

MODUL HEALTH DATA SCIENCE



Disusun Oleh:

TIM DOSEN PRODI SARJANA TERAPAN MANAJEMEN INFORMASI KESEHATAN

UNIVERSITAS INDONESIA MAJU

PROGRAM STUDI SARJANA TERAPAN MANAJEMEN INFORMASIKESEHATAN

FAKULTAS VOKASI

UNIVERSITAS INDONESIA MAJU

JAKARTA

2022



Modul Health Data Science

Nama Mahasiswa : _____
NPM : _____

Program Studi Sarjana Terapan Manajemen Informasi Kesehatan

Fakultas Vokasi

Universitas Indonesia Maju

2022

KATA PENGANTAR

Buku petunjuk praktikum ini disusun untuk memenuhi kebutuhan mahasiswa sebagai panduan dalam melaksanakan praktikum Health Data Science, untuk mahasiswa program studi D4 Manajemen Informasi Kesehatan (MIK) UIMA. Dengan adanya buku petunjuk praktikum ini diharapkan akan membantu dan mempermudah mahasiswa dalam memahami dan melaksanakan praktikum Health Data Science sehingga akan memperoleh hasil yang baik.

Materi yang dipraktikkan merupakan materi yang selaras dengan materi kuliah Health Data Science. Untuk itu dasar teori yang didapatkan saat kuliah juga akan sangat membantu mahasiswa dalam melaksanakan praktikum ini.

Buku petunjuk ini masih dalam proses penyempurnaan. Insha Allah perbaikan akan terus dilakukan demi kesempurnaan buku petunjuk praktikum ini dan disesuaikan dengan perkembangan ilmu pengetahuan. Semoga buku petunjuk ini dapat dipergunakan sebagaimana mestinya.

Jakarta, September 2022

Penyusun

DAFTAR ISI

HALAMAN SAMPUL.....	i
KATA PENGANTAR.....	iii
DAFTAR ISI	iv
BAB I SISTEM BENAM	1
BAB II KOMUNIKASI DATA.....	15
BAB III PROJECT IoT	35
BAB IV AKUISISI DATA: IMPLEMENTASI DAN PENGAMBILAN DATA DEN- GAN PEMROGRAMAN PHYTHON DAN C	51
BAB V TEKNOLOGI BIG DATA: IMPLEMENTASI HADOOP	65
DAFTAR PUSTAKA.....	93

BAB I

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)

A. 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+.

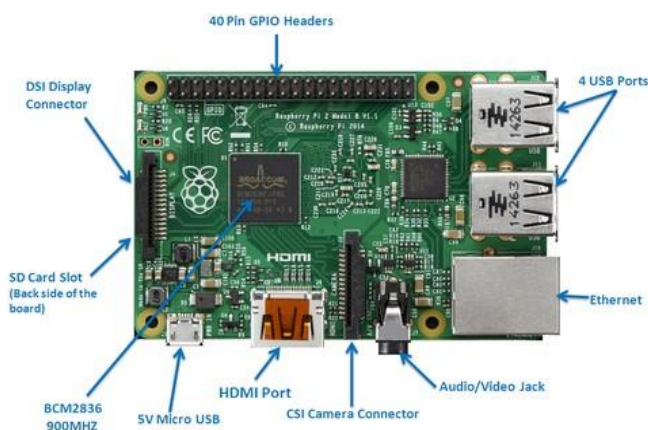
February 2015, Raspberry Pi 2 dirilis di pasaran. Board komputer baru tersedia hanya satu dalam sekali konfigurasi (model B) dan menyediakan fitur baru diantaranya Broadcom BMC2836 SoC dengan CPU ARM Cortex-A7 quad-core dan dual-core GPU Video IV. RAM sebesar 1 GB dengan spesifikasi yang mirip dengan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

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 1.1: 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. 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.

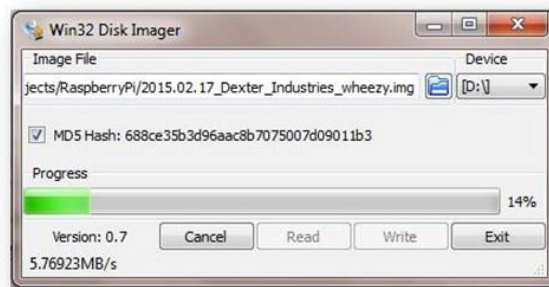
3. Image Sistem Operasi

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/macosex-apple-pi-baker/>.

4. Koneksi Internet

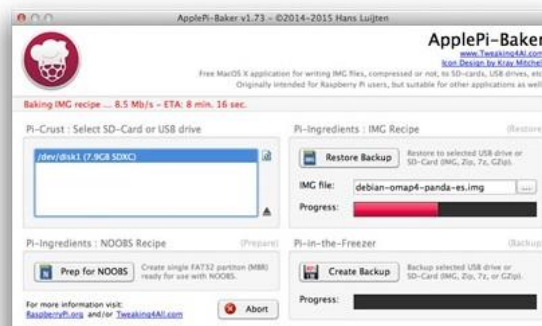
Langkah-langkah instalasi:

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 1.2: Win 32 Disk Imager

Untuk sistem operasi Macintosh dengan ApplePi-Baker:



Gambar 1.3: ApplePi Baker

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

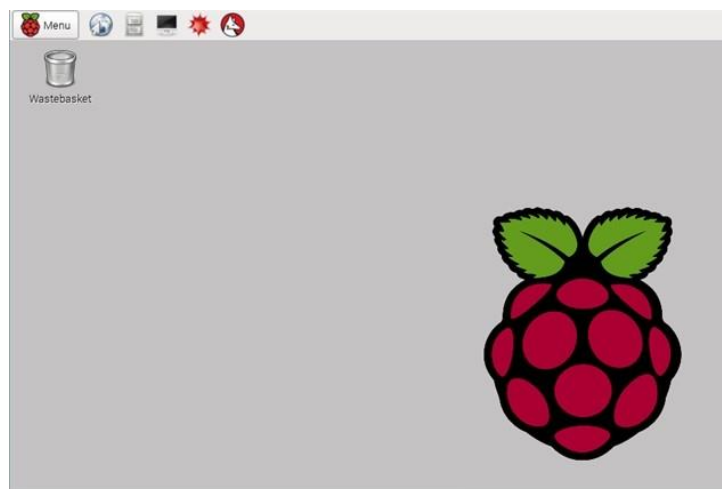


Gambar 1.4: Raspberry Pi dan perangkat-perangkat tambahannya

5. Nyalakan Raspberry Pi 2

6. Proses instalasi dimulai

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



Gambar 1.5: Raspbian Desktop

2. 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 > namafile.py*:

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

Gambar 1.7: Pemrograman Python

B. 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 mikrokontroler sering dikenal dengan simbol nC, uC, atau MCU. Terjemahan bebas dari pengertian tersebut, bisa dikatakan bahwa mikrokontroler adalah komputer yang berukuran mikro dalam 1 chip IC (integrated circuit) yang terdiri dari prosesor, memory, dan

I/O yang bisa kita kontrol dengan memprogramnya. I/O juga sering disebut dengan GPIO (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 kelebihanannya 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.

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].

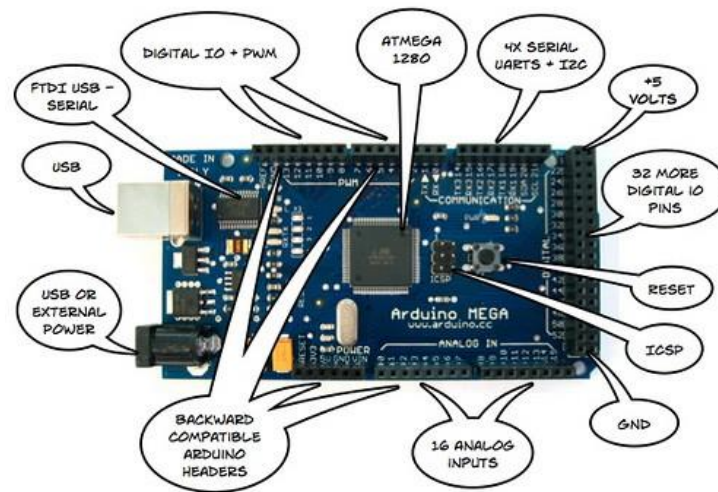
Table 1.1: Spesifikasi Arduino Mega 2560

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

3. Instalasi Sistem Operasi

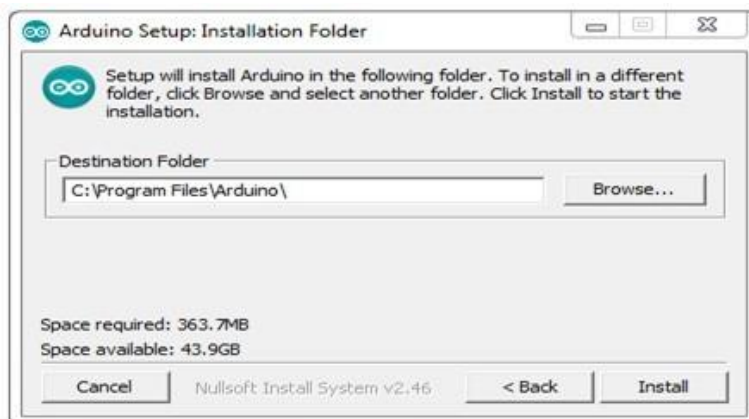
IDE Arduino Untuk memprogram board Arduino, kita butuh aplikasi IDE (Integrated Development Environment) bawaan dari Arduino. Aplikasi ini berguna untuk membuat, membuka, dan mengedit source code Arduino (Sketches, para programmer menyebut source code arduino dengan istilah "sketches"). Sketch merupakan source code yang berisi logika dan algoritma yang akan diupload ke dalam IC mikrokontroler (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. Masukkan folder instalasi untuk arduino atau biarkan default di C: \



Gambar 1.9: Gambar Proses Instalasi

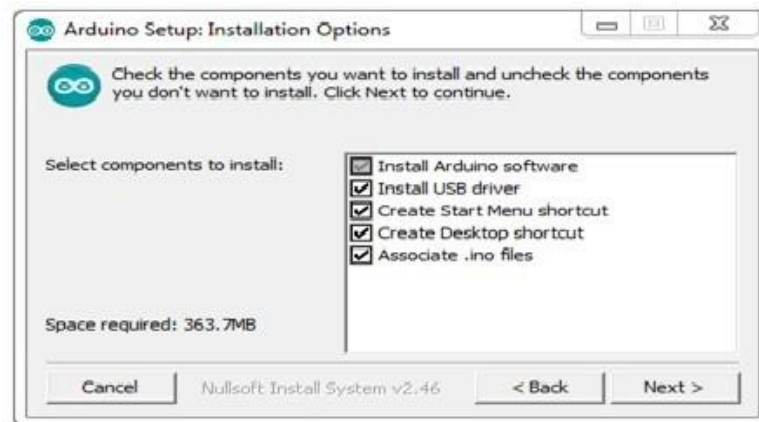
Program Files\ Arduino



Gambar 1.10: Gambar Proses Instalasi

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

6. Selanjutnya proses instalasi akan dimulai sampai selesai

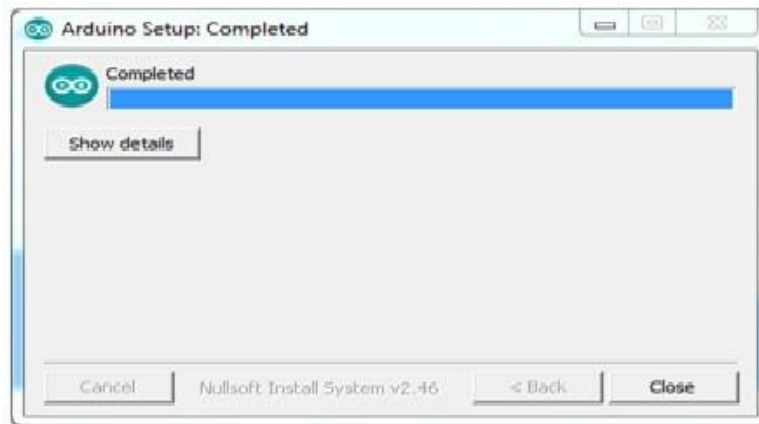


Gambar 1.11: Gambar Proses Instalasi

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, biasanya 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 mikrokontroler.
- 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

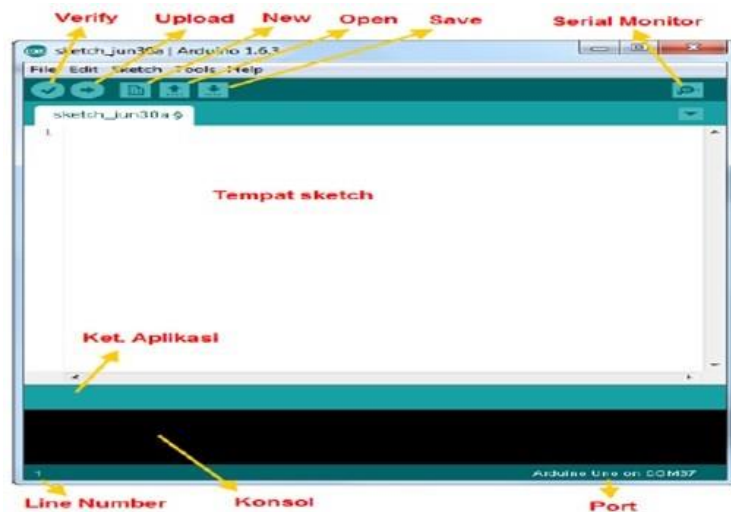


Gambar 1.12: Gambar Proses Instalasi

- 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.

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 1.13: 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 II

KOMUNIKASI DATA

A. 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 teknologi 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 memungkinkan setiap komputer untuk saling terhubung serta melakukan komunikasi, baik itu pada LAN atau internet.



Gambar 2.1 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

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 mengalami 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 alamat IP versi 4 tersebut adalah 255.255.255.255 dimana nilai dihitung dari nol sehingga nilai host yang dapat ditampung adalah $256 \times 256 \times 256 \times 256 = 4.294.967.296$ 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 mengidentifikasi alamat host (dapat berupa workstation, server atau sistem lainnya yang berbasis teknologi TCP/IP) di dalam jaringan

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 2.1.Tabel Oktet Kelas IP

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 proyek 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
```

B. 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 socket), 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 socket dalam bahasa seperti Python adalah di kelas pembantu dan metode yang dapat menyederhanakan pemrograman socket.

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'`):

```
1 [camus]$ <strong>python</strong>
2 Python 2.4 (#1, Feb 20 2005, 11:25:45)
3 [GCC 3.2.2 20030222 (Red Hat Linux 3.2.2-5)] on linux2
4 Type "help", "copyright", "credits" or "license" for more
5 information.
6 >>> <strong>import socket</strong>
7 >>> <strong>socket.gethostbyname('www.ibm.com')</strong>
8 '129.42.19.99'
9 >>>
```

Gambar 2.2: Pemrograman Socket API

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

2. Membuat dan destroy Socket

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

```
1 streamSock = <strong>socket.socket</strong>( socket.AF_INET, socket.SOCK_STREAM )
2
3 dgramSock = <strong>socket.socket</strong>( socket.AF_INET, socket.SOCK_DGRAM )
```

Gambar 2.3 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 socket TCP dan SOCK_DGRAM untuk socket UDP). Jika sistem operasi yang mendasari Anda mendukung IPv6, Anda juga dapat menentukan socket.AF_INET6 untuk membuat socket IPv6.

Untuk memutus sambungan 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 socket akan menghasilkan kesalahan.

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. Server Socket

Server socket biasanya mengekspos layanan pada jaringan. Karena server dan client socket dibuat dengan cara yang berbeda, saya mendiskusikannya secara independen.

Setelah Anda membuat socket, Anda menggunakan metode mengikat untuk mengikat alamat itu, metode listed dalam menempatkannya 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


```

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

```

Gambar 2.4: Server socket

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

5. Client socket

Mekanisme untuk menciptakan dan menghubungkan socket klien mirip dengan setup socket Server. Setelah menciptakan socket, alamat diperlukan - lokal tidak mengikat socket (seperti yang terjadi dengan server) melainkan untuk mengidentifikasi mana socket 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

```

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

```

Gambar 2.5: 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 socket data-gram, Anda harus memberikan informasi tujuan dengan data yang ingin Anda kirim. Dengan menggunakan koneksi, saya sudah cache informasi ini dengan klien dan metode send dapat terjadi seperti versi aliran socket (tidak ada alamat tujuan diperlukan). Anda dapat panggilan terhubung lagi untuk kembali menentukan target pesan datagram klien.

6. Socket options

Socket diset standar, tapi itu mungkin untuk mengubah perilaku socket 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 2.6: Opsi socket

nan server socket. Anda dapat meningkatkan socket 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

7. Asynchronous I/O

Python menawarkan asynchronous I / O sebagai bagian dari modul `select`. Fitur ini mirip dengan mekanisme `select` 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 socket 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 socket 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 = select.select( [sys.stdin], [], [] )
2
3 print sys.stdin.read()
```

Gambar 2.7: 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 socket. Pada contoh berikut dibawah, dua socket 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 = socket.socket( socket.AF_INET, socket.SOCK_STREAM )
5 sock2 = socket.socket( socket.AF_INET, socket.SOCK_STREAM )
6
7 sock1.connect( ('192.168.1.1', 25) )
8 sock2.connect( ('192.168.1.1', 25) )
9
10 while 1:
11     # Await a read event
12     rlist, wlist, elist = select.select( [sock1, sock2], [], [], 5
13
14     # Test for timeout
15     if [rlist, wlist, elist] == [ [], [], [] ]:
16
17         print "Five seconds elapsed.\n"
18
19     else:
20
21     # Loop through each socket in rlist, read and print the available data
22     for sock in rlist:
23
24         print sock.recv( 100 )
```

Gambar 2.8: Asynchronous I/O

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

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.

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

C. Restful

REST, singkatan bahasa Inggris dari representational state transfer atau transfer keadaan representasi, adalah suatu gaya arsitektur perangkat lunak untuk untuk pendistribusian 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 2.9: Restfull

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)

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

```

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.

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 ekstensi-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 menginterpretasi 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 environment aktif. Kita bisa mengaktifkan virtual environment 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://ipserverurl:/ flask` akan mengeksekusi fungsi home() dan menampilkan teks Hello, world!
3. Perintah ini digunakan untuk menjalankan 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_template`. 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`.

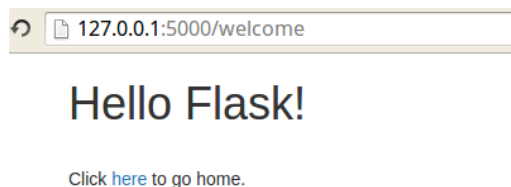
welcome.html

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 III PROJECT IoT

A. 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

B. 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>

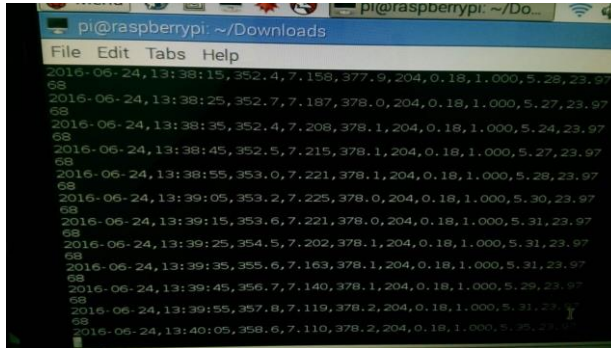
```

C. Tampilan Hasil

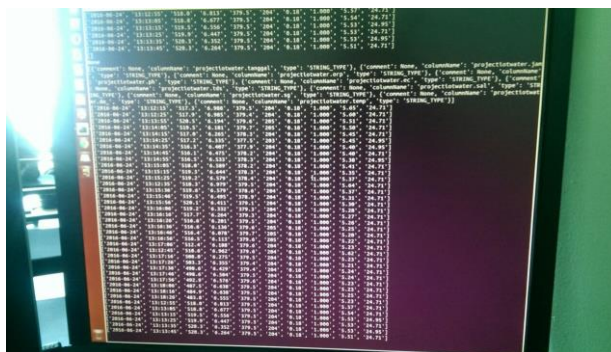
Dari percobaan di atas dapat kami hasilkan output data seperti berikut.

- (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.



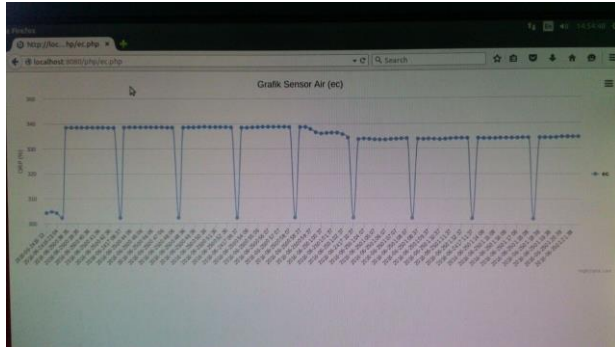
Gambar 3.1: Data Sensor yang diterima oleh Raspberry



Gambar 3.2: Data Sensor yang diterima oleh Server



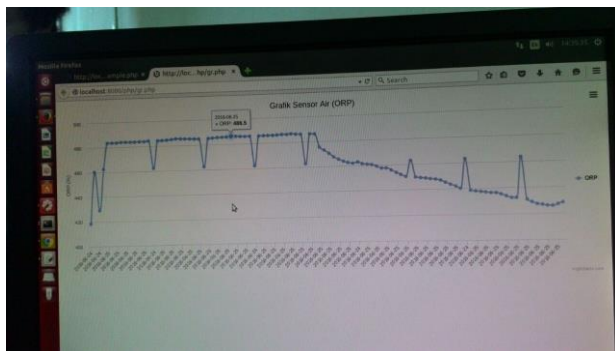
Gambar 3.3: Grafik pH



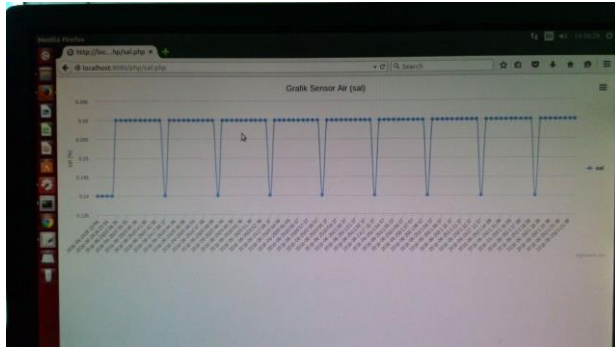
Gambar 3.4: Grafik Ec



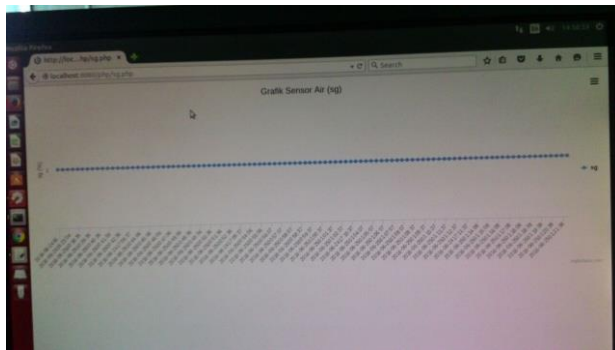
Gambar 3.5: Grafik DO



Gambar 3.6: Grafik ORP



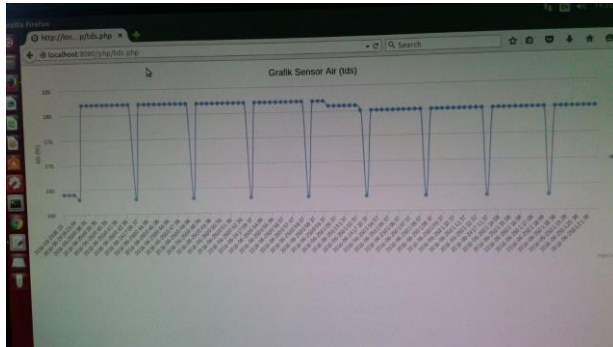
Gambar 3.7: Grafik Sal



Gambar 3.8: Grafik Sg



Gambar 3.9: Grafik Temp



Gambar 3.10: Grafik TDS

BAB IV

AKUISISI DATA: IMPLEMENTASI DAN PENGAMBILAN DATA DENGAN PEMROGRAMAN PYTHON DAN C

A. 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
}
```

B. Membaca Input Digital dengan Arduino

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

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

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
}
}
```

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 frequency
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();           // default 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);
}

```

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();
}

```

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);
}

```

Membaca Input Sensor Air Atlas dengan Ar-duino

```

//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 = Serial1.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;
}

```

Membaca Input secara GPIO dengan Raspberry 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();
```

Membaca Input secara I2C dengan Raspberry 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 po-  
hon 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 Rasp-
berry 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)
```

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 V

TEKNOLOGI BIG DATA: 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 :

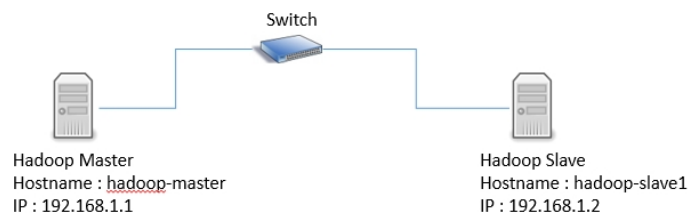
Table 5.1: Spesifikasi Perangkat Keras

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
- PostgreSQL 9.3
- Apache Hive 1.2.1
- OpenSSH-Server
- Apache Spark 1.6.1
- Python 2.7.0
- Scala 2.4.0
- Java 7 Open-JDK

Model infrastruktur dari Hadoop yang dibangun adalah



Gambar 5.1: Desain Infrastruktur Server Hadoop

A. Instalasi Hadoop

Instalasi dan konfigurasi dilakukan pertama kali pada komputer yang bertindak sebagai hadoop-master (single node), untuk menjadikan multinode cluster akan tinggal melakukan synchronize 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 :

```

hduser@HadoopMaster:~$ start-all.sh
This script is deprecated. Instead use start-dfs.sh and start-yarn.sh
Starting namenodes on [HadoopMaster]
HadoopMaster: starting namenode, logging to /usr/local/hadoop/logs/hadoop-hduser-namenode-HadoopMaster.out
HadoopMaster: starting datanode, logging to /usr/local/hadoop/logs/hadoop-hduser-datanode-HadoopMaster.out
HadoopSlave1: starting datanode, logging to /usr/local/hadoop/logs/hadoop-hduser-datanode-HadoopSlave1.out
Starting secondary namenodes [0.0.0.0]
0.0.0.0: starting secondarynamenode, logging to /usr/local/hadoop/logs/hadoop-hduser-secondarynamenode-HadoopMaster.out
Starting yarn daemons
starting resourcemanager, logging to /usr/local/hadoop/logs/yarn-hduser-resourcemanager-HadoopMaster.out
HadoopMaster: starting nodemanager, logging to /usr/local/hadoop/logs/yarn-hduser-nodemanager-HadoopMaster.out
HadoopSlave1: starting nodemanager, logging to /usr/local/hadoop/logs/yarn-hduser-nodemanager-HadoopSlave1.out

```

Gambar 5.2: Tampilan eksekusi start-all.sh

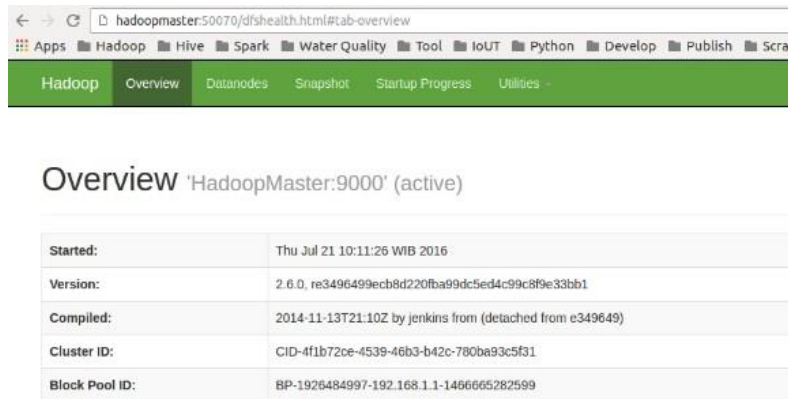
Untuk melihat daemon hadoop yang berjalan gunakan perintah jps (dari hduser)

```

hduser@HadoopMaster:~$ jps
4085 NodeManager
3448 NameNode
3798 SecondaryNameNode
4433 Jps
3580 DataNode
3953 ResourceManager

```

Gambar 5.2: Hasil Eksekusi jps yang memperlihatkan daemon hadoop yang berjalan di hadoop master



Gambar 5.3: Hadoop Web Interface

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,

```
192.168.11      .1      hadoop-master
192.168.12      .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 synchronize file hadoop pada semua node

```
$sudo apt-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 selanjutnya 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
```

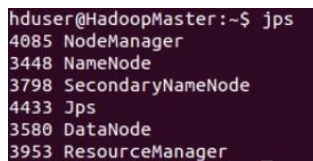
Selanjutnya lakukan format pada namenode

```
$hadoop namenode -format
```

Lalu menjalankan hadoop framework pada master node

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

Hasil eksekusi jps pada master node



```
hduser@HadoopMaster:~$ jps
4085 NodeManager
3448 NameNode
3798 SecondaryNameNode
4433 Jps
3580 DataNode
3953 ResourceManager
```

Gambar 5.4: Hasil Eksekusi jps di hadoop master

Hasil eksekusi jps pada slave node

Informasi pada Hadoop Web Interface

B. Instalasi Hive

langkah-langkah instalasi Apache Hive 1.2.1 :

```
hduser@HadoopSlave1:~$ jps
3201 NodeManager
3557 Jps
3061 DataNode
hduser@HadoopSlave1:~$
```

Gambar 5.5: Hasil Eksekusi jps di hadoop slave

Datanode Information

In operation

Node	Last contact	Admin State	Capacity	Used	Non DFS Used	Remaining	Blocks	Block pool used	Failed Volumes	Version
HadoopSlave1 (192.168.1.2:50020)	0	In Service	47.04 GB	15.37 MB	9.66 GB	37.36 GB	1876	15.37 MB (0.03%)	0	2.6.0
HadoopMaster (192.168.1.1:50010)	1	In Service	191.17 GB	15.43 MB	17.48 GB	173.67 GB	1876	15.43 MB (0.01%)	0	2.6.0

Gambar 5.6: 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
```

```
hduser@HadoopMaster:~/usr/local$ hive --service metastore
Starting Hive Metastore Server
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/hive/lib/hive-jdbc-2.0.0-standalone.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/hive/lib/log4j-slf4j-impl-2.4.1.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
```

Gambar 5.7: Menjalankan service metastore hive

```
hduser@HadoopMaster:~/usr/local$ hive --service hiveserver2
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/hive/lib/hive-jdbc-2.0.0-standalone.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/hive/lib/log4j-slf4j-impl-2.4.1.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
```

Gambar 5.8: Menjalankan service hiveserver2 hive

Kemudian jalankan hive jika ingin mengakses hive shell :

\$hive

```
hduser@HadoopMaster:~/usr/local$ hive
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/hive/lib/hive-jdbc-2.0.0-standalone.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/hive/lib/log4j-slf4j-impl-2.4.1.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]

Logging initialized using configuration in file:/usr/local/hive/conf/hive-log4j2.properties
Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. te
z, spark) or using Hive 1.x releases.
hive> █
```

Gambar 5.9: Mengakses hive shell

C. 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
```

Machine Learning dengan Spark

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

```
16/05/20 17:05:36 INFO Remoting: Remoting started; listening on addresses :[akka.
16/05/20 17:05:36 INFO util.Utils: Successfully started service 'sparkDriverActor
16/05/20 17:05:36 INFO spark.SparkEnv: Registering MapOutputTracker
16/05/20 17:05:36 INFO spark.SparkEnv: Registering BlockManagerMaster
16/05/20 17:05:36 INFO storage.DiskBlockManager: Created local directory at /tmp/
16/05/20 17:05:37 INFO storage.MemoryStore: MemoryStore started with capacity 511
16/05/20 17:05:37 INFO spark.SparkEnv: Registering OutputCommitCoordinator
16/05/20 17:05:42 INFO server.Server: jetty-8.y.z-SNAPSHOT
16/05/20 17:05:42 INFO server.AbstractConnector: Started SelectChannelConnector@0
16/05/20 17:05:42 INFO util.Utils: Successfully started service 'SparkUI' on port
16/05/20 17:05:42 INFO ui.SparkUI: Started SparkUI at http://127.0.1.1:4040
16/05/20 17:05:44 INFO executor.Executor: Starting executor ID driver on host loc
alhost
16/05/20 17:05:44 INFO util.Utils: Successfully started service 'org.apache.spark
.network.netty.NettyBlockTransferService' on port 37095.
16/05/20 17:05:44 INFO netty.NettyBlockTransferService: Server created on 37095
16/05/20 17:05:44 INFO storage.BlockManagerMaster: Trying to register BlockManage
r
16/05/20 17:05:44 INFO storage.BlockManagerMasterEndpoint: Registering block mana
ger localhost:37095 with 511.1 MB RAM, BlockManagerId(driver, localhost, 37095)
16/05/20 17:05:44 INFO storage.BlockManagerMaster: Registered BlockManager
Welcome to

  _____
 /_ _ _ _ _ \
| | | | | |
| |_|_|_|_|
|_|_|_|_|_|

version 1.6.1

Using Python version 2.7.3 (default, Sep 26 2013 20:08:41)
SparkContext available as sc, SQLContext available as sqlContext.
>>> █
```

Gambar 5.10: 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
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 approximately 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. prediction
AndLabel = 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/                               my
  GradientBoostingClassificationModel")

```

- 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.toString())

# 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()

```


DAFTAR PUSTAKA

- Ansari, S., Mohanlal, R., Poncela, J., Ansari, A., & Mohanlal, K. (2015). Importance of big data. In *Handbook of research on trends and future directions in big data and web intelligence* (pp. 1-19). IGI Global.
- Nugroho, F. P., Abdullah, R. W., Wulandari, S., & Hanafi, H. (2019). Keamanan Big Data di Era Digital di Indonesia. *Jurnal Informa: Jurnal Penelitian dan Pengabdian Masyarakat*, 5(1), 28-34.
- Oussous, A., Benjelloun, F. Z., Lahcen, A. A., & Belfkih, S. (2018). Big Data technologies: A survey. *Journal of King Saud University-Computer and Information Sciences*, 30(4), 431-448.
- Pal, S. K., Meher, S. K., & Skowron, A. (2015). Data science, big data and granular mining. *Pattern Recognit. Lett.*, 67(P2), 109-112.
- Yaqoob, I., Hashem, I. A. T., Gani, A., Mokhtar, S., Ahmed, E., Anuar, N. B., & Vasilakos, A. V. (2016). Big data: From beginning to future. *International Journal of Information Management*, 36(6), 1231-1247.