Content :

* 1. Perancangan dan pembuatan software
     1. [File management](Software-Development.docx#FileManagement)
     2. [Software Structure](#SoftwareStucture)
        1. Logsheet (data gathering, collection and persistenace from sensor)
        2. Web services
        3. Data Model
        4. Time keeping for synchronizing sampling
        5. Start up
     3. Pejelasan File
        1. [dataLogger.ino, datalogger.h](#StrukturArduino)
           1. Importing libraries
           2. Global variable, constant and objects declaration
           3. Setup
           4. Loop
        2. logsheet.h, logsheet.cpp
           1. Function execute()
           2. Function pendukung WEB
           3. Function private
        3. model.h, model.cpp
        4. sequenceTimer.h, sequenceTimer.cpp
        5. start\_up.h
           1. Menampilkan logo Garuda, seperti pada gambar berikut.
           2. Menampilkan welcome, seperti pada urutan gambar berikut.
           3. Menampilkan Step start up.
     4. Flowchart – datalogger
     5. Pelayanan WEB / WEB Services
     6. Authentication & Authorization (Auth)
     7. Static File
     8. URL Controller
     9. Perancangan WEB Page / Laman WEB.
        1. Home Page – laman utama/induk
           1. Struktur file html
           2. Mekanisme Auth
           3. Bagian Laman induk/utama

Topnav - Top navigasi (menu)

Element yang menjadi satu dengan id wrapper

Top container

Dial container

Graph container

Window modal dialog saat menu About active, seperti gambar berikut ini.

* + - * 1. Tabulasi dari Auth
        2. Daftar API laman utama
        3. Fungsi Javascript

userAccess (untuk Auth)

setupWidgets (untuk menggambar Dial - gauge, trending)

InitTrendingData – data awal trending.

* + - * 1. AJAX - Asynchronous **JavaScript** And XML
        2. Laman About
      1. Report Page – laman report (reporting)
         1. userAccess (untuk Auth)
         2. fillDataTable
         3. generateReportTable.
         4. Bagian laman report

Topnav

Element yang menjadi satu dengan id container

Report Header

Body table

Selection hari lengkap dengan button Dowload as PDF

* + - * 1. Window modal dialog saat menu About active
        2. Daftar API yang digunakan untuk laman report.
        3. Penggunaan JSON
        4. Fungsi Javascript

userAccess (untuk Auth)

fillUpParameter

getSamplingTime.

* + - 1. Laman config
         1. Bagian laman config
         2. Topnav
         3. Element yang menjadi satu dengan div content

Sampling Time

Radio Button

Parameter

Submit Button

* + - * 1. Window modal dialog saat menu About active
        2. Daftar API pada laman Config
      1. Login Page – laman Login (untuk mendukiung Auth)
         1. Fungsi javascript

userAccess (untuk Auth)

loginStatus

* + - * 1. Bagian laman login

Topnav

Element yang menjadi satu dengan div content

Username & Password

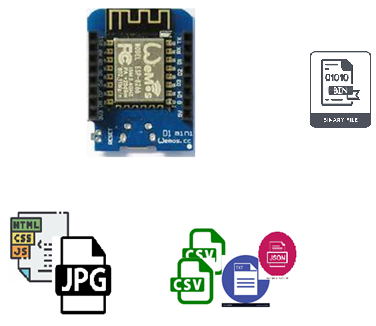
Login Button

* + - * 1. Window modal dialog saat menu About active
        2. Daftar API untuk laman login
      1. Logout

3.2.1. File management

Dalam pembuatan software ini, ESP-8266 terbagi dalam beberapa kelompok penyimpana file :

* File program 2 MB
* Local storage LittleFS 2 MB
  + Konfiguration – Sensor, User
  + Logsheet – data trending, hourly average selama 1 minggu (7 hari), daily average selama 1 tahun
  + Web file – HTML, CSS, JS dan Images



File: /logsheet/Friday\_ls.csv

File: /logsheet/Monday\_ls.csv

File: /logsheet/Saturday\_ls.csv

File: /logsheet/Sunday\_ls.csv

File: /logsheet/Thursday\_ls.csv

File: /logsheet/Tuesday\_ls.csv

File: /logsheet/Wednesday\_ls.csv

File: /logsheet/sensors.cfg

File: /modal-dialog.js

File: /report.html

File: /report.js

File: /style.css

File: /widgetsClass.js

**Log-sheet**

**compiled**

**root**

File: /authentication.js

File: /config.html

File: /config.js

File: /gmf.png

File: /index.html

File: /index.js

File: /login.html

File: /login.js

File: /logoGMF.png

File: /users.cfg

3.2.2. Software Structure

1. Logsheet (data gathering, collection and persistenace from sensor)
   * Get sensor data complete with filtering (alfa ema)
   * Display to OLED
   * Averaging – minute, hourly and daily
   * Saving to local storage (LittleFS) hourly base on day of week, daily base on year
   * Data Services for web
     + Sensor configuration
     + Logsheet hourly
     + Trending data (temperature and humidity)
2. Web services :
   * Server side (back end)
     + Static files services (Images, CSS and JS)
     + HTML
     + AJAX
   * Client side (Front end) pages / User Interface
     + Home
     + Report
     + Configuration
     + Login
     + Logout
     + About
3. Data Model
   * Sensor parameters (High, Low, Unit, Alarm, alfa ema)
   * Loghsheet (temperature and humidity)
   * User (username, password, email, level)
4. Time keeping for synchronizing sampling
5. Start up
   * Display logo during start up
   * Display welcome messages

Adapun fungsi tersebut di atas, dibagi menjadi beberapa bagian software dengan memisahkan sesuai dengan fungsinya (OOP principles) sebagai berikut :

* dataLogger.ino, datalogger.h

Sebagai program utama berikut sebagai pintu masuk (entry point) yang di dalamnya terdapat tulang punggung (skeleton) program Arduino – ESP . Struktur program Arduino minial sperti tampak dalam cuplikan program yang dibuat dari IDE Arduino berikut ini :

void setup() {

// put your setup code here, to run once:

}

void loop() {

// put your main code here, to run repeatedly:

}

Struktur minimal pada program Arduino (yang dipakai oleh esp8266 ini) setidaknya memuat fungsi void setup() dan void loop().

Karena keperluan selanjutnya, diperlukan importing beberapa library, serta penggunaan global variable, maka diperlukan tambahan seperti pejelasan pada sub bab berikut ini.

dataLogger.h berisi seluruh definisi global variables dan konstanta yang digunakan dalam keseluruhan program ESP-8266 ini.

* Importing libraries

Diperlukan beberapa libraries untuk mengembangkan software IoT Data Logger ini, baik yang langsung di-import melalui datalogger.ino ataupun saat pembuatan class (Logsheet, Model, StartUp). Libraries yang dimaksud adalah untuk mempermudah dalam pengembangan software sehingga tidak harus membuat coding sendiri mulai dari awal (scratch). Libraries ini dibagi menjadi pada kepentinganya sbb:

* Core Arduino – ESP8266
  + #include <Arduino.h>
* WIFI and WEB
  + #include <ESP8266WiFi.h>
  + #include <ESPAsyncTCP.h>
  + #include <ESPAsyncWebServer.h>
  + #include <ESP8266WiFiMulti.h>
* Library dibuat sendiri :
  + #include "dataLogger.h"
  + #include "SequenceTimer.h"
  + #include "logsheet.h"
  + #include "start\_up.h"
* Libraries untuk pembuatan Logsheet :
  + #include <Arduino.h>
  + #include <ArduinoJson.h>
  + #include <DHT.h>
  + #include <Wire.h> // Include Wire if you're using I2C
  + #include <SPI.h>
  + #include <Adafruit\_GFX.h>
  + #include <Adafruit\_SSD1306.h>
  + #include <LittleFS.h>
  + #include "model.h"
* Libraries untuk pembuatan SequenceTimer:
  + #include <Arduino.h>
* Libraries untuk pembuatan StartUp:
  + #include <Arduino.h>
  + #include <Wire.h> // Include Wire if you're using I2C
  + #include <SPI.h>
  + #include <Adafruit\_SSD1306.h>
* Global variable, constant and objects declaration

Diperlukan juga variable dan object yang mempunyai cakupan yang luas agar bisa dipakai (akses) oleh semua fungsi dalam IoT Data Logger ini. Variable itu tempat menyimpan status/data dari satu operasi fungsi dan digunakan untuk fungsi yang lain sehingga update status tadi tidak hilang.

Global variable dan constants bisa dideklarasikan pada progam utama (datalogger.ino) atau datalogger.h, bahkan lewat libraries yang di import (logsheet.h, model.h etc…). Untuk Mempermudah pelacakan dan pembuatan IoT Data Logger ini semua constants menggunakan huruf capatital (misalnya SAMPLING\_TIME\_MIN, SAMPLING\_TIME\_MAX, SIMULATION etc….).

Global objects adalah implementasi dari class yang telah dibuat ataupun class dari third parties (pihak ketiga). Sama seperti global variable object ini bisa digunakan untuk operasi yang bersifat umum untuk keseluruhan program. Diberikan contoh beberapa object dari class yang dipakai :

DHT dhtSensor(DHTPIN, DHTTYPE), dhtSensor adalah object (bentuk nyata) dari class DHT. Sehingga dhtSensor selanjutnya dapat menggunakan semua properties dan method/function yang telah didefinisikan secara global pada class DHT. Misal untuk mendapatkan pengukuran humidity, bisa menggunakan perintah dhtSensor.readHumidity(). Demikian juga untuk mendapatkan temperature dengan memanggil nama-object.nama-method.

Hal ini juga berlaku untuk object dari class yang dibuat sendiri, missal class Logsheet untuk membuat object logsheet (dengan perintah Logsheet logsheet("logsheet")). Selanjutnya untuk memanggil salah satu method/function global dari class Logsheet dengan menggunakan perintah logsheet.getValues(). Demikian seterusnya untuk object-object yang lainya.

* Setup

Dalam function setup ini, ada beberapa hal yang dikerjakan sbb:

* Init OLED 48x64 pixel dengan I2C
* Display logo GMF
* Display Welcome dan program step
* Init pin mode
* Init serial communication dengan baud rate 11520 bit/sec
* Init local storage (LittleFS) untuk server files, configuration dan logsheet
* List files dalam LittleFS
* Load User data (engineer dan operator)
* Setup default user, active user = guest
* Init sensor DHT11 untuk temperature dan humidity dengan one wire communication
* Setup logsheet meliputi :
  + Attach parameter temperature dan humidity
  + Attach sensor DHT11
  + Attach display OLED
  + Set time
* Setup urlController meliputi :
  + Static url (meliputi image files, CSS, JS)
  + HTTP url (meliputi HTML files)
  + AJAX url (data files)
* Start WIFI (MODE\_AP atau Multifiwi)

Mode yang dipakai dalam datalogger ini menggunakan MultiFiwi. Dalam mode ini datalogger bisa melakukan koneksi signal terkut yang ada pada daftar WIFI yang telah dimasukkan dalam setup datalogger.

Mode MultiWifi ini dengan membuat constanta yang ada pada file datalogger.h berikut ini.

bool AP\_MODE = false; // Do yo want the ESP as AP?

Daftar WIFI ini meliputi :

// Replace with your network credentials

* + - * const char \*ssid1 = "Sam-Laptop";
      * const char \*password1 = "sampon170466";
      * const char \*ssid2 = "SamMobile";
      * const char \*password2 = "sampon170466";
      * const char \*ssid3 = "SamFamily";
      * const char \*password3 = "muwa7x06";
      * const char \*ssid4 = "Fariyo Ganteng";
      * const char \*password4 = "Faryganteng";

Data tersebut di atas terdapat pada file datalogger.h.

* Start server (server.begin())
* Loop

Function loop ini yang dijalankan (execution) secara terus-menerus selama ESP8266 ini jalan. Adapun tugas/function yang dijalankan adalah :

* Execute logsheet serta memberikan nilai sampling time
* Execute mainSequence
* Memberikan seting time setiap 1 menit ke logsheet

Berikut cuplikan code pada datalogger.ino terkait penjelasan fungsi loop () di atas :

void loop(){

  //Logsheet action

  logsheet.execute(samplingTime);

  mainSequence.execute();

  if (mainSequence.isAMinuteEvent())

    logsheet.setTime(getTimeNtp());

}

Adapun detail dari masing-masing fungsi di atas dijelaskan dalam sub bab terpisah.

* logsheet.h, logsheet.cpp

Sebagaimana dijelaskan sepintas dalam Software structure point i. di atas, logSheet dengan entry point function execute, mempunyai tugas yang dijalankan sesuai dengan periode yang berikan pada nilai samplingTime. Nilai ini di-setting pada dataLogger.ino (function loop). Logsheet.h mendefinisikan class Logsheet yang selanjutnya dalam program utama (dataLogger.ino) digunakan untuk membuat object dengan nama logsheet(“logsheet”). Selanjutnya object ini (logsheet) digunakan untuk operasi selanjutnya. Dalam class Logsheet ada beberapa fungsi/method utama yang mendukungnya.

* Function execute()

Fungsi ini sebagai titik masuk (entry point) pada class Logsheet (class Logsheet yang selanjutnya digunakan untuk mendefinisakan object logsheet) dengan beberapa tugas sebagai berikut :

* + Melakukan sinkronisasi waktu antara dataLogger.ino (program utama).
  + Melakukan perhitungan jumlah sampling data per menit, dan membatasi 60 sampling per menit.
  + Melakukan beberapa kegiatan saat sampling time telah sesuai (tiba):
    - Mengambil data dari sensor (\_getsensorValue())

Fungsi ini akan mengambil data dari DHT11 berupa data temperature dan humidity serta melakukan filtering pada data yang diambil berdasarkan parameter alfa ema (default 80%). Pengambilan data DHT11 ini dilakukan ketika tidak pada mode SIMULATION. Saat mode SIMULATION data diambil dari random data.

Data temperature dan humidity selanjutnya disimpan dalam data parameter glogal yang di-attach ke logsheet object. Parameter ini baik untuk temperature dan humidity.

Selanjuntya, data filtering ini juga dibandingkan/dikomparasi dengan data konfigurasi untuk masing-masing konfigurasi sensor, baik untuk alarm high, low serta memberikan status untuk alarm yang dimaksud.

* + - Menampilkan data pengukuran pada OLED 64x48 pixel, lengkap dengan unit untuk Temperature dan Humidity.
    - Melakukan perhitungan event/kejadian mulai dari samplingTrending, samplingSecond, samplingMinute, samplingHour
    - Melakukan kegiatan logsheet sesuai dengan event yang terjadi seperti : perhitungan averaging, penyimpanan data ke dalam local storage (LittleFS) sesuai dengan file ynag dimaksud. Penyimpanan file dengan type text dengan format CSV (Coma Seprated Value).

Berikut cuplikan fungsi execute() pada class Logsheet :

void Logsheet::execute(unsigned long samplingTime){

  //set samplingTime

  \_samplingTime = samplingTime;

  //synchronize sampling time

  if (!\_synchronized)  {

    \_samplingSec = 0;

    \_synchronized = true;

  }

  //calculate sampling per minute

  \_nbrSamplingSec = 60000 / \_samplingTime;

  if (\_nbrSamplingSec < 1) \_nbrSamplingSec = 1;

  else if (\_nbrSamplingSec > SECOND\_60) \_nbrSamplingSec = SECOND\_60;

  unsigned long currMilli = millis();

  if ((currMilli - \_prevMilli) >= samplingTime){

    \_prevMilli = currMilli;

    //get sensor data and update parameters

    this->\_getSensorValue();

    //display to Oled

    this->\_oledDisplay(\_paramTemperature->getParam(PARAMETER\_VALUE),

                       \_paramHumidity->getParam(PARAMETER\_VALUE));

    //record event

    this->\_recordEvent();

    //record logsheet

    this->\_recordLogsheet();

  }

}

* Function pendukung WEB
  + Mengambil data rata-rata jam (hourlyAverage) selanjutnya dikirim ke WEB untuk ditampilkan sebagai data harian dalam setiap minggunya (Sunday – Saturday).
  + Mengambil data trending per sampling, data ini selanjutnya dikirim ke WEB untuk graphic trending temperature dan humidity (24 sampling data).
  + Mengambil data konfigurasi parameter sensor (range high/low, alarm high/low, unit) selanjutnya dikirim ke WEB untuk mebuat graphic dial baik temperature dan humidity.
  + Mengambil data pengukuran per-sampling untuk menampilkan nilai dalam graphic pada WEB.

Data yang diambil tersebut di atas disimpan dalam local storage (LittleFS) kecuali data pengukuran per-sampling.

Fungsi tersebut didefinisakn pada file Logsheet.h sebagai berikut :

public:

  Logsheet(String);

  String getHourlyAvg(int);

  String getTrendingData();

  String getCfgParameter();

  String getValues();//return string in format Json for Temp, Humidity

Selanjutnya, fungsi tersebut di atas menjadi salah satu fungsi pendukung API dalam pengiriman data ke client (browser). Berikut cuplikan dari fungsi/function di atas yang terdapat pada class Logsheet (pada file Logsheet.cpp) :

String Logsheet::getTrendingData(){

  /\*

    {

    "Time": [0, 1, 2, 3, 4, 5, 6, 7, 8, 9,

                 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23],

    "Temperature": [20.00, 20.01, 20.02, 20.03, 20.04, 20.05, 20.06, 20.07, 20.08, 20.09,

                 20.10, 20.11, 20.12, 20.13, 20.14, 20.15, 20.16, 20.17, 20.18, 20.19, 20.20, 20.21, 20.22, 20.23],

    "Humidity": [70.00, 70.01, 70.02, 70.03, 70.04, 70.05, 70.06, 70.07, 70.08, 70.09,

                 70.10, 70.11, 70.12, 70.13, 70.14, 70.15, 70.16, 70.17, 70.18, 70.19, 70.20, 70.21, 70.22, 70.23]

    }

    DynamicJsonDocument doc(1536);

  \*/

  //Json assembling

  String output;

  DynamicJsonDocument doc(1536);

  float h, t;

  doc.clear();

  JsonArray time = doc.createNestedArray("time");

  JsonArray temperature = doc.createNestedArray("temperature");

  JsonArray humidity = doc.createNestedArray("humidity");

  for (int row = 0; row < TRENDING\_24; row++)  {

    String Ti = String(\_logsheetTrending[row].time);

    String T = String(\_logsheetTrending[row].temperature);

    String H = String(\_logsheetTrending[row].humidity);

    time.add(Ti.toInt());

    temperature.add(T.toFloat());

    humidity.add(H.toFloat());

  }

  serializeJson(doc, output);

  return output;

}

Fungsi getTrendingData() adalah fungsi untuk memberikan data sampling (24 sampling data – temperature, humidity) digunakan pada laman utama/induk (index.html). Selanjutnya digunakan untuk menggambar trending pada graphic pada laman utama tadi.

* Function private untuk keperluan internal loghseet, fungsi ini hanya bisa digunakan dalam lingkup class logsheet sendiri untuk mendukung kegiatan yang diperlukan meliputi :
  + Setup default parameter – untuk setup default saat pengembangan software
  + Setup parameter sensor dari file berupa data parameter (range high/low, alarm high/low, unit).
  + Init random JSON – untuk setup JSON saat pengembangan software
  + Fungsi penyimpanan dan pembacaan file ke dan dari local storage (LittleFS)

Berikut daftar private function yang dipakai pada class Logsheet (didefinisikan pada file Logsheet.h) :

private:

  void \_setupDefaultParameter();

  void \_setupFileCfgParameter(String);

  void \_getSensorValue();

  void \_print(logsheetData);

  String \_initRandomJson();

  void \_oledDisplay(float, float);

  String \_getTimeStr(int);

  String \_getDayOfWeek(int);

  void \_recordEvent();

  void \_recordLogsheet();

  void \_minuteLogsheet();

  void \_hourlyLogsheet();

  void \_dailyLogsheet();

  String \_getCsv(logsheetData data);

  void \_shiftArray(int size, logsheetData last);

  logsheetData \_calculateAverage(logsheetData data[], int size);

  void \_appendFile(const char \*path, const char \*message);

  void \_writeFile(const char \*path, const char \*message);

  String \_readFile(const char \*path);

  String \_readFileJson(int); //read file day of week, return json

* model.h, model.cpp

Secara umum dijelaskan pada Software structure point iii di atas meliputi :

* + Sensor parameters (High, Low, Unit, Alarm, alfa ema)
  + Loghsheet (temperature and humidity)
  + User (username, password, email, level)

Data model ini selanjutnya digunakan untuk penyimpanan data secara global untuk operasional program. Data model yang dimaksud dibentuk dalam class ataupun data struktur biasa. Data model yang dibentuk dalam class juga dilengkapi dengan API (Aplication Program Interfacing) untuk mempermudah penggunaanya.

Class AccesParameter adalah class yang didedikasikan untuk akses parameter sensor dengan API sbb:

* Init
* getJSON – mengambil parameter dalam format JSON
* getOperation – mengambil data operasi untuk temperature dan humidity
* getParameter – mengambil data dalam bentuk struktur
* setParameter – seting parameter dalam bentuk struktur

Berikut cuplikan dari class AccessParam (akses parameter sensor ) :

  class AccessParam{

  public:

      AccessParam(String);

      void init(String, param);

      String getJson();

      JsonObject getOperation();

      void setOperationJson(JsonObject);

      param getParam();

      float getParam(int);

      void setParamJson(JsonObject);

      void setParam(param);

      void setParam(int, float);

      boolean isChangeAble(int);

      void setAlarm(int);

      String toString();

      String getId();

      void info();

  private:

      String    \_id;

      param \_dataParam;

  };//end of class

String AccessParam::getJson(){

  /\*

  {

    "unit":"%",

    "value":51.5,

    "highRange":100.0,

    "lowRange":0.0,

    "highLimit":80.0,

    "lowLimit":40.0,

    "alfaEma":80.0,

    "alarm":2

  }

  StaticJsonDocument<192> doc;

  doc["unit"] = "%";

  doc["value"] = 51.5;

  doc["highRange"] = 100;

  doc["lowRange"] = 0;

  doc["highLimit"] = 80;

  doc["lowLimit"] = 40;

  doc["alfaEma"] = 80;

  doc["alarm"] = 2;

  serializeJson(doc, output);

  \*/

  // Get a reference to the root object

  StaticJsonDocument<192> doc;

  String output;

  doc["unit"] = \_dataParam.unit;

  doc["value"] = \_dataParam.value;

  doc["highRange"] = \_dataParam.highRange;

  doc["lowRange"] = \_dataParam.lowRange;

  doc["highLimit"] = \_dataParam.highLimit;

  doc["lowLimit"] = \_dataParam.lowLimit;

  doc["alfaEma"] = \_dataParam.alfaEma;

  doc["alarm"] = \_dataParam.alarm;

  serializeJson(doc, output);

  return output;

}

* sequenceTimer.h, sequenceTimer.cpp

Sebagaimana dala Software structure point iv, class ini mengatur waktu utamanya setiap menit. Jika durasi satu menit telah tercapai, akan memberikan peringatan berupa event/kejadian sehingga dapat digunakan untuk proses eksekusi kegiatan yang ada pada object logsheet (dari class Logsheet) seperti dijelaskan dalam paparan logsheet di atas.

* start\_up.h

Secara garis besar dijelaskan dalam sub bab Software structure point v, StartUp merupakan Class (yang selanjutnya menadi object startUp) yang bertugas untuk :

* Menampilkan logo Garuda, seperti pada gambar berikut.

Gambar ini merupakan konversi dari file PNG menjadi bitmap dengan ukuran pixel 64x48 sesuai dengan ukuran OLED yang dipakai. Selanjutnya hasil konversi dijadikan konstanta dengan nama logo\_gmf. Untuk menampilkan logo ini menggunakan fungsi logoDisplay() yang ada pada class tersebut.

Berikut data logo yang sudah dikonversi ke bitmap dan disimpan dalam bentuk konstanta dalam start\_up.h :

//logo gmf

// 'gmf', 64x48px

const unsigned char logo\_gmf[] PROGMEM = {

    0x00, 0x7f, 0xc0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x40, 0x00, 0x7f, 0xc0, 0x00, 0x00, 0x00, 0x00,

    0x00, 0x00, 0x03, 0xf8, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xfe, 0x00, 0x00, 0x00, 0x00,

    0x00, 0x00, 0x00, 0x3f, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0f, 0xc0, 0x00, 0x00, 0x00,

    0x00, 0x00, 0x00, 0x07, 0xe0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0xf0, 0x00, 0x00, 0x00,

    0x1f, 0xff, 0xfe, 0x01, 0xf8, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0f, 0xe0, 0xfc, 0x00, 0x00, 0x00,

    0x00, 0x00, 0x01, 0xf8, 0x7e, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x7e, 0x3f, 0x00, 0x00, 0x00,

    0x00, 0x00, 0x00, 0x1f, 0x9f, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0f, 0xcf, 0xc0, 0x00, 0x00,

    0x00, 0x00, 0x00, 0x07, 0xe7, 0xe0, 0x00, 0x00, 0x03, 0xff, 0xe0, 0x03, 0xf3, 0xf2, 0x00, 0x00,

    0x00, 0x00, 0x7f, 0x81, 0xf9, 0xfb, 0xff, 0x80, 0x00, 0x00, 0x03, 0xe0, 0xfc, 0xf9, 0xff, 0xc0,

    0x00, 0x00, 0x00, 0xfc, 0x7e, 0xfc, 0xfe, 0x00, 0x00, 0x00, 0x00, 0x3e, 0x3f, 0x7e, 0x7e, 0xf8,

    0x00, 0x00, 0x00, 0x1f, 0x9f, 0xbf, 0x3f, 0xfc, 0x00, 0x00, 0x00, 0x0f, 0xcf, 0xdf, 0x9f, 0xfe,

    0x00, 0x00, 0x00, 0x07, 0xe7, 0xcf, 0xcf, 0xfe, 0x00, 0x7f, 0xfe, 0x03, 0xf7, 0xe7, 0xe7, 0x82,

    0x00, 0x00, 0x0f, 0xc1, 0xfb, 0xf3, 0xf7, 0x00, 0x00, 0x00, 0x03, 0xf0, 0xfd, 0xf9, 0xf8, 0x00,

    0x00, 0x00, 0x00, 0xfc, 0x7e, 0xfc, 0xfc, 0x00, 0x00, 0x00, 0x00, 0x3f, 0x3e, 0x7e, 0x78, 0x00,

    0x00, 0x00, 0x00, 0x1f, 0x9f, 0x3f, 0x30, 0x00, 0x00, 0x00, 0x00, 0x0f, 0xdf, 0x9f, 0x80, 0x00,

    0x00, 0x0f, 0xf0, 0x07, 0xef, 0xcf, 0xc0, 0x00, 0x00, 0x00, 0x1f, 0x83, 0xf7, 0xe7, 0xc0, 0x00,

    0x00, 0x00, 0x03, 0xe1, 0xfb, 0xf3, 0x80, 0x00, 0x00, 0x00, 0x00, 0xf8, 0xf9, 0xf8, 0x00, 0x00,

    0x00, 0x00, 0x00, 0x7c, 0xfc, 0xfc, 0x00, 0x00, 0x00, 0x00, 0x00, 0x3f, 0x7e, 0x7c, 0x00, 0x00,

    0x00, 0x00, 0x00, 0x1f, 0xbf, 0x3c, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0f, 0xdf, 0x90, 0x00, 0x00,

    0x00, 0x00, 0x00, 0x07, 0xcf, 0xc0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0xe7, 0xe0, 0x00, 0x00,

    0x00, 0x00, 0x00, 0x01, 0xf3, 0xc0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0xf9, 0x80, 0x00, 0x00,

    0x00, 0x00, 0x00, 0x00, 0xfc, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x7e, 0x00, 0x00, 0x00,

    0x00, 0x00, 0x00, 0x00, 0x3e, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1c, 0x00, 0x00, 0x00,

    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};

* Menampilkan welcome, seperti pada urutan gambar berikut.

Pesan welcome ditampilkan secara berurutan oleh fungsi welcomeDisplay(). Pesan welcome mulai dari GMF AeroAsia, lalu IoT DataLogger lengkap dengan versinya dan diakhiri dengan pesan Salman Alfarisi serta bulan pembuatan software ini.



Berikut cuplikan coding terkait dengan menampilkan pesan welcome di atas :

void StartUp::welcomeDisplay(){

    int x, y, totalH;

    //for oled 64x48

    int oledWidth = 64;

    int oledHeight = 48;

    int maxLine = 6; //for texSize 1;

    int maxCol = 10; //for texSize 1

    int hight = oledHeight / maxLine;

    String msg;

    //welcomeMsg [year+1] = {"GMF","AeroAsia","IoT", "DataLogger", "Ver: ", "Salman", "Alfarisi", "Oct,2021"} ;

    for (int msgNbr = 0; msgNbr <= year; msgNbr++)    {

        //y = line \* hight;

        //Show GMF AeroAsia

        //clear display

        if (msgNbr == gmf)        {

            \_display->clearDisplay();

            //Characters are rendered in the ratio of 7:10.

            //Meaning, passing font size 1 will render the text at 7×10 pixels per character,

            //passing 2 will render the text at 14×20 pixels per character and so on.

            \_display->setTextSize(1);

            \_display->setTextColor(WHITE);

            totalH = 2 \* 8;

            y = (oledHeight - totalH) / 2 + 4 / 2 - 4;

            x = (maxCol - 3) \* 6 / 2;

            \_display->setCursor(x, y); //x,y on pixel

            msg = welcomeMsg[msgNbr];

            \_display->print(msg);

        }

        //show to oled display

        else if (msgNbr == AeroAsia)        {

            // Show the display buffer on the hardware.

            // NOTE: You \_must\_ call display after making any drawing commands

            // to make them visible on the display hardware!

            //AeroAsia y = 14 + 8 + 4 = 26

            y += 14;

            x = (maxCol - 9) \* 6 / 2;

            \_display->setCursor(x, y); //x,y on pixel

            msg = welcomeMsg[msgNbr];

            \_display->print(msg);

            \_display->display();

            delay(2000);

        }

        //Show IoT, DataLogger, Version

        //clear display

        else if (msgNbr == IoT){

            \_display->clearDisplay();

            \_display->setTextSize(1);

            \_display->setTextColor(WHITE);

            totalH = 3 \* 8;

            y = (oledHeight - totalH) / 2 - 4;

            x = (maxCol - 3) \* 6 / 2;

            \_display->setCursor(x, y); //x,y on pixel

            msg = welcomeMsg[msgNbr];

            \_display->print(msg);

        }

        else if (msgNbr == dataLogger)        {

            y += 14;

            x = (maxCol - 10) \* 6 / 2;

            \_display->setCursor(x, y); //x,y on pixel

            msg = welcomeMsg[msgNbr];

            \_display->print(msg);

        }

        else if (msgNbr == version)        {

            y += 14;

            x = (maxCol - 10) \* 6 / 2;

            \_display->setCursor(x, y); //x,y on pixel

            msg = welcomeMsg[msgNbr];

            \_display->print(msg);

            \_display->print(VERSION);

            \_display->display();

            delay(2000);

        }

        //Show Salman, Alfarisi, year

        //clear display

        else if (msgNbr == salman)        {

            \_display->clearDisplay();

            \_display->setTextSize(1);

            \_display->setTextColor(WHITE);

            totalH = 3 \* 8;

            y = (oledHeight - totalH) / 2 - 4;

            x = (maxCol - 6) \* 6 / 2;

            \_display->setCursor(x, y); //x,y on pixel

            msg = welcomeMsg[msgNbr];

            \_display->print(msg);

        }

        else if (msgNbr == alfarisi)        {

            y += 14;

            x = (maxCol - 7) \* 6 / 2;

            \_display->setCursor(x, y); //x,y on pixel

            msg = welcomeMsg[msgNbr];

            \_display->print(msg);

        }

        //Oct,2021

        else if (msgNbr == year)        {

            y += 14;

            x = (maxCol - 7) \* 6 / 2;

            \_display->setCursor(x, y); //x,y on pixel

            msg = welcomeMsg[msgNbr];

            \_display->print(msg);

            \_display->display();

            delay(2000);

        }

    }

}

* Menampilkan Step start up.

Untuk mengetahui urutan sebagaimana dijelaskan pada fungsi setup di atas (lihat penjelasan datalogger.h dan datalogger.ino pada sub bab setup) diberikan display pada OLED sehingga tiap Langkah/step start up bisa diketahui. Step mulai dari 00 yaitu pada saat init OLED 64x48 pixel berikut I2C. seperti pada gambar di samping dan dilanjutkan pada urutan yang ada pada fungsi setup.

Pada STEP 05, IoT DataLogger berupaya untuk connect dengan WIFI yang telah dikonfigurasi pada datalogger.h. Jika koneksi berhasil, maka step dilanjutkan sampai dengan step 09.



Step 09 adalah dimulainya aktivasi server sehingga memungkinkan IoT DataLogger bisa diakses via internet/WEB.

Selanjutnya IoT DataLogger bertugas untuk melakukan pengambilan data dari sensor DHT11, menampilkan nilai pada OLED 64x48 pixel serta tugas yang lain seperti dijelaskan dalam logsheet.h dan logsheet.cpp di atas.

Berikut cuplikan dari fungsi untuk menampilkan display value pada OLED :

void Logsheet::\_oledDisplay(float t, float h){

  int tInt = t \* 100;

  t = tInt / 100.0; //get 2 digits

  int hInt = h \* 100;

  h = hInt / 100.0; //get 2 digits

  \_display->clearDisplay();

  \_display->setTextSize(1);

  \_display->setTextColor(WHITE);

  \_display->setCursor(0, 0);

  \_display->print("Humidity ");

  \_display->setCursor(1, 11);

  \_display->print(h);

  \_display->print(" %");

  \_display->setCursor(1, 29);

  \_display->print("Temperatur");

  \_display->setCursor(1, 40);

  \_display->print(t);

  \_display->print(" \*C ");

  \_display->display();

}

Flowchart – datalogger

Start

Import required libraries

Declare Variables & Objects

Protype functions

Display.begin() – init OLED I2C for 64x48 pixel

startUp.AttachDisplay() – attach OLED to startUp

startUp.logoDisplay – display logo to OLED

startUp.welcomeDisplay – display welcome messages

setup pin for LED

Serial.begin(11520) – baud rate 11520 bit/sec

LittleFS.begin() – init little FS (local storage ESP8266)

listAllFilesInDir() – list file in LittleFS

loadUser() – for engineer and operator

setupDefaultUser() – active user is guest

dhtSensor.begin() – init DHT11 sensor

logsheet.AttachParameter() – temperature & humidity

logsheet.AttachSensor() – sensor DHT11

logsheet.AttachDisplay() – OLED

logsheet.AttachLed() – LED

logsheet.setTime() -synchronize

N

Y

StartWIFI\_AP(); - WIFI AP init

startMultiWifi(); - Multi WIFI init

WIFI\_AP

urlController() – WEB handling (req – resp)

server.begin() – start server

1

1

isMinute

logSheet.setTime() – set time from NTP

Y

N

logSheet.execute() – per sampling time

mainSequence.execute()

End

Selain dari itu, IoT DataLogger juga siap sebai server untuk menangani permintaan via internet/WEB. Penjelasan terkait dengan ini dipaparkan dalam sub ba tersendiri. Fungsi yang menangani penampilan step ini adalah stepDisplay() dengan memberikan parameter step yang sedang berjalan padanya.

Pelayanan WEB / WEB Services

WEB services adalah bentuk pelayanan/tugas IoT DataLogger terkait dengan tanggapan/respon permintaan dari user (via devices lain : Laptop, Deskto, HP atau gadgets yang lainya). Sebagaimana dijelaskan secara umum pada sub bab Software Structure point ii (WEB Services), Pelayanan WEB ini meliputi :

* Authentication & Authorization user/pengguna
* Menampilkan laman WEB / WEB Page yang responsive

Pelayanan WEB ini terbagi menjadi 2 tempat yaitu (Server Side dan Client Side), server side adalah pelayanan yang dikerjakan oleh IoT DataLogger itu sendiri untuk mendukung permintaan (request dari pengguna/user via internet). Sedangkan client side adalah pelayanan IoT DataLogger yang dikerjakan bersamaan dengan browser yang digunakan (chrome, firefox etc…). Secara fungsi pelayanan ini dibagi menjadi Authentication & Authorization (Auth) dan menampilkan laman WEB/WEB page yang responsive.

Authentication & Authorization (Auth) adalah mekanisme untuk memberikan batasan akses kepada pengguna IoT Data Logger, hal ini diperlukan untuk keamanan program itu sendiri. Tidak setiap user yang akses bisa melakukan pemilihan menu, apalagi menu yang bersifat mengubah data atau reporting ke pejabat yang berwenang. Untuk itu IoT Data Logger juga mempunyai mekanisme Auth yang dimaksud yaitu membatasi pengguna dengan 3 tingkatan :

* Pengguna umum/tamu/Guest : GST – Pengguna ini adalah setiap pengguna yang bisa akses ke laman IoT Data Logger, dengan hanya bisa menampilkan laman induk dan login
* Pengguna Operator : OPR – Pengguna ini bisa melakukan reporting dan login, logout
* Pengguna Engineer : ENG – Pengguna ini bisa melakukan semua menu yang tesedia meliputi reporting, configuration sensor, login, logout

Mekanisme Auth ini berlaku untuk semua laman (index, reporting, login, logout, config).

Pelayanan WEB ini dimulai dengan setup API (Application Program Interface) yang dikerjakan pada fungsi setup (lihat penjelasan dataLogger.h dan dataLogger.cpp pada sub bab setup) saat menjalankan fungsi urlController(). API akan mengatur tatacara melakukan request via WEB, bagaimana IoT melakukan respon terhadapnya serta data apa yang harus dikirimkan terkait request yang dimaksud. urlController membagi respon ini menjadi 2 bagian yaitu :

* Respon terkait dengan permintaan file static (Images, JS, CSS)
* Respon terkait dengan permintaan dynamic (HTML dan data)

Permintaan semua files itu baik static ataupun dynamic tersimpan dalam local storage (LittleFS) lihat penjelasan sub bab Management File di atas.

Masing-masing tugas itu dikelompokkan dalam fungsi yang terpisah yaitu :

* loadStaticFiles()
* urlController()

loadStaticFiles() akan dipanggil oleh urlController() saat dijalankan/dieksekusi pada fungsi setup() pada dataLogger.ino (lihat penjelasan dataLogger.ino di atas serta flowchart terkait).

loadStaticFiles() mempunyai API seperti dalam daftar berikut :

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **API** | **HTTP Method** | **Description** |
| 1 | /index.js | HTTP\_GET | File javascript digunakan oleh index.html |
| 2 | /authentication.js | HTTP\_GET | File javascript digunakan oleh semua html files |
| 3 | /widgetsClass.js | HTTP\_GET | File javascript digunakan oleh index.html untuk graphic |
| 4 | /login.js | HTTP\_GET | File javascript digunakan oleh login.html |
| 5 | /modal-dialog.js | HTTP\_GET | File javascript digunakan oleh semua file html untuk menampilkan dialog windows untuk merespon permintaan About |
| 6 | /login.js | HTTP\_GET | File javascript digunakan oleh login.html |
| 7 | /report.js | HTTP\_GET | File javascript digunakan oleh report.html |
| 8 | /config.js | HTTP\_GET | File javascript digunakan oleh config.html |
| 9 | /logoGMF | HTTP\_GET | File image untuk menampilkan logo GMF pada index.html dan report.html |
| 10 | /report.js | HTTP\_GET | File javascript digunakan oleh report.html |
| 11 | /style.css | HTTP\_GET | File CSS digunakan untuk semua file html |

urlController() secara aktif merespon permintaan user yang diberikan lewat browser melalui laman yang tertera dalam alamat IP. Laman akan ditampilkan sebagaimana permintaan user melalui menu navigasi yang ada di atas setiap laman (top navigation). Selain itu saat pertama kali user masuk dalam alamat IP, maka dataLogger menampilkan home page (laman awal) dan memberikan file yang tersimpan dalam local storage (LittleFS) ke browser untuk ditampilkan dalam display devices user.

Berikut cuplikan dari bagian fungsi loadStaticFiles() yang diambil dari file dataLogger.ino :

void loadStaticFile(){

  // Route to load style.css file

  server.on("/style.css", HTTP\_GET, [](AsyncWebServerRequest \*request)

            { request->send(LittleFS, "/style.css", "text/css"); });

  // Route to load authentication.js file

  server.on("/authentication.js", HTTP\_GET, [](AsyncWebServerRequest \*request)

            { request->send(LittleFS, "/authentication.js", "text/js"); });

  // Route to load report.js file

  server.on("/report.js", HTTP\_GET, [](AsyncWebServerRequest \*request)

            { request->send(LittleFS, "/report.js", "text/js"); });

  // Route to load index.js file

  server.on("/index.js", HTTP\_GET, [](AsyncWebServerRequest \*request)

            { request->send(LittleFS, "/index.js", "text/js"); });

  // Route to load login.js file

  server.on("/login.js", HTTP\_GET, [](AsyncWebServerRequest \*request)

            { request->send(LittleFS, "/login.js", "text/js"); });

  // Route to load config.js file

  server.on("/config.js", HTTP\_GET, [](AsyncWebServerRequest \*request)

            { request->send(LittleFS, "/config.js", "text/js"); });

  // Route to load widgetsClass.js file

  server.on("/widgetsClass.js", HTTP\_GET, [](AsyncWebServerRequest \*request)

            { request->send(LittleFS, "/widgetsClass.js", "text/js"); });

  // Route to load modal-dialog.js file

  server.on("/modal-dialog.js", HTTP\_GET, [](AsyncWebServerRequest \*request)

            { request->send(LittleFS, "/modal-dialog.js", "text/js"); });

  // Route to load logoGMF file

  server.on("/logoGMF", HTTP\_GET, [](AsyncWebServerRequest \*request)

            { request->send(LittleFS, "/logoGMF.png", "image/png"); });

}

API yang ada dalam urlController sebagai berikut :

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **API** | **HTTP Method** | **Description** |
| 1 | / | HTTP\_GET | Menampilkan home page (index.html) |
| 2 | /report | HTTP\_GET | Menampilkan report page (report.html) |
| 3 | /login | HTTP\_GET | Menampilkan login page (login.html) |
| 4 | /login | HTTP\_POST | Kirim data username & password untuk user |
| 5 | /loginStatus | HTTP\_GET | AJAX data untuk mengirimkan status login user |
| 6 | /logout | HTTP\_GET | Menampilkan dialog logout selanjutnya home page |
| 7 | /config | HTTP\_GET | Menampilkan config page (config.html) |
| 8 | /config | HTTP\_POST | Update data konfigurasi sensor |
| 9 | /getTrendingData | HTTP\_GET | AJAX data trending pada graphic home page |
| 10 | /getSensorConfig | HTTP\_GET | AJAX data konfigurasi sensor pada graphic home page dan config page |
| 11 | /getActiveUser | HTTP\_GET | AJAX data aktive user digunakan pada semjua home page untuk authentication dan authorization |
| 12 | /haourlyAvgDay | HTTP\_GET | AJAX data untuk report page |
| 13 | /getSensor | HTTP\_GET | AJAX data untuk menampilkan data pengukuran sensor pada home page |
| 14 | /getSamplingTime | HTTP\_GET | AJAX data untuk sampling time |

Berikut cuplikan API yang dipakai pada fungsi urlController() pada file dataLogger.ino :

  // Route for root / web page

  server.on("/", HTTP\_GET, [](AsyncWebServerRequest \*request)

            { request->send(LittleFS, "/index.html", "text/html"); });

  // route to trending data

  server.on("/getTrendingData", HTTP\_GET, [](AsyncWebServerRequest \*request){

              String trendData = logsheet.getTrendingData();

              request->send(200, "application/json", trendData);});

  // route to active user

  server.on("/getActiveUser", HTTP\_GET, [](AsyncWebServerRequest \*request){

              String activeUsr = activeUser.getJson();

              request->send(200, "application/json", activeUsr);

              Serial.println(activeUsr);});

  // route to config sensor - temperature and humidity

  server.on("/getSensorCfg", HTTP\_GET, [](AsyncWebServerRequest \*request){

              String sensorCfg = logsheet.getCfgParameter();

              request->send(200, "application/json", sensorCfg);

              Serial.println(sensorCfg); });

urlController() dan loadStaticFiles() dikerjakan pada sisi server (Server Side).

Perancangan WEB Page / Laman WEB.

Perancangan WEB page memegang perana yang penting karena melalui WEB page ini pengguna berhubungan dengan IoT DataLogger ini, sehinnga beberapa kriteria WEB page harus diperhatikan :

* Kemudahan akses
* Ramah pengguna (user friendly)
* Sederhana
* Keindahan
* Kecepatan respon

Untuk keperluan di atas, setiap laman yang dibuat melibatkan 3 pilar utama bahasa program WEB yaitu (HTML, CSS, JS). HTML (Hyper Text Markup Language) digunakan untuk keperluan layout laman, mengatur setiap element yang akan ditampilkan. Agar laman tampak indah stylist diperlukan file CSS (Cascading Style Sheet) sehinga tampilan text bisa mempunyai model font serta ukuran yang menarik, berikut warna background dan warna font.

File CSS akan menjadi bagian dari HTML sekalipun ditulis dalam file yang berbeda/terpisah (dan dimungkinkan bisa dalam satu file). Pemisahan file ini untuk kemudahan melakukan perancangan, pengembangan serta pelacakan kesalahan (debuging). Pemisahan ini juga terkait dengan OOP principles dimana file dipisahkan berdasarkan BoC (Business of Concern) – berdasarkan tugasnya masing-masing.

File JS (javascript) menentukan perilaku (behavior) dari laman yang ditampilkan. Dengan JS memungkinkan laman menjadi lebih dinamis dalam merespon permintaan user IoT DataLogger. Hal ini dimungkinkan karena JS dapat melakukan hal-hal berikut ini :

* Memberikan respon tanpa terhubung dengan server lebih dulu
* Melakukan validasi dan memberikan feedback kepada user sebelum dikirim ke server
* Meminta data sebagian dari laman keseluruhan (AJAX)
* Memperkuat fungsi CSS sehingga bisa melakukan perubahan style
* Bisa melakukan animasi sehingga laman bisa sangat impresive

Sebagaimana file CSS, file JS merupakan bagian dari HTML dan bisa ditulis menjadi satu atau terpisah dalam file yang berbeda. Bahkan file CSS dan JS bisa merujuk pada link pada laman WEB.

Home Page – laman utama/induk

Home page akan tampil pertama kali ketika user menulis pada WEB browser alamat yang tertera dalam IP. Misalkan dalam simulasi yang dilakukan kita mengetikkan <http://192.168.137.235/> atau sekedar <http://192.168.137.235> atau bahkan 192.168.137.235 saja. Web browser akan mengirimkan pesan ini melalui jaringan internet via WIFI pada alamat terdekat (utamanya yang terhubung dalam satu jaringan LAN – Local Area Network). Jika ditemukan, maka server dalam hal ini ESP-8266 yang berfungsi sebagai server akan memberikan respon terkait permintaan itu. Respon yang diberikan sesuai dengan daftar API tersebut di atas (yang tertera dalam urlController lihat penjelasan WEB Services di atas).

Dalam API, permintaan itu artinya server (ESP-8266 – IoT Server) merespon dengan memberikan file index.html yang berada dalam local storage (LittleFS pada root folder) ke browser untuk diproses lebih lanjut oleh browser peminta.

Secara umum file html mempunyai struktur sbb :

<!DOCTYPE html>

<html>

    <head>

        <title>Page Title</title>

    </head>

    <body>

        <h1>This is a Heading</h1>

        <p>This is a paragraph.</p>

    </body>

</html>

Type dokumen, dalam hal ini adalah html. Ditandai dengan tag <!DOCTYPE html>

Seluruh dokument html berada dalam cakupan tag pembuka <html> dan tag penutup </html>. Selanjutnya tag <html> terbagi menjadi dua bagian yang ditandai dengan tag <head> …… </head> dan <body> ……. </body>. Untuk file index.html tampak seperti susunan berikut ini :

<!DOCTYPE html>

<html>

  <head>

    <meta content='text/html;charset=utf-8' http-equiv='Content-Type'>

    <meta name="viewport" content="width=device-width, initial-scale=1">

    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">

    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>

    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>

    <link rel="stylesheet" type="text/css" href="style.css">

    <title>IoT DATA LOGGER - Home</title>

  </head>

Dokument yang tampak dalam laman adalah yang berada pada tag <body> …… </body>. Pada gambar di atas menampilkan tag <head>, dalam bagian itu menampilkan beberapa hal meliputi meta data link CSS, JS yang digunakan berikut title dari laman yang ditampilkan ( IoT DATA LOGGER – HOME). Sebagaimana penjelasan file CSS, JS di atas, file-file itu bisa terpisah dalam file yang berbeda atau link ke alamat web.

Untuk mempercantik penampilan CSS merujuk pada link web dengan tag <link……> (<link rel = "stylesheet" href = "https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">) serta local storage (<link rel="stylesheet" type="text/css" href="style.css">).

File JS untuk memberikan respons yang cepat (responsive), merujuk dari WEB dengan tag yang ditandai dengan <srcipt….>

Selain dari WEB, file JS juga merujuk dari local storage yang dipanggil pada akhir sebelum tag penutup </html>. Peletakan pada awal atau akhir itu tergantung dari apakah file itu dikerjakan lebih dulu atau kemudian. Body html, merupakan bagian htlm yang ditampilkan dalam laman web.

  <body >

    <div class="topnav">

      <ul>

        <li id = "home"><a href="/" class="disabled-link">Home</a></li>

        <li id = "report"><a href="/report">Report</a></li>

        <li id = "config"><a href="/config">Config</a></li>

        <li id = "login"><a href="/login">Login</a></li>

        <li id = "logout"><a href="/logout" onclick="return confirm('Are you sure to logout ?')">Logout</a></li>

        <li id="about"><a href="#">About</a></li>

        <li id="activeUser">User</li>

      </ul>

    </div>

    <div id='wrapper'>

      <div>

        <img src="logoGMF" alt="Logo GMF" /><br/>

        IoT Data Logger<br />

        Temporary Store Room<br />

        Line : 7

      </div>

      <div id='topContainer'>

          <div id='topContainer2'>

          <div id='label'>Time : </div>

        </div>

        <span id='time'>16:45:41</span>

      </div>

      <div id='dialContainer'>

        <div id='dial\_1'>

          <h4>Temperature</h4>

          <canvas id='canvasTemp' width='150' height='125'  style='border:1px solid #000000;' > </canvas>

      </div>

        <div id='dial\_2'>

          <h4>Humidity</h4>

          <canvas id='canvasHumid' width='150' height='125'  style='border:1px solid #000000;' > </canvas>

        </div>

      </div>  <!--  'dialContainer'  -->

      <div id='graphContainer'>

        <h4>Trending</h4>

        <canvas id='graph' width='340' height='150'  style='border:1px solid #000000; background-color:white;' > </canvas>

        <div id='key'>

          <div> <div class='box red'></div> Temperature</div>

          <div> <div class='box blue'></div> Humidity</div>

        </div>

      </div>

      <!-- The Modal -->

      <div id="myModal" class="modal">

        <!-- Modal content -->

        <div class="modal-content">

          <div class="modal-header">

            <span class="close">&times;</span>

            <h2>IoT Data Logger</h2>

          </div>

          <div class="modal-body">

            <p>It's for Final Task of</p>

            <p>Mercu Buana - University</p>

            <p>Electronic - Engineering</p>

          </div>

          <div class="modal-footer">

            <h5>By : Salman Alfarisi - 583379</h5>

          </div>

        </div>

      </div>

    </div> <!--  'wrapper'  -->

  </body>

  <script src="authentication.js"></script>

  <script src="widgetsClass.js"></script>

  <script src="modal-dialog.js"></script>

  <script src="index.js"></script>

</html>

Seluruh file yang terlibat html, CSS dan JS sebagian besar fokus pada penampilan element yang ada pada tag body ini.

Pada bagian akhir html terdapat rujukan file JS yang berada pada local storage (LittleFS) yaitu :

  <script src="authentication.js"></script>

  <script src="widgetsClass.js"></script>

  <script src="modal-dialog.js"></script>

  <script src="index.js"></script>

Pada index.html, terbagi menjadi 2 bagian penting yaitu bagian atas yang memuat seluruh menu pada laman index.html dan laman web yang lain pada IoT Data Logger ini. Menu ini menggunakan class topnav pada botstrap. Masing-masing menu merujuk pada API pada urlController sebagaimana dijelaskan pada sub bab sebelumnya, dan mengimplementasikan mekanisme Auth seperti dijalaskan pada sub bab Authentication & Authorization di atas.

Berikut tabulasi dari Auth yang dimaksud :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Menu - Topnavigation** | **GST – Guest** | **OPR - Operator** | **ENG - Engineer** |
| 1 | Home | V | V | V |
| 2 | Report |  | V | V |
| 3 | Config |  |  | V |
| 4 | Login | V | V | V |
| 5 | Logout |  | V | V |
| 6 | About | V | V | V |

Mekanisme Auth dikerjakan oleh javascript yang tertuang dalam file authentication.js, file ini tersimpan dalam local storage (LittleFS) dan dikirim ke browser setelah ada permintaan dari laman utama/induk melalui file index.html. File authentication.js akan dipanggil oleh fungsi yang terdapat dalam file index.js saat pertama kali laman tampil. Mekanisme Auth ini akan dipakai oleh keseluruhan laman yang ada pada IoT Data logger ini.

function userAccess(data) {

    if (SIMULATION) activeUser = data;

    else activeUser = JSON.parse(data);

    let user = 'GST';

    let level = parseInt(activeUser.level)

    if (level == ENGINEER\_LEVEL){

        user = 'ENG';

        // Selecting element

        var elem = document.getElementById("report");

        elem.className = "enable-link";  // Add or replace all classes with note class

        // Selecting element

        var elem = document.getElementById("login");

        elem.className = "disabled-link";  // Add or replace all classes with note class

        // Selecting element

        var elem = document.getElementById("config");

        elem.className = "enable-link";  // Add or replace all classes with note class

        // Selecting element

        var elem = document.getElementById("logout");

        elem.className = "enable-link";  // Add or replace all classes with note class

    }

    else if (level == OPERATOR\_LEVEL){

        user = 'OPR';

        // Selecting element

        var elem = document.getElementById("report");

        elem.className = "enable-link";  // Add or replace all classes with note class

        // Selecting element

        var elem = document.getElementById("login");

        elem.className = "disabled-link";  // Add or replace all classes with note class

        // Selecting element

        var elem = document.getElementById("config");

        elem.className = "disabled-link";  // Add or replace all classes with note class

        // Selecting element

        var elem = document.getElementById("logout");

        elem.className = "enable-link";  // Add or replace all classes with note class

    }

    //default user = guest

    else {

        // Selecting element

        var elem = document.getElementById("report");

        elem.className = "disabled-link";  // Add or replace all classes with note class

        // Selecting element

        var elem = document.getElementById("login");

        elem.className = "enable-link";  // Add or replace all classes with note class

        // Selecting element

        var elem = document.getElementById("config");

        elem.className = "disabled-link";  // Add or replace all classes with note class

        // Selecting element

        var elem = document.getElementById("logout");

        elem.className = "disabled-link";  // Add or replace all classes with note class

    }

    //display active user

    document.getElementById('activeUser').innerHTML = user;

}

Fungsi userAccess() melakukan tugas Auth seperti diuraikan dalam sub bab sebelumnya dengan membagi user/pengguna yang aktif menjadi 3 kategori (GST, OPR, ENG – lihat pejelasan pada sub bab sebelumnya) dan memberikan otoritas (akses) ke beberapa menu yang ada pada topnav pada setiap laman WEB pada IoT Data Logger ini.

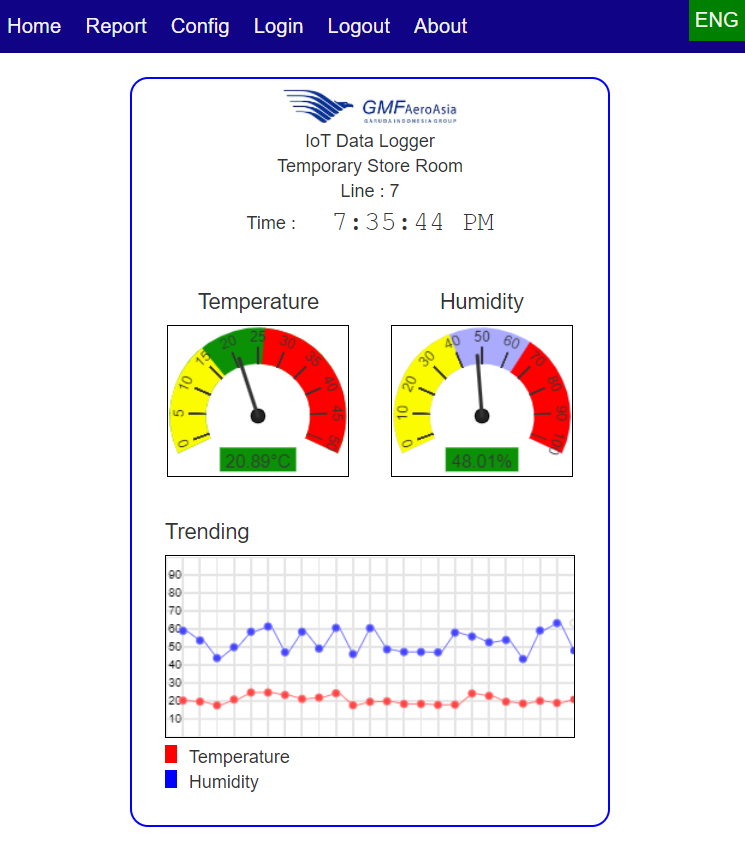
Jika semua file yang terlibat di atas (mulai dari html, css, js dan png) telah terkirim ke browser, selanjutnya tugas browser untuk menampilkan dalam layar/display user, sehingga tampak seperti gambar berikut (pada browser chrome).

Laman induk/utama terdiri dari beberapa bagian :

* Topnav (1 navigasi atas) terdapat 2 bagian (menu 1a dan active user 1b)
* Element yang menjadi satu dengan id wraper (2) terdiri dari 3 element :
  + Top container (2a) – menampilkan logo, IoT Data Logger sampai dengan waktu
  + Dial container (2b) – menampilkan indikasi (gauge) untuk temperature dan humidity
  + Graph container (2c) – menampilkan trending temperature dan humidity
* Window modal dialog saat menu About active, seperti gambar berikut ini.

Topnav – Topnavigation berikut berfungsi sebagai menu (kecuali indikasi User yang aktif).

Menu yang aktif pada topnav ini mengikuti mekanisme Auth (seperti dijelaskan di atas) dan dikerjakan oleh authentication.js dikerjakan saat laman pertama kali muncul. Saat menu dipilih, maka akan memberikan perintah/request ke server untuk memberikan file html yang bersesuaian, sesuai dengan API yang telah didefinisikan dalam fungsi urlController.



**1a**

**1**

**1b**

**2a**

**2b**

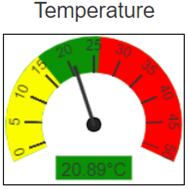
**2c**

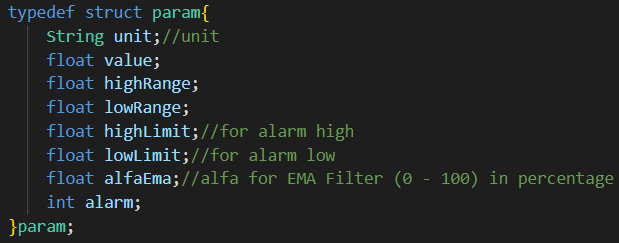
**2**

Berikut API yang dimaksud :

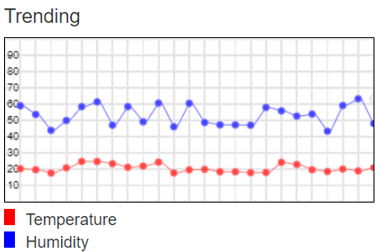
|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Menu - Topnavigation** | **API - urlController** | **Description** |
| 1 | Home | / | Laman utama |
| 2 | Report | /report | Laman report |
| 3 | Config | /config | Laman config sensor |
| 4 | Login | /login | Laman login |
| 5 | Logout | Dialog window | Dialog confirmation |
| 6 | About | Modal dialog window | Modal dialog |

Topcontainer terdapat logo, title sampai dengan waktu. Waktu diupdate setiap 1000 mili second oleh funsi interfal yang terdapat pada file index.js. File index.js ini sebagai induk seluruh file JS (javascript) yang digunakan pada laman ini.

Dialcontainer terdapat 2 dial untuk pengukuran temparature dan humidity, untuk menampilkan gambar (graphic), html menggunakan tag canvas yang selanjutnya dikerjakan oleh fungsi yang terdapat dalam file widgetsClass.js. Untuk melakukan gambar di samping diperlukan data konfigurasi sensor yang terdapat dalam IoT Data Logger (server) disusun dalam data terstruktur yang telah menjadi kelas AccessParam (Access Parameter sensor). Definisi itu ada pada file model.h dan model.cpp seperti dijelaskan di sub bab di atas. Pemodelan seperti memudahkan pengembangan program lebih lanjut sebagaimana OOP principles. Adapun parameter yang digunakan untuk kedua sensor tampak dalam gambar berikut ini :

Data yang dikirim ke browser dalam bentuk JSON format dan akan diparsing oleh browser sebelum digunakan untuk menggambar Dial. Data alarm digunakan untuk memberikan warna background pada indikasi. Jika terdapat alarm baik low/high diberikan warna merah, sedangkan jika normal diberikan warna hijau seperti gambar di atas.

Data configurasi sensor ini, diterima oleh browser dari server saat laman utama pertama kali tampil melalui API yang telah didefinisikan dalam urlController. Pengiriman data seperti di atas dapat dilakukan dengan technologi AJAX yang memungkinkan pengiriman data sebagian dari seluruh laman yang ada. Dan ini membuat laman lebih responsive.

Graphic trending, graphic ini juga dikerjakan pada fungsi yang terdapat dalam widgetsClass.js dan nilai diupdate setiap peride nilai sampling. Jumlah data trending adalah 24 sampling, artinya jika periode tiap 10000 ms (10 detik) maka akan menampung data 240 detik (4 menit).

Selanjutnya update nilai pengukuran berikut graphic trending dikerjakan oleh fungsi interval yang terdapat dalam index.js. Periode update ditentukan oleh nilai sampling time yang diberikan dari 2000 ms sampai dengan 60000 ms (1 menit), fungsi yang dimaksud seperti dalam cuplikan code berikut (dari index.js).

setInterval(function ( ) {

  var xhttp = new XMLHttpRequest();

  xhttp.onreadystatechange = function() {

    if (this.readyState == 4 && this.status == 200) {

      updateValues(this.responseText);

    }

  };

  xhttp.open("GET", "/getSensor", true);

  xhttp.send();

}, 10000 ) ;

Daftar API yang diperlukan pada laman utama (index.html):

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **API** | **HTTP Method** | **Description** |
| 1 | / | HTTP\_GET | Menampilkan home page (index.html) |
| 2 | /getTrendingData | HTTP\_GET | AJAX data trending pada graphic home page |
| 3 | /getSensorConfig | HTTP\_GET | AJAX data konfigurasi sensor pada graphic home page dan config page |
| 4 | /getActiveUser | HTTP\_GET | AJAX data aktive user digunakan pada semjua home page untuk authentication dan authorization |
| 5 | /getSensor | HTTP\_GET | AJAX data untuk menampilkan data pengukuran sensor pada home page |
| 6 | /style.css | HTTP\_GET | CSS file for styling |
| 7 | /index.js | HTTP\_GET | File javascript digunakan oleh index.html |
| 8 | /authentication.js | HTTP\_GET | File javascript digunakan oleh semua html files |
| 9 | /widgetsClass.js | HTTP\_GET | File javascript digunakan oleh index.html untuk graphic |
| 10 | /modal-dialog.js | HTTP\_GET | File javascript digunakan oleh semua file html untuk menampilkan dialog windows untuk merespon permintaan About |

Seluruh komando javascript ada pada file index.js, dari sini fungsi-fungsi pada file JS yang lain dipanggil (dijalankan). Utamanya saat pertama kali laman aktif, index.js akan menjalankan fungsi setupIndex() seperti dalam gambar di samping. Dari sini akan dijalankan fungsi :

* userAccess (untuk Auth)
* setupWidgets (untuk menggambar Dial - gauge, trending)
* InitTrendingData – data awal trending.

Fungsi yang dimaksud seperti dalam cuplikan code berikut (dari index.js).

// This is executed after the document has finished loading.

function setupIndex() {

  if (SIMULATION) {

    userAccess(activeUser);

    setupWidgets(sensorCfg);

    initTrendingData('dummy-data-simulation');

  }

  else {

    var index\_url = ["getActiveUser", "getSensorCfg", "getTrendingData"];

    var request = new XMLHttpRequest();

    (function loop(i, length) {

        if (i>= length) {

            return;

        }

        var url = "/" + index\_url[i];

        request.open("GET", url, true);

        request.onreadystatechange = function() {

            if(request.readyState === XMLHttpRequest.DONE && request.status === 200) {

              if(i == 0)userAccess(this.responseText);

              else if(i == 1)setupWidgets(this.responseText);

              else if(i == 2)initTrendingData(this.responseText);

              loop(i + 1, length);

            }

        }

        request.send();

    })(0, index\_url.length);

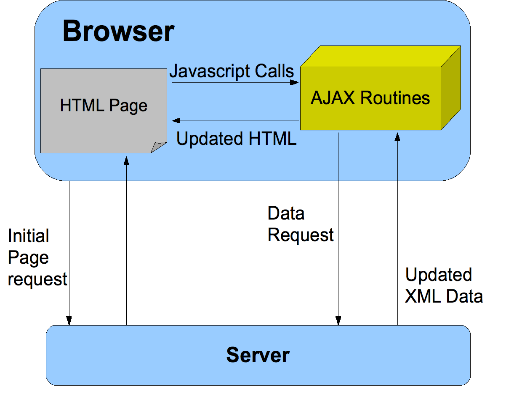
  }

}

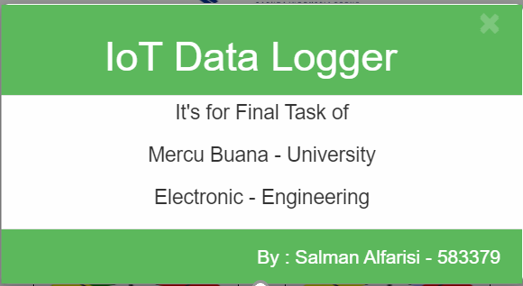
document.addEventListener('DOMContentLoaded', setupIndex, false);

Agar laman menjadi responsive terhadap pengguna/user, penggunaan teknik yang dikembangkan dengan nama AJAX sangat diperlukan. Dengan AJAX memungkinkan pengiriman data parsial (dibandingkan dengan total) yang perlu di-update saja. Hal ini membuat laman menjadi cepat ter-update demikian juga hemat dalam jumlah data yang terkirim. Laman secara keseluruhan hanya dikirim sekali saat laman mulai aktif (lihat gambar berikut ini).

AJAX - Asynchronous **JavaScript** And XML

AJAX berfungsi untuk:

* Mengambil data dari server secara *background*;
* Update tampilan web tanpa harus relaod browser;
* Mengirim data ke server secara *background*.

Dalam setiap laman IoT Data Logger, diberikan menu About yang jika dipilih/di-click/diaktifkan akan menampilkan gambar dialog modal window di samping. Untuk menampilkan window yangdimaksud tanpa melibatkan server, hal inilah yang membuat dengan peranan JS laman menjadi lebih responsive.

Keseluruhan mekanisme yang diuraikan di atas adalah kolaborasi antara server (server side/back end) dan client (client side/front end).

Report Page – laman report (reporting)

Laman report dapat muncul setelah memilih Report pada menu topnav, namun menu ini hanya bisa diakses oleh user yang login dengan level OPR ataupun ENG (lihat pembahasan Auth pada sub bab di atas). Laman ini memberikan fungsi pelaporan (reporting) berupa rerata data per jam yang dikemas dalam hari tiap pekan (Sunday – Saturday). Dengan penyimpanan data selama satu pekan (7 hari) memberikan kehandalan bahwa tidak ada data logsheet yang tidak terecord. Bentuk report dalam file PDF dan bisa di-share via WA atau email dalam bentuk attachment. Reporting dalam bentuk pdf ini memberikan keunggulan di antaranya :

* Tidak bisa diubah/diedit
* Lebih flexible dalam pengiriman data
* Telah terformat dalam bentuk tabel.

Laman reporting didapat dari file report.html yang mempunyai format umum seperti dijelaskan pada file index.html (Dokumen type, Head, Body dll – lihat penjelasan pada sub bab laman utama). Pada element head serta bagian atas dari body untuk keseluruhan laman IoT Data Logger ini adalah sama, sehingga tidak perlu dijelaskan ulang.

Javascript terkait dengan laman report ini dimulai dengan file report.js sebagai pusat komando JS yang lain. Saat pertama kali laman ini aktif, ada beberapa fungsi yang dijalankan lebih awal :

* userAccess (untuk Auth)
* fillDataTable
* generateReportTable.

userAcces seperti penjelasan pada sub bab sebelumnya (Auth).

Fungsi fillTable adalah pengambilan data dari server berupa data rerata jam yang dikemas dalam hari selama 24 jam (jam 00 – 23) sesuai dengan nama hari dalam pekan (Sunday – Saturday) data ini dikirm dalam format JSON untuk kemudahan dalam proses selanjutnya (parsing), dan pembuatan table report. Fungsi yang dimaksud seperti dalam cuplikan report.js berikut :

// This is executed after the document has finished loading.

function setupReport(){

  if (SIMULATION) {

    userAccess(activeUser);

  }

  else {

    var url\_day = "hourlyAvgDay?days=" + document.getElementById("days").value;

    var index\_url = ["getActiveUser", url\_day];

    var request = new XMLHttpRequest();

    (function loop(i, length) {

      if (i >= length) {

        return;

      }

      var url = "/" + index\_url[i];

      request.open("GET", url, true);

      request.onreadystatechange = function () {

        if (request.readyState === XMLHttpRequest.DONE && request.status === 200) {

          if (i == 0) userAccess(this.responseText);

          else if (i == 1) {

            fillDataTable(this.responseText);

            generateReportTable(table);

          }

          loop(i + 1, length);

        }

      }

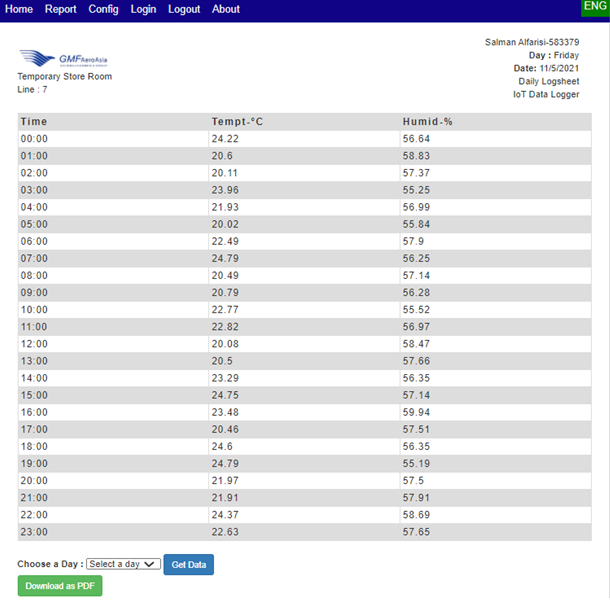
      request.send();

    })(0, index\_url.length);

  }

}

document.addEventListener('DOMContentLoaded', setupReport, false);



**1**

**1a**

**1b**

**2a**

**2**

**2b**

**2c**

**2d**

Laman report terdiri dari beberapa bagian :

* Topnav (1 navigasi atas) terdapat 2 bagian (menu 1a dan active user 1b)
* Element yang menjadi satu dengan id container (2) terdiri dari 4 element :
  + Report Header (2a, 2b) – menampilkan logo, lengkap dengan reporter
  + Body table (2c) – menampilkan tabulasi data rerata jam untuk temperature dan humidity
  + Selection hari lengkap dengan button Dowload as PDF (2d) – menampilkan pilihan hari yang dilaporkan dan export to PDF.
* Window modal dialog saat menu About active – liohat penjelasan pada sub bab sebelumnya.

Pejelasan topnav lihat pada sub bab sebelumnya.

Selain topnav, report.html terdiri dari beberapa element diantarnanya:

Daftar API yang digunakan untuk laman report.

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **API** | **HTTP Method** | **Description** |
| 1 | /report | HTTP\_GET | Menampilkan report page (report.html) |
| 2 | /haourlyAvgDay | HTTP\_GET | AJAX data untuk report page |
| 3 | /getActiveUser | HTTP\_GET | AJAX data aktive user digunakan pada semjua home page untuk authentication dan authorization |
| 4 | /style.css | HTTP\_GET | CSS file for styling |
| 5 | /report.js | HTTP\_GET | File javascript digunakan oleh report.html |
| 6 | /authentication.js | HTTP\_GET | File javascript digunakan oleh semua html files |
| 7 | /modal-dialog.js | HTTP\_GET | File javascript digunakan oleh semua file html untuk menampilkan dialog windows untuk merespon permintaan About |

Config Page – laman config (sensor configuration)

Laman Config dapat muncul setelah memilih Config pada menu topnav, namun menu ini hanya bisa diakses oleh user yang login dengan level ENG (lihat pembahasan Auth pada sub bab di atas). Laman ini memberikan fungsi configurasi sensor berikut samplingTime. Konfigurasi sensor meliputi perubahan untuk masing-masing parameter sensor yang terdapat pada model parameter sbb :

  typedef struct param{

      String unit;//unit

      float value;

      float highRange;

      float lowRange;

      float highLimit;//for alarm high

      float lowLimit;//for alarm low

      float alfaEma;//alfa for EMA Filter (0 - 100) in percentage

      int alarm;

  }param;

Format data pengiriman server – browser (client) menggunakan JSON format seperi tampak dalam format berikut :

{

    "Temperature": {

      "unit":"°C",

      "value":20.25,

      "highRange":50.0,

      "lowRange":0.0,

      "highLimit":26.0,

      "lowLimit":16.0,

      "alfaEma":80.0,

      "alarm":0

    },

    "Humidity": {

      "unit":"%",

      "value":80.75,

      "highRange":100.0,

      "lowRange":0.0,

      "highLimit":65.0,

      "lowLimit":40.0,

      "alfaEma":80.0,

      "alarm":0

    }

  }

Penggunaan JSON format mempunyai keunggulan mudah untuk dibaca ulang serta didukung banyak library (dan bahkan JSON menjadi standard format pengiriman data pada WEB). Sehingga IoT Data Logger dapat mudah dikembangkan lebih lanjut.

Laman config didapat dari file config.html yang mempunyai format umum seperti dijelaskan pada file index.html (Dokumen type, Head, Body dll – lihat penjelasan pada sub bab laman utama). Pada element head serta bagian atas dari body untuk keseluruhan laman IoT Data Logger ini adalah sama, sehingga tidak perlu dijelaskan ulang.

Javascript terkait dengan laman config ini dimulai dengan file config.js sebagai pusat komando JS yang lain. Saat pertama kali laman ini aktif, ada beberapa fungsi yang dijalankan lebih awal :

* userAccess (untuk Auth)
* fillUpParameter
* getSamplingTime.

userAcces seperti penjelasan pada sub bab sebelumnya (Auth).

Fungsi fillUpParameter adalah mengisi form pada display layar dengan data dari server dengan JSON format (seperti penjelasan sebelumnya), jenis parameter yang diisikan sejumlah parameter pada penjelasan sebelumnya. Demikian juga untuk sampling time untuk kecepatan pengukuran masing-masing sensor (temperature dan humidity). Berikut disertakan cuplikan dari file config.js terkait penjelasan di atas.

// This is executed after the document has finished loading.

function setupConfig(){

  if (SIMULATION) {

    userAccess(activeUser);

  }

  else {

    //init prevCfg

    if (document.getElementById("configT").checked) prevCfg = "configT";

    else if (document.getElementById("configH").checked) prevCfg = "configH";

    var index\_url = ["getActiveUser", "getSensorCfg", "getSamplingTime"];

    var request = new XMLHttpRequest();

    (function loop(i, length) {

      if (i >= length) {

        return;

      }

      var url = "/" + index\_url[i];

      request.open("GET", url, true);

      request.onreadystatechange = function () {

        if (request.readyState === XMLHttpRequest.DONE && request.status === 200) {

          var respText = this.responseText;

          if (i == 0) userAccess(respText);

          else if (i == 1) fillUpParameter(respText);

          else if (i == 2) getSamplingTime(respText);

          loop(i + 1, length);

        }

      }

      request.send();

    })(0, index\_url.length);

  }

}

document.addEventListener('DOMContentLoaded', setupConfig, false);

Laman config terdiri dari beberapa bagian (lihat gambar berikut):

* Topnav (1 navigasi atas) terdapat 2 bagian (menu 1a dan active user 1b)
* Element yang menjadi satu dengan div content (2) terdiri dari 4 element :
  + Sampling Time (2a) – Input sampling time (ms)
  + Radio Button (2b) – Input untuk pilihan temperature dan humidity
  + Parameter (2c) – Input parameter sensor
  + Submit Button (2d) – Button untuk submit configuration ke server.
* Window modal dialog saat menu About active – liohat penjelasan pada sub bab sebelumnya.

Pejelasan topnav lihat pada sub bab sebelumnya.

Sebelum data dikirimkan ke server, config.js melakukan validasi data sehingga data yang terkirim bebas dari kesalahan baik jenis datanya (numerik/angka ataupun nilainya). Validasi ini sangat diperlukan agar data yang terkirim tidak sia-sia dan menghabiskan bandwidth. Fungsi yang bertugas untuk itu dapat dilihat dalam cuplikan file dari config.js berikut ini :

function validateForm() {

  //sampling time

  var data = document.forms["configForm"]["samplingTime"];

  if (data.value =="")data.value = data.placeholder;

  var x = parseInt(data.value);

  if (x < 2000) {

    alert("Sampling time min 2000");

    return false;

  }

  //afaEma

  data = document.forms["configForm"]["alfaEma"];

  if (data.value =="")data.value = data.placeholder;

  x = parseInt(data.value);

  if (x < 10) {

    alert("afaEma time min 10");

    return false;

  }

  else if (x > 100) {

    alert("afaEma time max 100");

    return false;

  }

  //alarm

  data = document.forms["configForm"]["alarmH"];

  if (data.value =="")data.value = data.placeholder;

  var aH = parseInt(data.value);

  data = document.forms["configForm"]["alarmL"];

  if (data.value =="")data.value = data.placeholder;

  var aL = parseInt(data.value);

  data = document.forms["configForm"]["indicatorH"];

  if (data.value =="")data.value = data.placeholder;

  var rH = parseInt(data.value);

  data = document.forms["configForm"]["indicatorL"];

  if (data.value =="")data.value = data.placeholder;

  var rL = parseInt(data.value);

  if (aL > aH) {

    alert("alarmL GT alarmH");

    return false;

  }

  else if (aL < rL) {

    alert("alarmL LT indicatorL");

    return false;

  }

  else if (aH > rH) {

    alert("alarmH GT indicatorH");

    return false;

  }

  else if (rL > rH) {

    alert("indicatorL GT indicatorH");

    return false;

  }

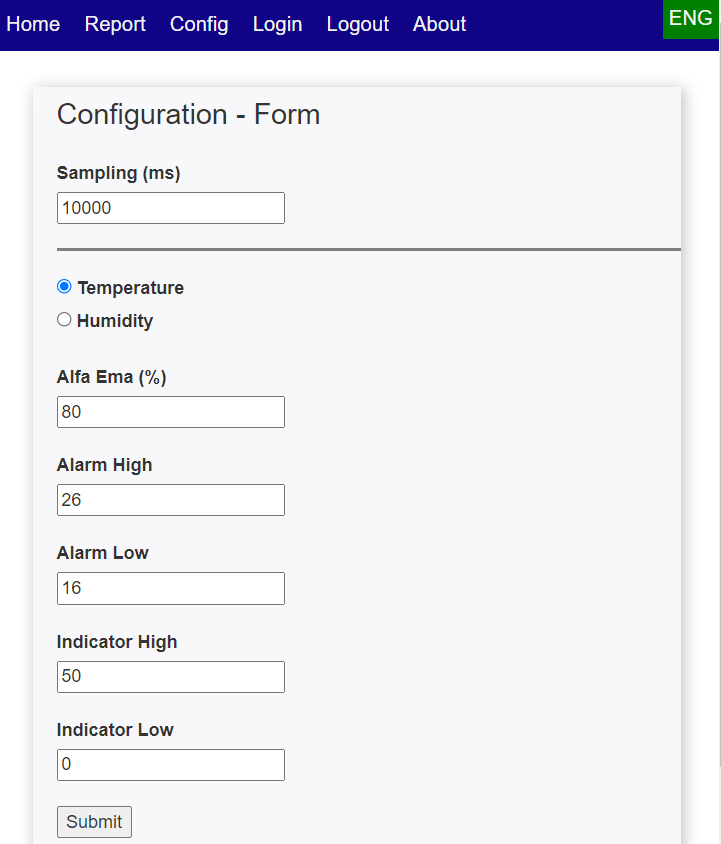
  //confirmation submit

  return confirm('Are you, sure you want to submit this parameters ?');

}

Fungsi validateForm() dipanggil menjelang dikirimnya data. Jika validasi OK (tidak ada kesalahan), maka data akan dikirim ke server dengan format JSON yang dimaksud di atas.

Berikut adalah laman config yang dimaksud :



**1**

**1a**

**1b**

**2a**

**2b**

**2**

**2c**

**2d**

Daftar API pada laman Config :

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **API** | **HTTP Method** | **Description** |
| 1 | /config | HTTP\_GET | Menampilkan report page (config.html) |
| 2 | /getSensorConfig | HTTP\_GET | AJAX data konfigurasi sensor pada graphic home page dan config page |
| 3 | /getActiveUser | HTTP\_GET | AJAX data aktive user digunakan pada semjua home page untuk authentication dan authorization |
| 4 | /getSamplingTime | HTTP\_GET | AJAX data untuk sampling time |
| 5 | /style.css | HTTP\_GET | CSS file for styling |
| 6 | /config.js | HTTP\_GET | File javascript digunakan oleh config.html |
| 7 | /authentication.js | HTTP\_GET | File javascript digunakan oleh semua html files |
| 8 | /modal-dialog.js | HTTP\_GET | File javascript digunakan oleh semua file html untuk menampilkan dialog windows untuk merespon permintaan About |

Login Page – laman Login (untuk mendukiung Auth)

Laman Login dapat muncul setelah memilih Login pada menu topnav, menu ini untuk semua user GST, OPR dan ENG (lihat pembahasan Auth pada sub bab di atas). Laman ini memberikan fasilitas user untuk bisa masuk pada level pengguna OPR atau ENG, dengan mengisikan username serta password pada form input pada laman. Parameter user ini sesuai dengan model user yang didefinikan pada file model.h sbb:

  typedef struct userData{

      String username;//engineer

      String password;//123456

      String email;//exampla@example.com

      int level;

  }userData;

Ada parameter email yang saat ini belum digunakan dan bisa dikembangkan pada untuk pengembangan lebih lanjut. Format user data pengiriman server – browser (client) menggunakan JSON format seperi tampak dalam format berikut :

  {

    "Engineer": {

      "username":"engineer",

      "password":"123456",

      "email":"engineer@example.com",

      "level":9

    },

    "Operator": {

      "username":"operator",

      "password":"123",

      "email":"operator@example.com",

      "level":1

    }

  }

Laman Login didapat dari file login.html yang mempunyai format umum seperti dijelaskan pada file index.html (Dokumen type, Head, Body dll – lihat penjelasan pada sub bab laman utama). Pada element head serta bagian atas dari body untuk keseluruhan laman IoT Data Logger ini adalah sama, sehingga tidak perlu dijelaskan ulang.

Javascript terkait dengan laman login ini dimulai dengan file login.js sebagai pusat komando JS yang lain. Saat pertama kali laman ini aktif, ada beberapa fungsi yang dijalankan lebih awal :

* userAccess (untuk Auth)
* loginStatus

userAcces seperti penjelasan pada sub bab sebelumnya (Auth).

Seperti namanya fungsi loginStatus, akan memberikan status pada user apakah sudah login atau belum.

// This is executed after the document has finished loading.

function setupLogin() {

  if (SIMULATION) {

    userAccess(activeUser);

  }

  else {

    var index\_url = ["getActiveUser", "loginStatus"];

    var request = new XMLHttpRequest();

    (function loop(i, length) {

      if (i >= length) {

        return;

      }

      var url = "/" + index\_url[i];

      request.open("GET", url, true);

      request.onreadystatechange = function () {

        if (request.readyState === XMLHttpRequest.DONE && request.status === 200) {

          if (i == 0) userAccess(this.responseText);

          if (i == 1) loginStatus(this.responseText);

          loop(i + 1, length);

        }

      }

      request.send();

    })(0, index\_url.length);

  }

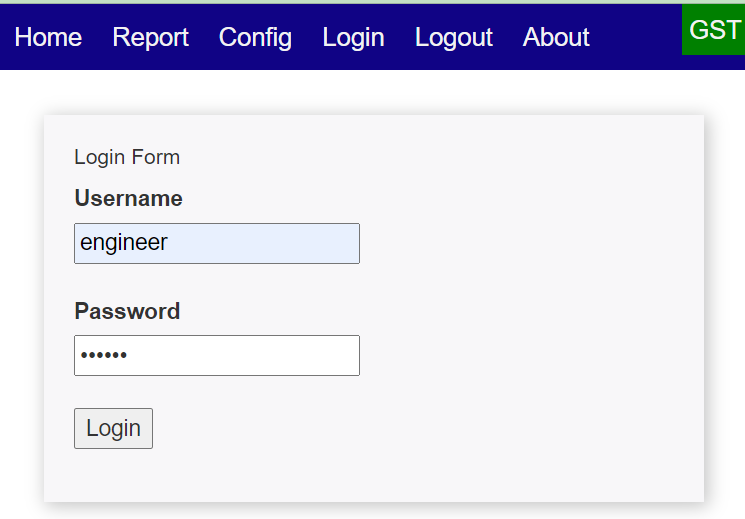
}

document.addEventListener('DOMContentLoaded', setupLogin, false);

Laman login terdiri dari beberapa bagian (lihat gambar berikut):

* Topnav (1 navigasi atas) terdapat 2 bagian (menu 1a dan active user 1b)
* Element yang menjadi satu dengan div content (2) terdiri dari 1 element :
  + Username & PAssword (2a) – Input untuk Username & Password
  + Login Button (2d) – Button untuk submit login data ke server.
* Window modal dialog saat menu About active – liohat penjelasan pada sub bab sebelumnya.

Pejelasan topnav lihat pada sub bab sebelumnya.



**1**

**1a**

**1b**

**2**

**2a**

Saat button Login ditekan (di-click) data dikirim ke server untuk divalidasi kesamaan dengan data yang tersimpan dalam server (lihat penjelasan Auth). Jika sesuai maka, user akan diarahkan ke laman utama dan bisa melakukan akses ke beberapa menu dalam topnav.

Daftar API untuk laman login

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **API** | **HTTP Method** | **Description** |
| 1 | /login | HTTP\_GET | Menampilkan report page (login.html) |
| 2 | /loginStatus | HTTP\_GET | AJAX data user status |
| 3 | /getActiveUser | HTTP\_GET | AJAX data aktive user digunakan pada semjua home page untuk authentication dan authorization |
| 4 | /style.css | HTTP\_GET | CSS file for styling |
| 5 | /login.js | HTTP\_GET | File javascript digunakan oleh login.html |
| 6 | /authentication.js | HTTP\_GET | File javascript digunakan oleh semua html files |
| 7 | /modal-dialog.js | HTTP\_GET | File javascript digunakan oleh semua file html untuk menampilkan dialog windows untuk merespon permintaan About |

