrograman Socket di python

Dalam bahasa pemrograman socket apapun secara universal sama yaitu socket sebagai saluran antara dua aplikasi yang dapat berkomunikasi satu sama lain baik di Python, Perl, Ruby, Scheme, atau bahasa lain (bahasa yang memiliki antarmuka soket), secara universal socket sama dimana socket digunakan sebagai saluran antara dua aplikasi yang dapat berkomunikasi satu sama lain (baik secara lokal pada mesin tunggal atau antara dua mesin di lokasi yang terpisah). Bedanya dengan pemrograman soket dalam bahasa seperti Python adalah di kelas pembantu dan metode yang dapat menyederhanakan pemrograman soket.

4.2.1 Pemrograman Socket API

Contoh berikut mengilustrasikan berinteraksi dengan interpreter Python. Di sini, saya menggunakan metode kelas socket gethostbyname untuk menyelesaikan nama domain berkualifikasi lengkap (www.ibm.com) ke string alamat quad-dotted IP Address ('129.42.19.99'):

```
[camus]$ <strong>python</strong>
Python 2.4 (#1, Feb 20 2005, 11:25:45)
[GCC 3.2.2 20030222 (Red Hat Linux 3.2.2-5)] on linux2
Type "help", "copyright", "credits" or "license" for more
   information.
>>> <strong>import socket</strong>
>>> <strong>socket.gethostbyname('www.ibm.com')</strong>
'129.42.19.99'
>>>
```

Gambar 4.1: Pemrograman Socket API

Setelah modul socket diimpor, saya memanggil metode kelas gethostbyname untuk menyelesaikan nama domain ke alamat IP.

4.2.2 Membuat dan destroy Socket

Untuk membuat socket baru, Anda dapat menggunakan metode socket dari kelas socket. Metode socket mirip dengan BSD API.

```
streamSock = <strong>socket.socket</strong>( socket.AF_INET, socket.SOCK_STREAM )

dgramSock = <strong>socket.socket</strong>( socket.AF_INET, socket.SOCK_DGRAM )
```

Gambar 4.2: Membuat socket baru

Dalam setiap kasus, objek socket dikembalikan. The AF_INET simbol - argumen satu menunjukkan bahwa Anda meminta sebuah Internet Protocol

(IP) socket, khususnya IPv4. Argumen kedua adalah jenis protokol transport (SOCK_STREAM untuk soket TCP dan SOCK_DGRAM untuk soket UDP). Jika sistem operasi yang mendasari Anda mendukung IPv6, Anda juga dapat menentukan socket.AF_INET6 untuk membuat soket IPv6.

Untuk memutus sambugan socket, kita dapat menggunakan close method
streamSock.close()
Terakhir, kita dapat menghapus socket dengan statemen del:
del streamSock
Pernyataan ini secara permanen menghapus objek socket. Mencoba untuk referensi soket akan menghasilkan kesalahan.

4.2.3 Penanganan alamat IP

Alamat endpoint untuk socket adalah tuple yang terdiri dari alamat interface dan nomor port. Karena Python dapat mewakili tupel dengan mudah, alamat dan port diwakili seperti itu. Berikut menggambarkan titik akhir untuk alamat antarmuka 192.168.1.1 dan port 80:
(192.168.1.1, 80)
Anda juga dapat menggunakan nama domain berkualifikasi lengkap, seperti:
(www.ibm.com, 80)

4.2.4 Server Socket

Server soket biasanya mengekspos layanan pada jaringan. Karena server dan client soket dibuat dengan cara yang berbeda, saya mendiskusikannya secara independen.
Setelah Anda membuat soket, Anda menggunakan metode mengikat untuk mengikat alamat itu, metode listed dalam menempatkanya di listen state, dan akhirnya metode diterima untuk menerima sambungan klien baru. Hal ini ditunjukkan di bawah ini:

Untuk server, alamat (”, 2525) menunjukkan wildcard yang digunakan untuk alamat antarmuka (”), memungkinkan menerima koneksi masuk dari antarmuka pada host. Anda juga mengikat nomor port 2525.
Perhatikan di sini metode menerima kembali tidak hanya objek socket baru

```
sock = <strong>socket.socket</strong>( socket.AF_INET, socket.SOCK_STREAM )
sock.<strong>bind</strong>( ('', 2525) )
sock.<strong>listen</strong>( 5 )
newsock, (remhost, remport) = sock.<strong>accept</strong>()
```

Gambar 4.3: Server soket

yang mewakili koneksi client (newsock) tetapi juga tupel alamat (alamat terpencil dan nomor port dari ujung rekan dari soket). modul SocketServer Python dapat menyederhanakan proses ini lebih jauh, seperti yang ditunjukkan program di atas.

4.2.5 Client socket

Mekanisme untuk menciptakan dan menghubungkan soket klien mirip dengan setup soket Server. Setelah menciptakan socket, alamat diperlukan - lokal tidak mengikat soket (seperti yang terjadi dengan server) melainkan untuk mengidentifikasi mana soket yang akan dituju. Katakanlah ada server pada host dengan alamat IP interface dari '192.168.1.1' dan port 2525. Kode berikut menciptakan socket baru dan menghubungkan ke server: Apa yang

```
sock = <strong>socket.socket</strong>( socket.AF_INET, socket.SOCK_STREAM )
sock.<strong>connect</strong>( ('192.168.1.1', 2525) )
```

Gambar 4.4: Client socket

berbeda di sini adalah bahwa meskipun saya telah menggunakan metode connect, tidak ada hubungan nyata antara klien dan server. The connect di sini adalah penyederhanaan untuk nanti I / O. Biasanya di soket datagram, Anda harus memberikan informasi tujuan dengan data yang ingin Anda kirim. Dengan menggunakan koneksi, saya sudah cache informasi ini dengan klien dan metode sent dapat terjadi seperti versi aliran soket (tidak ada alamat tujuan diperlukan). Anda dapat panggilan terhubung lagi untuk kembali menentukan target pesan datagram klien.

4.2.6 Socket options

Soket diset standar, tapi itu mungkin untuk mengubah perilaku soket menggunakan opsi. Anda memanipulasi pilihan socket dengan metode setsockopt dan memanggil mereka dengan metode getsockopt.

berikut opsi socket sederhana dengan Python
Opsi SO _REUSEADDR yang paling sering digunakan dalam pembangu-

```
1  sock = <strong>socket.socket</strong>( socket.AF_INET, socket.SOCK_STREAM )
2
3  # Get the size of the socket's send buffer
4  bufsize = sock.<strong>getsockopt</strong>( socket.SOL_SOCKET, socket.SO_SNDBUF )
5
6  # Get the state of the SO_REUSEADDR option
7  state = sock.<strong>getsockopt</strong>( socket.SOL_SOCKET, socket.SO_REUSEADDR )
8
9  # Enable the SO_REUSEADDR option
10 sock.<strong>setsockopt</strong>( socket.SOL_SOCKET, socket.SO_REUSEADDR, 1 )
```

Gambar 4.5: Opsi socket

nan server socket. Anda dapat meningkatkan soket mengirim dan menerima buffer untuk kinerja yang lebih besar, tetapi mengingat bahwa Anda beroperasi di sini interpreted scripting language, hal ini tidak dapat memberikan Anda banyak manfaat

4.2.7 Asynchronous I/O

Python menawarkan asynchronous I / O sebagai bagian dari modul select. Fitur ini mirip dengan mekanisme sesect pada pemrograman C, tetapi select pada python memiliki beberapa penyederhanaan. Pada bagian ini, akan diperkenalkan modul select pada dan kemudian menunjukkan cara menggunakannya dengan Python.

Metode select memungkinkan untuk digunakan pada peristiwa yang multiplex dengan penggunaan beberapa soket dan untuk beberapa event yang berbeda. Misalnya, Anda dapat menginstruksikan select untuk memberikan notifikasi ketika data telah tersedia, memungkinkan untuk menulis data melalui socket, dan juga bila terjadinya kesalahan pada socket. Disinilah anda dapat menggunakan banyak soket pada saat yang sama.

C bekerja dengan bitmap sedangkan Python menggunakan daftar untuk mewakili deskripsi yang digunakan untuk memonitoring dan juga mengembalikan deskripsi. Perhatikan contoh berikut : Argumen dikirimkan un-

```
1 rlist, wlist, elist = <strong>select.select</strong>( [sys.stdin], [], [] )
2
3 print sys.stdin.read()
```

Gambar 4.6: Contoh input standar

tuk memilih daftar yang mewakili peristiwa membaca, menulis, dan kesalahan. metode select mengembalikan tiga daftar yang berisi object (membaca, menulis an exception). Dalam contoh ini, setelah rlist akan menjadi [sys.stdin], yang menunjukkan bahwa data tersedia untuk dibaca pada stdin. Data ini kemudian dapat dibaca dengan menggunakan metode read.

Metode select juga bekerja pada deskriptor soket. Pada contoh berikut dibawah, dua soket klien dibuat dan terhubung ke remote peer. Metode select kemudian digunakan untuk mengidentifikasi bahwa data pad socket telah tersedia untuk dibaca. Data kemudian dibaca dan ditampilkan ke stdout.

```
1  import socket
2  import select
3
4  sock1 = <strong>socket.socket</strong>( socket.AF_INET, socket.SOCK_STREAM )
5  sock2 = <strong>socket.socket</strong>( socket.AF_INET, socket.SOCK_STREAM )
6
7  sock1.<strong>connect</strong>( ('192.168.1.1', 25) )
8  sock2.<strong>connect</strong>( ('192.168.1.1', 25) )
9
10 while 1:
11
12   # Await a read event
13   rlist, wlist, elist = <strong>select.select</strong>( [sock1, sock2], [], [], 5
14
15   # Test for timeout
16   if [rlist, wlist, elist] == [ [], [], [] ]:
17
18     print "Five seconds elapsed.\n"
19
20   else:
21
22     # Loop through each socket in rlist, read and print the available data
23     for sock in rlist:
24
25       print sock.<strong>recv</strong>( 100 )
```

Gambar 4.7: Asynchronous I/O

4.2.8 Memulai membuat socket client-server

Untuk membuat sistem client-server dibutuhkan 2 komputer berbasis linux dimana satu komputer dijadikan server dan satu komputer di jadikan client

4.2.8.1 komputer server

berikut code socket server berbasis python yang digunakan :

```
import time
import pyhs2
import socket
TCP_IP = '192.168.10.99'
TCP_PORT = 1987
BUFFER_SIZE = 256
i = 1
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM
   )
s.bind((TCP_IP, TCP_PORT))
s.listen(1)
con = pyhs2.connect(host='HadoopMaster', port=10000,
authMechanism="PLAIN", user='hduser',password='
   qwerty',
database='default')
cur = con.cursor()
print 'Listening :'
conn, addr = s.accept()
print 'Connection address:', addr
print cur.getDatabases()

try:
    while i:
        dataBase = conn.recv(BUFFER_SIZE)
        if not dataBase: continue
        dataSplit = dataBase.split(',')
        tanggal = str(dataSplit[0])
        jam = str(dataSplit[1])
        orp = str(dataSplit[2])
```

```
        ph = str(dataSplit[3])
        ec = str(dataSplit[4])
        tds = str(dataSplit[5])
        sal = str(dataSplit[6])
        sg = str (dataSplit[7])
        do = str (dataSplit[8])
        temp = str (dataSplit[9])
        commandSave="INSERT INTO TABLE
           projectiotwater VALUES ('{}','{}',
                '{}','{}','{}','{}','{}','{}','{}','{}')
                   ".format(tanggal,jam,orp,
                ph,ec,tds,sal,sg,do,temp)
        cur.execute(commandSave)

        print "Save Success"
except KeyboardInterrupt:
    conn.close()
```

dimana program ini melakukan penyimpanan ke server bigData dengan mengambil data dari komputer client.

4.2.8.2 komputer Client

berikut code client socket berbasis python yang digunakan :

```
import socket
import serial
import time
import sys
sensor = '/dev/ttyUSB0'
baud_= 9600
TCP_IP = '192.168.10.99'
TCP_PORT = 1987
BUFFER_SIZE = 512
ser = serial.Serial(sensor, baud_)
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM
   )
s.connect((TCP_IP, TCP_PORT))
try:
```

```
    while True:
        waktu = time.strftime ("%Y-%m-%d,%H:%M:%S",
                time.localtime())
        respon = ser.readline()
        cekbyte = sys.getsizeof(respon)
        print cekbyte
        if (cekbyte>67):
            dataSensor = respon[respon.find("#")
                        +1:respon.find("*")]
            dataBase=','.join([waktu,dataSensor])
            print dataBase
            MESSAGE = dataBase
            s.send(MESSAGE)
            f = open('./sensor.csv','a')
            f.write(dataBase)
            f.write("\n")
            time.sleep(30)
        else:
            continue
except KeyboardInterrupt:
    s.close()
    ser.close()
```

dimana program ini melakukan pengambilan data sensor melalui koneksi /dev/ttyUSB0./dev/ttyUSB0

4.3 Restful

REST, singkatan bahasa Inggris dari representational state transfer atau transfer keadaan representasi, adalah suatu gaya arsitektur perangkat lunak untuk untuk pendistibusian sistem hipermedia seperti WWW. Istilah ini diperkenalkan pertama kali pada tahun 2000 pada disertasi doktoral Roy Fielding, salah seorang penulis utama spesifikasi HTTP. Istilah ini selanjutnya dipergunakan secara luas pada komunitas jaringan.

REST secara spesifik merujuk pada suatu koleksi prinsip-prinsip arsitektur jaringan yang menggariskan pendefinisian dan pengalamatan sumber daya. Istilah ini sering digunakan dengan longgar untuk mendeskripsikan semua antarmuka sederhana yang menyampaikan data dalam domain spesi-

fik melalui HTTP tanpa tambahan lapisan pesan seperti SOAP atau pelacakan sesi menggunakan cookie HTTP. Dua pengertian ini dapat menimbulkan konflik dan juga tumpang tindih. Dimungkinkan untuk merancang suatu sistem perangkat lunak besar sesuai dengan gaya arsitektur REST Fielding tanpa menggunakan HTTP dan tanpa berinteraksi dengan WWW. Juga dimungkinkan untuk merancang antarmuka XML+HTTP sederhana yang tidak mengikuti prinsip-prinsip REST, tapi sebaliknya mengikuti model dari RPC (remote procedure call). Perbedaan penggunaan istilah REST ini cukup menyebabkan permasalahan dalam diskusi-diskusi teknis.

Sistem yang mengikuti prinsip REST Fielding sering disebut sebagai "RESTful".

Gambar 4.8: Restfull

4.3.1 RESTful Web Service pada Java menggunakan Jersey (JAX-RS 1.1)

Restfull dapat pada java dengan menggunakan Jersey pada kali ini kami menggunakan versi jersey (JAX-RS 1.1)

4.3.1.1 install Jersey

untuk menginstall Jersey, pertama download installer di http://jersey.java.net, kemudian buat dynamic web project di eclipse, lalu copy semua library

pada folder Jersey yang baru saja kamu download ke folder /WEB-INF/lib/. Sekarang coba buat class Hello.java seperti dibawah ini :

```
package com.kusandriadi.ws;

import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;

@Path("/hello")
public class Hello {

        @GET
        @Produces(MediaType.TEXT_HTML)
        public String helloHtml(){
                return "<html> " +
                "<title>" +
        "Hello Kus Andriadi :D" +
        "</title>"
        + "<body><h1>" + "Hello Kus Andriadi :D" +
        "</body></h1>" +
        "</html> ";
        }

        @GET
        @Produces(MediaType.TEXT_PLAIN)
        public String helloText(){
                return "Hello Kus Andriadi";
        }

        @GET
        @Produces(MediaType.TEXT_XML)
        public String helloXML(){
                return "<?xml version=\"1.0\"?>" +
        "<hello> Kus Andriadi" + "</hello>";
        }
}
```

Anotasi @Path(/hello) diatas maksudnya adalah jika ada request ke path /hello akan diolah oleh class ini, anotasi @GET akan menjawab request dari client (begitu juga dengan @POST, @DELETE dan @PUT jika digunakan) sesuai dengan HTTP Standard Methods. Anotasi @Produce berguna sebagai hasil respons dari server, apakah berbentuk text, xml, html, dst.

Jangan lupa untuk mendefinisikan Servlet Jersey di web.xml seperti ini :

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema
   -instance" xmlns="http://java.sun.com/xml/ns/
   javaee" xmlns:web="http://java.sun.com/xml/ns/
   javaee/web-app_2_5.xsd" xsi:schemaLocation="http
   ://java.sun.com/xml/ns/javaee http://java.sun.com
   /xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID"
   version="3.0">
  <display-name>HelloWS</display-name>
  <servlet>
        <servlet-name>Jersey</servlet-name>
        <servlet-class>com.sun.jersey.spi.container.
           servlet.ServletContainer</servlet-class>

        <init-param>
                <param-name>com.sun.jersey.config.
                   property.packages</param-name>
                <param-value>com.kusandriadi</param-
                   value>
        </init-param>
        <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet-mapping>
        <servlet-name>Jersey</servlet-name>
        <url-pattern>/rest/*</url-pattern>
  </servlet-mapping>
</web-app>
```

Pada ¡param-value¿ arahkan ke package dimana class Hello tadi berasal. ¡url-pattern¿ disitu maksudnya adalah setiap url yang masuk dengan pattern /rest/* akan diolah oleh Servlet Jersey. Setelah selesai, mari kita membuat class Client untuk request ke class REST tadi. Buat project baru, copy

semua library Jersey tadi ke project yang baru dibuat, lalu cukup buat class main seperti dibawah ini.

```
import com.sun.jersey.api.client.Client;
import com.sun.jersey.api.client.ClientResponse;
import com.sun.jersey.api.client.WebResource;
import com.sun.jersey.api.client.config.ClientConfig
   ;
import com.sun.jersey.api.client.config.
   DefaultClientConfig;
import java.net.URI;
import javax.ws.rs.core.MediaType;
import javax.ws.rs.core.UriBuilder;

/**
 *
 * @author Kus Andriadi
 */
public class ClientWS {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        ClientConfig clientConfig = new
           DefaultClientConfig();
        Client client = Client.create(clientConfig);

        WebResource webResource = client.resource(
           getURI());

        //respon dari uri
        System.out.println(webResource.accept
        (MediaType.TEXT_PLAIN).get(ClientResponse.
        class).toString());

        //xml
        System.out.println(webResource.accept
```

```
        (MediaType.TEXT_XML).get(String.class).
        toString());

        //html
        System.out.println(webResource.accept
        (MediaType.TEXT_HTML).get(String.class).
        toString());

        //text
        System.out.println(webResource.accept
        (MediaType.TEXT_PLAIN).get(String.class).
        toString());
    }

    private static URI getURI() {
        return UriBuilder.fromUri
        ("http://localhost:8080/HelloWS/rest/hello")
           .build();
    }
}
```

Jalankan class main tersebut, lalu akan output seperti dibawah ini :

```
GET http://localhost:8080/HelloWS/rest/
hello returned a response status of 200 OK
<?xml version="1.0"?><hello> Kus Andriadi</hello>
<html> <title>Hello word</title><body>
<h1>Hello Kus Andriadi :D</body></h1></html>
Hello Kus Andriadi
```

4.3.2 Membuat REST API dengan Flask (Python Microframework)

Restfull dapat pada python dengan menggunakan flask, dimana Flask merupakan salah satu microframework Python yang banyak digunakan untuk pembuatan RESTFul API. Pada tulisan kali ini dan selanjutnya -insyaAllah-, kita akan berkenalan dengan flask dan mengimplementasikannya dalam membuat REST API.

4.3.2.1 Install Flask

Untuk dapat menggunakan flask, kita harus menginstall Python terlebih dahulu. Pada tulisan kali ini sengaja tidak dibahas bagaimana instalasi python agar mempersingkat pembahasan. Bagi pengguna Linux, biasanya Python sudah otomatis terinstal ketika melakukan instalasi operating system. Pada seri belajar kali ini, penulis akan membawakan contoh instalasi dan konfigurasi sebagai pengguna Ubuntu. Ok, setelah python, terinstall, yang akan kita lakukan adalah menyiapkan perlengkapan development, yaitu flask dan ektensi-ekstensinya. Oya, flask merupakan script python yang akan berjalan dengan interpreter. Oleh karena itu sebagaimana bahasa pemrograman yang lain seperti PHP dan yang semisal, kita membutuhkan server environment yang akan meng-host dan menginterpreter script ketika dipanggil.

Mari kita buka terminal, kemudian pindah ke direktori project kita, misalnya.

```
cd /home/projects/flask-learning
```

Kemudian yang kita lakukan pertama kali adalah membuat virtual environment

```
$ sudo virtualenv --no-site-packages venv
```

Virtual environment ini semacam environment di mana program kita akan berjalan, tanpa mempengaruhi environment python sistem komputer kita. Setelah perintah di atas kita eksekusi, kita akan melihat ada direktori venv muncul pada root direktori project. Langkah berikutnya adalah melakukan instalasi flask di atas environment tersebut, setelah virtual enviromnent akfif. Kita bisa mengaktifkan virtual environtment dengan mengeksekusi perintah berikut

```
$ source venv/bin/activate
```

kemudian install flask dengan menggunakan perintah

```
$ pip install flask
```

buatlah program simple -hello world- dengan membuat file bernama app.py di root directory.

```
from flask import Flask #1

app = Flask(__name__) #2
```

```
@app.route('/')  #3
def home():
   return 'Hello, world!'

if __name__ == '__main__': #4
   app.run(debug=True)
```

Yang kita lakukan pada script di atas adalah dengan cara melakukan impor modul flask terlebih dahulu.

1. Melakukan impor modul flask

2. Membuat application object app dari class FlaskMembuat router yang akan melakukan mapping URLs / ke fungsi home(). Artinya ketika client mengakses http://¡apiserverurl¿/ flask akan mengeksekusi fungsi home() dan menampilkan teks Hello, world!

3. Perintah ini digunakan untuk menjalakan server flask dengan mode debug. Artinya ketika sewaktu-waktu terjadi error, maka pesan error akan ditampilkan. Ini hanya dilakukan pada fase development. Adapun pada fase live, mode debug harusnya bernilai True.

lakukan pengetesan program dengan perintah :

```
$ python app.py
```

Jika berhasil maka console akan menampilkan status bahwa server flask sudah berjalan

```
Running on http://127.0.0.1:5000/ (Press CTRL+C to
   quit)
* Restarting with stat
```

Alamat URL di atas adalah base endpoint dari program yang telah kita buat. Ketika kita mencoba membuka URL http://127.0.0.1:5000/ pada browser, maka pesan Hello World kita akan muncul.

Terakhir pada tulisan ini, mari kita coba menambahkan sebuah fungsi welcome yang akan diakses di URL http://127.0.0.1:5000/welcome, seperti ini:

```
@app.route('/welcomeef welcome():
    return render_template('welcome.html')
```

Pada fungsi welcome, kita melakukan pemanggilan fungsi render_tempalte. Fungsi ini digunakan untuk merender suatu html file, dalam kasus ini halaman welcome.html yang ada pada direktori templates. Sebelumnya, kita perlu menambahkan import statement untuk meload fungsi render_template tersebut dari modul flask.

```
from flask import Flask, render_template
```

Berikutnya, karena kita belum membuat file welcome.html, jika alamat URL welcome diakses akan memunculkan error not found. Oleh karena itu, mari kita buat direktori statics dan templates.

Direktori statics berisi file-file statik seperti gambar, css, atau java script. Sedangkan direktori templates berisi file-file template html yang akan digunakan dengan memanggil fungsi render_template. Kita tempatkan file welcome.html berikut pada direktori templates.

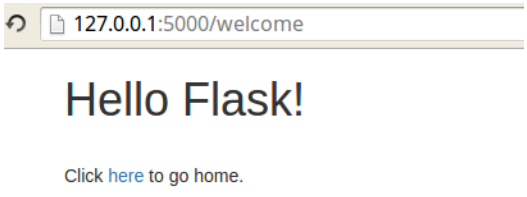
welcome.html

```
sa
```

Agar terlihat menarik, kita tambahkan style untuk halaman tersebut dengan menggunakan bootstrap css. Bootstrap CSS ini bisa diunduh di halaman getbootstrap. Download, ekstrak, dan letakan file bootstrap.min.css pada direktori statics.

Oya, jika kita cermati, file app.py dan file welcome.html berada pada direktori yang terpisah. Akan tetapi pemanggilan file welcome.html pada app.py cukup menggunakan render_template(welcome.html) dan bukan render_template(/templates/welcome.html) karena fungsi render_template ini sendiri sudah merujuk ke direktori templates.

Nah, sekarang kita dapat mencoba melihat hasilnya dengan membuka http://127.0.0.1:5000/welcome pada browser.



Bab 5

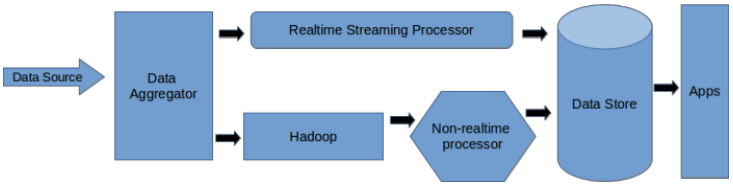
Teknologi Big Data

5.1 Big Data

Hingga saat ini, definisi resmi dari istilah Big Data belum ada. Namun demikian, latar belakang dari munculnya istilah ini adalah fakta yang menunjukkan bahwa pertumbuhan data yang terus berlipat ganda dari waktu ke waktu telah melampaui batas kemampuan media penyimpanan maupun sistem database yang ada saat ini. Kemudian, McKinseyGlobal Institute (MGI), dalam laporannya yang dirilis pada Mei 2011, mendefinisikan bahwa Big Data adalah data yang sudah sangat sulit untuk dikoleksi, disimpan, dikelola maupun dianalisa dengan menggunakan sistem database biasa karena volumenya yang terus berlipat. Tentu saja definisi ini masih sangat relatif, tidak mendeskripsikan secara eksplisit sebesar apa big data itu. Tetapi, untuk saat sekarang ini, data dengan volume puluhan terabyte hingga beberapa petabyte kelihatannya dapat memenuhi definis MGI tersebut. Di lain pihak, berdasarkan definisi dari Gartner, big data itu memiliki tiga atribute yaitu : volume, variety dan velocity. Ketiga atribute ini dipakai juga oleh IBM dalam mendifinisikan big data. Volume berkaitan dengan ukuran, dalam hal ini kurang lebih sama dengan definisi dari MGI. Sedangkan variety berarti tipe atau jenis data, yang meliputi berbagai jenis data baik data yang telah terstruktur dalam suatu database maupun data yang tidak terorganisir dalam suatu database seperti halnya data teks pada web pages, data suara, video, click stream, log file dan lain sebagainya. Yang terakhir, velocity dapat diartikan sebagai kecepatan dihasilkannya suatu data dan seberapa cepat data itu harus diproses agar dapat memenuhi permintaan pengguna. Dari

segi teknologi, dipublikasikannya Google Bigtable pada 2006 telah menjadi moment muncul dan meluasnya kesadaran akan pentingnya kemampuan untuk memproses Big Data. Berbagai layanan yang disediakan Google, yang melibatkan pengolahan data dalam skala besar termasuk search engine-nya, dapat beroperasi secara optimal berkat adanya Bigtable yang merupakan sistem database berskala besar dan cepat. Semenjak itu, teknik akses dan penyimpanan data KVS (Key-Value Store) dan teknik komputasi paralel yang disebut MapReduce mulai menyedot banyak perhatian. Lalu, terinspirasi oleh konsep dalam GoogleFile System dan MapReduce yang menjadi pondasi Google Bigtable, seorang karyawan Yahoo! bernama Doug Cutting kemudian mengembangkan software untuk komputasi paralel terdistribusi (distributed paralel computing) yang ditulis dengan menggunakan Java dan diberi nama Hadoop. Saat ini Hadoop telah menjadi project open source-nya Apache Software. Salah satu pengguna Hadoop adalah Facebook, SNS (Social Network Service) terbesar dunia dengan jumlah pengguna yang mencapai 800 juta lebih. Facebook menggunakan Hadoop dalam memproses big data seperti halnya content sharing, analisa access log, layanan message / pesan dan layanan lainnya yang melibatkan pemrosesan Big Data. Jadi, yang dimaksud dengan Big Data bukanlah semata-mata hanya soal ukuran, bukan hanya tentang data yang berukuran raksasa. Big data adalah data berukuran raksasa yang volumenya terus bertambah, terdiri dari berbagai jenis atau varietas data, terbentuk secara terus menerus dengan kecepatan tertentu dan harus diproses dengan kecepatan tertentu pula. Momen awal ketenaran istilah Big Data adalah kesuksesan Google dalam memberdayakan Big Data dengan menggunakan teknologi canggihnya yang disebut Bigtable beserta teknologi-teknologi pendukungnya.

5.1.1 Arsitektur Big Data



Gambar 5.1: Arsitektur Big Data

Data source adalah sumber data untuk big Data. Data umumnya dipompa

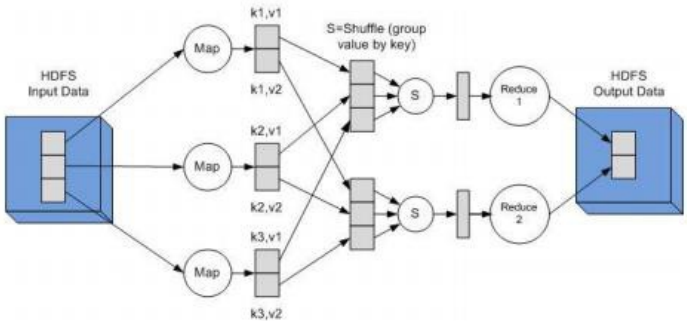
masuk Big Data dengan menggunakan API ataupun dengan operasional file system seperti transfer file. Ada dua jenis data source yaitu streaming data source dan bulk data source. Contoh streaming data source misalnya adalah tweets dari twitter API. Sedangkan Bulk data misalnya adalah file teks biasa yang sangat besar seperti file log dari suatu aplikasi ataupun file yang berisi data yang di dump dari database. Data aggregator adalah tool atau software yang mengumpulkan dan manyalurkan data dari sumber ke beberapa jenis pengolahan data di Big data. Ada dua jenis data aggregator berdasarkan cara kerjanya. Jenis pertama adalah Pull-based data aggregator. Jenis ini mengumpulkan data dan memberikan data tersebut kepada siapa saja yang meminta tanpa registrasi sebelumnya, mirip seperti Java Messaging Queue. Contohnya adalah Apache Kafka, RabbitMQ. Jenis kedua adalah Push-based data aggregator. Jenis kedua ini mengumpulkan data dan mengirim data ke sistem lain yang sudah di set terhubung dan menerima data dari data aggregator. Sistem yang mau mendapatkan data harus terdaftar di data aggregator dulu dan biasanya diperlukan effort lebih jika ada sistem baru yg ingin mendapatkan data dari data aggregator jenis ini dibanding jenis yang pertama. Contoh Push-Based Data Aggregator adalah Apache Flume dan Spring-XD. Realtime streaming Processor adalah salah satu sistem pengolahan di Big Data yang umum ditemukan. Fungsinya adalah untuk menganalisis data yang bersifat realtime dan streaming. Contohnya adalah menghitung hashtag yang muncul di semua tweet di twitter. Sifat dari pemrosesan ini haruslah ringa, dan cepat. Oleh karena itu analisis data secara kompleks jarang sekali dilakukan. Output dari pemrosesan ini adalah gambaran umum dari data yang didapatkan dan tidak terlalu detil. Outputnya pun sebaiknya disimpan di datastore sehingga bisa digunakan oleh aplikasi yang membutuhkan. Untuk hasil analisis data yang sangat detil bisa di lihat di Non-realtime processor. Contoh tool yang digunakan di realtime streaming misalnya adalah Apache Storm, Apache Spark Streaming dan Spring-XD. Meskipun hasilnya tidal detil, tetapi pemrosesan ini diperlukan mengingat pemrosesan secara bulk / non-realtime membutuhkan waktu yang cukup lama. Dengan demikian user bisa melihat secara garis besar data yang diolah meskipun tidak detil sembari menunggu pemrosesan non realtime selesai. Hadoop yang dimaksud di sini adalah HDFS. Disini hadoop lebih ditekankan sebagai tempat penyimpanan data yang sangat besar. Hadoop menjadi tempat semua data sehingga bisa dianalisis oleh berbagai tools untuk berbagai kepentingan sehingga bisa didapatkan hasil yang cukup detil dan bisa memenuhi kebutuhan dari user. Non-realtime processor adalah proses

pemrosesan data di Big Data untuk data besar yang terdapat di HDFS. Pemrosesan ini menggunakan berbagai jenis tool sesuai kebutuhan. sebuah data bisa dianalisis lebih dari satu tools. Contoh tool yang sering digunakan antara lain Hive dan Pig untuk Map Reduce, Apache Mahout dan Apache Spark untuk machine learning dan artificial intelligence. Hasil dari pemrosesan ini dimasukkan ke dalam data store untuk kemudian bisa di lihat di level aplikasi. Sistem pemrosesan ini umumnya memerlukan waktu yang relatif lebih lama mengingat data yang diproses relatif sangat besar. Data store adalah tools untuk menyimpan data hasil pemrosesan baik realtime maupun non-realtime. Datastore disini bisa berupa RDBMS ataupun jenis NoSQL lainnya. RDBMS sangat jarang digunakan sebagai data store mengingat keterbatasan dalam sisi ukuran yang bisa ditampung tanpa kehilangan kinerja. Datastore yang umumnya dipakai adalah NoSQL yang berbasis Document (mis. MongoDB), Column-oriented seperti HBase dan Cassandra, dan juga key-value pair seperti couchDB. Beberapa data store yang jarang kedengaran juga dipakai seperti misalnya Voldemort dan Druid. Apps adalah aplikasi yang berinteraksi langsung dengan user. Aplikasi disini mengakses data yang berada di data store untuk kemudian disajikan kepada user. Jenis aplikasi disini sangat bervariasi bisa berupa web, desktop ataupun mobile. Pada umumnya aplikasi disini hanyalah untuk melakukan visualisasi dari data yang sudah dianalisis sebelumnya.

5.1.2 Konsep Map Reduce

MapReduce adalah model pemrograman yang digunakan untuk mendukung distributed computing yang dijalankan di atas data yang sangat besar dan dijalankan secara paralel di banyak komputer. Mapreduce memungkinkan programmer untuk melakukan komputasi yang sederhana dengan menyembunyikan kompleksitas dan detail dari paralelisasi, distribusi data, load balancing dan fault tolerance. Program Mapreduce ini dapat dijalankan pada berbagai bahasa pemrograman termasuk Java, Phyton, dan C++[45]. Konsep Mapreduce membagi proses menjadi dua tahapan, yaitu Map dan Reduce. Map merupakan proses yang berjalan secara paralel, sedangkan reduce merupakan penggabungan hasil dari proses Map.Map memiliki fungsi yang dipanggil untuk setiap input yang menghasilkan output pasangan ¡key, value¿. Pada kondisi ini Map melakukan transformasi setiap data elemen input menjadi data elemen output. Reduce adalah tahapan yang dilakukan setelah mapping selesai. Reduce akan memeriksa semua value input dan

mengelompokkannya menjadi satu value output. Sebelum memasuki tahap reduce, pasangan ¡key, value¿ dikelompokkan berdasarkan key. Tahap ini dinamakan tahap shuffle. Tahap shuffle dilakukan sebagai persiapan menuju tahap reduce. Reduce juga memiliki fungsi yang dipanggil untuk setiap key. Fungsi reduce memperhatikan setiap value dari key bersangkutan. Untuk setiap key akan dihasilkan sebuah ringkasan value-nya. Proses skema Mapreduce dapat dilihat pada gambar :.



Gambar 5.2: Konsep Map Reduce

Pada gambar 5.2 dapat menunjukkan bagaimana proses mapping menghasilkan suatu pasangan key dan value, bagaimana key dan value tersebut dikelompokkan berdasarkan key yang sama kemudian dilakukan proses reducing, sehingga mendapatkan suatu output.

5.1.3 Hadoop

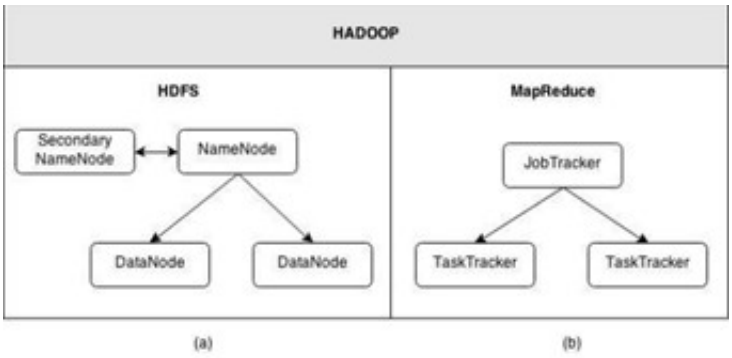
Hadoop adalah framework perangkat lunak berbasis Java dan opensource yang berfungsi untuk mengolah data yang besar secara terdistribusi dan berjalan di atas Cluster yang terdiri dari beberapa komputer yang saling terhubung. Hadoop dibuat oleh Doug Cutting yang pada asalnya Hadoop ini adalah sub project dari Nutch yang digunakan untuk search engine. Hadoop bersifat open source dan berada di bawah bendera Apache Software Foundation.

5.1.3.1 Arsitektur Hadoop

Hadoop terdiri dari common Hadoop yang berguna dalam menyediakan akses ke dalam file system yang didukung oleh Hadoop. Common Hadoop ini berisi

paket yang diperlukan oleh JAR file, skrip yang dibutuhkan untuk memulai Hadoop dan dokumentasi pekerjaan yang telah dilakukan oleh Hadoop. Berdasarkan dua inti dari Hadoop adalah terdiri dari:

- Hadoop Distributed File System (HDFS) untuk data yang terdistribusi.
- MapReduce Framework untuk aplikasi dan programming yang terdistribusi.



Gambar 5.3: Inti dan Komponen Hadoop

Gambar 5.3 menggambarkan bagian inti Hadoop yang terdiri dari HDFS dan MapReduce. Pada Gambar 1.3 (a) menggambarkan komponen dari HDFS yang terdiri dari NameNode, DataNode, dan Secondary NameNode dan Gambar 1.3 (b) menggambarkan komponen dari MapReduce yang terdiri dari JobTracker dan TaskTracker. Sebuah cluster kecil pada Hadoop dapat terdiri dari satu master node dan beberapa slave node. Master node ini terdiri dari NameNode dan JobTracker, sedangkan slave node terdiri dari DataNode dan TaskTracker. Hadoop membutuhkan JRE 1.6 atau JRE dengan versi yang lebih tinggi. Dalam menjalankan dan menghentikan sistem pada Hadoop dibutuhkan ssh yang harus dibentuk antar node pada sebuah cluster [46].

5.1.3.2 Kelebihan Hadoop

Komputasi terdistribusi merupakan bidang yang sangat beragam dan luas, namun Hadoop memiliki beberapa kelebihan yang dapat membedakannya dengan yang lain, berdasarkan (2) kelebihan Hadoop adalah sebagai berikut:

- Mudah untuk di akses Hadoop dapat berjalan pada jumlah cluster yang besar ataupun pada layanan komputasi awan seperti Amazon Elastic Compute Cloud (EC2).

- Stabil Hadoop sangat baik dalam menangani sebuah masalah yang muncul ketika sedang memproses sebuah pekerjaan, hal ini dikarenakan dari awalnya Hadoop memang ditunjukan untuk di jalankan pada komuditas perangkat keras.

- Memiliki skala yang besar Hadoop memiliki jangkauan skala yang besar, sehingga dapat menghandle ketika adanya pertambahan jumlah node dalam sebuah cluster.

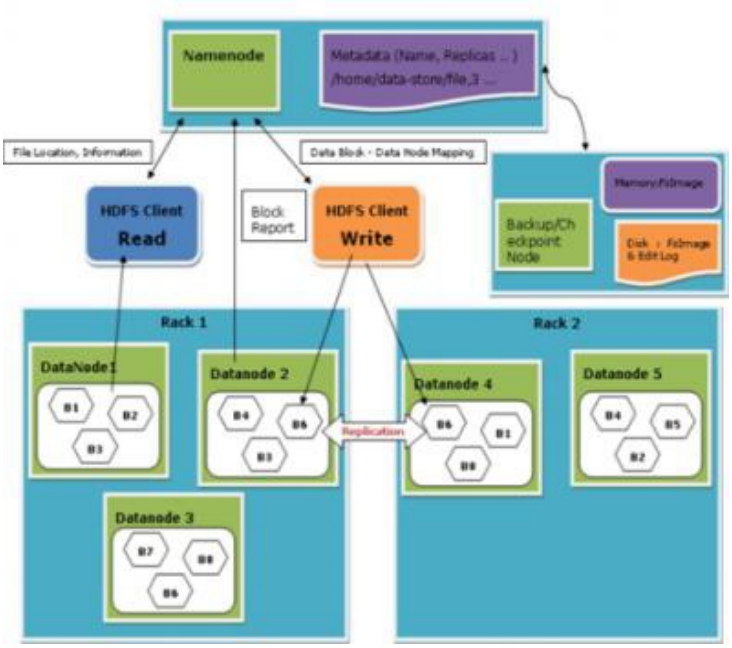
- Mudah digunakan Hadoop sangat mudah dijalankan dan digunakan pada single node maupun multi node.

5.1.3.3 Hadoop Distributed File System (HDFS)

HDFS adalah distributed file sistem berbasis Java yang menyimpan file dalam jumlah yang besar dan disimpan secara terdistribusi di dalam banyak komputer yang saling berhubungan[45]. File sistem ini membutuhkan server induk yang dinamakan Namenode yang berfungsi untuk menyimpan Metadata dari data yang ada di dalam HDFS. Namenode juga melakukan pemecahan (splitting) file menjadi block file dan mendistribusikannya pada komputerkomputer dalam Cluster. Hal ini bertujuan untuk replikasi dan fault tolerance. Namenode juga berfungsi sebagai antar muka yang memberikan informasi melalui Metadata terhadap file yang tersimpan agar data block file tersebut terlihat seperti file yang sesungguhnya. Sedangkan block data disimpan di dalam komputer-komputer yang dinamakan Datanode yang merupakan slave dari HDFS. Datanode dapat berkomunikasi satu sama lain untuk menjaga konsistensi data dan memastikan proses replikasi data berjalan dengan baik.

Suatu arsitektur HDFS terdiri dari satu Namenode yang menyimpan Metadata dari file dan satu atau banyak Datanode yang menyimpan blok-blok file dan hasil replikasi blok file sebagaimana terlihat pada gambar 5.4.

- NameNode NameNode terdapat pada komputer yang bertindak sebagai master yang mengkoordinasi DataNode untuk melakukan beberapa tugas (jobs)[48]. NameNode ini adalah pusat dari sistem berkas pada

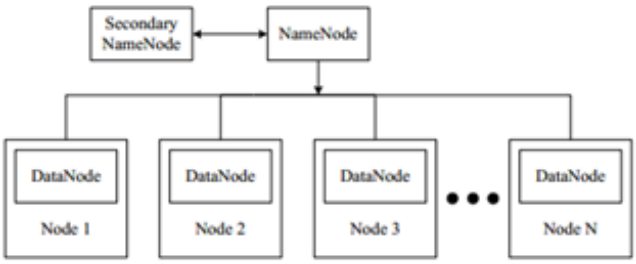


Gambar 5.4: Arsitektur HDFS

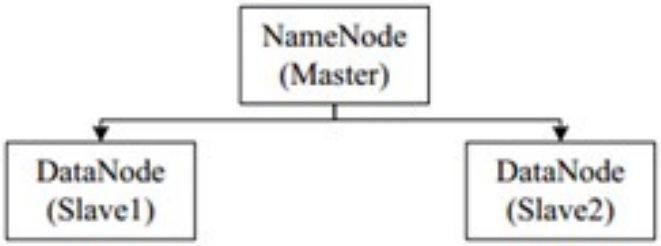
HDFS. Gambaran NameNode yang berada pada master sebagai pusat sistem berkas HDFS dapat dilihat pada Gambar 5.5.

NameNode membuat sistem direktori dari semua file yang ada di dalam sistem dan dapat mengetahui bagaimana file tersebut di pecah-pecah menjadi beberapa blocks data serta mengetahui nodes yang menyimpan blocks data tersebut[45].

- DataNode DataNode adalah salah satu komponen dari HDFS yang berfungsi untuk menyimpan dan mengambil kembali data pada slave node pada setiap permintaan yang dilakukan oleh NameNode. DataNode berada pada setiap slave node pada sebuah cluster yang telah dibuat. DataNode juga berfungsi untuk membaca dan menulis block pada HDFS ke file yang sebenarnya pada local file system. Sebagai contoh apabila user ingin membaca atau menulis file ke HDFS, file tersebut akan dipecah menjadi beberapa block, kemudian NameNode akan memberitahu dimana blocks tersebut berada sehingga DataNode dapat membaca dan menulis blocks tersebut ke file yang sebenarnya pada file system[45].



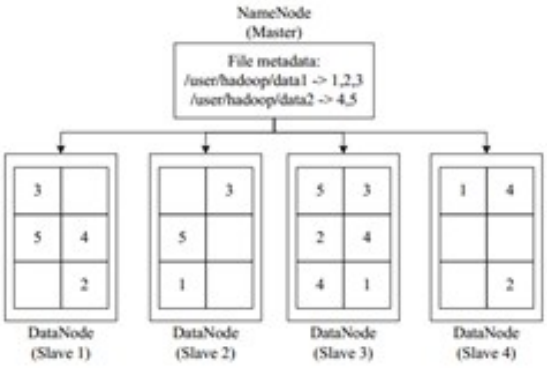
Gambar 5.5: Komponen HDFS



Gambar 5.6: Komponen HDFS

Pada Gambar 5.6 terlihat bahwa NameNode menjaga jalur dari file metadata dimana setiap file tersebut adalah sebuah sistem yang dipecah-pecah menjadi beberapa block [45]. DataNode menyimpan backup dari pecahan-pecahan block tersebut dan secara berkala memberitahu kepada NameNode untuk tetap menjaga jalur dari file metadata. Selama sistem berjalan, DataNode terhubung dengan NameNode dan melakukan sebuah handshake. Bedasarkan handshake ini bertujuan untuk melakukan verifikasi terhadap namespace ID dan juga software version pada sebuah DataNode.

Namespace ID adalah sebuah ID yang muncul ketika pertama kali melakukan format pada NameNode. Namespace ID ini disimpan pada semua node yang ada pada sebuah cluster. Jika ada node yang memiliki namespace ID yang berbeda maka node tersebut tidak akan dapat bergabung pada sebuah cluster. Tujuan adanya namespace ID ini adalah untuk menjaga integritas dari HDFS. Sofware version adalah versi software yang digunakan oleh Hadoop[48]. Konsistensi pada soft-



Gambar 5.7: Interaksi antara Namenode dan Datanode

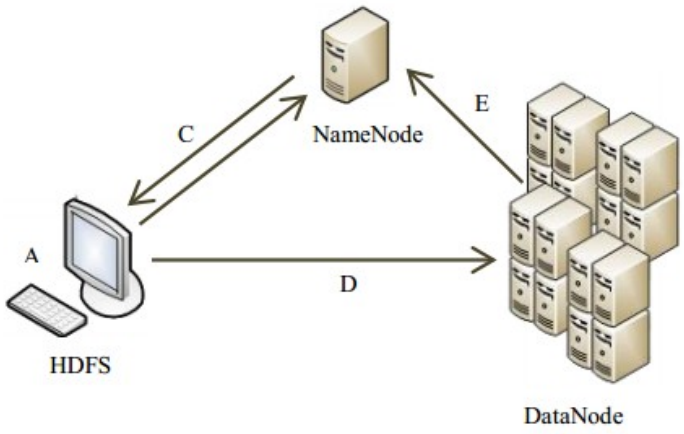
ware version ini sangat penting, karena jika software version yang digunakan berbeda maka akan menyebabkan file corrupt pada sebuah sistem. Jika salah satu node memiliki namespace ID dan juga software version tidak sama dengan nodes yang lain, maka node tersebut tidak akan terdaftar pada sistem cluster yang ada[48].

- Secondary NameNode Secondary NameNode adalah daemon yang berfungsi melakukan monitoring keadaan dari cluster HDFS. Sama seperti NameNode, pada setiap cluster yang ada terdapat satu Secondary NameNode, yang berada pada master node. Secondary NameNode ini juga berfungsi untuk membantu dalam meminimalkan down time dan hilangnya data yang terjadi pada HDFS[45]. Secondary NameNode ini sering menimbulkan kesalahpahaman pengertian bahwa apabila NameNode down maka akan langsung digantikan oleh Secondary NameNode padahal Secondary NameNode ini hanya menyimpan informasi terbaru dari struktur direktori pada NameNode. Jadi jika terjadi kegagalan yang dilakukan oleh NameNode maka dibutuhkan konfigurasi yang dilakukan oleh user untuk menjadikan Secondary NameNode sebagai NameNode yang utama.

5.1.3.4 Prosedur Menyimpan Data

Untuk prosedur menyimpan data harus ada sebuah komputer client yang terhubung dengan sebuah Hadoop cluster

Langkah-Langkah prosedur menyimpan data sebagai berikut:



Gambar 5.8: Arsitektur Penyimpanan Data

- User memasukan perintah masukan pada komputer client
- Komputer client berkomunikasi dengan NameNode memberitahu bahwa ada data yang akan disimpan dan menanyakan lokasi blok-blok tempat menyimpan data.
- Komputer client mendapat jawaban dari NameNode berupa lokasi blok-blok untuk penyimpanan data.
- Komputer client langsung berkomunikasi dengan DataNode untuk memasukan data data pada lokasi blok-blok yang sudah diatur NameNode. Data sudah otomatis terbelah-belah sesuai dengan ukuran yang di setting sehingga dapat langsung menempati blok-blok yang sudah ditentukan.
- DataNode memberikan laporan kepada NameNode bahwa data-data telah masuk dan menempati blok-blok yang sudah ditentukan oleh NameNode.

5.1.3.5 Prosedur Membaca Data

Untuk prosedur membaca data harus ada sebuah komputer client yang terhubung dengan sebuah Hadoop cluster.

Langkah prosedur membaca data sebagai berikut:

- User memasukan perintah untuk mengambil data pada komputer client.
- Komputer client berkomunikasi dengan NameNode untuk menanyakan alamat DataNode penyimpan data yang diinginkan.
- Komputer client mendapat jawaban dari NameNode berupa lokasi blok-blok tempat penyimpanan data yang inginkan.
- Komputer client secara langsung berhubungan dengan DataNode untuk mengakses lokasi blok-blok tempat penyimpanan data yang diinginkan.
- DataNode akan memberikan data yang diinginkan dan data secara otomatis akan ditampilkan pada layar komputer client.

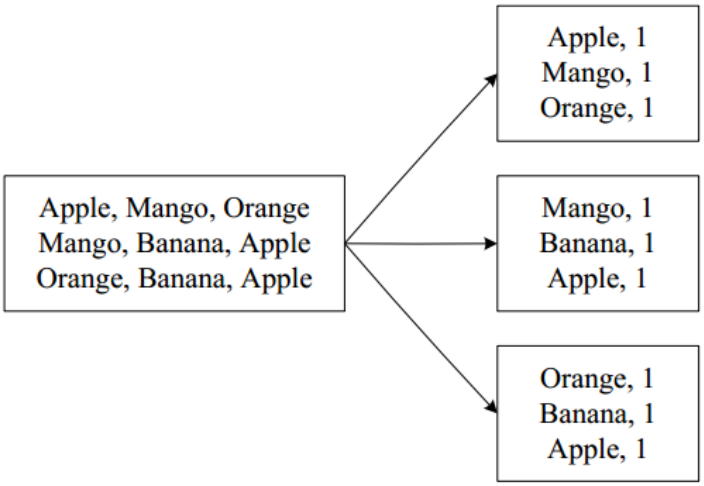
5.1.3.6 Hadoop Map Reduce

Di Hadoop MapReduce engine ini terdiri dari satu Jobtracker dan satu atau banyak Tasktracker. Jobtracker adalah Mapreduce master. Jobtracker meminta kepada Namenode tentang lokasi dari blok data sekaligus akan mendistribusikan blok data tersebut ke Tasktracker yang lokasinya paling dekat dengan data. Namenode menerima input, kemudian input tersebut dipecah menjadi beberapa sub-problem yang kemudian didistribusikan ke Datanode ini akan memproses sub-problem yang diterimanya, setelah sub-problem tersebut sudah diselesaikan maka akan dikembalikan ke-Namenode. Namenode menerima jawaban dari semua sub-problem dari banyak Datanode, kemudian menggabungkan jawaban-jawaban tersebut menjadi satu jawaban untuk mendapatkan penyelesaian dari permasalahan utama. Keuntungan dari Mapreduce ini adalah proses map dan reduce dijalankan secara terdistribusi. Dalam setiap proses mapping bersifat independent, sehingga proses ini dapat dijalankan secara simultan dan paralel. Demikian pula dengan proses reduce dapat dilakukan secara paralel di waktu yang sama, selama output dari operasi mapping mengirimkan key-value yang sesuai dengan proses reducernya.

5.1.3.7 Konsep Dasar Map Reduce

Hadoop menyediakan dua jenis slot untuk melakukan MapReduce yaitu slot map dan slot reduce. Secara default Hadoop telah menentukan jumlah slot map dan slot reduce untuk setiap node yaitu dua slot map dan satu slot reduce. Pada saat memproses data, Hadoop terlebih dahulu melakukan proses

mapping pada task yang terdapat pada slot map sampai selesai kemudian dilanjutkan dengan proses reduce pada slot reduce. Proses mapping: pertama WordCount menginput file plaintext yang tersimpan pada direktori HDFS. Kemudian WordCount akan membagi file plaintext tersebut menjadi beberapa bagian yang berisikan kata yang muncul pada file input dan nilai 1 pada setiap kata yang ada. Gambaran pada saat WordCount melakukan proses mapping ini dapat dilihat pada Gambar 5.8.



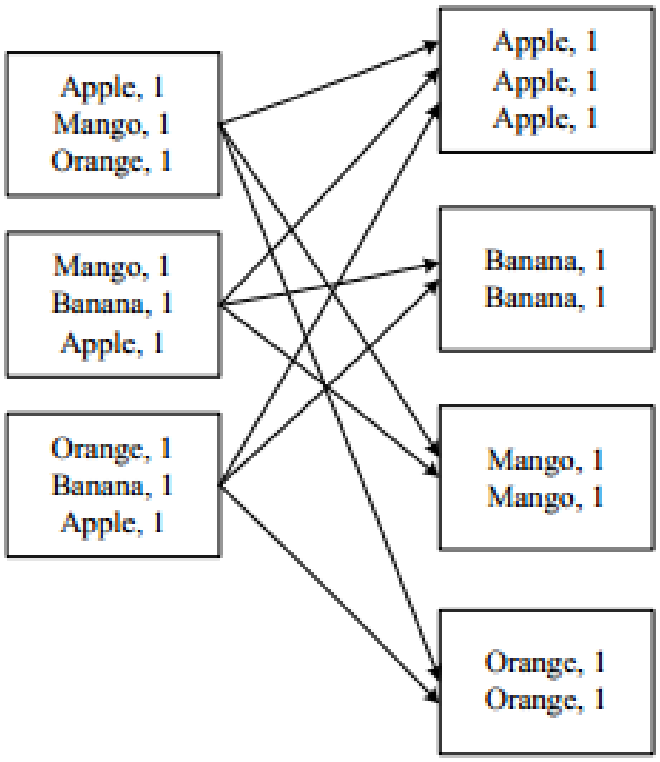
Gambar 5.9: Proses Mapping

Pada Gambar 1.8 terlihat sebuah file input yang berisikan kata-kata yang dibagi menjadi beberapa bagian yang berisikan kata dan nilai 1 pada setiap kata yang ada. Setelah proses mapping ini selesai maka akan dilanjutkan dengan proses shuffle yang berfungsi untuk menggabungkan kata-kata yang sama untuk mempersiapkan proses reducing. Gambaran dari proses shuffle ini dapat dilihat pada Gambar 5.9.

Proses reducing: pada proses ini terjadi penggabungan kata yang sama setelah proses shuffle dan menghitung jumlah kata yang sama tersebut. Gambaran proses reducing ini dapat dilihat pada Gambar 5.10.

Gambaran proses MapReduce yang terjadi secara keseluruhan dapat dilihat pada Gambar 5.11.

Gambar 1.11 menggambarkan sebuah data yang dibagi menjadi beberapa bagian yang kemudian pada setiap bagian dilakukan proses mapping, dan setelah proses mapping selesai bagian-bagian data tersebut di acak (shuffle)



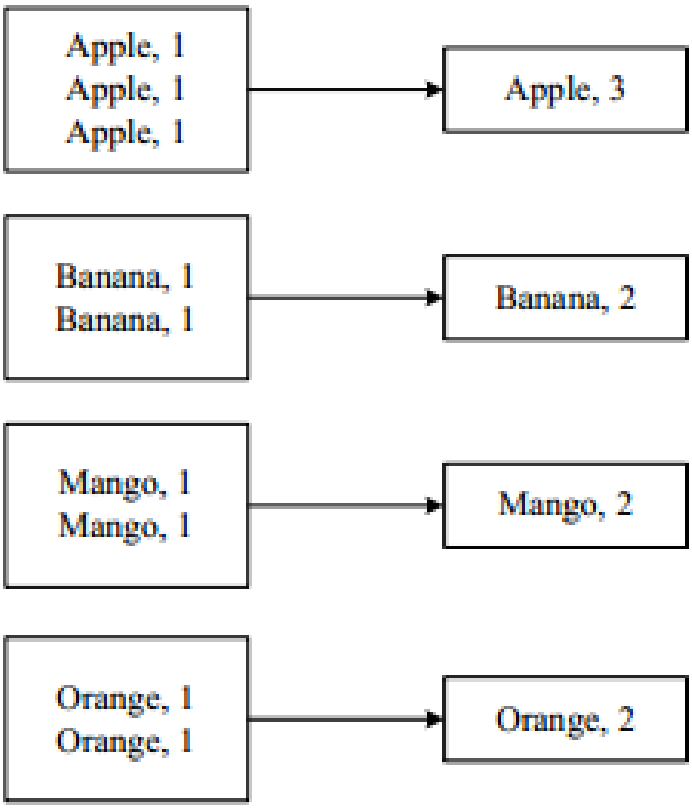
Gambar 5.10: Proses Shuffle

untuk melalui proses reducing. Keuntungan dari MapReduce adalah proses map dan reduce yang dapat diterapkan secara terdistribusi. Pada setiap proses mapping dan proses reducing bersifat independent sehingga proses dapat dijalankan secara paralel pada waktu yang sama, selama output dari proses mapping mengirimkan key value yang sesuai dengan proses reducingnya. Didalam Hadoop, MapReduce ini terdiri dari satu JobTracker dan beberapa TaskTracker pada sebuah cluster.

5.1.3.8 Komponen Map Reduce

MapReduce yang terdapat pada Hadoop memiliki 2 komponen utama penting yaitu:

- JobTracker JobTracker adalah sebuah komponen dari MapReduce yang berfungsi untuk memecah pekerjaan (job) yang diberikan ke HDFS

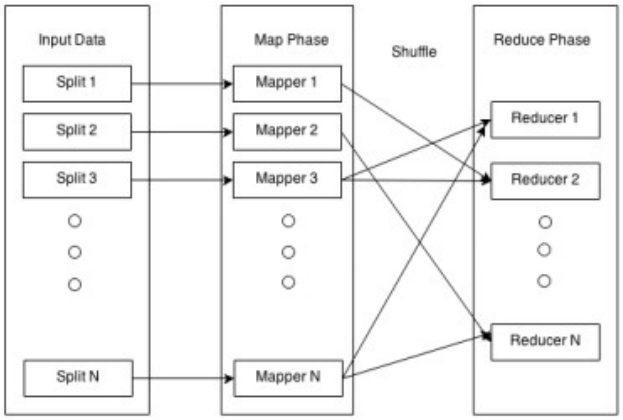


Gambar 5.11: Proses Reduceing

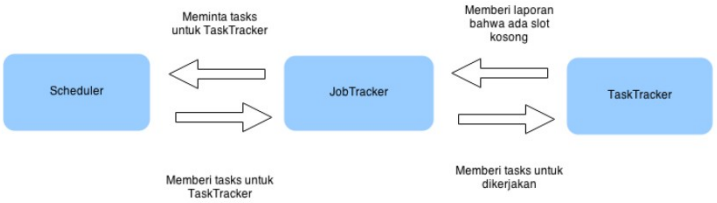
menjadi beberapa tasks yang lebih kecil berdasarkan jumlah slave yang ada[45]. Setelah pekerjaan (job) tersebut dipecah-pecah menjadi beberapa tasks, JobTracker akan memberikan pekerjaan-pekerjaan tersebut kepada setiap slave node yang terdapat di dalam cluster tersebut. JobTracker secara berkala mengkoordinasi semua tasks yang diberikan kepada TaskTracker menggunakan scheduler task (pengatur tugas), kemudian TaskTracker akan mengerjakan tasks tersebut. Setelah TaskTracker menyelesaikan task yang diberikan, maka TaskTracker akan meminta task yang baru kepada JobTracker. Gambaran kerja dari JobTracker dapat dilihat pada Gambar 5.12.

- TaskTracker

  TasTracker adalah sebuah daemon yang berfungsi untuk menerima tu-



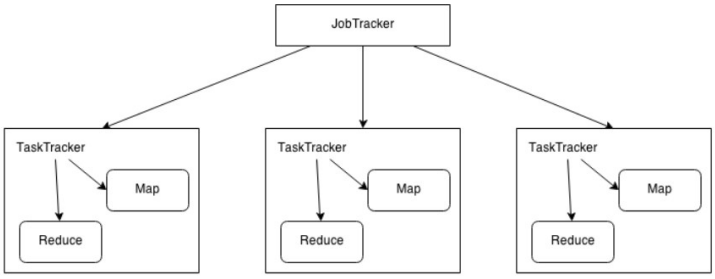
Gambar 5.12: Proses Map and Reduce



Gambar 5.13: Kerja JobTracker

gas (task) yang diberikan oleh JobTracker dan kemudian mengerjakan task tersebut ke dalam Java Virtual Machine (JVM) yang terpisah, dengan menjalakan task tersebut ke dalam Java Virtual Machine (JVM) yang terpisah, maka hal ini akan mengurangi beban pekerjaan yang dilakukan secara paralel yang diberikan oleh JobTracker.

Gambar 5.13 menggambarkan bagaimana sebuah JobTracker yang berkomunikasi dengan beberapa TaskTracker yang melakukan proses MapReduce. Secara konstan TaskTracker terus bekomunikasi dengan JobTracker dengan memberikan laporan setiap proses yang telah dilakukan. Jika JobTracker gagal menerima hasil task yang dikerjalan oleh TaskTracker, maka JobTracker akan mengirimkan kembali task tersebut kepada nodes lain pada cluster tersebut untuk dikerjakan ulang.



Gambar 5.14: Kerja Task Tracker

5.1.3.9 Hadoop Single-Node Cluster dan Multi-Node Cluster

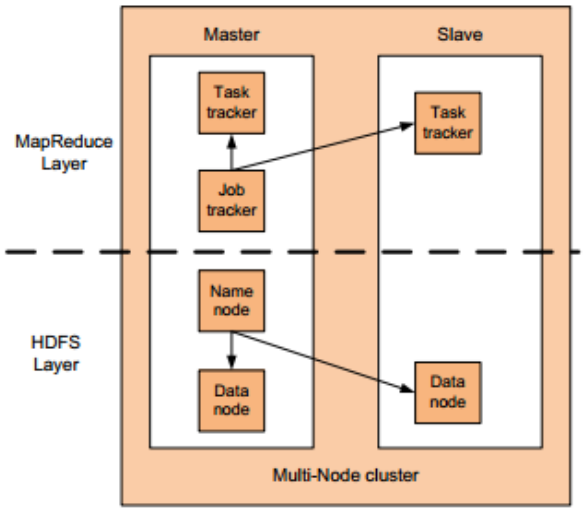
Hadoop single-node diimplementasikan pada satu mesin. Mesin tersebut didesain menjadi master tetapi dapat bekerja juga sebagai slave dan semua proses distribusi dilakukan dalam satu mesin tersebut. Seperti pada Gambar 1.14, dalam Hadoop terbagi menjadi dua layer yaitu layer HDFS yang menjalankan Namenode dan Datanode dan layer Mapreduce yang menjalankan Jobtracker dan Tasktracker. Kedua layer ini sangat penting terutama Namenode dan Jobtracker, karena apabila dua bagian ini tidak berjalan maka kerja HDFS dan Mapreduce tidak bisa dijalankan. Pada mesin single node, Datanode dan Tasktracker hanya ada satu, jika memiliki mesin yang banyak maka kedua bagian ini akan terbentuk pada setiap mesin(multi-node).

Gambar 5.15 Ilustrasi layer HDFS dan Mapreduce dalam Hadoop

Identifikasi pada sebuah Cluster Hadoop multi-node menggunakan dua mesin atau lebih, adalah satu untuk master dan satu atau yang lain sebagai slave. Untuk mengkonfigurasi mesin tersebut adalah berupa mesin-mesin single-node yang akan digabung menjadi satu multi-node dimana satu mesin akan didesain menjadi master tapi dapat bekerja juga sebagai slave, sedangkan komputer yang lain akan menjadi slave. Pada gambar 1.15 merupakan sistem Multi-Node Cluster yang menggunakan dua mesin.

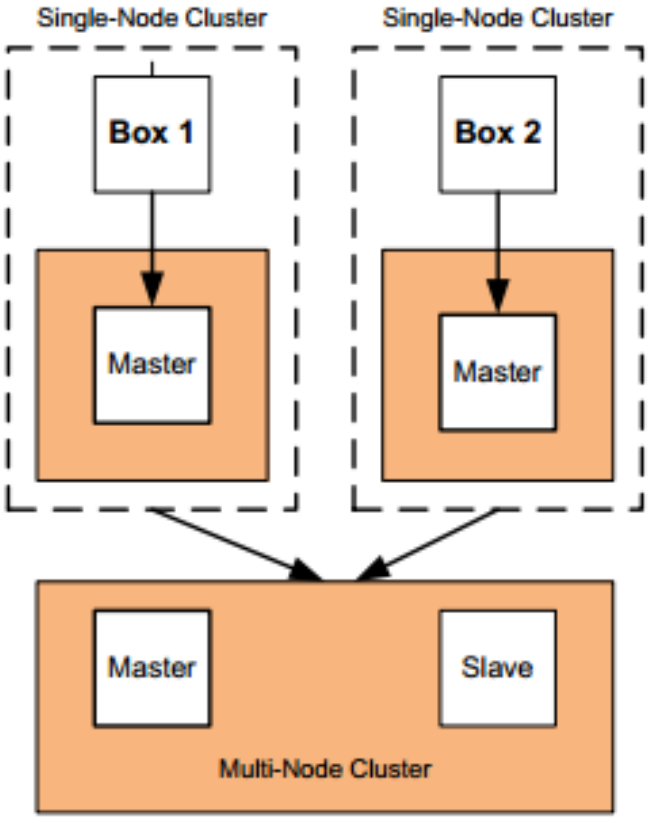
5.1.3.10 Aliran Data Hadoop MapReduce

Input Mapreduce berasal dari file-file yang akan diproses dalam cluster HDFS. File-file ini akan didistribusikan pada semua node yang ada. Jika menjalankan sebuah program Mapreduce maka akan menjalankan mapping task pada semua node. Semua mapping task adalah sama dan setiap mapping tidak memiliki identitas tertentu dalam mengeksekusi task, oleh karenanya



Gambar 5.15: Multi Node Cluster

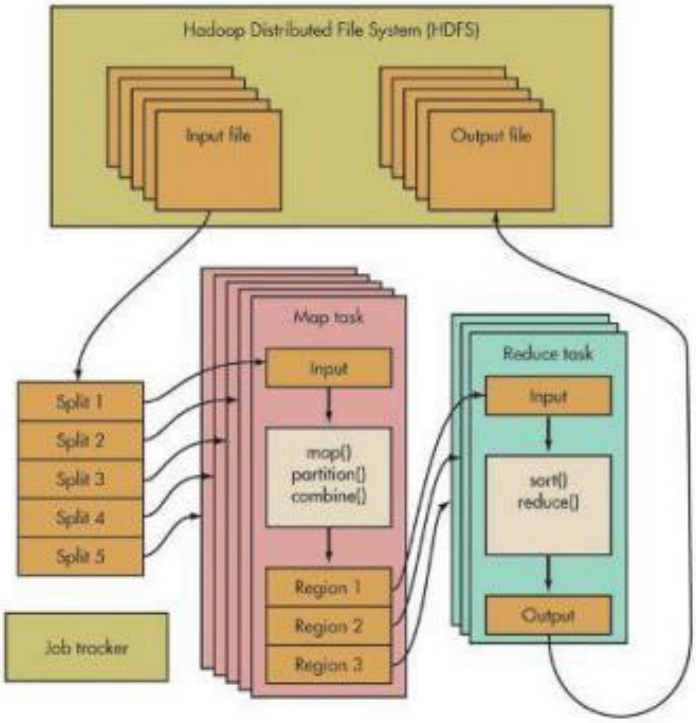
semua mapping dapat memproses semua input file yang manapun. Proses mapping tersebut tersebar dalam semua node yang ada dalam Cluster. Pasangan intermediate (key, value) akan didistribusikan untuk mendapatkan semua value dengan key yang sama saat semua proses mapping selesai dilakukan. Semua pasangan intermediate (key, value) dengan key yang sama tersebut akan diproses oleh satu reducer. Reducing task juga tersebar pada semua node yang terdapat dalam Cluster sebagaimana mapping task. Masing-masing mapping task mengabaikan mapping task yang lain dan tidak saling bertukar informasi mengenai proses mapping, demikian juga dengan reducing juga mengabaikan reducing task yang lain dan tidak bertukar informasi mengenai proses reducing. Salah satu kelebihan Hadoop Mapreduce, user tidak perlu memberikan informasi mekanisme transfer data dari node yang satu ke node yang lain karena telah dilakukan oleh Hadoop Mapreduce itu sendiri secara implisit menggunakan key dan value sebagai informasi. Di dalam Mapreduce terdapat beberapa proses yang terjadi hingga suatu output dapat dihasilkan sesuai dengan yang diinginkan oleh user. Mapreduce job adalah sebuah unit kerja yang ingin diimplementasikan. Mapreduce job ini terdiri dari input data, program Mapreduce dan informasi tentang konfigurasi. Hadoop menjalankan job tersebut dengan membaginya menjadi task-task. Task-task tersebut dibagi lagi menjadi dua yaitu map task dan reduce task. Ada dua tipe node yang akan mengontrol proses eksekusi job tersebut



Gambar 5.16: Sistem Multi Node Cluster

yaitu sebuah Jobtracker dan sejumlah Tasktracker. Jobtracker mengkoordinasi semua job yang akan dijalankan di sistem dengan menjadwalkan eksekusi task-task tersebut pada Tasktracker. Tasktracker menjalankan task dan memberikan laporan progres kepada Jobtracker. Jobtracker menyimpan record dari semua progress dari masing-masing job. Jika sebuah task gagal dieksekusi maka Jobtracker akan mengatur kembali eksekusi task tersebut pada Tasktracker yang lain. Hadoop membagi input untuk Mapreduce job menjadi bagian-bagian yang memiliki besaran tetap yang disebut input split. Hadoop membuat satu map task untuk setiap input split. Input split ini dijalankan oleh fungsi map yang telah dibuat oleh user untuk setiap record dalam input split.

Seperti pada Gambar 5.20, Aliran data dari suatu proses Mapreduce dimulai dari suatu input data. Input data tersebut kemudian dipecah menjadi

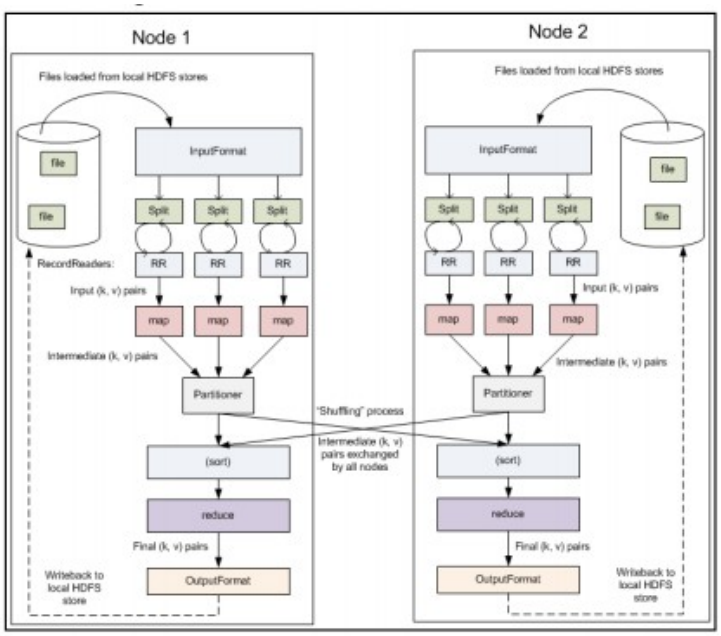


Gambar 5.17: Proses aliran data Hadoop Mapreduce

beberapa blok bagian. Data tersebut direplikasi dan disimpan dalam node-node (Datanode) sedangkan Metadata (nama file, jumlah replikasi, jumlah blok file) dari input tersebut tersimpan dalam satu Namenode. Untuk melakukan proses mapping, input tersebut dipanggil dari Datanode kemudian dilakukan proses splitting. Dari file input yang telah di-split tersebut akan dilakukan proses mapping. Proses mapping akan menghasilkan suatu output pasangan intermediate key dan value, dari semua key yang sama akan dikelompokkan menjadi satu. Pasangan intermediate key-value tersebut kumudian di-shuffle untuk dilakukan proses reducing yang tersebar di sejumlah komputer. Dari proses reducing tersebut akan dihasilkan suatu output yang diinginkan. Berikut proses detail yang terjadi dalam proses Mapreduce terangkum dalam gambar 5.21

Sebagaimana pada gambar 5.21 bagian-bagian aliran data Mapreduce dapat dijelaskan sebagai berikut.

- Input File Input file adalah tempat dimana data sebagai input dari



Gambar 5.18: Ilustrasi secara detail proses Mapreduce

Mapreduce pertama kali disimpan. Input file ini tersimpan dalam HDFS.

- Input format adalah cara bagaimana input tersebut akan dipecah (split) dan dibaca. Dalam Hadoop InputFormat memiliki class abstract yang dinamakan FileInputFormat. Saat mengeksekusi sebuah job, FileInputFormat menyediakan path yang berisi file-file yang akan dibaca. FileInputFormat akan membagi file-file ini menjadi satu atau lebih untuk masing-masing InputSplit. InputFormat membaca record melalui implementasi dari RecordReader. InputFormat mendefinisikan daftar task yang akan dieksekusi pada tahap mapping. Setiap task sesuai untuk sebuah satu input split. Standar InputFormat yang telah disediakan oleh Hadoop sebagai berikut:

  – Text Input Format TextInputFormat adalah default dari InputFormat,membaca baris-baris dari file teks. TextInputFormat sangat berguna untuk data yang tidak terformat seperti logfile. TextInputFormat mengambil nilai byte (byte offset of the line) dari setiap baris sebagai key dan isi dari baris tersebut sebagai value.

  - Key Value Input Format KeyInputFormat mem-parsing baris-baris menjadi pasangan key dan value berdasarkan karakter tab. Key yang diambil adalah semua karakter sampai ditemukannya karakter tab pada suatu baris dan sisanya adalah sebagai value.
  - Sequence File Input Format SequenceFileInputFormat membaca file biner yang spesifik untuk Hadoop. Key dan value pada SequenceFileInputFormat ini ditentukan oleh user.

- Input Split Input split mendeskripsikan sebuah unit kerja yang meliputi sebuah map task dalam sebuah program Mapreduce. File yang akan diproses dalam task sebelumnya melalui proses pemecahan (splitting) file menjadi beberapa bagian. Besaran pemecahan file ini mencapai 64 MB, nilai ini sama dengan besaran block data dalam HDFS.

- Record Reader Record Reader mengatur bagaimana cara mengakses data, menampung data dari sumber, dan mengubah data-data tersebut menjadi pasangan ¡key, value¿ yang dapat dibaca oleh mapper. Instance dari RecordReader disediakan oleh InputFormat.

- Mapper Mapper sebagai tahap pertama dari Mapreduce yang didefinisikan oleh user. Saat diberikan sebuah key dan sebuah value maka method map() akan memberikan pasangan ¡key, value¿ yang akan diteruskan kepada Reducer.

- Partition dan Shuffle Setelah map task pertama selesai dieksekusi, ada node-node yang masih memproses map task yang lain, disamping itu pula terjadi pertukaran intermediate output dari map task yang dibutuhkan oleh reducer. Proses pertukaran output map task dan diteruskannya output map task tersebut kepada reducer dinamakan proses shuffle. Ada sebagian dataset yang telah dieksekusi melalui map task namun ada juga yang masih dalam proses mapping oleh map task. Bagian data set yang telah melalui map task tersebut menghasilkan intermediate key value yang akan menjadi input dari reduce taks. Sebagian data set inilah yang dinamakan partition.

- Sort Setiap reduce task bertanggung jawab melakukan reducing terhadap value yang sesuai dengan intermediate key. Kumpulan intermediate key pada single-node secara otomatis akan di sorting oleh Hadoop sebelum akhirnya dilakukan proses reduce.

- Reducer Reducer adalah tahap kedua dari Mapreduce yang didefinisikan oleh user. Instance reducer akan dibuat untuk setiap reduce task, instance reducer ini sesuai dengan program yang dibuat oleh user. Untuk setiap key yang ada method reduce() akan dipanggil sekali.

- Output Format Outputformat berfungsi seperti InputFormat hanya saja OutputFormat berfungsi memberikan output file. Instance dari OutputFormat disediakan oleh Hadoop akan memberikan output file pada lokal disk HDFS.

- RecordWriter Sebagaimana RecordReader, RecordWriter berguna untuk menuliskan record ke file sebagaimana diinstruksikan oleh OutputFormat. User perlu mengkonfigurasi dan menentukan Mapreduce job yang akan dieksekusi, menentukan input, menentukan lokasi input. HDFS akan mendistribusikan job tersebut kepada Tasktracker dan memecah job tersebut ke dalam sejumlah map task, shuffle, sort dan reduce task. HDFS akan menempatkan output pada output directory dan memberitahu user jika job telah selesai dieksekusi.

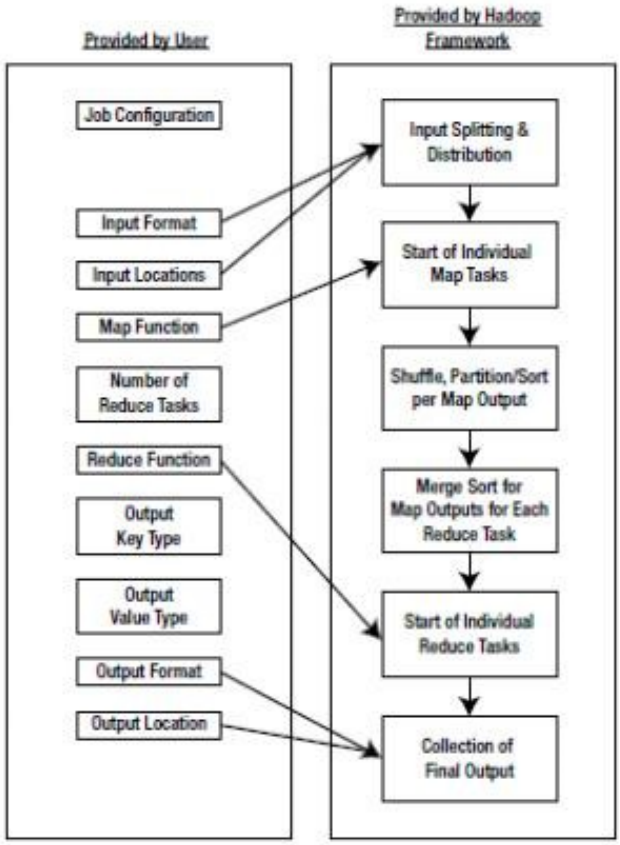
Pada gambar 5.22 di atas memberitahukan bagian-bagian Mapreduce job yang harus dispesifikasikan dan dibuat oleh user dan yang telah disediakan oleh Hadoop. Berikut adalah tabel penjelasan mengenai gambar tersebut.

Tabel 5.1 Identifikasi job configuration antara user dan hadoop framework

5.1.4 Apache Hive

Apache hive adalah tool selain untuk membentuk program Map Reduce. Apache Hive pertama kali dikembangkan oleh Facebook untuk melakukan data warehouse pada cluster Hadoop mereka yang sangat banyak. Selanjutnya Hive disumbangkan ke Apache Foundation untuk dikembangkan oleh komunitas open source. Hive lebih ditujukan untuk proses data warehouse diatas HDFS. Pada Apache Hive proses Map Reduce dituliskan dengan gaya yang sangat mirip dengan SQL yang pada umumnya ada di RDBMS. Contoh Script Hive yang menghitung kemunculan huruf

```
CREATE TABLE word_text(word STRING) COMMENT 'This is
    the word table'
ROW FORMAT DELIMITED FIELDS TERMINATED BY ' '
```



Gambar 5.19: Identifikasi job configuration dalam Mapreduce

```
LINES TERMINATED BY '\n';
LOAD DATA INPATH        'hdfs:/user/hue/
   word_count_text.txt' INTO TABLE word_text;
SELECT word, count(*) as count FROM word_text GROUP
   BY word;
```

Seperti terlihat pada script Hive diatas, tampak sekali kemiripan antara Script Hive dengan SQL. Script Hive diatas terdiri dari tiga statement yang masing-masing diakhiri dengan semi-colon (;). Bagian pertama adalah membentuk tabel yang akan menampung semua kata. Bagian kedua adalah menarik semua kata dari file di HDFS ke dalam tabel. Sedangkan bagian ketiga adalah

SQL query yang bisa dilakukan terhadap data yang sudah dimasukkan

Configuration	*Handled By*
Konfigurasi *Job*	*User*
Input split dan *distribution*	*Hadoop Framework*
Map and *Reduce Function <key, value>*	*User*
Proses *shuffle, partition,* dan *sort*	*Hadoop Framework*
Lokasi *distributed file system* untuk *input job*	*User*
Lokasi *distributed file system* untuk *output job*	*User*
Input format dan *output format*	*User*
Class fungsi *Mapper* dan *Reducer*	*User*
File *jar* yang berisi fungsi *map* dan *reduce*	*User*
Informasi *Job*	*Hadoop Framework*

ke tabel. Kita bisa melakukan berbagai query seperti SQL pada tabel yang sudah kita bentuk sehingga tidak hanya terbatas pada satu query saja. Output dari query tersebut bisa langsung ke layar, bisa ke file atau ke sistem eksternal menggunakan tool tambahan seperti Apache Thrift atau Apache Avro.

Hive cukup modular sehingga bisa di gabungkan dengan banyak tool lain seperti Spring, Apache Thrift dan Apache Avro. Pig[48]

5.1.5 Apache Spark

Spark adalah sebuah framework untuk komputasi terdistribusi, framework ini memudahkan dalam menulis program yang dapat berjalan secara paralel pada banyak node dalam sebuah cluster komputer. Memiliki kemampuan untuk penjadwalan sumber daya, pendelegasian tugas, eksekusi, pelacakan, dan komunikasi antar node, serta pengolahan data secara paralel. Juga merupakan API yang dapat bekerja pada data terdistribusi. Hal ini membuat Spark mirip dengan Apache Hadoop namun dengan dasar arsitektur yang berbeda. Spark dimulai sebagai sebuah proyek penelitian di University of California, Berkeley. Penelitian itu difokuskan pada kasus penggunaan algoritma machine learning terdistribusi. Penelitian tersebut berhasil merancang sebuah aplikasi yang memiliki kinerja tinggi dan dapat bekerja secara iteratif/berulang untuk data yang sama. Kinerja tersebut diperolah dari teknik dataset caching di memory, dikombinasikan dengan latency rendah dan overhead untuk memulai tugas komputasi paralel. Selain itu beberapa fitur lainnya seperti toleransi kesalahan, struktur data memory terdistribusi yang fleksibel, dan powerful API, Spark telah terbukti berguna untuk berbagai skala besar tugas pengolahan data, berdasarkan pada mechine learning dan analisis berulang.[49] Spark inilah yang kemudian disumbangkan ke Apache Software

Foundation.

Spark menyediakan programmer dengan antar muka pemrograman aplikasi yang berpusat pada struktur data yang disebut Resilient Distributed Dataset (RDD), sebuah multiset read-only data yang terdistribusi pada sebuah cluster komputer dengan fitur toleransi kesalahan.[50] Hal ini dikembangkan guna mengatasi keterbatasan pada paradigma komputasi Map Reduce yang memaksa penggunaan struktur dataflow linear khusus pada program terdistribusi, dimana pada program Map Reduce yaitu membaca data masukan dari disk, memetakan (Map) fungsi seluruh data, Mengurasi (Reduce) hasil pemetaan, dan menyimpan hasilnya ke disk. Spark RDD berfungsi sebagai working set untuk program terdistribusi yang menggunakan shared memory terdistribusi. Hal ini yang menyebabkan secara performa spark lebih unggul dalam hal kecepatan pemrosesan data.

Library utama pada Spark :

- Spark SQL Spark SQL adalah komponen di atas Spark yang memperkenalkan abstraksi data baru yang disebut DataFrame, yang menyediakan dukungan untuk data terstruktur dan semi-terstruktur. Spark SQL menyediakan bahasa domain-spesifik untuk memanipulasi DataFrame menggunakan Scala, Java, atau Python. Spark SQL menyediakan dukungan pada bahasa SQL sebagai antarmuka baris perintah dan juga menyediakan dukungan terhadap server ODBC / JDBC.

- Spark Streaming Spark Streaming memanfaatkan penjadwalan cepat untuk melakukan analisis streaming. Hal ini dibutuhkan untuk menganalisis data secara realtime. Spark Streaming support data dari Kafka, Flume, Twitter, ZeroMQ, Kinesis, dan TCP/IP Socket.

- MLlib (Machine Learning) Spark MLlib adalah kerangka mesin pembelajaran dan statistik terdistribusi di atas Spark. Dukungan Spark MLlib sebagai algoritma machine learning dan statistik meliputi banyak hal yaitu:

  - Summary Statistics, Correlation, Stratified Sampling, Hypothesis Testing, Random Data Generation.
  - Klasifikasi dan Regresi : Support Vector Machine, Logistic Regression, Decision Trees, naive Bayes Classification.
  - Teknik Collaborative Filtering termasuk Alternating Least Squares (ALS)

  - Metode Cluster Analysis termasuk K-Means dan Latent Dirichlet Allocation (LDA)
  - Teknik Dimensionality Reduction seperti Singular Value Decomposition (SVD), dan Principal Component Analysis (PCA)
  - Ekstraksi Fitur dan Fungsi Transformasi
  - Algoritma Optimasi seperti Stochastic Gradient Descent, Limited-memry BFGS (-L-BFGS)

- GraphX (graph) GraphX adalah kerangka pengolahan grafik terdistribusi di atas Apache Spark. Karena didasarkan pada RDDS, yang berubah, grafik yang berubah dan dengan demikian GraphX tidak cocok untuk grafik yang perlu diperbarui, apalagi dengan cara transaksional seperti database grafik. GraphX menyediakan dua API terpisah untuk pelaksanaan secara besar-besaran algoritma paralel (seperti PageRank): abstraksi Pregel, dan gaya MapReduce API yang lebih umum. Tidak seperti pendahulunya Bagel, yang secara resmi tidak digunakan dalam Spark 1.6, GraphX memiliki dukungan penuh untuk grafik properti (grafik di mana properti dapat dilampirkan untuk tepi dan titik). GraphX dapat dilihat sebagai basis memory dari Apache Giraph, yang dimanfaatkan Hadoop MapReduce berbasis disk. Seperti Apache Spark, GraphX awalnya dimulai sebagai sebuah proyek penelitian di UC Berkeley AMPLab dan Databricks, dan kemudian disumbangkan ke Apache Software Foundation dan proyek Spark.

Beberapa keunggulan Spark : [51]

- Kecepatan Program Spark berjalan hingga 100 kali lebih cepat dibanding Hadoop MapReduce di memory, atau 10 kali lebih cepat pada disk. Apache Spark memiliki mesin eksekusi DAG canggih yang mendukung aliran data siklik dan komputasi in-memory.

- Mudah Digunakan Menulis aplikasi secara cepat di Java, Scala, Python, dan R. Spark memiliki lebih dari 80 operator level tinggi yang membuatnya mudah untuk membangun aplikasi paralel. Dan dapat digunakan secara interaktif dari shell Scala, shell Python dan shell R.

- Umum Menggabungkan SQL, Streaming dan analisis kompleks. Kekuatan Spark karena memiliki banyak library termasuk SQL dan DataFrame,

MLlib untuk machine learning, GraphX, dan Spark Streaming. Semua library ini dapat digunakan dalam satu aplikasi yang sama.

- Berjalan di mana saja Spark dapat berjalan di atas Hadoop, Mesos, standalone, ataupun di cloud. Spark juga dapat mengakses sumber data yang beragam seperti HDFS, Cassandra, HBase, dan S3. Spark dapat dijalankan pada mode Standalone, EC2, Hadoop YARN, atau Apache Mesos. Akses data di HDFS, Cassandra, HBase, Hive, Tachyon, dan berbagai Data Source Hadoop.

5.1.6 Machine Learning

Sejak komputer ditemukan, para peneliti telah berfikir adakah kemungkinan dapat belajar. Jika kita mengerti bagaimana cara memprogram komputer agar mereka dapat belajar, dan berkembang dari pengalaman secara otomatir, hasilnya akan luar biasa dramatis. Bayangkan komputer belajar dari data-data medical untuk menemukan cara baru menangani suatu penyakit, belajar dari pengalaman untuk mengoptimumkan energy yang dibutuhkan untuk melakukan pekerjaan rumah tangga, dan lain-lain. Sejak tahun 1980-an, bidang ilmu soft computing mulai muncul dan berkembang berdampingan dengan ilmu konvensional hard computing. Adapun yang membedakan antara kedua ilmu ini adalah setelah deprogram, hard computing akan memberikan hasil yang sama untuk input yang sama, sementara soft computing akan belajar dari input-input yang diberikan sebelumnya untuk memberikan hasil yang lebih akurat di masa depan.

1. Definisi Machine Learning adalah salah satu disiplin ilmu dari Computer Science yang mempelajari bagaimana membuat komputer atau mesin itu mempunyai suatu kecerdasan (Artificial Intellegence).

2. Tipe Pembelajaran Tipe pembelajaran dalam Machine Learning terbagi dalam tiga kategori yang tergantung pada sinyal atau umpan balik yang tersedia dalam sistem pembelajaran.

   (a) Supervised Learning Supervised Learning adalah teknik pembelajaran mesin dengan membuat suatu fungsi dari data latihan. Data latihan terdiri dari pasangan nilai input (biasa dalam bentuk vector), dan output yang diharapkan untuk input yang bersangkutan. Tugas dari mesin supervised learning adalah memprediksi

nilai fungsi untuk semua nilai input yang mungkin, setelah mengalami sejumlah data latihan. Untuk tujuan ini, mesin harus dapat melakukan proses generalisasi dari data latihan yang diberikan, untukmemprediksikan nilai output dari input yang belum pernah diberikan sebelumnya dengan cara yang masuk akal. Supervised Learning biasanya digunakan untuk klasifikasi (classification) yaitu algoritma yang belajar berdasarkan sekumpulan contoh pasangan input-output yang diinginkan dalam jumlah yang cukup besar. Algoritma ini mengamati contoh-contoh tersebut dan kemudian menghasilkan sebuah model yang mampu memetakan input yang baru menjadi output yang tepat. Salah satu contoh yang paling sederhana adalah terdapat sekumpulan contoh masukan berupa umur seseorang dan contoh keluaran yang berupa tinggi badan orang tersebut. Algoritma pembelajaran melalui contoh, mengamati contoh tersebut dan kemudian mempelajari sebuah fungsi yang pada akhirnya dapat memperkirakan tinggi badan seseorang berdasarkan inputan umur orang tersebut.

(b) Unsupervised Learning Teknik ini menggunakan prosedur yang berusaha untuk mencari partisi dari sebuah pola. Unsupervised learning mempelajari bagaimana sebuah sistem dapat belajar untuk merepresentasikan pola input dalam cara yang menggambarkan struktur statistical dari keseluruhan pola input. Berbeda dari supervised learning, unsupervised learning tidak memiliki target output yang eksplisit atau tidak ada pengklasifikasian input. Dalam machime learning, teknik unsupervised sangat penting. Hal ini dikarenakan cara bekerjanya mirip dengan cara bekerja otak manusia. Dalam melakukan pembelajaran, tidak ada informasi dari contoh yang tersedia. Oleh karena itu, unsupervised learning menjadi esensial. Unsupervised learning biasanya digunakan untuk pengelompokan (clustering). Salah satu contoh dalam dunia nyata misalnya supir taksi yang secara perlahan-lahan menciptakan konsep macet dan tidak macet tanpa pernah diberikan contoh oleh siapapun.

(c) Reinforcement Learning Reinforcement Learning adalah sebuah model pembelajaran terhadap apa yang dilakukan, dan sebagai umpan balik bagaimana memetakan perubahan situasi lingkungan terhadap aksi yang dilakukan untuk memaksimalkan keuntun-

#	Cuaca	Temperatur	Kecepatan Angin	Berolah-raga
1	Cerah	Normal	Pelan	Ya
2	Cerah	Normal	Pelan	Ya
3	Hujan	Tinggi	Pelan	Tidak
4	Cerah	Normal	Kencang	Ya
5	Hujan	Tinggi	Kencang	Tidak
6	Cerah	Normal	Pelan	Ya

Gambar 5.20: Tabel Cuaca

gan. Model proses pembelajar ini tidak harus mendikte aksi yang mana yang harus dilakukan dulu seperti umumnya pada model mesin learning. Tetapi sistem harus menjelajahi dulu seluruh aksi yang mana yang menghasilkan reward terbesar dengan cara mencobanya dan mengerjakannya. Sebagai contoh, sangatlah sulit untuk mengajarkan seseorang untuk mengemudikan mobil, tetapi dengan memberikan beberapa nilai negative untuk menabrak, melencang dari jalur perlahan-lahan seseorang dapat mengemudikan mobil dengan baik.

3. Ide Machine Learning Ide dasar adanya konsep machine learning adalah adanya penyajian keputusan berdasarkan fakta seperti beriktu ini: Fakta harian dalam 6 hari dan keputusan untuk berolah raga sebagai berikut:

   Sehingga muncul kasus seperti berikut ini yang membutuhkan suatu penyelesaian Ketika cuaca cerah, apakah akan berolahraga? Ketika cuaca cerah dan temperatur normal, apakah akan berolahraga?

   (a) Data Training Dari pengolahan table cuaca akan diperoleh data training sebagai berikut:

   Keterangan:

   i. Attribut adalah kolom data, terdapat atribut itu sendiri dan target
   ii. Instance adalah isi dari atribut sebagai contoh seperti atribut cuaca mempunyai instance cerah dan hujan, sering ditulis dengan cuaca = cerah, hujan.
   iii. Record atau tuple adalah baris data

Key Attribut Target

Day	Cuaca	Temperatur	Kecepatan Angin	Berolah-raga
D1	Cerah	Normal	Pelan	Ya
D2	Cerah	Normal	Pelan	Ya
D3	Hujan	Tinggi	Pelan	Tidak
D4	Cerah	Normal	Kencang	Ya
D5	Hujan	Tinggi	Kencang	Tidak
D6	Cerah	Normal	Pelan	Ya

Gambar 5.21: Data training table cuaca

#	Cuaca	Temperatur	Kecepatan Angin	Berolah-raga
3	Hujan	Tinggi	Pelan	Tidak
5	Hujan	Tinggi	Kencang	Tidak

Gambar 5.22: Data konsisten

(b) Macam-macam data Dari data cuaca, macam-macam data yang ada dalam machine learning meliputi berikut ini:

i. Data konsisten
Atribut cuaca dan temperature mempunyai nilai yang sama dalam satu keputusan (berolah raga), maka data ini adalah data yang konsisten.

ii. Data tidak konsisten
Tidak satupun atribut yang mempunyai nilai yang sama dalam satu keputusan (berolah raga), maka data ini adalah data yang tidak konsisten.

iii. Data Bias
Dari data di atas, data ke-4. Data ini mempunyai keputusan

#	Cuaca	Temperatur	Kecepatan Angin	Berolah-raga
1	Cerah	Normal	Pelan	Ya
2	Cerah	Tinggi	Pelan	Ya
4	Hujan	Normal	Kencang	Ya
6	Cerah	Normal	Pelan	Ya

Gambar 5.23: Data tidak konsisten

#	Cuaca	Temperatur	Kecepatan Angin	Berolah-raga
1	Cerah	Normal	Pelan	Ya
2	Cerah	Normal	Pelan	Ya
3	Hujan	Normal	Pelan	Ya
4	Cerah	Normal	Pelan	Tidak

Gambar 5.24: Data Bias

yang berbeda dengan data 1 dan data 2. Tetapi instance pada semua attributnya sama, sehingga data ini disebut dengan data bias.

5.2 Implementasi Hadoop

Pada buku ini Hadoop diimplementasikan pada multi node cluster. Adapun peralatan yang digunakan sebagai berikut :

1. 2 PC Server dengan spesifikasi sebagai berikut :

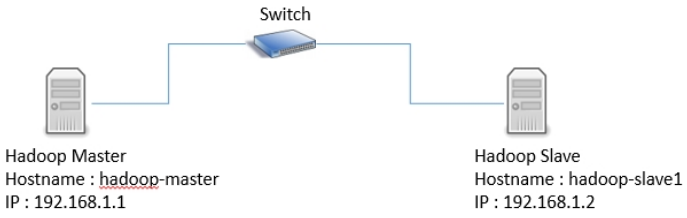
Table 5.1: Spesifikasi Perangkat Keras

PC1 (Master and Slave)	**PC 2 (Slave)**
Ubuntu GNU/Linux 14.04	Ubuntu GNU/Linux 14.04
Intel(R)Core(TM)2 Quad CPU Q8400 @ 2.66GHz	Intel(R)Core(TM)2 Quad CPU Q8400 @ 2.66GHz
Memory 8 GB	Memory 4 GB
HD 1 TB	HD 1 TB

2. Perangkat Lunak yang digunakan dalam Ubuntu GNU/Linux 14.04 :
   - Apache Hadoop 2.6.0
   - Apache Hive 1.2.1
   - Apache Spark 1.6.1
   - Python 2.7.0
   - Scala 2.4.0
   - Java 7 Open-JDK

- PostgreSQL 9.3
- OpenSSH-Server

Model infrastruktur dari Hadoop yang dibangun adalah



Gambar 5.25: Desain Infrastruktur Server Hadoop

5.2.1 Instalasi Hadoop

5.2.1.1 Instalasi dan Konfigurasi Hadoop

Instalasi dan konfigurasi dilakukan pertama kali pada komputer yang bertindak sebagai hadoop-master (single node), untuk menjadikan multinode cluster akan tinggal melakukan syncronize dengan komputer slave nantinya dan dengan melakukan perubahan konfigurasi sedikit maka multinode cluster akan terkonfigurasi. Tahapan instalasi dan konfigurasi Hadoop di 192.168.1.1. Pertama lakukan instalasi java open=jdk pada ubuntu linux, karena hadoop merupakan framework yang berbasis pada java.

```
$sudo apt-get install openjdk- 7-jdk
```

Pastikan java terinstall dengan mengecek versi java dengan perintah pada terminal java -version. Selanjutnya install OpenSSH Server.

```
$sudo apt-get install openssh-server
```

Selanjutnya menambahkan user dan group baru untuk hadoop

```
$sudo addgroup hadoop
$sudo adduser -ingroup hadoop hduser
$sudo adduser hduser sudo
#sudo adduser hduser
```

Selanjutnya lakukan konfigurasi pada SSH Access bertujuan dalam melakukan remote antar node dalam hadoop cluster oleh hadoop master tanpa perlu memasukkan autentikasi SSH.

- pindah user ke hduser yang telah dibuat, su hduser
- Ketik "ssh-keygen -t rsa -P"
- Untuk mengaktifkan SSH Access, copy keys ke home pada folder .ssh menggunakan perintah :

```
cat $HOME/.ssh/id_rsa.pub >> $HOME/.ssh/
   authorized_keys
```

Lakukan pengetesan dengan melakukan "ssh localhost", jika tidak ada permintaan password berarti konfigurasi berhasil.

Disable dukungan ipv6 dengan cara :

- Buka sysctl.conf di /etc/sysctl.conf
- Tambahkan 3 baris berikut :

```
net.ipv6.conf.all.disable_ipv6 = 1
net.ipv6.conf.default.disable_ipv6 = 1
net.ipv6.conf.lo.disable_ipv6 = 1
```

Langkah selanjutnya adalah melakukan instalasi hadoop 2.6.0.

- Extract hadoop.2.6.0.tar.gz
- Pindahkan hasil extract ke folder /usr/local/hadoop
- ganti ke pemilikan folder hadoop kepada hduser

```
$sudo chown -R hduser:hadoop /usr/local/hadoop
```

- ganti permission ke mode all x

```
$chmod +x -R /usr/local/hadoop
```

Langkah selanjutnya adalah melakukan konfigurasi pada global variable dengan mengedit file .bashrc:

```
nano ~/.bashrc
```

Tambahkan path Java pada baris terakhir:

```
export  JAVA_HOME='/usr/lib/jvm/java-7-openjdk-amd64
   '
#Add Hadoop bin/ directory to PATH export
PATH=$PATH:$HADOOP_HOME/bin:$JAVA_PATH/bin:
   $HADOOP_HOME/sbin
```

Tambahkan variable untuk hadoop environment :

```
export HADOOP_PREFIX=/usr/local/hadoop
export HADOOP_HOME=/usr/local/hadoop
export HADOOP_MAPRED_HOME=${HADOOP_HOME}
export HADOOP_COMMON_HOME=${HADOOP_HOME}
export HADOOP_HDFS_HOME=${HADOOP_HOME}
export YARN_HOME=${HADOOP_HOME}
export HADOOP_CONF_DIR=${HADOOP_HOME}/etc/hadoop
export HADOOP_COMMON_LIB_NATIVE_DIR=${HADOOP_PREFIX
   }/lib/native export
HADOOP_OPTS="Djava.library.path=$HADOOP_PREFIX/lib"
```

Selain itu set juga java home pada file hadoop-env.sh pada /usr/local/hadoop/conf/hadoop-env.sh tambahkan :

```
export JAVA_HOME="/usr/lib/jvm/java-7-openjdk-amd64"
```

Setelah itu reload setting dengan perintah "source /.bashrc

Selanjutnya melakukan konfigurasi pada Hadoop yang berada pada direktori /usr/local/hadoop/etc/hadoop/ :

- Edit yarn-site.xml dan replace dengan konfigurasi di bawah ini :

```
<?xml version="1.0"?>
<configuration>
<property>
        <name>yarn.nodemanager.aux-services</
           name>
        <value>mapreduce_shuffle</value>
</property>
<property>
```

```
        <name>yarn.nodemanager.aux-services.
           mapreduce.shuffle.class</name>
        <value>org.apache.hadoop.mapred.
           ShuffleHandler</value>
</property>
<property>
        <name>yarn.resourcemanager.resource-
           tracker.address</name>
        <value>hadoop-master:8025</value>
</property>
<property>
        <name>yarn.resourcemanager.scheduler.
           address</name>
        <value>hadoop-master:8035</value>
</property>
<property>
        <name>yarn.resourcemanager.address</name
           >
        <value>hadoop-master:8050</value>
</property>
</configuration>
```

- Selanjutnya edit file core-site.xml dan replace dengan konfigurasi berikut :

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="
   configuration.xsl"?>
<configuration>
<property>
        <name>fs.default.name</name>
        <value>hdfs://hadoop-master:9000</value>
</property>
<property>
        <name>hadoop.proxyuser.hduser.hosts</
           name>
        <value>*</value>
</property>
```

```
<property>
        <name>hadoop.proxyuser.hduser.groups</
           name>
        <value>*</value>
</property>
</configuration>
```

- Selanjutnya edit file mapred-site.xml dan tambahkan konfigurasi berikut :

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="
   configuration.xsl"?>
<configuration>
<property>
        <name>mapreduce.job.tracker</name>
        <value>hadoop-master:9001</value>
</property>
<property>
        <name>mapred.framework.name</name>
        <value>yarn</value>
</property>
</configuration>
```

- Selanjutnya edit hdfs-site.xml dan tambahkan konfigurasi berikut :

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="
   configuration.xsl"?>
<configuration>
<property>
        <name>dfs.replication</name>
        <value>2</value>
</property>
<property>
        <name>dfs.namenode.name.dir</name>
        <value>/usr/local/hadoop_store/hdfs/
           namenode</value>
</property>
```

```
<property>
<name>dfs.datanode.data.dir</name>
<value>/usr/local/hadoop_store/hdfs/namenode/
   datanode</value>
</property>
</configuration>
```

Kemudian buat beberapa folder sesuai konfigurasi pada hdfs-site.xml

```
sudo mkdir -p /usr/local/hadoop_store/hdfs/namenode
sudo mkdir -p /usr/local/hadoop_store/hdfs/namenode/
   datanode
```

Setelah itu lakukan format terhadap namenode

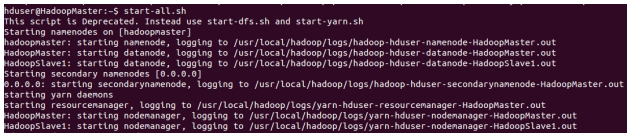
```
$hadoop namenode -format
```

Jika berhasil maka akan ada pesan bahwa namenode has been successfully formatted.

Selanjutnya jalankan hadoop dengan perintah :

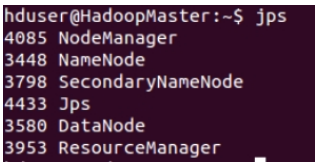
```
start-all.sh
```

Akan muncul tampilan seperti berikut :

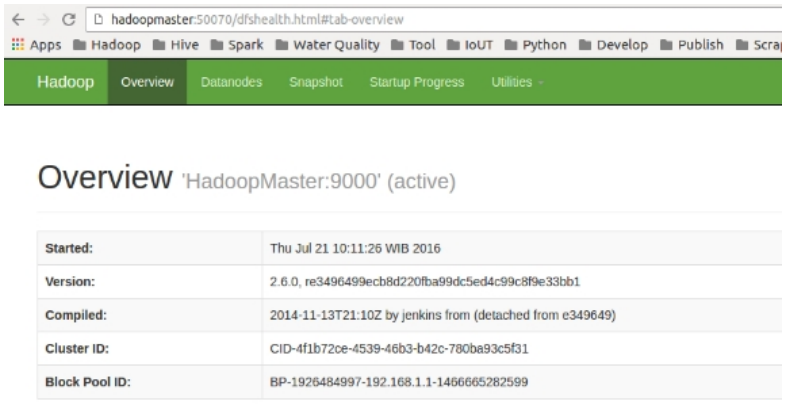


Gambar 5.26: Tampilan eksekusi start-all.sh

Untuk melihat daemon hadoop yang berjalan gunakan perintah jps (dari hduser)



Gambar 5.27: Hasil Eksekusi jps yang memperlihatkan daemon hadoop yang berjalan di hadoop master



Gambar 5.28: Hadoop Web Interface

5.2.1.2 Instalasi dan Konfigurasi Multi-Node Cluster

Langkah pertama adalah mengidentifikasi secara benar hostname dari tiap-tiap node. Hostname dapat di cek pada file /etc/hostname,

```
192.168.1.1    hadoop-master
192.168.1.2    hadoop-slave1
```

Konfigurasi ini diterapkan pada kedua node. Membuat group dan user hadoop dan hduser pada semua node

```
$sudo addgroup hadoop
$sudo adduser -ingroup hadoop hduser
```

Kemudian menambahkan hduser ke sudoers, dengan perintah :

```
$sudo usermod -a -G sudo hduser
```

Kemudian menginstall rsync untuk syncronize file hadoop pada semua node

```
$sudo at-get install rsync
```

Selanjutnya mengedit core-site.xml, hdfs-site.xml, yarn-site.xml, mapred-site.xml. Selanjutnya file masters pada /usr/local/hadoop/etc/hadoop/masters tambahkan

```
hadoop-master
```

dan file slaves pada /usr/local/hadoop/etc/hadoop/slaves tambahkan

```
hadoop-master
hadoop-slave1
```

Langkah selajutnya mendistribusikan hadoop file dari master node ke slave node

```
$sudo rsync -avxP /usr/local/hadoop/hduser@hadoop-
   slave1:/usr/local/hadoop/
```

Dengan perintah rsync di atas akan berbagi file yang tersimpan pada /usr/local/hadoop pada master node ke slave node. Jadi pada sisi slave tidak perlu mendownload lagi. Selanjutnya mengupdate file .bashrc pada sisi slave.

Selanjutnya mengatur SSH Passwordless pada slave node untuk memudahkan proses menjalankan dan menghentikan hadoop daemon pada slave node dari master node.

```
ssh-copy-id    i  /home/hduser/.ssh/id_rsa.pub
   hduser@hadoop-master ssh-copy-id    i  /home/
   hduser/.ssh/id_rsa.pub hduser@hadoop-slave1
```

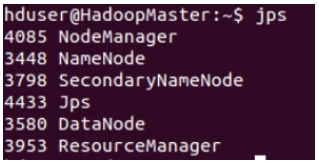
Selanjutnya lakukan format pada namenode

```
$hadoop namenode -format
```

Lalu menjalankan hadoop framework pada master node

```
$start-all.sh
```

Hasil eksekusi jps pada master node

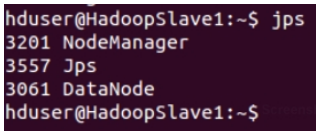


Gambar 5.29: Hasil Eksekusi jps di hadoop master

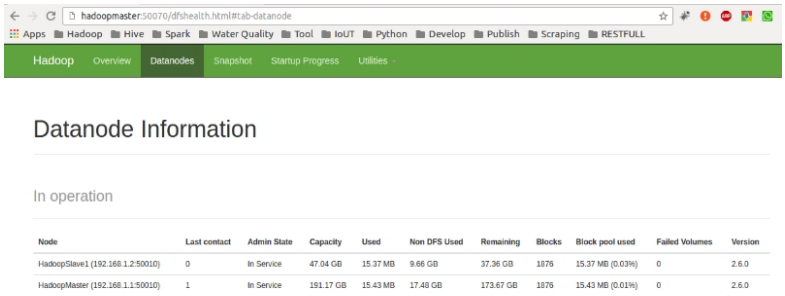
Hasil eksekusi jps pada slave node
Informasi pada Hadoop Web Interface

5.2.2 Instalasi Hive

angkah-langkah instalasi Apache Hive 1.2.1 :



Gambar 5.30: Hasil Eksekusi jps di hadoop slave



Gambar 5.31: Informasi master and slave node pada hadoop web interface

- extract apache-hive-1.2.1-bin.tar.gz
- pindahkan apache-hive-1.2.1-bin ke /usr/local/hive
- seth path environment pada file .bashrc

```
export HIVE_HOME=/usr/local/hive export PATH=
   $PATH:$HIVE_HOME/bin
export CLASSPATH=$CLASSPATH:/usr/local/hadoop/
   hadoop/lib/*:.
export CLASSPATH=$CLASSPATH:/usr/local/hadoop/
   hive/lib/*:
```

- reload seth path environmentnya

```
$source ~/.bashrc
```

Jalankan hive dengan terlebih dahulu menjalankan service metastore nya :

```
$hive --service metastore
```

Kemudian jalankan service hiveserver2 :

```
$hive --service hiveserver2
```

Gambar 5.32: Menjalankan service metastore hive

Gambar 5.33: Menjalankan service hiveserver2 hive

Kemudian jalankan hive jika ingin mengakses hive shell :

```
$hive
```

Gambar 5.34: Mengakses hive shell

5.2.3 Instalasi Spark

Langkah-langkah instalasi Apache Spark sebagai berikut:

- Download scala-2.10.5.tgz
- Download spark-1.6.1-bin-hadoop2.6.tgz dari www.spark.apache.org
- extract scala-2.10.5.tgz dan spark-1.6.1-bin-hadoop2.6.tgz
- pindahkan hasil extract scala-2.10.5 ke /usr/local/scala
- pindahkan hasil extract spark-1.6.1-bin-hadoop2.6 ke /usr/local/spark
- seth path environment pada file .bashrc

```
#SCALA
export SCALA_HOME=/usr/local/scala
export PATH=$PATH:$SCALA_HOME/bin
#SPARK
export SPARK_HOME=/usr/local/spark
export PATH=$PATH:$SPARK_HOME/bin
export PYTHONPATH=$SPARK_HOME/python/:
   $PYTHONPATH
export PYTHONPATH=$SPARK_HOME/python/lib/py4j
   -0.9-src.zip:$PYTHONPATH
```

- Edit spark-env.sh pada /usr/local/spark/conf

```
export SCALA_HOME=/usr/local/scala
export SPARK_WORKER_MEMORY=1g
export SPARK_WORKER_INSTANCES=2
export SPARK_WORKER_DIR=/usr/local/spark/
   sparkdata
export SPARK_MASTER_IP=hadoop-master
```

- Edit slaves

```
hadoop-master
hadoop-slave1
```

- Edit spark-defaults.conf

```
spark.master spark://hadoop-master:7077
```

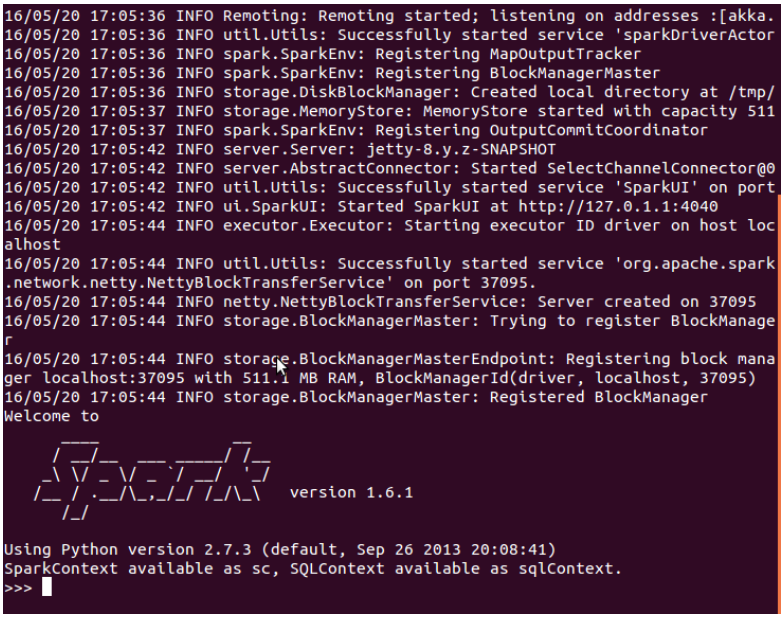
- Terapkan pada semua node

Jalankan python spark shell dengan mengetikkan

```
$pyspark
```

5.2.4 Machine Learning dengan Spark

Spark mendukung berbagai macam metode untuk klasifikasi biner, klasifikasi multiclass, dan analisis regresi.



Gambar 5.35: pyspark shell

- Klasifikasi Biner (Binary Classification) Metode yang disupport adalah linear SVMs, logistic regression, decision trees, random forests, gradient-boosted trees, naive Bayes.

- Klasifikasi Multiclass (Multiclass Classification) Metode yang disupport adalah Logistic regression, decision trees, random forest, naive Bayes.

- Regression Metode yang disupport adalah linear least squares, Lasso, ridge regression, decision trees, random forests, gradient-boosted trees, isotonic regression.

Clustering merupakan penyelesaian kasus untuk unsupervised learning. Model yang disupport oleh spark adalah

- K-means

- Gaussian mixture

- Power iteration clustering (PIC)

- Latent Dirichlet Allocation (LDA)
- Bisecting k-means
- Streaming k-means

Contoh source classification, regression, dan clustering pada spark dengan bahasa python:[56]

- Linear Support Vector Machine (SVMs) (Classification)

```
from pyspark.mllib.classification import
   SVMWithSGD, SVMModel
from pyspark.mllib.regression import
   LabeledPoint

# Load and parse the data
def parsePoint(line):
    values = [float(x) for x in line.split(' ')]
    return LabeledPoint(values[0], values[1:])

data = sc.textFile("data/mllib/sample_svm_data.
   txt")
parsedData = data.map(parsePoint)

# Build the model
model = SVMWithSGD.train(parsedData, iterations
   =100)

# Evaluating the model on training data
labelsAndPreds = parsedData.map(lambda p: (p.
   label, model.predict(p.features)))
trainErr = labelsAndPreds.filter(lambda (v, p):
   v != p).count()/ float(parsedData.count())
print("Training Error = " + str(trainErr))

# Save and load model
model.save(sc, "myModelPath")
sameModel = SVMModel.load(sc, "myModelPath")
```

- Logistic Regression (Classification)

```
from pyspark.mllib.classification import
   LogisticRegressionWithLBFGS,
   LogisticRegressionModel
from pyspark.mllib.regression import
   LabeledPoint

# Load and parse the data
def parsePoint(line):
    values = [float(x) for x in line.split(' ')]
    return LabeledPoint(values[0], values[1:])

data = sc.textFile("data/mllib/sample_svm_data.
   txt")
parsedData = data.map(parsePoint)

# Build the model
model = LogisticRegressionWithLBFGS.train(
   parsedData)

# Evaluating the model on training data
labelsAndPreds = parsedData.map(lambda p: (p.
   label, model.predict(p.features)))
trainErr = labelsAndPreds.filter(lambda (v, p):
   v != p).count() / float(parsedData.count())
print("Training Error = " + str(trainErr))

# Save and load model
model.save(sc, "myModelPath")
sameModel = LogisticRegressionModel.load(sc, "
   myModelPath")
```

- Naive Bayes (Classification)

```
from pyspark.mllib.classification import
   NaiveBayes, NaiveBayesModel
from pyspark.mllib.linalg import Vectors
```

```
from pyspark.mllib.regression import
   LabeledPoint

def parseLine(line):
    parts = line.split(',')
    label = float(parts[0])
    features = Vectors.dense([float(x) for x in
       parts[1].split(' ')])
    return LabeledPoint(label, features)

data = sc.textFile('data/mllib/
   sample_naive_bayes_data.txt').map(parseLine)

# Split data aproximately into training (60%)
   and test (40%)
training, test = data.randomSplit([0.6, 0.4],
   seed=0)

# Train a naive Bayes model.
model = NaiveBayes.train(training, 1.0)

# Make prediction and test accuracy.
predictionAndLabel = test.map(lambda p: (model.
   predict(p.features), p.label))
accuracy = 1.0 * predictionAndLabel.filter(
   lambda (x, v): x == v).count() / test.count()

# Save and load model
model.save(sc, "target/tmp/myNaiveBayesModel")
sameModel = NaiveBayesModel.load(sc, "target/tmp
   /myNaiveBayesModel")
```

- Decision Trees (Classification)

```
from pyspark.mllib.tree import DecisionTree,
   DecisionTreeModel
from pyspark.mllib.util import MLUtils
```

```
# Load and parse the data file into an RDD of
   LabeledPoint.
data = MLUtils.loadLibSVMFile(sc, 'data/mllib/
   sample_libsvm_data.txt')
# Split the data into training and test sets
   (30% held out for testing)
(trainingData, testData) = data.randomSplit
   ([0.7, 0.3])

# Train a DecisionTree model.
#  Empty categoricalFeaturesInfo indicates all
   features are continuous.
model = DecisionTree.trainClassifier
(trainingData, numClasses=2,
   categoricalFeaturesInfo={},
impurity='gini', maxDepth=5, maxBins=32)

# Evaluate model on test instances and compute
   test error
predictions = model.predict(testData.map(lambda
   x: x.features))
labelsAndPredictions = testData.map(lambda lp:
   lp.label).zip(predictions)
testErr = labelsAndPredictions.filter(lambda (v,
    p): v != p).count() / float(testData.count()
   )
print('Test Error = ' + str(testErr))
print('Learned classification tree model:')
print(model.toDebugString())

# Save and load model
model.save(sc, "target/tmp/
   myDecisionTreeClassificationModel")
sameModel = DecisionTreeModel.load(sc, "target/
   tmp/myDecisionTreeClassificationModel")
```

- Random Forest (Classification)

```
from pyspark.mllib.tree import RandomForest,
   RandomForestModel
from pyspark.mllib.util import MLUtils

# Load and parse the data file into an RDD of
   LabeledPoint.
data = MLUtils.loadLibSVMFile(sc, 'data/mllib/
   sample_libsvm_data.txt')
# Split the data into training and test sets
   (30% held out for testing)
(trainingData, testData) = data.randomSplit
   ([0.7, 0.3])

# Train a RandomForest model.
#  Empty categoricalFeaturesInfo indicates all
   features are continuous.
#  Note: Use larger numTrees in practice.
#  Setting featureSubsetStrategy="auto" lets the
    algorithm choose.
model = RandomForest.trainClassifier
(trainingData, numClasses=2,
   categoricalFeaturesInfo={},
numTrees=3, featureSubsetStrategy="auto",
impurity='gini', maxDepth=4, maxBins=32)

# Evaluate model on test instances and compute
   test error
predictions = model.predict(testData.map(lambda
   x: x.features))
labelsAndPredictions = testData.map(lambda lp:
   lp.label).zip(predictions)
testErr = labelsAndPredictions.filter(lambda (v,
    p): v != p).count() / float(testData.count()
   )
print('Test Error = ' + str(testErr))
print('Learned classification forest model:')
print(model.toDebugString())
```

```
# Save and load model
model.save(sc, "target/tmp/
   myRandomForestClassificationModel")
sameModel = RandomForestModel.load(sc, "target/
   tmp/myRandomForestClassificationModel")
```

- Gradient-Boosted Trees (GBTs) (Classification)

```
from pyspark.mllib.tree import
   GradientBoostedTrees,
   GradientBoostedTreesModel
from pyspark.mllib.util import MLUtils

# Load and parse the data file.
data = MLUtils.loadLibSVMFile(sc, "data/mllib/
   sample_libsvm_data.txt")
# Split the data into training and test sets
   (30% held out for testing)
(trainingData, testData) = data.randomSplit
   ([0.7, 0.3])

# Train a GradientBoostedTrees model.
#  Notes: (a) Empty categoricalFeaturesInfo
   indicates all features are continuous.
#         (b) Use more iterations in practice.
model = GradientBoostedTrees.trainClassifier(
   trainingData,categoricalFeaturesInfo={},
numIterations=3)

# Evaluate model on test instances and compute
   test error
predictions = model.predict(testData.map(lambda
   x: x.features))
labelsAndPredictions = testData.map(lambda lp:
   lp.label).zip(predictions)
testErr = labelsAndPredictions.filter(lambda (v,
    p): v != p).count() / float(testData.count()
   )
```

```
print('Test Error = ' + str(testErr))
print('Learned classification GBT model:')
print(model.toDebugString())

# Save and load model
model.save(sc, "target/tmp/
   myGradientBoostingClassificationModel")
sameModel = GradientBoostedTreesModel.load(sc,"
   target/tmp/
   myGradientBoostingClassificationModel")
```

- Linear least squares, Lasso, and ridge regression (Regression)

```
from pyspark.mllib.regression import
   LabeledPoint, LinearRegressionWithSGD,
   LinearRegressionModel

# Load and parse the data
def parsePoint(line):
    values = [float(x) for x in line.replace
       (',', ' ').split(' ')]
    return LabeledPoint(values[0], values[1:])

data = sc.textFile("data/mllib/ridge-data/lpsa.
   data")
parsedData = data.map(parsePoint)

# Build the model
model = LinearRegressionWithSGD.train(parsedData
   , iterations=100, step=0.00000001)

# Evaluate the model on training data
valuesAndPreds = parsedData.map(lambda p: (p.
   label, model.predict(p.features)))
MSE = valuesAndPreds.map(lambda (v, p): (v - p)
   **2).reduce(lambda x, y: x + y) /
   valuesAndPreds.count()
print("Mean Squared Error = " + str(MSE))
```

```
# Save and load model
model.save(sc, "myModelPath")
sameModel = LinearRegressionModel.load(sc, "
   myModelPath")
```

- Decision Trees (Regression)

```
from pyspark.mllib.tree import DecisionTree,
   DecisionTreeModel
from pyspark.mllib.util import MLUtils

# Load and parse the data file into an RDD of
   LabeledPoint.
data = MLUtils.loadLibSVMFile(sc, 'data/mllib/
   sample_libsvm_data.txt')
# Split the data into training and test sets
   (30% held out for testing)
(trainingData, testData) = data.randomSplit
   ([0.7, 0.3])

# Train a DecisionTree model.
#  Empty categoricalFeaturesInfo indicates all
   features are continuous.
model = DecisionTree.trainRegressor
(trainingData, categoricalFeaturesInfo={},
impurity='variance', maxDepth=5, maxBins=32)

# Evaluate model on test instances and compute
   test error
predictions = model.predict(testData.map(lambda
   x: x.features))
labelsAndPredictions = testData.map(lambda lp:
   lp.label).zip(predictions)
testMSE = labelsAndPredictions.map(lambda (v, p)
   : (v - p) * (v - p)).sum() /\
    float(testData.count())
print('Test Mean Squared Error = ' + str(testMSE
```

```
    ))
print('Learned regression tree model:')
print(model.toDebugString())

# Save and load model
model.save(sc, "target/tmp/
    myDecisionTreeRegressionModel")
sameModel = DecisionTreeModel.load(sc, "target/
    tmp/myDecisionTreeRegressionModel")
```

- Random Forest (Regression)

```
from pyspark.mllib.tree import RandomForest,
    RandomForestModel
from pyspark.mllib.util import MLUtils

# Load and parse the data file into an RDD of
    LabeledPoint.
data = MLUtils.loadLibSVMFile(sc, 'data/mllib/
    sample_libsvm_data.txt')
# Split the data into training and test sets
    (30% held out for testing)
(trainingData, testData) = data.randomSplit
    ([0.7, 0.3])

# Train a RandomForest model.
#  Empty categoricalFeaturesInfo indicates all
    features are continuous.
#  Note: Use larger numTrees in practice.
#  Setting featureSubsetStrategy="auto" lets the
     algorithm choose.
model = RandomForest.trainRegressor
(trainingData, categoricalFeaturesInfo={},
numTrees=3, featureSubsetStrategy="auto",
impurity='variance', maxDepth=4, maxBins=32)

# Evaluate model on test instances and compute
    test error
```

```
predictions = model.predict(testData.map(lambda
   x: x.features))
labelsAndPredictions = testData.map(lambda lp:
   lp.label).zip(predictions)
testMSE = labelsAndPredictions.map(lambda (v, p)
   : (v - p) * (v - p)).sum() /\
    float(testData.count())
print('Test Mean Squared Error = ' + str(testMSE
   ))
print('Learned regression forest model:')
print(model.toDebugString())

# Save and load model
model.save(sc, "target/tmp/
   myRandomForestRegressionModel")
sameModel = RandomForestModel.load(sc, "target/
   tmp/myRandomForestRegressionModel")
```

- Gradient-Boosted Trees (GBTs) (Regression)

```
from pyspark.mllib.tree import
   GradientBoostedTrees,
   GradientBoostedTreesModel
from pyspark.mllib.util import MLUtils

# Load and parse the data file.
data = MLUtils.loadLibSVMFile(sc, "data/mllib/
   sample_libsvm_data.txt")
# Split the data into training and test sets
   (30% held out for testing)
(trainingData, testData) = data.randomSplit
   ([0.7, 0.3])

# Train a GradientBoostedTrees model.
#  Notes: (a) Empty categoricalFeaturesInfo
   indicates all features are continuous.
#         (b) Use more iterations in practice.
model = GradientBoostedTrees.trainRegressor
```

```
(trainingData,categoricalFeaturesInfo={},
   numIterations=3)

# Evaluate model on test instances and compute
   test error
predictions = model.predict(testData.map(lambda
   x: x.features))
labelsAndPredictions = testData.map(lambda lp:
   lp.label).zip(predictions)
testMSE = labelsAndPredictions.map(lambda (v, p)
   : (v - p) * (v - p)).sum() /\
    float(testData.count())
print('Test Mean Squared Error = ' + str(testMSE
   ))
print('Learned regression GBT model:')
print(model.toDebugString())

# Save and load model
model.save(sc, "target/tmp/
   myGradientBoostingRegressionModel")
sameModel = GradientBoostedTreesModel.load(sc, "
   target/tmp/myGradientBoostingRegressionModel
   ")
```

- isotonic regression (Regression)

```
import math
from pyspark.mllib.regression import
   IsotonicRegression, IsotonicRegressionModel

data = sc.textFile("data/mllib/
   sample_isotonic_regression_data.txt")

# Create label, feature, weight tuples from
   input data with weight set to default value
   1.0.
parsedData = data.map(lambda line: tuple([float(
   x) for x in line.split(',')]) + (1.0,))
```

```
# Split data into training (60%) and test (40%)
   sets.
training, test = parsedData.randomSplit([0.6,
   0.4], 11)

# Create isotonic regression model from training
    data.
# Isotonic parameter defaults to true so it is
   only shown for demonstration
model = IsotonicRegression.train(training)

# Create tuples of predicted and real labels.
predictionAndLabel = test.map(lambda p: (model.
   predict(p[1]), p[0]))

# Calculate mean squared error between predicted
    and real labels.
meanSquaredError = predictionAndLabel.map(lambda
    pl: math.pow((pl[0] - pl[1]), 2)).mean()
print("Mean Squared Error = " + str(
   meanSquaredError))

# Save and load model
model.save(sc, "target/tmp/
   myIsotonicRegressionModel")
sameModel = IsotonicRegressionModel.load(sc, "
   target/tmp/myIsotonicRegressionModel")
```

- K-means (Clustering)

```
from pyspark.mllib.clustering import KMeans,
   KMeansModel
from numpy import array
from math import sqrt

# Load and parse the data
data = sc.textFile("data/mllib/kmeans_data.txt")
```

```
parsedData = data.map(lambda line: array([float(
   x) for x in line.split(' ')]))

# Build the model (cluster the data)
clusters = KMeans.train(parsedData, 2,
   maxIterations=10,
        runs=10, initializationMode="random")

# Evaluate clustering by computing Within Set
   Sum of Squared Errors
def error(point):
    center = clusters.centers[clusters.predict(
       point)]
    return sqrt(sum([x**2 for x in (point -
       center)]))

WSSSE = parsedData.map(lambda point: error(point
   )).reduce(lambda x, y: x + y)
print("Within Set Sum of Squared Error = " + str
   (WSSSE))

# Save and load model
clusters.save(sc, "myModelPath")
sameModel = KMeansModel.load(sc, "myModelPath")
```

- Gaussian Mixture (Clustering)

```
from pyspark.mllib.clustering import
   GaussianMixture
from numpy import array

# Load and parse the data
data = sc.textFile("data/mllib/gmm_data.txt")
parsedData = data.map(lambda line: array([float(
   x) for x in line.strip().split(' ')]))

# Build the model (cluster the data)
gmm = GaussianMixture.train(parsedData, 2)
```

```
# output parameters of model
for i in range(2):
    print ("weight = ", gmm.weights[i], "mu = ",
        gmm.gaussians[i].mu,
        "sigma = ", gmm.gaussians[i].sigma.
           toArray())
```

- Power iteration clustering (PIC) (Clustering)

```
from __future__ import print_function
from pyspark.mllib.clustering import
   PowerIterationClustering,
   PowerIterationClusteringModel

# Load and parse the data
data = sc.textFile("data/mllib/pic_data.txt")
similarities = data.map(lambda line: tuple([
   float(x) for x in line.split(' ')]))

# Cluster the data into two classes using
   PowerIterationClustering
model = PowerIterationClustering.train(
   similarities, 2, 10)

model.assignments().foreach(lambda x: print(str(
   x.id) + " -> " + str(x.cluster)))

# Save and load model
model.save(sc, "myModelPath")
sameModel = PowerIterationClusteringModel.load(
   sc, "myModelPath")
```

- Latent Dirichlet allocation (LDA) (Clustering)

```
from pyspark.mllib.clustering import LDA,
   LDAModel
from pyspark.mllib.linalg import Vectors
```

```
# Load and parse the data
data = sc.textFile("data/mllib/sample_lda_data.
   txt")
parsedData = data.map(lambda line: Vectors.dense
   ([float(x) for x in line.strip().split(' ')])
   )
# Index documents with unique IDs
corpus = parsedData.zipWithIndex().map(lambda x:
    [x[1], x[0]]).cache()

# Cluster the documents into three topics using
   LDA
ldaModel = LDA.train(corpus, k=3)

# Output topics. Each is a distribution over
   words (matching word count vectors)
print("Learned topics (as distributions over
   vocab of " + str(ldaModel.vocabSize()) + "
   words):")
topics = ldaModel.topicsMatrix()
for topic in range(3):
    print("Topic " + str(topic) + ":")
    for word in range(0, ldaModel.vocabSize()):
        print(" " + str(topics[word][topic]))

# Save and load model
model.save(sc, "myModelPath")
sameModel = LDAModel.load(sc, "myModelPath")
```

- Streaming k-means(Clustering)

```
from pyspark.mllib.linalg import Vectors
from pyspark.mllib.regression import
   LabeledPoint
from pyspark.mllib.clustering import
   StreamingKMeans
def parse(lp):
    label = float(lp[lp.find('(') + 1: lp.find
```

```
        (',')])
    vec = Vectors.dense(lp[lp.find('[') + 1: lp.
        find(']')].split(','))
    return LabeledPoint(label, vec)

trainingData = ssc.textFileStream("/training/
    data/dir").map(Vectors.parse)
testData = ssc.textFileStream("/testing/data/dir
    ").map(parse)
model = StreamingKMeans(k=2, decayFactor=1.0).
    setRandomCenters(3, 1.0, 0)
model.trainOn(trainingData)
print(model.predictOnValues(testData.map(lambda
    lp: (lp.label, lp.features))))

ssc.start()
ssc.awaitTermination()
```

5.3 Web Semantic

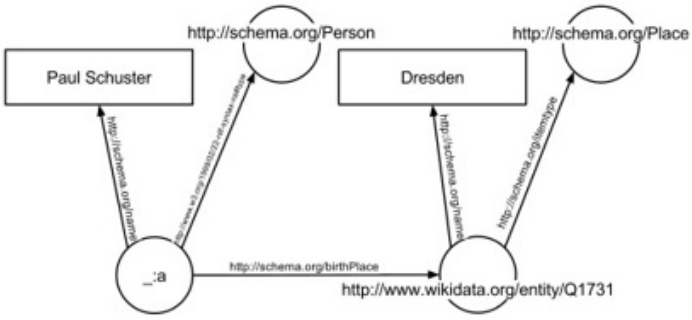
Semantic Web adalah perpanjangan dari Web melalui standar World Wide Web Consortium (W3C). [1] Standar ini mempromosikan format data umum dan protokol pertukaran di Web, yang paling mendasar adalah Resource Description Framework (RDF).

Menurut W3C, " Semantic Web memberikan kerangka umum yang memungkinkan data akan dibagi dan digunakan kembali antar aplikasi, perusahaan, dan batas-batas komunitas". Istilah ini diciptakan oleh Tim Berners-Lee untuk web data yang dapat diproses oleh mesin.

Tahun 2001 Scientific American artikel yang dipublish oleh Berners-Lee, Hendler, dan Lassila dijelaskan evolusi yang diharapkan dari Web yang ada untuk Semantic Web. Pada tahun 2006, Berners-Lee dan rekan menyatakan bahwa:. "Ide sederhana ini ... sebagian besar masih belum direalisasi" . Pada tahun 2013, lebih dari empat juta Web domain yang terkandung markup Semantic Web.

contoh :
contoh mendefinisikan lima kali lipat berikut (ditampilkan di Turtle Syntax).

```
<div vocab="http://schema.org/" typeof="Person">
  <span property="name">Paul Schuster</span> was born in
  <span property="birthPlace" typeof="Place"
href="http://www.wikidata.org/entity/Q1731">
    <span property="name">Dresden</span>.
  </span>
</div>
```



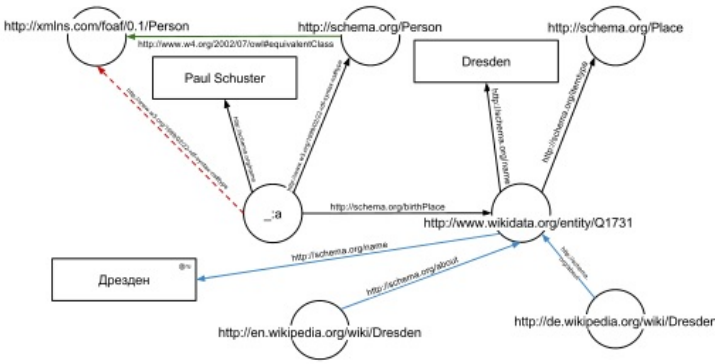
Gambar 5.36: Semantic Web

Setiap tiga mempresentasikan satu tepi yang dihasilkan oleh graph: elemen pertama dari triple (subjek) adalah nama dari simpul mana tepi dimulai, elemen kedua (predikat) jenis tepi, dan elemen terakhir dan ketiga (objek) baik nama node di mana tepi berakhir atau nilai literal (misalnya teks, angka, dll).

5.3.1 URI (Uniform Resource Identifier)

Salah satu keuntungan menggunakan Uniform Resource Identifier (URI) adalah bahwa mereka dapat dereferenced menggunakan protokol HTTP. Menurut disebut Linked prinsip Data Terbuka, seperti URI dereferenced harus menghasilkan sebuah dokumen yang menawarkan data lebih lanjut tentang URI diberikan. Dalam contoh ini, semua URI, baik untuk tepi dan node. misalnya :
http:/schema.org/Person, http:/schema.org/birthPlace,http:/www.wikidata.org /entity/Q1731) Akan menghasilkan grafik RDF dan menggambarkan URI, misalnya yang Dresden adalah sebuah kota di Jerman, atau bahwa seseorang, dalam arti bahwa URI, dapat fiksi.

Grafik kedua menunjukkan contoh sebelumnya, tapi sekarang diperkaya dengan beberapa kali lipat dari dokumen yang dihasilkan dari dereferencing




```
_:a <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://schema.org/Person> .
_:a <http://schema.org/name> "Paul Schuster" .
_:a <http://schema.org/birthPlace> <http://www.wikidata.org/entity/Q1731> .
<http://www.wikidata.org/entity/Q1731> <http://schema.org/itemtype> <http://schema.org/Place> .
<http://www.wikidata.org/entity/Q1731> <http://schema.org/name> "Dresden" .
```

Gambar 5.37: Uniform Resource Identifier

http:/ schema.org/Person (tepi hijau) dan http://www.wikidata.org/entity/Q1731 (blue edges).

Selain itu ke tepi diberikan dalam dokumen terlibat secara eksplisit, tepi dapat secara otomatis disimpulkan: triple dari fragmen RDFa asli dan

```
_:a <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://schema.org/Person> .
```

triple dari dokumen di http://schema.org/Person (tepi hijau dalam gambar)

```
<http://schema.org/Person> <http://www.w3.org/2002/07/owl#equivalentClass> <http://xmlns.com/foaf/0.1/Person> .
```

memungkinkan untuk menyimpulkan berikut tiga, yang diberikan semantik OWL (merah garis putus-putus

5.3.2 Latar Belakang

Konsep Semantic Jaringan Model dibentuk pada awal 1960-an oleh ilmuwan kognitif Allan M. Collins, ahli bahasa M. Ross Quillian dan psikolog Elizabeth F. Loftus sebagai bentuk untuk mewakili pengetahuan semantik terstruktur. Bila diterapkan dalam konteks internet modern, hal ini

```
_:a <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://xmlns.com/foaf/0.1/Person> .
```

menghubungkan halaman web, terbaca-manusia dengan memasukkan mesin-dibaca metadata tentang halaman dan bagaimana mereka berhubungan satu sama lain. Hal ini memungkinkan agen otomatis untuk mengakses Web dengan lebih cerdas dan melakukan lebih banyak tugas atas nama pengguna. Istilah "Semantic Web" diciptakan oleh Tim Berners-Lee, [3] penemu World Wide Web dan direktur dari World Wide Web Consortium ("W3C"), yang mengawasi pengembangan standar Semantic Web yang diusulkan. Dia mendefinisikan Semantic Web sebagai "web data yang dapat diproses secara langsung dan tidak langsung oleh mesin".

Banyak teknologi yang diusulkan oleh W3C sudah ada sebelum mereka ditempatkan di bawah payung W3C. Ini digunakan dalam berbagai konteks, terutama yang berhubungan dengan informasi yang mencakup domain tertentu dan terbatas, dan di mana berbagi data merupakan kebutuhan umum, seperti penelitian ilmiah atau pertukaran data antar bisnis. Selain itu, teknologi lainnya dengan tujuan yang sama telah muncul, seperti Microformats.

Tim Berners-Lee awalnya mengungkapkan visi Semantic Web sebagai berikut:

Saya punya mimpi untuk Web [di mana komputer] menjadi mampu menganalisis semua data di Web - isi, link, dan transaksi antara orang dan komputer. Sebuah "Semantic Web", yang membuat ini mungkin, belum muncul, tetapi ketika itu terjadi, mekanisme sehari-hari perdagangan, birokrasi dan kehidupan sehari-hari akan ditangani oleh mesin berbicara dengan mesin. The "agen cerdas" orang telah disebut-sebut untuk usia akhirnya akan terwujud.

Semantic Web dianggap sebagai integrator di konten yang berbeda, aplikasi informasi dan sistem. Ini memiliki aplikasi dalam penerbitan, blogging, dan banyak daerah lainnya.

5.3.3 Limitations of HTML)

Banyak file pada komputer biasa juga bisa longgar dibagi menjadi dokumen yang dapat dibaca manusia dan mesin data yang dibaca. Dokumen

seperti mail, laporan, dan brosur dibaca oleh manusia. Data, seperti kalender, buku alamat, playlist, dan spreadsheet disajikan dengan menggunakan program aplikasi yang memungkinkan mereka dilihat, dicari dan dikombinasikan.

Saat ini, World Wide Web terutama didasarkan pada dokumen tertulis dalam Hypertext Markup Language (HTML), sebuah konvensi markup yang digunakan untuk pengkodean tubuh teks diselingi dengan objek multimedia seperti gambar dan bentuk interaktif. tag metadata menyediakan metode yang komputer dapat mengkategorikan konten halaman web, misalnya: Den-

```
<meta name="keywords" content="computing, computer studies, computer" />
<meta name="description" content="Cheap widgets for sale" />
<meta name="author" content="John Doe" />
```

Gambar 5.38: Tag metadata

gan HTML dan alat untuk membuat itu (mungkin software web browser, mungkin lain agen pengguna), seseorang dapat membuat dan menyajikan halaman yang berisi daftar barang untuk dijual. HTML halaman katalog ini dapat membuat, pernyataan dokumen-tingkat sederhana seperti "judul dokumen ini adalah 'Widget Superstore'", tetapi tidak ada kemampuan dalam HTML itu sendiri untuk menegaskan jelas bahwa, misalnya, nomor item X586172 adalah Acme Gizmo dengan harga eceran 199, atau bahwa itu adalah produk konsumen. Sebaliknya, HTML hanya bisa mengatakan bahwa rentang teks "X586172" adalah sesuatu yang harus diposisikan dekat "Acme Gizmo" dan "199 ", dll Tidak ada cara untuk mengatakan "ini adalah katalog" atau bahkan untuk menetapkan bahwa "Acme Gizmo" adalah semacam judul atau bahwa "199 " adalah harga. Juga tidak ada cara untuk mengungkapkan bahwa potongan-potongan informasi terikat bersama dalam menggambarkan item diskrit, berbeda dari item lainnya mungkin tercantum pada halaman.

HTML semantik mengacu pada praktek HTML tradisional markup berikut niat, daripada menetapkan rincian tata letak langsung. Misalnya, penggunaan ¡em¿ yang menunjukkan "penekanan" bukan ¡i¿, yang menentukan miring. Rincian tata letak yang tersisa hingga browser, dalam kombinasi dengan Cascading Style Sheets. Tapi praktek ini jatuh pendek dari menentukan semantik objek seperti barang untuk dijual atau harga.
Microformats memperpanjang sintaks HTML untuk membuat mesin yang

dapat dibaca markup semantik tentang obyek termasuk orang, organisasi, peristiwa dan produk. [9] inisiatif serupa termasuk RDFa, Microdata dan Schema.org.

5.3.4 Semantic Web solutions)

Semantic Web membutuhkan solusi lebih lanjut. Ini melibatkan penerbitan dalam bahasa khusus dirancang untuk data: Resource Description Framework (RDF), Web Ontologi Language (OWL), dan Extensible Markup Language (XML). HTML menggambarkan dokumen dan hubungan antara mereka. RDF, OWL, dan XML, sebaliknya, dapat menggambarkan hal-hal yang sewenang-wenang seperti orang, pertemuan, atau bagian pesawat.
Teknologi ini digabungkan untuk memberikan deskripsi yang melengkapi atau mengganti isi dokumen Web. Dengan demikian, konten dapat memanifestasikan dirinya sebagai data deskriptif yang disimpan dalam database diakses Web, [10] atau sebagai markup dalam dokumen (terutama, di HTML Extensible (XHTML) diselingi dengan XML, atau, lebih sering, murni dalam XML, dengan tata letak atau render isyarat disimpan secara terpisah). Deskripsi dibaca mesin memungkinkan manajer konten untuk menambah arti konten, yaitu, untuk menggambarkan struktur pengetahuan yang kita miliki tentang konten itu. Dengan cara ini, mesin dapat memproses pengetahuan itu sendiri, bukan teks, menggunakan proses yang sama dengan penalaran deduktif manusia dan inferensi, sehingga memperoleh hasil yang lebih bermakna dan membantu komputer untuk melakukan pengumpulan informasi otomatis dan penelitian.
Contoh tag yang akan digunakan dalam halaman web non-semantik: Pengkodean

```
<item>blog</item>
```

Gambar 5.39: Tag non Semantic

informasi sejenis di laman web semantik mungkin terlihat seperti ini: Tim

```
<item rdf:about="http://example.org/semantic-web/">Semantic Web</item>
```

Gambar 5.40: Tag Semantic

Berners-Lee menyebut jaringan yang dihasilkan dari Linked data Giant global Grafik, kontras dengan berbasis HTML World Wide Web. Berners-Lee berpendapat bahwa jika masa lalu adalah berbagi dokumen, masa depan adalah berbagi data. jawabannya terhadap pertanyaan "bagaimana" memberikan tiga poin dari instruksi. Satu, URL harus menunjuk ke data. Dua, ada yang mengakses URL harus mendapatkan kembali data. Tiga, hubungan dalam data harus menunjuk ke URL tambahan data.

5.3.5 Web 3.0)

Tim Berners-Lee menggambarkan web semantik sebagai komponen dari "Web 3.0".
Orang terus bertanya apa Web 3.0. Saya pikir mungkin ketika Anda punya overlay dari scalable vector graphics - semuanya beriak dan melipat dan mencari berkabut - pada Web 2.0 dan akses ke Web semantik terintegrasi di ruang besar data, Anda akan memiliki akses ke sumber daya data yang luar biasa - Tim Berners-Lee, 2006

"Semantic Web" kadang-kadang digunakan sebagai sinonim untuk "Web 3.0", meskipun definisi setiap istilah bervariasi. Beberapa tantangan untuk Web Semantic termasuk luasnya, ketidakjelasan, ketidakpastian, inkonsistensi, dan penipuan. sistem penalaran otomatis akan harus berurusan dengan semua masalah ini dalam rangka untuk memenuhi janji dari Semantic Web.
Luasnya: The World Wide Web berisi banyak miliaran halaman. The SNOMED CT terminologi medis ontologi sendiri berisi 370.000 nama kelas, dan teknologi yang ada belum mampu menghilangkan semua hal semantik digandakan. Setiap sistem penalaran otomatis akan harus berurusan dengan input yang benar-benar besar.
Ketidakjelasan: Ini adalah konsep tepat seperti "muda" atau "tinggi". Ini muncul dari ketidakjelasan permintaan pengguna, konsep yang diwakili oleh penyedia konten, dari yang sesuai dengan query untuk hal penyedia dan mencoba untuk menggabungkan basis pengetahuan yang berbeda dengan tumpang tindih tapi konsep agak berbeda. logika fuzzy adalah teknik yang paling umum untuk berurusan dengan ketidakjelasan.
Ketidakpastian: Ini adalah konsep yang tepat dengan nilai-nilai yang tidak pasti. Misalnya, seorang pasien mungkin menyajikan serangkaian gejala yang sesuai dengan sejumlah diagnosa yang berbeda yang berbeda masing-masing dengan probabilitas yang berbeda. teknik penalaran probabilistik umumnya

digunakan untuk mengatasi ketidakpastian.
Inkonsistensi: Ini adalah kontradiksi logis yang pasti akan timbul selama pengembangan ontologi besar, dan ketika ontologi dari sumber terpisah digabungkan. penalaran deduktif gagal serempak ketika dihadapkan dengan inkonsistensi, karena "sesuatu mengikuti dari kontradiksi". penalaran yg dpt dibatalkan dan penalaran paraconsistent dua teknik yang dapat digunakan untuk menangani inkonsistensi.
Tipu: Ini adalah ketika produsen informasi yang sengaja menyesatkan konsumen informasi. teknik kriptografi yang saat ini digunakan untuk mengurangi ancaman ini.
Daftar ini tantangan adalah ilustrasi ketimbang lengkap, dan berfokus pada tantangan untuk "logika pemersatu" dan "bukti" lapisan Semantic Web. World Wide Web Consortium (W3C) Inkubator Group untuk Ketidakpastian Penalaran untuk World Wide Web (URW3-XG) laporan akhir benjolan masalah ini bersama-sama di bawah judul tunggal "ketidakpastian". Banyak teknik yang disebutkan di sini akan memerlukan ekstensi untuk Web Ontologi Language (OWL) misalnya untuk membubuhi keterangan probabilitas kondisional. Ini merupakan area penelitian aktif.

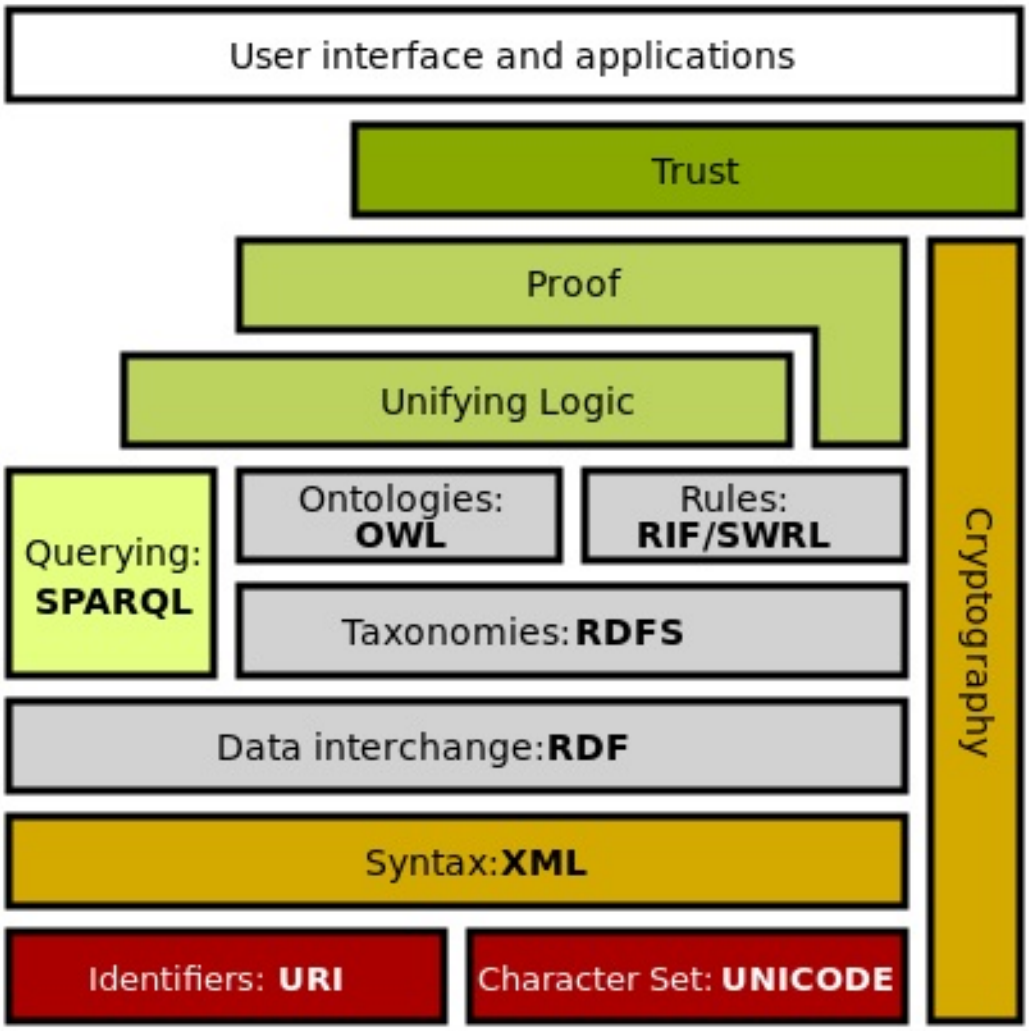
5.3.6 Standart

Standardisasi untuk Semantic Web dalam konteks Web 3.0 adalah di bawah perawatan W3C.

5.3.7 komponen

Istilah "Semantic Web" sering digunakan lebih khusus untuk merujuk pada format dan teknologi yang memungkinkan hal itu. [2] Pengumpulan, penataan dan pemulihan data terkait diaktifkan oleh teknologi yang memberikan deskripsi formal konsep, istilah, dan hubungan dalam satu domain pengetahuan tertentu. Teknologi ini ditetapkan sebagai standar W3C dan meliputi:
Resource Description Framework (RDF), metode umum untuk menggambarkan informasi RDF Schema (RDFS) Simple Knowledge Organization System (SKOS) SPARQL, sebuah bahasa RDF query Notation3 (N3), dirancang dengan manusia-dibaca dalam pikiran N-Triples, format untuk menyimpan dan mengirim data Turtle (Terse RDF Triple Language) Web Ontology Language (OWL),keluarga bahasa representasi pengetahuan Rule

Interchange Format (RIF), kerangka dialek bahasa aturan web yang mendukung aturan interchange di Web The Semantic Web Stack menggambarkan



Gambar 5.41: Komponen Semantic WEB

arsitektur Semantic Web. Fungsi dan hubungan komponen dapat diringkas sebagai berikut: XML menyediakan sintaks unsur untuk struktur konten dalam dokumen, namun rekan ada semantik dengan makna dari isi yang terkandung dalam. XML tidak saat ini komponen penting dari teknologi Semantic Web dalam kebanyakan kasus, seperti sintaks alternatif ada, seperti

kura-kura. Kura-kura adalah standar de facto, tapi belum melalui proses standarisasi formal. XML Schema adalah bahasa untuk menyediakan dan membatasi struktur dan isi dari unsur yang terkandung dalam dokumen XML. RDF adalah bahasa yang sederhana untuk mengekspresikan model data, yang merujuk pada objek ("sumber web") dan hubungan mereka. Model berbasis RDF dapat direpresentasikan dalam berbagai sintaks, misalnya, RDF / XML, N3, Turtle, dan RDFa. RDF adalah standar fundamental dari Semantic Web. [15] [16] RDF Schema meluas RDF dan kosa kata untuk menggambarkan sifat dan kelas sumber daya berbasis RDF, dengan semantik untuk umum-hierarki dari properti dan kelas tersebut. OWL menambahkan lebih kosakata untuk menjelaskan sifat dan kelas: antara lain, hubungan antara kelas (misalnya disjointness), kardinalitas (misalnya "tepat satu"), kesetaraan, mengetik lebih kaya dari properti, karakteristik sifat (misalnya simetri), dan kelas disebutkan. SPARQL adalah protokol dan permintaan bahasa untuk sumber web data semantik. RIF adalah Format W3C Peraturan Interchange. Ini adalah bahasa XML untuk mengekspresikan aturan Web bahwa komputer dapat mengeksekusi. RIF menyediakan beberapa versi, yang disebut dialek. Ini termasuk RIF Basic Logic Dialek (RIF-BLD) dan RIF Produksi Aturan Dialek (RIF PRD).

5.3.8 Yang sudah distandarisasi

- RDF
- RDFS
- Rule Interchange Format (RIF)
- SPARQL
- Unicode
- Uniform Resource Identifier
- Web Ontology Language (OWL)
- XML

5.3.9 Belum teralisasi

- Unifying Logic and Proof layers
- Semantic Web Rule Language (SWRL)

5.3.10 Aplikasi

Tujuannya adalah untuk meningkatkan kegunaan dan kegunaan Web dan sumber daya yang saling berhubungan dengan menciptakan Semantic Web Services, seperti:

- Server yang mengekspos sistem data yang ada menggunakan standar RDF dan SPARQL. Banyak konverter untuk RDF ada dari aplikasi yang berbeda. database relasional merupakan sumber penting. Web server semantik menempel pada sistem yang ada tanpa mempengaruhi operasi.

- Dokumen "ditandai" dengan informasi semantik (perpanjangan dari HTML ¡meta¿ tag yang digunakan dalam halaman Web saat ini untuk memberikan informasi untuk mesin pencari Web menggunakan web crawler). Ini bisa menjadi informasi mesin-dimengerti tentang isi manusia-dimengerti dokumen (seperti pencipta, judul, deskripsi, dll) atau bisa juga murni metadata mewakili seperangkat fakta (seperti sumber daya dan layanan di tempat lain di situs). Perhatikan bahwa apa pun yang dapat diidentifikasi dengan Uniform Resource Identifier (URI) dapat digambarkan, sehingga web semantik dapat alasan tentang binatang, orang, tempat, ide, dll Ada empat format penjelasan semantik yang dapat digunakan dalam dokumen HTML; Microformat, RDFa, Microdata dan JSON-LD. [17] Semantic markup sering dihasilkan secara otomatis, bukan manual.

- kosakata metadata umum (ontologi) dan peta antara kosakata yang memungkinkan pencipta dokumen untuk mengetahui bagaimana mark up dokumen mereka sehingga agen dapat menggunakan informasi dalam metadata yang disediakan (sehingga Penulis dalam arti 'Penulis halaman' memenangkan ' t menjadi bingung dengan Penulis dalam arti sebuah buku yang merupakan subjek dari tinjauan buku).

- agen otomatis untuk melakukan tugas-tugas bagi pengguna web semantik menggunakan data ini.
- layanan berbasis web (sering dengan agen mereka sendiri) untuk memberikan informasi khusus untuk agen, misalnya, layanan Percayalah bahwa agen bisa meminta jika beberapa toko online memiliki sejarah layanan yang buruk atau spamming.

layanan tersebut dapat berguna untuk mesin pencari umum, atau dapat digunakan untuk manajemen pengetahuan dalam suatu organisasi. aplikasi bisnis meliputi:

- Memfasilitasi integrasi informasi dari sumber campuran
- Melarutkan ambiguitas dalam terminologi perusahaan
- Meningkatkan pencarian informasi sehingga mengurangi informasi yang berlebihan
- Mengidentifikasi informasi yang relevan berkenaan dengan domain tertentu [18]
- Memberikan dukungan pengambilan keputusan

Dalam sebuah perusahaan, ada kelompok pengguna dan manajemen mampu menegakkan pedoman perusahaan seperti adopsi ontologi tertentu dan penggunaan anotasi semantik. Dibandingkan dengan Semantic Web publik ada persyaratan yang lebih rendah pada skalabilitas dan informasi yang beredar dalam perusahaan dapat lebih dipercaya secara umum; privasi adalah kurang dari sebuah isu di luar penanganan data pelanggan.

5.3.11 Reaksi skeptis - Kelayakan praktis

Kritik mempertanyakan kelayakan dasar pemenuhan lengkap atau bahkan sebagian dari Semantic Web, menunjukkan baik kesulitan dalam pengaturan itu dan kurangnya tujuan umum kegunaan yang mencegah upaya yang diperlukan dari yang diinvestasikan. Dalam sebuah makalah tahun 2003, Marshall dan Shipman menunjukkan overhead kognitif yang melekat dalam meresmikan pengetahuan, dibandingkan dengan authoring dari hypertext web tradisional:

Sambil belajar dasar-dasar HTML relatif mudah, belajar bahasa representasi pengetahuan atau alat membutuhkan penulis untuk belajar tentang metode representasi ini abstraksi dan efeknya pada penalaran. Misalnya, memahami hubungan kelas-misalnya, atau hubungan superclass-subclass, lebih dari pemahaman bahwa salah satu konsep adalah "jenis" konsep lain. [...] Abstraksi ini diajarkan untuk ilmuwan komputer umumnya dan insinyur pengetahuan khusus tetapi tidak cocok arti bahasa alami serupa menjadi "jenis" sesuatu. penggunaan efektif seperti representasi resmi membutuhkan penulis untuk menjadi insinyur pengetahuan terampil di samping keahlian lain yang dibutuhkan oleh domain. [...] Setelah satu telah belajar bahasa representasi formal, masih sering jauh lebih banyak usaha untuk mengekspresikan ide-ide dalam representasi yang dari dalam representasi kurang formal [...]. Memang, ini adalah bentuk pemrograman berdasarkan deklarasi data semantik dan membutuhkan pemahaman tentang bagaimana algoritma penalaran akan menafsirkan struktur menulis.

Menurut Marshall dan Shipman, sifat diam-diam dan perubahan banyak pengetahuan menambah masalah rekayasa pengetahuan, dan membatasi penerapan Semantic Web untuk domain tertentu. Masalah lebih lanjut bahwa mereka menunjukkan cara domain- atau organisasi tertentu untuk mengekspresikan pengetahuan, yang harus diselesaikan melalui kesepakatan masyarakat bukan hanya sarana teknis. [19] Ternyata, masyarakat khusus dan organisasi untuk proyek-proyek intra-perusahaan memiliki cenderung mengadopsi teknologi web semantik lebih besar dari masyarakat perifer dan kurang-khusus. [20] kendala praktis ke arah adopsi telah muncul kurang menantang di mana domain dan ruang lingkup yang lebih terbatas dibandingkan masyarakat umum dan World Wide Web. [20]

Akhirnya, Marshall dan Shipman melihat masalah pragmatis dalam gagasan (Knowledge Navigator-style) agen cerdas yang bekerja di Web Semantic sebagian besar yang dikuratori secara manual:

Dalam situasi di mana kebutuhan pengguna dikenal dan sumber daya didistribusikan informasi dijelaskan dengan baik, pendekatan ini dapat sangat efektif; dalam situasi yang tidak diramalkan dan yang menyatukan berbagai tak terduga dari sumber informasi, pendekatan Google adalah lebih kuat. Selanjutnya, Semantic Web mengandalkan rantai inferensi yang lebih rapuh; elemen yang hilang dari hasil rantai kegagalan untuk melakukan tindakan yang diinginkan, sementara manusia dapat menyediakan bagian yang hilang dalam pendekatan yang lebih Google seperti. [...] Pengorbanan biaya-manfaat dapat bekerja dalam mendukung khusus dibuat metadata Semantic

Web diarahkan pada tenun sumber informasi domain-spesifik yang terstruktur dengan baik bersama-sama masuk akal; memperhatikan kebutuhan pengguna / pelanggan akan mendorong federasi ini jika mereka ingin berhasil.

kritik Cory Doctorow ("metacrap") adalah dari perspektif perilaku manusia dan preferensi pribadi. Misalnya, orang mungkin termasuk metadata palsu ke dalam halaman Web dalam upaya untuk menyesatkan mesin Semantic Web yang naif menganggap kebenaran metadata ini. Fenomena ini dikenal dengan metatag yang tertipu Peringkat Altavista algoritma dalam mengangkat peringkat halaman Web tertentu: mesin pengindeksan Google khusus mencari upaya tersebut di manipulasi. Peter Grdenfors dan Timo Honkela menunjukkan bahwa berdasarkan logika-teknologi web semantik hanya mencakup sebagian kecil dari fenomena yang relevan terkait dengan semantik.

5.3.12 Sensor Dan Privasi

Antusiasme tentang web semantik dapat diredam oleh kekhawatiran mengenai penyensoran dan privasi. Misalnya, teks-teknik menganalisis sekarang dapat dengan mudah dilewati dengan menggunakan kata lain, metafora misalnya, atau dengan menggunakan gambar di tempat kata-kata. Sebuah implementasi lanjutan dari web semantik akan membuat lebih mudah bagi pemerintah untuk mengontrol melihat dan penciptaan informasi online, informasi ini akan menjadi jauh lebih mudah untuk mesin konten-blocking otomatis untuk memahami. Selain itu, masalah ini juga telah dikemukakan bahwa, dengan penggunaan file FOAF dan geolocation meta-data, akan ada sangat sedikit anonimitas terkait dengan penulis artikel tentang hal-hal seperti blog pribadi.

Kritik lain dari web semantik adalah bahwa hal itu akan jauh lebih memakan waktu untuk membuat dan mempublikasikan konten karena akan perlu dua format untuk satu bagian dari data: satu untuk melihat manusia dan satu untuk mesin. Namun, banyak aplikasi web dalam pembangunan mengatasi masalah ini dengan membuat format yang dapat dibaca mesin pada penerbitan data atau permintaan dari mesin untuk data tersebut. Perkembangan Microformats telah menjadi salah satu reaksi semacam ini kritik. Argumen lain dalam membela kelayakan web semantik adalah harga kemungkinan jatuh tugas kecerdasan manusia di pasar tenaga kerja digital, seperti Amazon Mechanical Turk.

Spesifikasi seperti ERDF dan RDFa memungkinkan data RDF sewenang-

wenang untuk dimasukkan dalam halaman HTML. The (Deskripsi Sumber Daya Mengumpulkan dari Dialek Bahasa) GRDDL mekanisme memungkinkan materi (termasuk Microformats) secara otomatis diartikan sebagai RDF yang ada, sehingga penerbit hanya perlu menggunakan format tunggal, seperti HTML.

5.3.13 Kegiatan Penelitian Pada Aplikasi Perusahaan

Kelompok riset pertama secara eksplisit berfokus pada Semantic Web Perusahaan itu tim ACACIA di INRIA-Sophia Antipolis-, didirikan pada tahun 2002. Hasil dari pekerjaan mereka termasuk dalam RDF (S) berdasarkan mesin pencari Corese, dan penerapan teknologi web semantik di ranah E-learning.

Sejak 2008, kelompok penelitian Web Korporat Semantic, terletak di Free University of Berlin, berfokus pada blok bangunan. Perusahaan Semantic Search, Kolaborasi Semantic Perusahaan, dan Corporate Ontologi Rekayasa

penelitian rekayasa Ontologi meliputi pertanyaan tentang bagaimana melibatkan pengguna non-pakar dalam menciptakan ontologi dan konten semantis dijelaskan. dan untuk mengekstraksi pengetahuan eksplisit dari interaksi pengguna dalam perusahaan.

5.3.14 Memulai membuat web semantic

Untuk membuat sistem web semantic, secara ada tiga tahapan, yaitu membuat file owl dengan menggunakan aplikasi Protege, menyimpan file owl di server openRDF dan untuk melakukan query menggunakan SPARQL

5.3.14.1 Protege

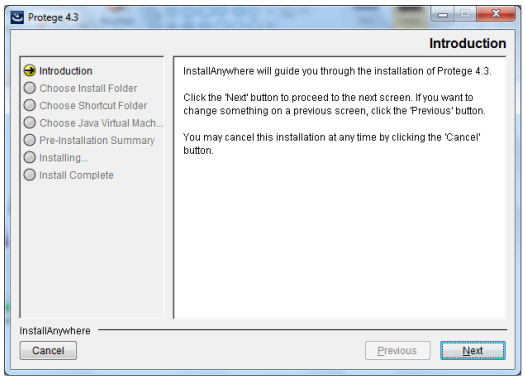
Protg adalah sebuah alat bantu yang berbentuk perangkat lunak yang digunakan untuk pengembang sistem untuk mengembangkan Knowledge-Base System.Aplikasi yang dikembangkan dengan Protg digunakan dalam pemecahan masalah dan pembuat keputusan dalam sebuah domain.

Protg dikembangkan oleh sebuah organisasi yang bernaung di bawah Standford,yang mengambil spesialisasi dibidang ontology.Segala sesuatu yang berhubungan dengan Protg dapat dilihat pada alamat http://Protege.stanford.edu/, termasuk tutorial dan komunitas pengguna Protg.

Protg merupakan sebuah alat yang digunakan untuk membuat sebuah domain ontology, menyesuaikan form untuk entry data, dan memasukan data. Berbagai format penyimpanan seperti OWL, RDF, XML, dan HTML. Protg menyediakan kemudahan plug and play yang membuatnya fleksibel untuk pengembangan prototype yang berkembang.

untuk melakukan instalasi :

1. download installer protege di http://protege.stanford.edu/
2. Jalan file exe yang telah didownload kemudian ikuti petunjuk hingga proses instalasi selesai

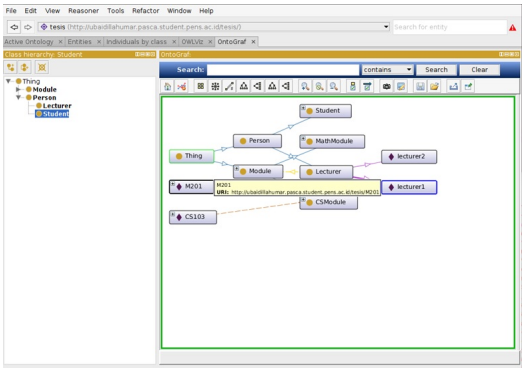


Gambar 5.42: Instalasi protege

5.3.14.2 OWL

OWL adalah salah satu bentuk ontology yang memang dirancang dengan tujuan untuk digunakan oleh aplikasi yang perlu memproses isi informasi ketimbang menampilkan informasi untuk konsumsi manusia. OWL merupakan rekomendasi W3C (World Wide Web Consortium) dalam penulisan ontology untuk web untuk web semantic. OWL dituliskan dalam syntax XML (eXtended Markup Language). Selain XML dalam OWL digunakan juga bahasa XMLS (XML Schema), RDF (Resource Description Framework) dan RDFS (RDF Schema) dan OWL itu sendiri.

pada matakuliah ini, kami membuat file owl dengan bantuan aplikasi protege



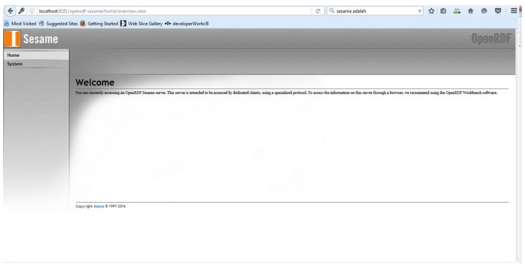
Gambar 5.43: Membuat file owl menggunakan protege

5.3.14.3 OpenRDF-Sesame

Sesame merupakan kerangka open source untuk query dan menganalisa data RDF. Itu diciptakan, dan masih dipertahankan, oleh Dutch software Aduna. Ini pada awalnya dikembangkan sebagai bagian dari "On-To-Knowledge", sebuah proyek web semantik yang bermula dari 1999 sampai 2002.

Sesame mendukung dua bahasa query: SeRQL dan SPARQL. Komponen lain dari Sesame adalah Alibaba, sebuah API yang memungkinkan untuk pemetaan kelas Java ke ontologi dan untuk menghasilkan file Java dari ontologi. Hal ini memungkinkan untuk menggunakan ontologi tertentu seperti RSS, FOAF dan Dublin Core, langsung dari Java.

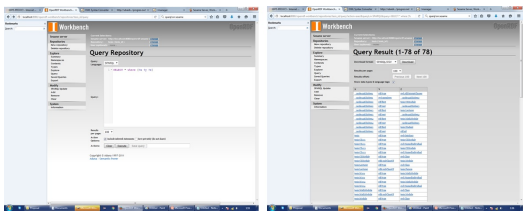
Triplestores lainnya dapat digunakan melalui Sesame, termasuk Mulgara, dan AllegroGraph.



Gambar 5.44: openRDF-Sesame

5.3.14.4 SPARQL

SPARQL merupakan bahasa query untuk RDF. Di dalam lembar rekomendasinya W3C menuliskan SPARQL menyediakan fasilitas untuk mengekstrak informasi dalam bentuk URI, blank nodedan literal, mengekstrak subgraf RDF, dan membangun graf RDF baru berdasar pada informasi dari graf yang di-query



Gambar 5.45: Penerapan SPARQL pada openRDF-Sesame

Bab 6

Project IoT

Project IoT ini merupakan Project yang Team penulis lakukan untuk simulasi dari pemaparan diatas. Bagaimana pemrosesan pengambilan data dari awal sampai ke tahap penyimpanan data pada Big Data (Hadoop) hingga ke tahap visualisasi data tersebut dengan basis web.

6.1 Alat dan Bahan Pelaksanaan Project

Dalam project ini kami menggunakan beberapa sensor dan beberapa peralatan lain, diantaranya yaitu :

1. Raspberry Pi 3
2. Access Point Router
3. 2 PC Server (Multinode Cluster)
    - Sistem Operasi Ubuntu GNU/Linux 14.04
    - Hadoop 2.6
    - Hive 1.2.1
    - Web Server Apache
4. Arduino Mega
5. Sensor Air Atlas yang terdiri dari 5 sensor :
    - Sensor pH

- Sensor EC
- Sensor DO
- Sensor ORP
- Sensor Temperature

6. Laptop
7. USB Wifi

6.2 Alur Kerja Project

Dalam proses pengambilan data menggunakan sensor digunakan Single Board Micro Computer (SBMC) Arduino Mega sebagai alat pemroses data yang diambil dari sensor. Komunikasi antara Sensor dan Arduino Mega menggunakan komunikasi UART. Pada Arduino ini digunakan sebuah program berbasis bahasa pemrograman C untuk proses pengambilan data dan pengiriman data ke luar dari Arduino. Selanjutnya dari Arduino dengan menggunakan komunikasi serial mengirim data ke Raspberry Pi 3 yang dilengkapi dengan program python untuk menerima data masukan yang bersumber dari Arduino Mega tersebut. Selanjutnya dengan menggunakan komunikasi soket antara raspberry Pi 3 (Socket Client) dan PC Server Hadoop (Socket Server), Raspberry Pi 3 mengirimkan data yang diterima dari Arduino dan langsung diterima oleh PC Server yang selanjutnya dengan skema map and reduce pada framework Apache Hive di simpan ke dalam Hadoop Distributed File System (HDFS). Program Socket Client maupun Socket Server ini juga menggunakan bahasa pemrograman python. Selanjutnya data yang tersimpan dalam HDFS divisualisasikan menggunakan bahasa pemrograman PHP + Java Script.

6.3 Source Code

Berikut Source Code dari Project yang kami lakukan :

1. Source Code C pada Arduino Mega.

```
#include <SoftwareSerial.h> //Include the
   software serial library
```

```
int pinX = 4;                 //Arduino pin 7 to
   control pin pinX
int pinY = 5;                 //Arduino pin 6 to
   control pin pinY

char computerdata[20];
char sensordata[30];
byte computer_bytes_received = 0;
byte sensor_bytes_received = 0;
unsigned long previousMillis = 0;
const long interval = 2000;
float temp;
int ubah;
String dataString = "";
String ORP, EC, PH, DO, suhu;
char kirimsensor[100];

void setup() {
  pinMode(pinY, OUTPUT);
  pinMode(pinX, OUTPUT);
  pinMode(A0,INPUT);
  Serial.begin(9600);
  Serial1.begin(9600);
  Serial2.begin(9600);
  ubah = 0;
}

void serialEvent() {}

void loop()
{
  unsigned long currentMillis = millis();
  temp = read_temp();
  open_channel();
  if (Serial1.available() > 0)
  {
    if (ubah == 0 )
```

```
      {
        ORP = "";
        ORP = String(sensordata);
      }
      else if (ubah == 1)
      {
        DO = "";
        DO = String(sensordata);
      }
      else if (ubah == 2)
      {
        EC = "";
        EC = String(sensordata);
      }
      else if (ubah == 3)
      {
        suhu = "";
        suhu = String(temp);
        PH = "";
        PH = String(sensordata);
      }
      ubah++;
      if (ubah > 3)
        ubah = 0;
    }
    if (currentMillis - previousMillis >= interval
       ) {
      // save the last time you blinked the LED
      previousMillis = currentMillis;
      dataString = "#" + ORP + "," + PH + "," + EC
          + "," + DO + "," + suhu + "*";
      Serial2.println(dataString);
    }
  }

  void open_channel()
  {
    switch (ubah)
```

```
  {   //Looking to see what channel to open
    case 0:
      digitalWrite(pinX, LOW);
      digitalWrite(pinY, LOW);
      break;
    case 1:
      digitalWrite(pinX, HIGH);
      digitalWrite(pinY, LOW);
      break;
    case 2:
      digitalWrite(pinX, LOW);
      digitalWrite(pinY, HIGH);
      break;
    case 3:
      digitalWrite(pinX, HIGH);
      digitalWrite(pinY, HIGH);
      break;
  }
}

float read_temp(void)
{
  float v_out;
  float temp;
  v_out = analogRead(0);
  v_out *= 0.0048;
  v_out *= 1000;
  temp = (0.0512 * v_out) - 20.5128;
  return temp;
}
```

2. Source Code Python pada bagian Raspberry Pi 3. Source Code ini berfungsi sebagai penerima data masukan dari Arduino Mega sekaligus menjadikan Raspberry Pi 3 sebagai Socket Client yang mana data yang diterima langsung dikirim ke Socket Server.

```
import socket
import serial
```

```
import time
import sys

sensor = '/dev/ttyUSB0'
baud_= 9600
TCP_IP = '192.168.10.99'
TCP_PORT = 1987
BUFFER_SIZE = 512

ser = serial.Serial(sensor, baud_)
s = socket.socket(socket.AF_INET, socket.
   SOCK_STREAM)
s.connect((TCP_IP, TCP_PORT))
try:
 while True:
  waktu = time.strftime ("%Y-%m-%d,%H:%M:%S",
     time.localtime())
  respon = ser.readline()
  cekbyte = sys.getsizeof(respon)
        print cekbyte
        if (cekbyte>67):
     dataSensor = respon[respon.find("#")+1:
        respon.find("*")]
            dataBase=','.join([waktu,dataSensor
               ])
            print dataBase
            MESSAGE = dataBase
            s.send(MESSAGE)
            f = open('./sensor.csv','a')
            f.write(dataBase)
            f.write("\n")
            time.sleep(30)
        else:
            continue
except KeyboardInterrupt:
    s.close()
    ser.close()
```

3. Source Code Python pada Server Big Data. Code ini untuk menjadikan PC Server sebagai Socket Server yang me-listening data kiriman dari Socket Client dan sekaligus sebagai code untuk melakukan penyimpanan data pada Hadoop Distributed File System (HDFS) melalui skema framework hive.

```
import time
import pyhs2
import socket

TCP_IP = '192.168.10.99'
TCP_PORT = 1987
BUFFER_SIZE = 256
i = 1

s = socket.socket(socket.AF_INET, socket.
   SOCK_STREAM)
s.bind((TCP_IP, TCP_PORT))
s.listen(1)

con = pyhs2.connect(host='HadoopMaster', port
   =10000, authMechanism="PLAIN",
user='hduser',password='qwerty',database='
   default')
cur = con.cursor()
print 'Listening :'
conn, addr = s.accept()
print 'Connection address:', addr
print cur.getDatabases()

try:
    while i:
        dataBase = conn.recv(BUFFER_SIZE)
        if not dataBase: continue
        dataSplit = dataBase.split(',')
        tanggal = str(dataSplit[0])
        jam = str(dataSplit[1])
        orp = str(dataSplit[2])
```

```
            ph = str(dataSplit[3])
            ec = str(dataSplit[4])
            tds = str(dataSplit[5])
            sal = str(dataSplit[6])
            sg = str (dataSplit[7])
            do = str (dataSplit[8])
            temp = str (dataSplit[9])
            commandSave="INSERT INTO TABLE
               projectiotwater VALUES
               ('{}','{}','{}',
            '{}','{}','{}','{}','{}','{}','{}')".
               format
            (tanggal,jam,orp,ph,ec,tds,sal,sg,do,
               temp)
            cur.execute(commandSave)
            print "Save Success"
    except KeyboardInterrupt:
        conn.close()
```

4. Source Code PHP + Java Script untuk visualisasi data sensor

   (a) Contoh code untuk visualisasi data pH

```
<?php
  error_reporting(E_ALL);
  // Load this lib
  require_once __DIR__ . '/ThriftSQL.phar';
  // Try out a Hive query
  $hive = new \ThriftSQL\Hive( 'HadoopMaster
     ', 10000, 'hduser', 'qwerty' );
  $hiveTables = $hive
  //->setSasl( false ) // To turn SASL auth
     off, on by default
    ->connect()
    ->queryAndFetchAll( 'select * from
       projectiotwater limit 100' );
  //clear the client and close socket.
  $hive->disconnect();
```

```
  //$impala->disconnect();

function getValueTanggal($data, $type)
{
        $temp = [];
        $type = "ph";
        switch($type)
        {
                case "orp": $type = 2; break;
                case "ph": $type = 3; break;
                case "ec": $type = 4; break;
                case "tds": $type = 5; break;
                case "sal": $type = 6; break;
                case "sg": $type = 7; break;
                case "do": $type = 8; break;
                case "temp": $type = 9; break
                   ;
                default: return $temp;
        };
  foreach($data as $d)
  array_push($temp, ["tanggal" => $d[0] . $d
     [1], "value" => $d[$type]]);
  return $temp;
}

$data = getValueTanggal($hiveTables, "temp");
//echo json_encode($data);
?>
<script src="js/jquery.min.js"></script>
<script src="js/highcharts.js"></script>
<script src="js/exporting.js"></script>
<script type="text/javascript">

$(function () {
  $('#view').highcharts({
  chart: {
        type: 'spline'
        },
```

```
        title: {
          text: 'Grafik Sensor Air (PH)',
          x: -20 //center
           },
        subtitle: {
           text: '',
           x: -20
           },
        xAxis: {
         categories: [<?php foreach($data as $d)
            { echo "'" .  $d['tanggal'] .
            "',";}?>]
           },
           yAxis: {
            title: {
             text: 'PH (%)'
             },
            plotLines: [{
              value: 0,
              width: 1,
              color: '#808080'
              }]
           },
        tooltip: {
          valueSuffix: ''
          },
       legend: {
         layout: 'vertical',
         align: 'right',
         verticalAlign: 'middle',
         borderWidth: 0
          },
       series: [{
         name: 'ORP',
         data: [<?php foreach($data as $d){echo
            $d['value'] . ",";}?>]
          }
]
```

```
    });
});
</script>
<div id="view" style="min-width: 310px;
   height: 400px; margin: 0 auto"></div>
```

(b) Contoh code untuk visualisasi data temperatur

```
<?php
  error_reporting(E_ALL);
  // Load this lib
  require_once __DIR__ . '/ThriftSQL.phar';
  // Try out a Hive query
  $hive = new \ThriftSQL\Hive( 'HadoopMaster
     ', 10000, 'hduser', 'qwerty' );
  $hiveTables = $hive
  //->setSasl( false ) // To turn SASL auth
     off, on by default
    ->connect()
    ->queryAndFetchAll( 'select * from
       projectiotwater limit 100' );
//clear the client and close socket.
  $hive->disconnect();
  //$impala->disconnect();

function getValueTanggal($data, $type)
{
        $temp = [];
        $type = "temp";
        switch($type)
        {
                case "orp": $type = 2; break;
                case "ph": $type = 3; break;
                case "ec": $type = 4; break;
                case "tds": $type = 5; break;
                case "sal": $type = 6; break;
                case "sg": $type = 7; break;
                case "do": $type = 8; break;
                case "temp": $type = 9; break
```

```
                    ;
                default: return $temp;
        };
foreach($data as $d)
array_push($temp, ["tanggal" => $d[0], "value
   " => $d[$type] ]);
return $temp;
}
$data = getValueTanggal($hiveTables, "temp");
//echo json_encode($data);
?>
<script src="js/jquery.min.js"></script>
<script src="js/highcharts.js"></script>
<script src="js/exporting.js"></script>
<script type="text/javascript">
$(function () {
$('#view').highcharts({
chart: {
type: 'spline'
 },
title: {

  text: 'Grafik Sensor Air (temp)',
       x: -20 //center
       },
  subtitle: {
       text: '',
       x: -20
  },
  xAxis: {
categories: [<?php foreach($data as $d){ echo
    "'" .  $d['tanggal'] . "',";}?>]
  },
  yAxis: {
       title: {
       text: 'temp (%)'
         },
  plotLines: [{
```

```
            value: 0,
            width: 1,
            color: '#808080'
            }]
          },
        tooltip: {
            valueSuffix: ''
            },
            legend: {
              layout: 'vertical',
              align: 'right',
              verticalAlign: 'middle',
              borderWidth: 0
            },
     series: [{
     name: 'temp',
     data: [<?php foreach($data as $d){echo $d['
        value'] . ",";}?>]
            }
             ]
        });
});
</script>

<div id="view" style="min-width: 310px;
   height: 400px; margin: 0 auto"></div>
```

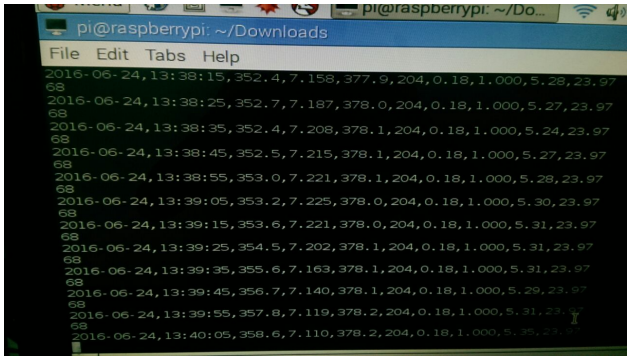
6.4 Tampilan Hasil

Dari percobaan di atas dapat kami hasilkan output data seperti berikut.

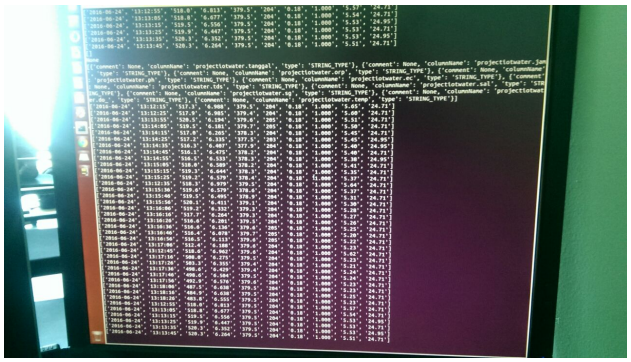
(a) Pengiriman data dari sensor ke Raspberry Pi

(b) Pengiriman data dari Raspberry Komputer Server

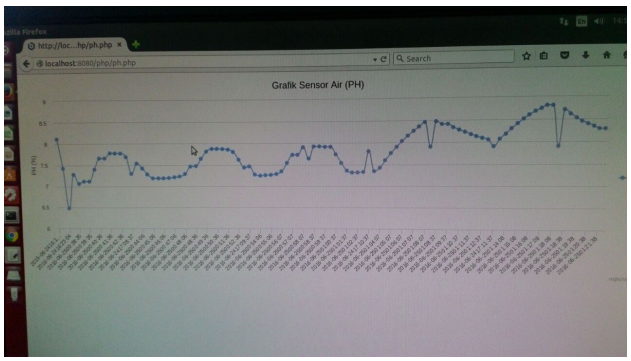
5. Tampilan grafik dari visualisasi data yang dikirim ke server.



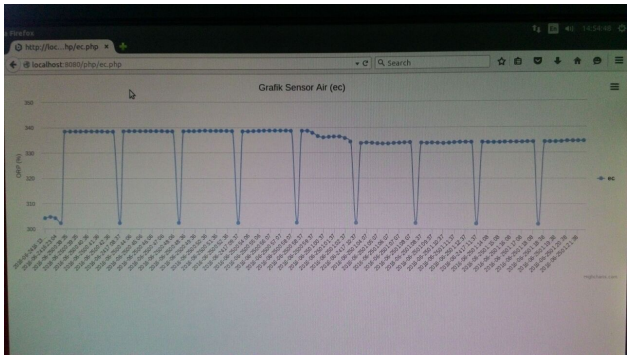
Gambar 6.1: Data Sensor yang diterima oleh Raspberry



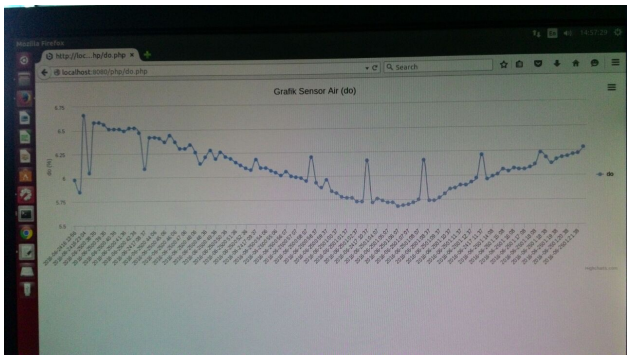
Gambar 6.2: Data Sensor yang diterima oleh Server



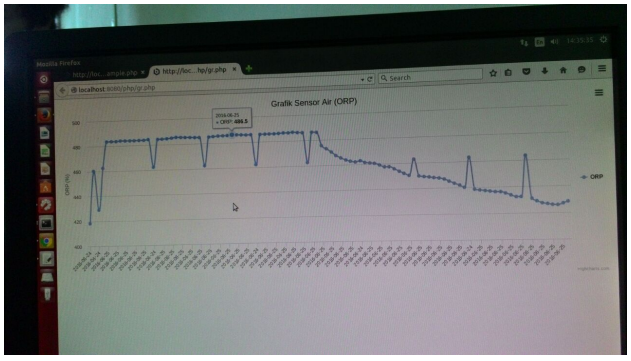
Gambar 6.3: Grafik pH



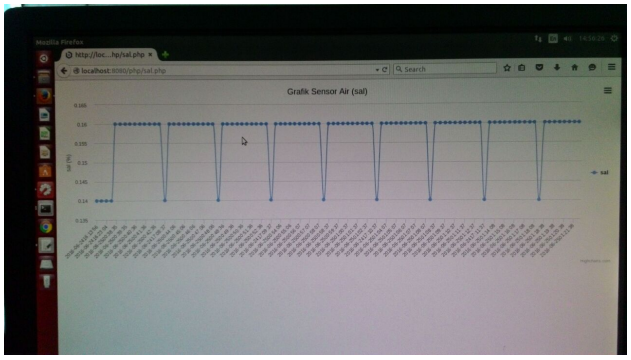
Gambar 6.4: Grafik Ec



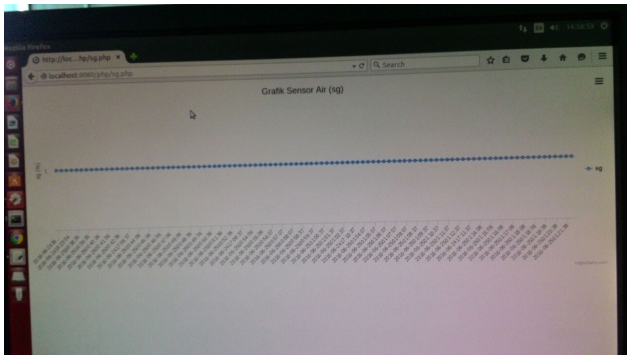
Gambar 6.5: Grafik DO



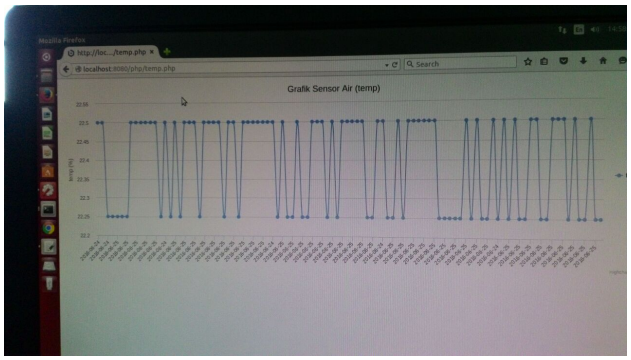
Gambar 6.6: Grafik ORP



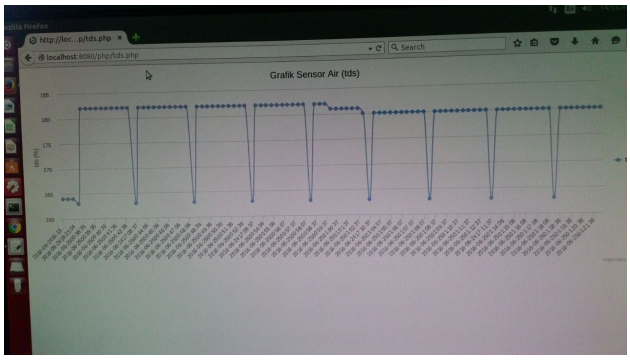
Gambar 6.7: Grafik Sal



Gambar 6.8: Grafik Sg



Gambar 6.9: Grafik Temp



Gambar 6.10: Grafik TDS

Daftar Pustaka

[1] Recommendation ITU-T Y.2060 Overview of Internet of Thing (06/2012).

[2] Atzori, Luigi, Antonio Iera, and Giacomo Morabito. "The internet of things: A survey." Computer networks 54.15 (2010): 2787-2805.

[3] https://www.raspberrypi.org/products/raspberry-pi-2-model-b/

[4] https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

[5] https://www.raspberrypi.org/downloads/

[6] https://www.raspberrypi.org/documentation/usage/python/

[7] Santoso, H. (2015) Panduan Praktis Arduino untuk Pemula.[e-book]. Tersedia di: http://www.elangsakti.com/2015/07/ebook-gratis-belajar-arduino-pemula.html.[Diakses pada: 5 Mei 2016].

[8] http://www.hendriono.com/blog/post/mengenal-arduino-mega2560

[9] https://id.wikipedia.org/wiki/Arduino

[10] http://techno-robotics.com/all-products/arduino-mega-2560/

[11] http://saptaji.com/2015/06/27/cara-menginstal-software-ide-arduino/

[12] https://www.robot-electronics.co.uk/htm/cmps3tech.htm

[13] http://www.waveshare.com/wiki/Liquid_Level_Sensor

[14] http://www.atlas-scientific.com/product_pages

[15] http://www.waveshare.com/wiki/MQ-5_Gas_Sensor

[16] http://www8.garmin.com/aboutGPS/

[17] https://www.sparkfun.com/products/10736

[18] http://www.waveshare.com/wiki/DHT11_Temperature-Humidity_Sensor

[19] http://www.waveshare.com/wiki/Sound_Sensor

[20] http://www.waveshare.com/wiki/Hall_Sensor

[21] http://www.waveshare.com/wiki/Laser_Sensor

[22] http://e-belajarelektronika.com/sensor-gerak-pir-passive-infra-red/

[23] http://akucintaelektronika.blogspot.co.id/2012/10/apa-sih-sensor-ultra-sonic-itu.html

[24] http://tokosuperelectronics.com/sensor-ultrasonic-srf02-murah-sensosr-jarak-srf02/

[25] https://www.robot-electronics.co.uk/htm/srf02techI2C.htm

[26] http://www.waveshare.com/wiki/Color_Sensor

[27] http://www.waveshare.com/wiki/Tilt_Sensor

[28] http://www.waveshare.com/wiki/Rotation_Sensor

[29] www.waveshare.com/wiki/Infrared_Reflective_Sensor

[30] http://www.waveshare.com/wiki/Moisture_Sensor

[31] http://www.waveshare.com/wiki/Flame_Sensor

[32] http://www.waveshare.com/wiki/UV_Sensor

[33] https://www.atlas-scientific.com/_files/code/pi_i2c_sample_code.pdf?

[34] https://patriciamantiri1128.files.wordpress.com/2012/06/sistel-makalah.pdf

[35] Sukaridhoto, Stritrusta. Jaringan Komputer. Politeknik Elektronika Negeri Surabaya : Surabaya, 2016

[36] http://www.serbacara.com/2015/05/pengertian-wifi-dan-bagaimana-cara-kerjanya.html

[37] http://www.pengertianku.net/2015/03/pengertian-gateway-dan-fungsinya.html

[38] http://woocara.blogspot.co.id/2015/05/pengertian-router-fungsi-router-dan-cara-kerja-router.html

[39] http://www.ibm.com/developerworks/linux/tutorials/l-pysocks/

[40] http://www.tutorialspoint.com/python/python_networking.htm

[41] Sukaridhoto S, ST.PhD(2016). "Jaringan Komputer. PENS. pp. 7184.

[42] https://docs.python.org/2/howto/sockets.html

[43] http://blog.kusandriadi.com/restful-web-service-pada-java-menggunakan-jersey-jax-rs-1-1/

[44] https://ixsan.wordpress.com/2015/11/23/membuat-rest-api-dengan-flask-python-microframework-1/

[45] Jason, Venner. Pro Hadoop. United States of America : Apress, 2009

[46] Noll, Michael G. Running Hadoop On Ubuntu Linux (Single-Node Cluster and Multi-Node Cluster). Germany : University of Luxembourg, 2012.

[47] Boedy, Cged. Pengertian Apache Hadoop. Pengertian Apache Hadoop. [Online] 05 2012. [Cited: 04 20, 2015.] http://cgeduntuksemua.blogspot.com/2012/04/pengertian- apache-hadoop.html

[48] Apache Hive [online] http://hive.apache.org/

[49] Pentreath, Nick. Machine Learning with Spark. UK : Packt Publishing Ltd, 2015

[50] Zaharia, Matei; Chowdhury, Mosharaf; Franklin, Michael J.; Shenker, Scott; Stoica, Ion. Spark: Cluster Computing with Working Sets (PDF). USENIX Workshop on Hot Topics in Cloud Computing (HotCloud)

[51] Apache Spark [online] http://spark.apache.org/

[52] Apache Hadoop [online] http://hadoop.apache.org/

[53] Noll, Michael G. Running Hadoop On Ubuntu Linux (Single-Node Cluster and Multi-Node Cluster). Germany : University of Luxembourg, 2012.

[54] Frampton, Mike. Mastering Apache Spark. UK : Packt Publishing Ltd, 2015

[55] Sankar, Krishna, Holden Karau. Fast Data Processing with Spark Second Edition. UK : Packt Publishing Ltd, 2015

[56] Apache Spark Online Documentation. http://spark.apache.org/docs/latest/

[57] Berners-Lee, Tim; James Hendler; Ora Lassila (May 17, 2001). "The Semantic Web". Scientific American Magazine. Retrieved March 26, 2008.

[58] Lee Feigenbaum (May 1, 2007). "The Semantic Web in Action". Scientific American. Retrieved February 24, 2010.

[59] Berners-Lee, Tim (May 17, 2001). "The Semantic Web". Scientific American. Retrieved March 13, 2008.

[60] Nigel Shadbolt, Wendy Hall, Tim Berners-Lee (2006). "The Semantic Web Revisited" (PDF). IEEE Intelligent Systems. Retrieved April 13, 2007.

[61] Ramanathan V. Guha (2013). "Light at the End of the Tunnel". International Semantic Web Conference 2013 Keynote. Retrieved March 8, 2015.

[62] https://en.wikipedia.org/wiki/Semantic_Web

[63] Buffa, Michel; Dehors, Sylvain; Faron-Zucker, Catherine; Sander, Peter (2005). "Towards a Corporate Semantic Web Approach in Designing Learning Systems: Review of the Trial Solutioins Project" (PDF). International Workshop on Applications of Semantic Web Technologies for E-Learning. Amsterdam, Holland. pp. 7376.

[64] http://www.scribd.com/doc/40522315/Semantic-Web

[65] wijayanto H. "Penerapan web semantic Dalam pencarian katalog buku Di perpustakaan stmik sinar nusantara surakarta". Surakarta 2012.