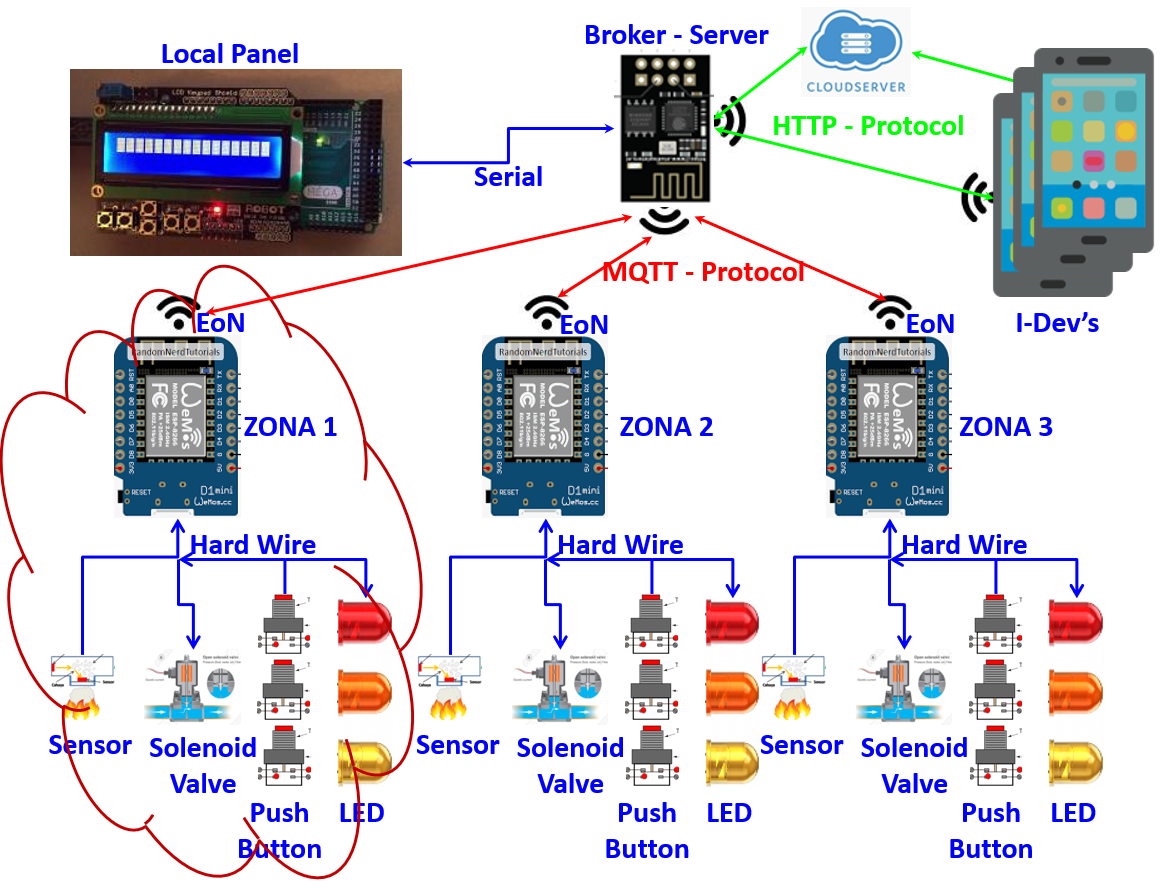
**Fire Protection System Project**

* **Spesifikasi** 
  1. System ini dirancang secara wireless serta mempunyai kemampuan diakses melalui internet sehingga bisa dibuka dari manapun (sepanjang fasilitas internet ada).
  2. Panel lokal juga dapat digunakan untuk operasional melihat status, parameter setting berikut merubahnya melalui keypad. Berikut LCD untuk menampilkan pesan ke pengguna.
  3. Pada ujung jaringan (EoN – Edge of Network) adalah node yang terdekat dengan zona dimana terpasang sensor, solenoid valve, push button, LED untuk keperluan pengamanan.
  4. Protokol komunikasi antar broker (Server) dan sensor/actuator menggunakan MQTT (Message Queuing Telemetry Transport), sedangkan kejaringan internet menggunakan HTTP (Hypertext Transfer Protocol).
  5. System ini menggunakan program dengan pendekatan OOP (Object Oriented Programming) serta menggunakan pattern/pola MVC (Model-View-Controller).
  6. Pendekatan ini ditempuh untuk mempermudah dalam pembuatan, mudah dikembangkan serta mudah melakukan pelacakan jika ada masalah.

Project ini terdiri dari 3 domain meliputi:

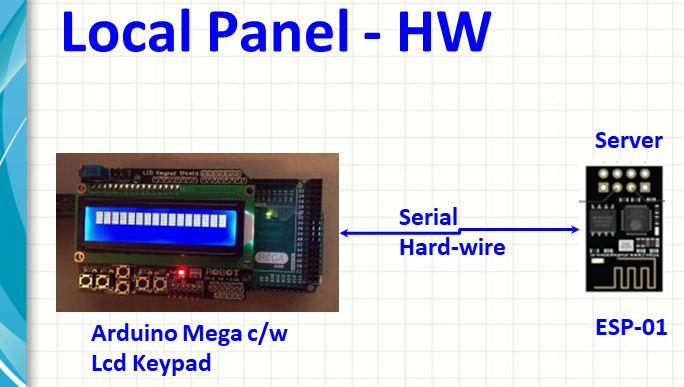
* **Local Panel**, dengan spesifikasi :
  1. System juga dilengkapi display local berupa lcd serta beberapa keypad (button) sehingga bisa diakses secara local.
  2. Pengguna (user) bisa melihat dari lcd local beberapa menu, setting parameter serta status masing-masing zona.
  3. Pengguna juga bisa melakukan perubahan setting parameter melalui keypad yang disediakan di local.
  4. Komunikasi lokal panel dengan server menggunakan serial hard-wire.
* **Server**, dengan spesifikasi:
  1. Merupakan pintu utama (gateway) untuk akses ke dan dari internet, berkomunikasi dengan Local panel (serial dengan hard-wire) serta dengan EoN.
  2. Protokol komunikasi dengan internet menggunakan HTTP (Hypertext Transfer Protocol).
  3. Protokol komunikasi dengan Eon menggunakan MQTT (Message Queuing Telemetry Transport).
  4. Pengguna juga dapat melakukan monitoring status dan perubahan secara remote melalui jaringan internet.
* **EoN**, dengan spesifikasi:
  + 1. Merupakan ujung terdepan (EoN – Edge of Network) yang terpasang sensor api (fire), solenoid valve, push button, LED untuk keperluan pengamanan.
    2. Fungsi solenoid valve (sebagai actuator) untuk membuka media pemadam api (air, powder, foam dll) dalam system ini menggunakan air bertekan (sehingga mampu melakukan penyiraman/spraying).
    3. Sensor memberikan data pengukuran ke EoN (yang selanjutnya bertidak sebagai pengendali/controller) jika nilai melebihi setting parameter, akan memberikan alarm.
    4. Controller ini bisa bekerja secara otomatis, manual-lokal, manual-remote.
    5. Jika terjadi alarm (ada api) dan controller bekerja otomatis, maka actuator bekerja dengan mengaktifkan solenoid valve untuk mengalirkan media pemadam (air bertekanan).
    6. Jika controller bekerja secara manual-lokal, actuator bekerja berdasar status dari push button.
    7. Jika controller bekerja secra manual-remote, actuator bekerja sesuai perintah yang diberikan dari remote (lokal panel ataupun lewat internet).
    8. Instalasi sensor, solenoid valve, push button serta LED menggunakan hard-wire.
    9. Komunikasi dengan server menggunakan nir-kabel (WIFI/Wireless) dengan protokol MQTT (Message Queuing Telemetry Transport).
* **Hardware**

****

**WlFPS – Wireless Protection System Hardware**

* **Domain Local Panel Hardware**

Local panel terdiri dari Arduino Mega lengkap dengan Lcd keypad untuk interface dengan pengguna (user). Pengguna dapat melakukan monitoring masing-masing zona fire protection, serta melakukan perubahan parameter setting padanya.



Komunikasi dengan server menggunakan serial (RS232) secara hard-wiring, karena lokasi keduanya berdekatan (satu tempat).

* **Domain Local Panel Software**

Pembagian program menjadi bagian-bagian kecil sesuai dengan fungsinya membuat program lebih sederhana dan dapat dikerjakan secara parallel oleh anggota tim. Pembagian ini menggunkan pattern/pola MVC (Model View Controller).

**Model** adalah bagian program yang memuat model data yang terlibat dalam program. Dalam WlFPS ini meliputi data-data yang ditampilkan dalam menu, data parameter setting masing-masing zona. Data tersusun berdasar struktur-data sehingga memudahkan dalam penangananya.

**Struktur data menu** :

**typedef struct dataMenu{**

**int id;//**

**boolean hasParameter;//is a Parameter**

**String pesan1;//pesan untuk baris pertama (1)**

**String pesan2;//pesan untuk baris kedua (2)**

**}dataMenu;**

**Struktur data parameter – Untuk masing-masing Zona :**

**typedef struct parameter{**

**String id;//**

**String unit;//unit**

**int indexMenu;//index menu**

**float value;**

**float highRange;**

**float lowRange;**

**float highLimit;//for alarm high**

**float lowLimit;//for alarm low**

**float increment;**

**boolean highAlarm = false;**

**boolean lowAlarm = false;**

**}parameter;**

Pada model ini juga dilengkapi dengan mothode/fungsi untuk melakukan akses pada masing-masing struktur data serta dijadikan dalam satu class agar menjadikan satu object. Seluruh model data dikumpulkan dalam file *Model.h* (header file) serta *Model.cpp* (uraian masing-masing method/fungsi).

**View** adalah bagian program yang bertugas menampilkan pesan ke LCD, Serial port, serta home-page (termasuk LED – jika diperlukan). Pesan ini meliputi tampilan menu utama, data parameter serta status dari WlFPS (alarm masing-masing zona). Dalam view ini juga bertugas untuk menampilkan operasional WlFPS dan menjadi antarmuka antara pengguna dan system. Seluruh class (object) view dikumpulkan ke dalam file View.h (header file) serta View.cpp (uraian masing-masing method/fungsi).

**Controller** adalah bagian program yang melakukan pengendalian/pengaturan dari seluruh kegiatan meliputi :

1. Memberikan respon pada setiap perintah yang diberikan oleh pengguna.
2. Memilih data yang ditampilkan dalam LCD, Serial Port, Web-page ataupun LED
3. Menentukan tindakan berdasar input yang masuk

Controller ini dilengkapi dengan method/fungsi beberapa menu :

1. Menu Utama
2. Menu Parameter
3. Menu Ubah Parameter

Controller juga bisa melakukan akses ke beberapa object/class :

**void attachCmdIn(command\*);//Pasang command, penggunaan pointer ini sangat mutlak (JIKA TIDAK ERROR !!!!)**

**void pasangModelMenu(AccessDataMenu\*);//Pasang Model menu, penggunaan pointer ini sangat mutlak (JIKA TIDAK ERROR !!!!)**

**void pasangModelParameter(AccessParameter\*);//Pasang Model parameter, penggunaan pointer ini sangat mutlak (JIKA TIDAK ERROR !!!!)**

**void pasangView(View\*);//Pasang View, penggunaan pointer ini sangat mutlak (JIKA TIDAK ERROR !!!!)**

**void setMenuIndex(int);**

**int getMenuIndex();**

**int getParameterIndex();**

**void menu();**

Perintah diakses melalui void attachCmdIn(command\*) – pasang input-command/perintah yang meliputi perintah dari Keypad, Serial, Bluetooth, Wifi ataupun via Web page serta dari sensor yang terpasang. Dari perintah ini kemudian controller melakukan proses untuk pemilihan data, memproses serta menampilkan dalam LCD, Serial Webpage serta LED.

Model diakses melalui void pasangModelMenu (AccessDataMenu\*), void pasangModelParameter (AccesssDataParameter\*) yang memungkinkan controller untuk melakukan akses (read/write).

View diakses melalui void pasangView(View\*) yang memungkinkan controller untuk memberikan erintah pada View untuk menampilkan data menu, parameter setelah dilakukan proses sesuai logic/algoritma yang dibuat dalam WlFPS.

Seluruh kegiatan controller dilakukan dalam void menu() didalamnya terdapat mekanisme pengambilan perintah/command dari masing-masing inputan (Keypad, Serial, Bluetooth, Web-page) menentukan menu-mana yang aktif berdasar perintah yang masuk. Menu ini meliputi Menu Utama, Menu Parameter, Menu Ubah Parameter. Dalam masing-masing menu itu ditentukan data yang ditampilkan dalam LCD, LED, Web-page serta fungsi untuk masing-masing Keypad.

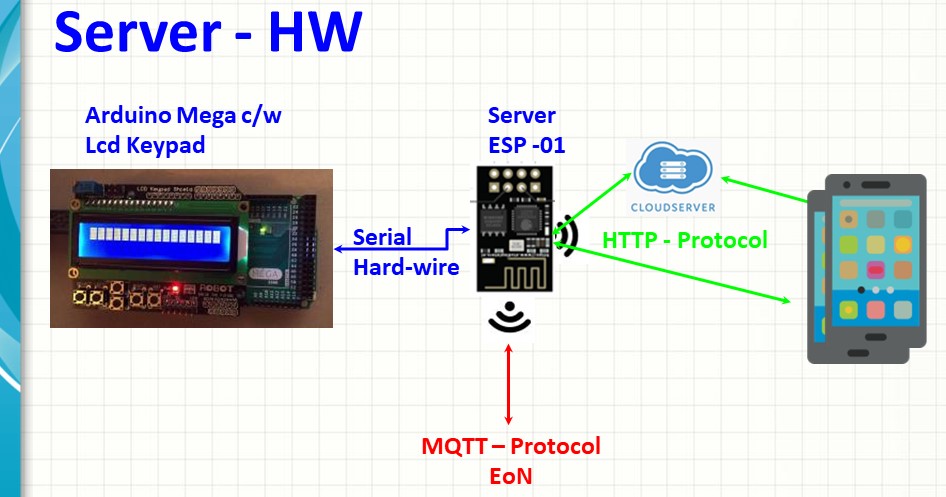
Seluruh mekanisme controller ini ada pada file Controller.h (header file) dan Controller.cpp (uraian masing-masing method/fungsi).

Selain hal di atas, WlFPS juga mempunyai object/class yang bertugas untuk mengatur beberapa fungsi Watch-dog (berupa LED yang berkedip setiap satu detik pada Built-in LED). Watch-dog ini berfungsi sebagai tanda bahwa system bekerja normal. Fungsi itu ada pada file SequenceTimer.h (header file) dan SequenceTimer.cpp (uraian masing-masing method/fungsi).

* **Domain Server – Hardware**

Server merupakan pintu gerbang yang menghubungkan WlPS dengan dunia internet, sekaligus menjadi jembatan local panel. Fungsi Server adalah :

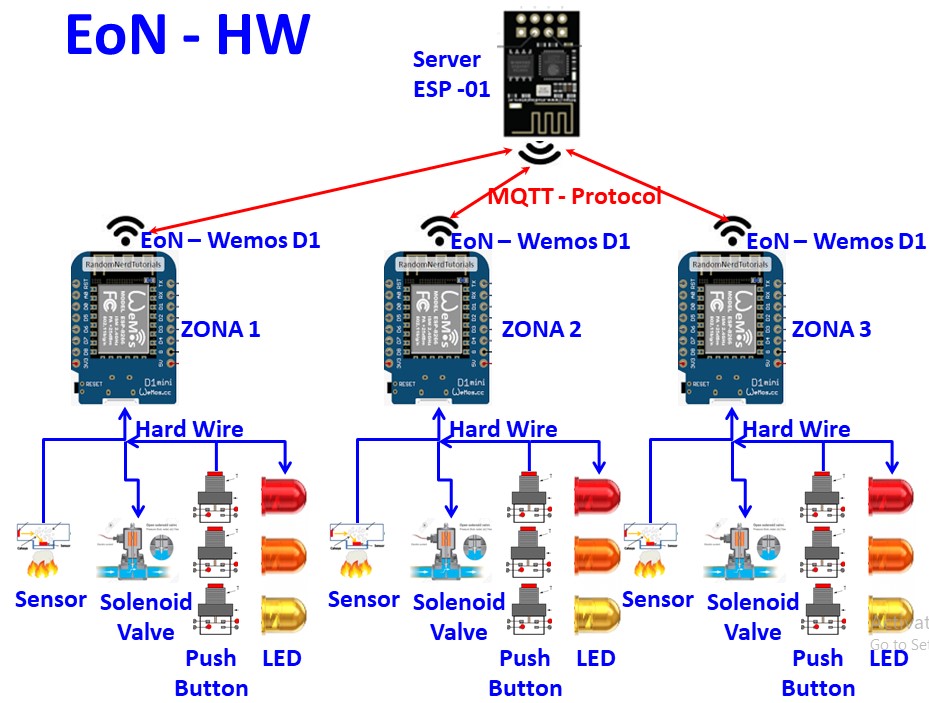
* Pintu gerbang dengan Internet menggunakan protocol komunikasi HTTP.
* Berkomunikasi dengan EoN secara nir-kable (WIFI) dengan protocol MQTT
* Berkomunikasi dengan local panel denga serial 232 secara hard-wire



* **Domain Server – Software**

Uraian Software Server

* **Domain EoN – Hardware**

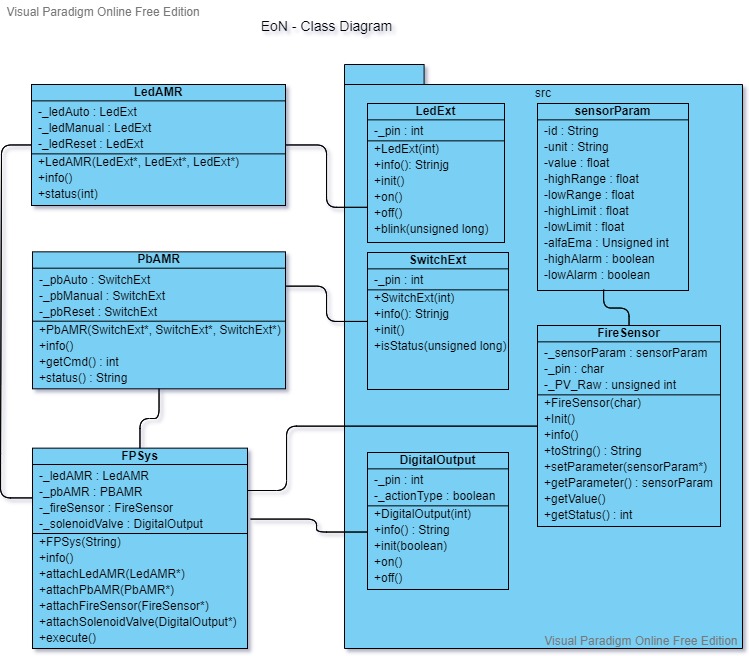


* **Domain EoN - Software**

EoN – merupakan ujung terdepan (EoN – Edge of Network) yang terpasang sensor api (fire), solenoid valve, push button, LED untuk keperluan pengamanan. Terdiri dari beberapa class yang bekerja mendukung fungsinya sbb:

* FPSys – Fire Protection System, object (instantiasi dari class) yang bertugas melakukan pengamanan/proteksi api.
* CommServer – Communication to server, object yang bertugas untuk komunikasi dengan server melalui WIFI dengan protokol MQTT.

Gambaran secara detail dari masing-masing class dapat dilihat dari diagram class berikut ini :



FPSys terdiri dari beberapa object (instantiasi dari class) dengan garis besar sbb:

* LedAMR : adalah class yang bertugas untuk menampilkan (View) berupa 3 LED sesuai dengan mode operation :
  + Mode Auto : LED Auto menyala
  + Mode Manual : LED Manual menyala
  + Mode Manual On : LED Manual berkedip/blink (menunjukkan solenoid bekerja/aktif saat mode manual)
  + Mode Reset : LED Reset menyala
  + Mode Ready : Semua LED mati (off)

Dengan memberikan informasi terkait mode operasi, maka object akan menetukan LED yang mana yang sedang aktif, melalui method/funtion ***void status (int modeOperation)***.

LedAMR ini merupakan class wrapper (pembungkus) dari class LedExt (LED Extended). LedExt merupakan object dari LED yang mengoperasikan LED dalam beberapa mode :

* Mode on : LED on
* Mode off : LED off
* Mode blink : LED berkedip/blink
* PbAMR : adalah push button dengan 3 switch/button untuk memberikan perintah ke FPSys. Perintah dari 3 switch ini menentukan mode opersi dari FPSys.
  + Button/switch Auto : Menentukan mode Auto
  + Button/switch Manual : Menentukan mode Manual
  + Button/switch Manual : Menentukan mode Manual On
  + Button/switch Reset : Menentukan mode Reset
  + Button/switch Reset : Menentukan mode Ready

Operationmode logic :

***int PbAMR::getCmd(unsigned long debounceTime){***

***int cmd;***

***if (\_pbReset->isStatus(debounceTime)){***

***if (\_prevCmd == PB\_RESET) cmd = PB\_READY;//highest priority***

***else if (\_prevCmd == PB\_READY) cmd = PB\_READY;//highest priority***

***else cmd = PB\_RESET;//highest priority***

***if(\_prevCmd != cmd) \_isInfo = false;***

***\_prevCmd = cmd;***

***}***

***else {***

***if((\_pbAuto->isStatus(debounceTime) && (\_prevCmd != PB\_MANUAL\_ON))){***

***cmd = PB\_AUTO;//lower priority***

***if(\_prevCmd != cmd) \_isInfo = false;***

***\_prevCmd = cmd;***

***}***

***if(\_pbManual->isStatus(debounceTime)){***

***if (\_prevCmd == PB\_MANUAL) cmd = PB\_MANUAL\_ON;//higher priority***

***else if (\_prevCmd == PB\_MANUAL\_ON) cmd = PB\_MANUAL\_ON;//higher priority***

***else cmd = PB\_MANUAL;//higher priority***

***if(\_prevCmd != cmd) \_isInfo = false;***

***\_prevCmd = cmd;***

***}***

***else cmd = \_prevCmd;***

***}***

***return cmd;***

***}***

To be continue…..

* FireSensor

To be continue…..

* DigitalOutput

To be continue…..

* Package src - Sources

To be continue…..