**INSTITUT TEKNOLOGI BANDUNG**

**PROGRAM STUDI TEKNIK ELEKTRO**

Jalan Ganesha No. 10 Gedung Labtek V Lantai 2 🕿 (022)2508135-36, 🖷 (022)2500940

Bandung 40132

**Dokumentasi Produk Tugas Akhir**

Lembar Sampul Dokumen

|  |  |  |
| --- | --- | --- |
| Judul Dokumen | TUGAS AKHIR TEKNIK ELEKTRO:  Sistem Detektor Gempa dan Peringatan Dini Tsunami *Decision Support System* | |
|  |  | |
| Jenis Dokumen | IMPLEMENTASI | |
|  | Catatan: Dokumen ini dikendalikan penyebarannya oleh Prodi Teknik Elektro ITB | |
| Nomor Dokumen | B400-01-TA1617.01. 069 | |
|  |  | |
| Nomor Revisi | 01 | |
|  |  | |
| Nama File | B400 .docx | |
|  |  | |
| Tanggal Penerbitan | 8 Maret 2017 | |
|  |  | |
| Unit Penerbit | Prodi Teknik Elektro - ITB | |
|  |  | |
| Jumlah Halaman | 52 | (termasuk lembar sampul ini) |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Data Pemeriksaan dan Persetujuan | | | | |
| Ditulis | Nama | Kevin Shidqi | Jabatan | Anggota |
| oleh | Tanggal | 8 March 2017 | Tanda Tangan |  |
|  | Nama | Bramantio Yuwono | Jabatan | Anggota |
|  | Tanggal | 8 March 2017 | Tanda Tangan |  |
|  | Nama | Christoporus Deo  Putratama | Jabatan | Anggota |
|  | Tanggal | 8 March 2017 | Tanda Tangan |  |
| Diperiksa | Nama | Dr. techn. Ary Setijadi  Prihatmanto | Jabatan | Dosen Pembimbing |
| oleh | Tanggal | 8 Maret 2017 | Tanda Tangan |  |
|  |  |  |  |  |
| Disetujui | Nama | Dr. techn. Ary Setijadi  Prihatmanto | Jabatan | **Dosen Pembimbing** |
| Oleh | Tanggal | 8 Maret 2017 | Tanda Tangan |  |
|  |  |  |  |  |

# DAFTAR ISI

[DAFTAR ISI 2](#_Toc449991383)

[Catatan Sejarah Perbaikan Dokumen 3](#_Toc449991384)

[Proposal Proyek Pengembangan “ Rancang Bangun *Flapping Wings* *Micro Aerial Vehicle*: Sistem Kendali, Sensor, dan Telemetri ” 4](#_Toc449991385)

[1 Pengantar 4](#_Toc449991386)

[1.1 Ringkasan Isi Dokumen 4](#_Toc449991387)

[1.2 Tujuan Penulisan dan Aplikasi/Kegunaan Dokumen 4](#_Toc449991388)

[1.3 Referensi 4](#_Toc449991389)

[1.4 Daftar Singkatan 5](#_Toc449991390)

[2 IMPLEMENTASI 7](#_Toc449991391)

[2.1 Implementasi pengiriman pesan 7](#_Toc449991392)

[**2.1.1** **Pendahuluan** 7](#_Toc449991393)

[**2.1.2** **Struktur** 10](#_Toc449991394)

[**2.1.3** **Environment** 12](#_Toc449991395)

[**2.1.4** **Prosedur Implementasi RabbitMQ** 12](#_Toc449991396)

[**2.1.5** **Prosedur Implementasi pengiriman** 15](#_Toc449991397)

[2.2 Sensor Earthquake Catcher Network 16](#_Toc449991398)

[**2.2.1** **Main Module** 16](#_Toc449991399)

[**2.2.2** **Komponen Elektrik** 35](#_Toc449991400)

[**2.2.3** **Algoritma** 38](#_Toc449991401)

[2.3 Pengolah Data dan GUI 48](#_Toc449991403)

[**2.3.1** **Pengolah Data** 51](#_Toc449991404)

[**2.3.2** **Antarmuka Grafis** 54](#_Toc449991405)

# Catatan Sejarah Perbaikan Dokumen

|  |  |  |  |
| --- | --- | --- | --- |
| Revisi | Tanggal | Oleh | Perbaikan |
|  |  |  |  |
|  |  |  |  |
|  |  |
|  |  |

# Implementasi Proyek Sistem Jaringan Detektor Gempa dan Tsunami Decision Support System

# Pengantar

## Ringkasan Isi Dokumen

Dokumen B400 ini merupakan dokumentasi hasil implementasi proyek pengembangan Sistem Jaringan Detektor Gempa dan Tsunami Decision Support System. Dokumen B400 ini terbagi dalam dua bab. Bab pertama adalah pengantar yang di dalamnya dibahas ringkasan isi dokumen, tujuan penulisan, kegunaan dokumen, referensi yang digunakan, dan daftar singkatan. Selanjutnya pada bab kedua dibahas hasil implementasi proyek pengembangan Sistem Jaringan Detektor Gempa dan Tsunami Decision Support System. Pembahasan hasil implementasi produk tersebut meliputi implementasi pada sensor gempa, messaging server, Seiscomp3, dan GUI berdasarkan desain sistem yang telah dipaparkan pada dokumen B100, B200, dan B300

## Tujuan Penulisan dan Aplikasi/Kegunaan Dokumen

Tujuan dari penulisan dokumen B400 ini adalah:

a. Pemaparan hasil implementasi tugas akhir Sistem Jaringan Detektor Gempa dan Tsunami Decision Support System.;

b. Dokumentasi dalam pengerjaan tugas akhir Sistem Jaringan Detektor Gempa dan Tsunami Decision Support System..

Dokumen B400 ini ditujukan untuk tim dosen pembimbing TA1617.01.069 dan tim tugas akhir Teknik Elektro ITB 2016.

## Referensi

[1] Rudloff, Alexander, *German-Indonesian Tsunami Early Warning System (GITEWS) Decision Support System (DSS),* Deutschen Zentrums für Luft- und Raumfahrt (DLR): Köln (2010)

## Daftar Singkatan

| Singkatan | Arti |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# Implementasi Proyek

## Implementasi Pengiriman Pesan menuju Messaging Server RabbitMQ

### Pendahuluan

Pengiriman pesan dilakukan dari sensor gempa menuju messaging server. Protokol yang digunakan oleh sensor untuk mengirim pesan adalah MQTT, sedangkan pesan diterima oleh messaging server RabbitMQ dengan protocol AMQP. Hal ini dapat dilakukan dengan menggunakan format pengiriman pesan berupa file dengan ekstensi .json, sehingga kedua protocol tersebut dapat saling berkomunikasi satu sama lain.

### Struktur Implementasi

Implementasi pengiriman pesan menuju messaging server RabbitMQ dilakukan berdasarkan perancangan produk yang sudah dilakukan. Sebuah sensor gempa akan mengirimkan data dengan format .json untuk dimasukkan ke dalam sebuah queue pada RabbitMQ. Namun untuk dapat memasukkan pesan dari sensor yang identik ke dalam sebuah queue, maka perlu dilakukan beberapa tahap implementasi, yaitu adalah sebagai berikut:

1. menentukan protokol pengiriman pesan yang dilakukan oleh sensor gempa
2. pembuatan akun server RabbitMQ umum (CloudAMQP) di internet atau akun server RabbitMQ local yang telah disediakan.
3. pengaturan server dengan melakukan konfigurasi queue, topic dan exchange melalui browser desktop sehingga pesan yang diterima mempunyai channel tersendiri.

### Environment

Perangkat keras dan piranti lunak yang dibutuhkan dalam mengimplementasikan RabbitMQ server adalah sebagai berikut:

1. perangkat keras Laptop Asus X450JE sebagai komputer untuk menjalankan lokal RabbitMQ dan juga sebagai tempat Microsoft Visual Studio 2013 dijalankan.
2. perangkat lunak Microsoft Visual Studio 2013 sebagai perangkat lunak IDE untuk mengembangkan GUI yang berfungsi sebagai consumer RabbitMQ.
3. perangkat keras Raspberry Pi 3 sebagai mini komputer untuk menjalankan lokal RabbitMQ
4. perangkat lunak Python 3.5 sebagai perangkat lunak IDE yang digunakan pada Raspberry Pi 3 yang berfungsi sebagai pengirim pesan maupun consumer pada RabbitMQ
5. perangkat keras Node MCU Amica sebagai mikrokontroller sensor gempa untuk mengirimkan pesan geospasial menuju server RabbitMQ
6. perangkat lunak Arduino IDE 1.8.1 sebagai perangkat lunak IDE untuk mengembangkan algoritma pengiriman pesan geospasial menuju RabbitMQ
7. JSON sebagai library dan syntax pengiriman pesan antara sensor gempa dengan RabbitMQ dan juga antara RabbitMQ dengan consumer

### Implementasi RabbitMQ

Penggunaan RabbitMQ pada sebuah devais bisa dilakukaan dengan melakukan instalasi dan konfigurasi pada platform tertentu. Dalam proyek ini, platform yang digunakan adalah dengan Windows, Linux, maupun Arduino. Selain menggunakan platform, dapat juga menggunakan RabbitMQ umum yang terdapat pada internet dengan menggunakan jasa CloudAMQP. Rincian implementasi RabbitMQ pada semua devais yang digunakan adalah sebagai berikut:

#### Implementasi RabbitMQ dengan menggunakan CloudAMQP

CloudAMQP digunakan sebagai messaging server online. Supaya dapat menggunakan CloudAMQP, dilakukan pendaftaran akun terlebih dahulu.

Pendaftaran CloudAMQP dapat dilakukan dengan membuka tautan dibawah ini pada browser internet.

|  |
| --- |
| [*https://www.cloudamqp.com/plans.html*](https://www.cloudamqp.com/plans.html) |

Apabila menelaah halaman web ini, disediakan beberapa plan untuk membuat sebuah instance pada CloudAMQP. Proyek ini menggunakan plan Little Lemur karena tidak berbayar. Klik tombol “Try Little Lemur now” untuk menggunakan plan ini.

|  |
| --- |
|  |

Kemudian akan terbuka halaman web untuk melakukan login atau pendaftaran. Pendaftaran bisa dilakukan dengan melakukan sign up atau dengan menggunakan akun Github maupun akun Google.

|  |
| --- |
|  |

Setelah melakukan pendaftaran, maka akan terbuka halaman web untuk membuat sebuah instance sesuai dengan plan yang dipilih. Masukan nama Instance yang diinginkan, plan yang digunakan, dan data center. Data center dapat dipilih dengan bebas namun disarankan dekat dengan negara tempat mengakses server agar dapat mengurangi latency.

|  |
| --- |
|  |

Akan muncul sebuah list instance ketika membuat instance lebih dari sekali. Artinya, ada beberapa server yang bisa digunakan sebagai messaging server. Namun, server tidak berbayar ini memiliki batasan yaitu jumlah message dan koneksi yang terbatas. Apabila hendak mengatur konfigurasi server, maka dapat dilakukan dengan klik tombol “RabbitMQ Manager di sebelah kanan layar browser.

|  |
| --- |
|  |

Setelah itu, pada browser akan terbuka halaman baru yang merujuk pada alamat server. Ketika pertama kali membuka halaman web server ini, maka tampilan yang akan dilihat adalah sebagai berikut:

|  |
| --- |
|  |

Hal yang dilakukan pertama kali adalah dengan mengklik tab Queues pada bagian atas layar.

|  |
| --- |
|  |

Pada tab Queues akan terdapat kolom name, durability, auto delete, dsb. Kolom name diisi dengan nama Queue (dalam proyek ini, menggunakan nama “ecn”), kolom durability diisi dengan pilihan “Durable”, sementara itu bagian lain tidak perlu diubah. Setelah itu, tekan tombol Add queue di bagian bawah halaman. Apabila sudah menekan tombol tersebut, maka akan muncul sebuah list queue dengan judul “ecn”.

|  |
| --- |
|  |

Kemudian hal selanjutnya yang harus dilakukan adalah klik tab Exchanges di bagian atas halaman.

|  |
| --- |
|  |

Akan ditampilkan layar exchanges pada halaman web. Klik kalimat amq.topic pada bagian bawah list exchange.

|  |
| --- |
|  |

Kemudian dilakukan binding exchange dengan queue. Isi kolom seperti gambar diatas. Hal ini dilakukan supaya queue dihubungkan dengan exchange melalui jalur routing key “earthquake”.

Apabila sudah melakukan bindings, maka akan terdapat tampilan seperti dibawah ini

|  |
| --- |
|  |

Konfigurasi CloudAMQP sebagai messaging server online bisa dikatakan cukup hingga tahap ini. Penggunaan server ini pada beberapa devais harus mengacu pada syntax yang telah disediakan oleh dokumentasi RabbitMQ sehingga akan diperoleh konfigurasi berbeda pada platform yang berbeda.

### Implementasi pengiriman pesan oleh Sensor Gempa

Pada proyek ini, sensor gempa dikembangkan pada mikrokontroller Node MCU Amica dengan menggunakan perangkat lunak Arduino IDE. Sensor gempa akan mengirimkan pesan pada RabbitMQ berupa string yang memiliki format extensi json. Pengiriman pesan dilakukan dengan menggunakan protokol MQTT. Hal ini disebabkan karena Arduino tidak mendukung AMQP dalam implementasinya, namun peran AMQP sebagai messaging protocol dapat disubtitusi oleh MQTT. Perbedaannya hanya terletak di kompleksitas dan fitur pengiriman pesan yang diberikan oleh kedua protokol tersebut. Algoritma dari prosedur pengiriman pesan yang dilakukan oleh Node MCU Amica adalah sebagai berikut:

|  |
| --- |
| //====================================================//  //======Wi-Fi Connection & MQTT Function Procedure====//  //====================================================//  void WiFiConnect**()**  **{**  // We start by connecting to a WiFi network  Serial**.**print**(**"Connecting to "**);**  Serial**.**println**(**ssid**);**  WiFi**.**begin**(**ssid**,** pass**);**  **while** **(**WiFi**.**status**()** **!=** WL\_CONNECTED**)**  **{**  delay**(**500**);**  Serial**.**print**(**"."**);**  **}**  randomSeed**(**micros**());**  Serial**.**println**(**""**);**  Serial**.**println**(**"WiFi connected"**);**  Serial**.**println**(**"IP address: "**);**  Serial**.**println**(**WiFi**.**localIP**());**  **}**  void reconnect\_server**()** **{**  // Loop until we're reconnected  **while** **(!**client**.**connected**())**  **{**  Serial**.**print**(**"Attempting MQTT connection..."**);**  // Create a random client ID  String clientId **=** "ESP8266Client-"**;**  clientId **+=** String**(**random**(**0xffff**),** HEX**);**  // Attempt to connect  //if you MQTT broker has clientID,username and password  //please change following line to if (client.connect(clientId,userName,passWord))  **if** **(**client**.**connect**(**mqtt\_clientID**.**c\_str**(),** mqtt\_user**.**c\_str**(),**mqtt\_password**.**c\_str**()))**  **{**  Serial**.**println**(**"connected"**);**  **}** **else** **{**  Serial**.**print**(**"failed, rc="**);**  Serial**.**print**(**client**.**state**());**  Serial**.**println**(**" try again in 5 seconds"**);**  // Wait 6 seconds before retrying  delay**(**6000**);**  **}**  **}**  **}** //end reconnect() |

Algoritma diatas merupakan algoritma untuk melakukan koneksi dengan Access Point terdekat (koneksi dengan menggunakan Wi-Fi). Koneksi dilakukan dari antara Node MCU menuju access point kemudian dilakukan access menuju messaging server. Secara singkat, kegunaan fungsi-fungsi diatas adalah sebagai berikut:

|  |  |
| --- | --- |
| WifiConnect() | Melakukan proses koneksi dengan akun ssid dan password |
| reconnect\_server() | Melakukan proses rekoneksi dan pengiriman data mqtt |

Namun, untuk dapat melakukan proses koneksi, dibutuhkan beberapa pengaturan. Konfigurasi yang dilakukan supaya messaging server (dalam hal ini, RabbitMQ yang digunakan adalah CloudAMQP) dapat diakses adalah:

|  |
| --- |
| const char**\*** mqtt\_server **=** "black-boar.rmq.cloudamqp.com"**;** //MQTT server  const char**\*** server\_topic **=** "amq.topic.ecn"**;** //MQTT server topic  String mqtt\_clientID **=** "ESP8266Client-1"**;**  String mqtt\_user **=** "lsowqccg:lsowqccg"**;**  String mqtt\_password **=** "kbLv9YbzjQwxz20NH7Rfy98TTV2eK17j"**;** |

Fungsi dari masing-masing variable pada konfigurasi tersebut adalah:

|  |  |
| --- | --- |
| mqtt\_server | Variable penampung alamat server yang akan diakses |
| server\_topic | Variable penampung nama exchange yang digunakan |
| mqtt\_clientID | Variable nama klien sensor gempa |
| mqtt\_user | Variable penampung user dan vhost dari messaging server |
| mqtt\_password | Variabel penampung password messaging server |

Sementara itu, pesan yang dikirimkan oleh prosedur MQTT adalah sebagai berikut:

|  |
| --- |
| //====================================================//  //==================ENCODE JSON FUNCTION==============//  //====================================================//  struct data InitJsonObject**(**struct data msg**)**  **{**  static const double init\_lat **=** **-**6.889916**,** init\_lon **=** 107.61133**;**  msg**.**pointTime **=** "test"**;**  msg**.**timeZone **=** "Asia/Jakarta"**;**  msg**.**interval **=** "500"**;**  msg**.**geometry**.**type **=** "Feature"**;**  msg**.**geometry**.**geo**.**type **=** "Point"**;**  msg**.**geometry**.**geo**.**coordinates**[**0**]** **=** String**(**init\_lat**);**  msg**.**geometry**.**geo**.**coordinates**[**1**]** **=** String**(**init\_lon**);**  msg**.**geometry**.**property**.**Name **=** "ITB"**;**  **return** msg**;**  **}**  String JsonToString**(**struct data msg**)**  **{**  String a **=** ""**;**  a **=** a **+** "{" **+** " \"pointTime\": " **+** "\"" **+** msg**.**pointTime **+** "\"" **+** ","**;**  a **=** a **+** "\"timeZone\":" **+** "\"" **+** msg**.**timeZone **+** "\"" **+** ","**;**  a **=** a **+** "\"interval\":" **+** msg**.**interval **+** ","**;**  a **=** a **+** "\"geojson\" :" **+** "{"**;**  a **=** a **+** "\"type\":" **+** "\"" **+** msg**.**geometry**.**type **+** "\"" **+** ","**;**  a **=** a **+** "\"geometry\":" **+** "{"**;**  a **=** a **+** "\"type\":" **+** "\"" **+** msg**.**geometry**.**geo**.**type **+** "\"" **+** ","**;**  a **=** a **+** "\"coordinates\": [ " **+** msg**.**geometry**.**geo**.**coordinates**[**0**]** **+** ","**;**  a **=** a **+** msg**.**geometry**.**geo**.**coordinates**[**1**]** **+** "]"**;**  a **=** a **+** "},"**;**  a **=** a **+** "\"properties\":" **+** "{"**;**  a **=** a **+** "\"name\":" **+** "\"" **+** msg**.**geometry**.**property**.**Name **+** "\""**;**  a **=** a **+** "}"**;**  a **=** a **+** "},"**;**  a **=** a **+** "\"accelerations\": ["**;**  **for** **(**int i **=** 0**;** i **<** 20**;** i**++)**  **{**  **if** **(**i **!=** 19**)**  **{**  a **=** a **+** "{"**;**  a **=** a **+** "\"x\": " **+** msg**.**accelerations**[**i**].**x **+** ","**;**  a **=** a **+** "\"y\": " **+** msg**.**accelerations**[**i**].**y **+** ","**;**  a **=** a **+** "\"z\": " **+** msg**.**accelerations**[**i**].**z **;**  a **=** a **+** "},"**;**  **}**  **else**  **{**  a **=** a **+** "{"**;**  a **=** a **+** "\"x\": " **+** msg**.**accelerations**[**i**].**x **+** ","**;**  a **=** a **+** "\"y\": " **+** msg**.**accelerations**[**i**].**y **+** ","**;**  a **=** a **+** "\"z\": " **+** msg**.**accelerations**[**i**].**z **;**  a **=** a **+** "}"**;**  **}**  **}**  a **=** a **+** "]"**;**  a **=** a **+** "}"**;**  **return** a**;**  **}** |

Format JSON ini dibuat supaya dapat dilakukan pemilahan pesan dengan mudah oleh consumer pada messaging server. Pesan yang dikirimkan adalah data geospasial dengan urutan variable tertentu.

Supaya bisa membentu urutan data yang baik pada JSON, pengaturan tipe data berbasis objek telah dilakukan. Algoritmanya adalah sebagai berikut:

|  |
| --- |
|  |

Apabila dikirimkan, maka bentuk pesan yang akan ditangkap oleh messaging server adalah sebagai berikut:

|  |
| --- |
| '{ "pointTime": "","timeZone":"Asia/Jakarta","interval":1000,"geojson" :{"type":"Feature","geometry":{"type":"Point","coordinates": [ 123.6,123.6]}"properties":{"name":"ITB"}},"accelerations": [{"x": 72,"y": 32,"z": 36},{"x": 220,"y": 60,"z": 172},{"x": 172,"y": 232,"z": 160},{"x": 168,"y": 56,"z": 48},{"x": 172,"y": 32,"z": 252},{"x": 172,"y": 20,"z": 100},{"x": 232,"y": 100,"z": 68},{"x": 4,"y": 132,"z": 200},{"x": 200,"y": 84,"z": 80},{"x": 152,"y": 16,"z": 96},{"x": 12,"y": 12,"z": 140},{"x": 136,"y": 132,"z": 220},{"x": 180,"y": 136,"z": 48},{"x": 16,"y": 28,"z": 244},{"x": 140,"y": 76,"z": 0},{"x": 168,"y": 120,"z": 56},{"x": 208,"y": 128,"z": 96},{"x": 52,"y": 112,"z": 108},{"x": 216,"y": 4,"z": 108},{"x": 184,"y": 56,"z": 44}]}' |

Pesan diatas dikirimkan menuju CloudAMQP setiap 1 detik. Artinya dalam satu detik, messaging server akan mendapatkan data lokasi sensor oleh GPS dan juga data perubahan kecepatan pada 3 buah sumbu (x, y, dan z) yang telah dipadatkan hingga 20 buah array per detik, sehingga frekuensi data gelombang perubahan kecepatan sensor gempa mencapai 20 HZ/sumbu koordinat.

### Implementasi pengambilan pesan oleh Consumer

Consumer yang dimaksud disini adalah GUI yang dirancang dengan menggunakan Microsoft Visual Studio 2013. Supaya GUI dapat mengambil pesan dari messaging server, maka diberikan algoritma pada seperti berikut:

|  |
| --- |
| ConnectionFactory factory**;**  **using** **(**StreamReader r **=** **new** StreamReader**(**"config1.json"**))**  **{**  string json **=** r**.**ReadToEnd**();**  Config config **=** JsonConvert**.**DeserializeObject**<**Config**>(**json**);**  factory **=** **new** ConnectionFactory**{**Uri **=** config**.**url**};**  **}**  factory**.**Protocol **=** Protocols**.**DefaultProtocol**;**  factory**.**Port **=** AmqpTcpEndpoint**.**UseDefaultPort**;**  **using** **(**var connection **=** factory**.**CreateConnection**())**  **using** **(**var channel **=** connection**.**CreateModel**())**  **{**  channel**.**QueueDeclare**(**queue**:** "ecn"**,**  durable**:** **true,**  exclusive**:** **false,**  autoDelete**:** **false,**  arguments**:** **null);**  channel**.**QueueBind**(**queue**:** "ecn"**,**  exchange**:** "amq.topic"**,**  routingKey**:** "amq.topic.ecn"  **);**  Console**.**WriteLine**(**"Queue Declare Emergency GUI"**);**  var consumer **=** **new** EventingBasicConsumer**(**channel**);**  consumer**.**Received **+=** **(**model**,** ea**)** **=>**  **{**  var body **=** ea**.**Body**;**  var message **=** Encoding**.**UTF8**.**GetString**(**body**);**  Console**.**WriteLine**(**" [x] Received {0}"**,** message**);**  Console**.**WriteLine**(**" ///////////////////////////////////////////////////////////////////////////////////////////////////"**);**  data **=** message**;**  **try**  **{**  data accelReport **=** JsonConvert**.**DeserializeObject**<**data**>(**message**);**  Console**.**WriteLine**(**"{0} {1} {2} {3} {4}"**,** accelReport**.**geojson**.**geometry**.**coordinates**[**0**],** accelReport**.**geojson**.**geometry**.**coordinates**[**1**],** accelReport**.**accelerations**[**0**].**x**,** accelReport**.**accelerations**[**0**].**y**,** accelReport**.**accelerations**[**0**].**z**);**  **for** **(**int i **=** 0 **;** i **<** 20**;** i**++)**  **{**  perfChart**.**AddValue**((**decimal**)**int**.**Parse**(**accelReport**.**accelerations**[**i**].**x**));**  perfChart1**.**AddValue**((**decimal**)**int**.**Parse**(**accelReport**.**accelerations**[**i**].**y**));**  perfChart2**.**AddValue**((**decimal**)**int**.**Parse**(**accelReport**.**accelerations**[**i**].**z**));**  **}**  gMapControl1**.**Position **=** **new** GMap**.**NET**.**PointLatLng**(**double**.**Parse**(**accelReport**.**geojson**.**geometry**.**coordinates**[**0**]),** double**.**Parse**(**accelReport**.**geojson**.**geometry**.**coordinates**[**1**]));**  **}**  **catch** **(**Exception ex**)**  **{**  Console**.**WriteLine**(**"ERROR: {0}"**,** ex**);**  **}**  **};**  channel**.**BasicConsume**(**queue**:** "ecn"**,** //"emergency\_gui",  noAck**:** **true,**  consumer**:** consumer**);**  Console**.**WriteLine**(**"Already BasicConsume"**);**  **}** |

Dalam algoritma tersebut, terdapat syntax yang bertugas membaca file json dalam suatu directory yang serupa dengan directory GUI. Setelah membaca file json tersebut, akan diambil isi dari file yang bertugas untuk mengakses sebuah messaging server.

|  |
| --- |
| **using** **(**StreamReader r **=** **new** StreamReader**(**"config1.json"**))**  **{**  string json **=** r**.**ReadToEnd**();**  Config config **=** JsonConvert**.**DeserializeObject**<**Config**>(**json**);**  factory **=** **new** ConnectionFactory**{**Uri **=** config**.**url**};**  **}** |

Kemudian dilakukan akses pada queue yang terdapat pada messaging server.

|  |
| --- |
| **using** **(**var connection **=** factory**.**CreateConnection**())**  **using** **(**var channel **=** connection**.**CreateModel**())**  **{**  channel**.**QueueDeclare**(**queue**:** "ecn"**,**  durable**:** **true,**  exclusive**:** **false,**  autoDelete**:** **false,**  arguments**:** **null);**  channel**.**QueueBind**(**queue**:** "ecn"**,**  exchange**:** "amq.topic"**,**  routingKey**:** "amq.topic.ecn"  **);** |

Setelah itu, dilakukan proses consume data pada queue yang berkaitan.

|  |
| --- |
| var consumer **=** **new** EventingBasicConsumer**(**channel**);**  consumer**.**Received **+=** **(**model**,** ea**)** **=>**  **{**  var body **=** ea**.**Body**;**  var message **=** Encoding**.**UTF8**.**GetString**(**body**);**  **…**  **}** |

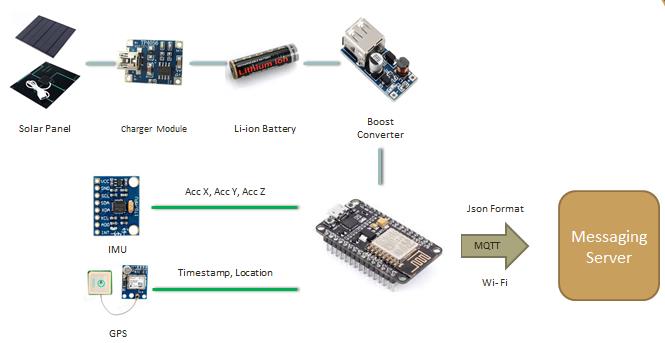
Kemudian dilakukan parsing dengan menambahakn syntax berikut ini:

|  |
| --- |
| data accelReport **=** JsonConvert**.**DeserializeObject**<**data**>(**message**);** |

Data merupakan sebuah tipe data object dengan deklarasi seperti ini:

|  |
| --- |
| class data  {  public string pointTime;  public string timeZone;  public string interval;  public geojson geojson;  public acc[] accelerations;  };  class geojson  {  public string type;  public geometry geometry;  public prop property;  };  class geometry  {  public string type;  public string[] coordinates;  };  class prop  {  public string name;  };  class acc  {  public string x;  public string y;  public string z;  };  } |

## Sensor Earthquake Catcher Network



Gambar Diagram Blok Sensor ECN

Pada sistem yang kita buat, Sensor ECN berfungsi untuk membaca data seismik, timestamp, dan lokasi dari sensor tersebut. Seperti yang sudah dijabarkan pada dokumen-dokumen sebelumnya, sensor kami memiliki beberapa spesifikasi yang harus dipenuhi yaitu harga sensor yang jauh lebih rendah dari harga seismometer(kisaran 1 juta), peforma sensor yang tidak terganggu dengan kondisi lingkungan sekitarnya, sensor dapat bekerja selama 24 jam tanpa henti, daya yang digunakan sensor rendah (<1W), dan akurasi dari pembacaan sensor getaran tinggi (ADC 14 bit).

Desain sensor ECN terdiri dari tiga bagian, yaitu modul utama, modul charge controller, dan modem Wi-Fi untuk komunikasi. Modul utama berfungsi untuk membaca data-data yang diperlukan seperti data getaran (percepatan), Timestamp, dan Lokasi dari sensor dan mengirimkannya menggunakan protokol MQTT ke Messaging server (cloudAMQP) dengan menggunakan Wi-Fi, Modul Charge Controller berfungsi sebagai penyedia sumber daya yang terus-menerus bekerja untuk modul utama, sedangkan modem Wi-Fi digunakan untuk menyediakan Wi-Fi access point untuk modul utama yang bekerja secara terus menerus.

2.1.1 Main Module

2.1.1.1 Elektrikal

Terdapat tiga komponen utama pada main module, yaitu NodeMCU v2, MPU6050, dan GPS Ublox Neo 6M.



Gambar NodeMCU v2

NodeMCU merupakan mikrokontroller yang sudah built-in modul Wi-Fi sehingga tidak diperlukan modul Wi-Fi tambahan. NodeMCU digunakan sebagai mikrokontroler yang bertugas untuk melakukan proses sampling data percepatan sebanyak 20 kali selama 0,5 detik, membaca data lokasi dan timestamp yang diberikan oleh GPS, lalu mengirimkan data-data tersebut ke server dengan menggunakan protokol MQTT melalui jaringan Wi-Fi.



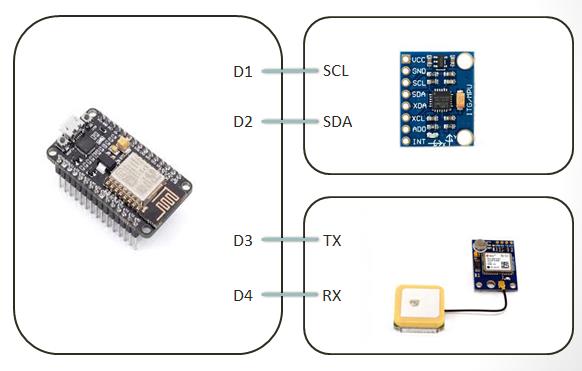
Gambar GPS Ublox Neo 6M

Untuk mendapatkan lokasi sensor dan data waktu yang dapat sinkron dengan sensor-sensor yang lain, digunakan modul GPS Ublox Neo 6M. Modul GPS ini terus menerus mendapat informasi dari satelit dan informasi-informasi tersebut dapat diperoleh NodeMCU dengan menggunakan komunikasi serial. Gelombang yang diterima oleh antena GPS ini memiliki daya tembus yang rendah sehingga modul GPS ini harus diletakkan di tempat yang tidak terhalang oleh tembok/atap. Data–data yang dapat diperoleh dari modul GPS ini adalah data latitude dan longitude serta data-data waktu yang tersinkronisasi seperti detik, menit, jam, tanggal, bulan, dan tahun



Gambar Sensor IMU

Modul yang digunakan untuk mendapatkan data getaran adalah modul MPU6050. Pada modul MPU6050 ini terdapat accelerometer sehingga dapat diperoleh data percepatan dari getaran. Komunikasi yang digunakan modul MPU6050 ini adalah I2C. Inisisasi pembacaan data dari MPU6050 dilakukan dengan cara wake-up MPU6050 atau dengan menuliskan bit 0 ke register 0x6B. Setelah dilakukan inisiasi, data percepatan x, y, dan z dapat diperoleh dengan membaca register 0x3B sampai 0x3F.

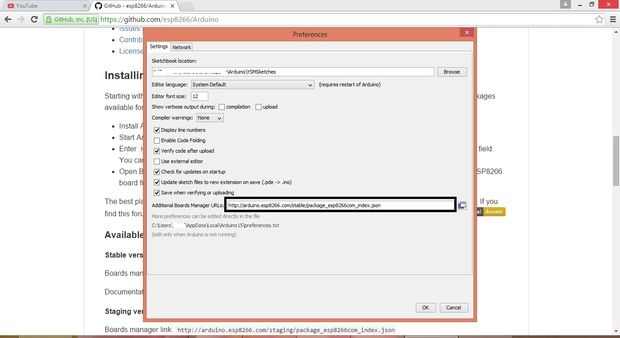


Gambar Skematik Modul Utama Sensor ECN

Untuk interkoneksi antar komponen dapat dilihat pada gambar diatas. Pin SCL pada MPU6050 dihubungkan dengan pin D1 pada NodeMCU dan pin SDA pada MPU6050 dihubungkan dengan pin D2 pada NodeMCU agar komunikasi I2C dapat dijalankan. Sedangkan pin Tx pada GPS dihubungkan dengan pin D3 pada NodeMCU dan pin Rx pada GPS dihubungkan dengan pin D4 pada NodeMCU. Pin D3 dan D4 merupakan pin yang dibuat menjadi pin Rx dan Tx dengan menggunakan software serial. Tidak digunakannya pin Rx dan Tx yang sudah ada pada NodeMCU karena pin Rx dan Tx tersebut sudah digunakan untuk serial monitor sehingga proses debugging lebih mudah dilakukan.

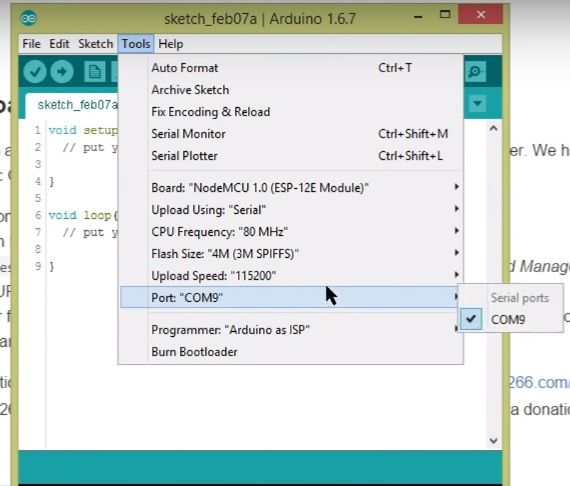
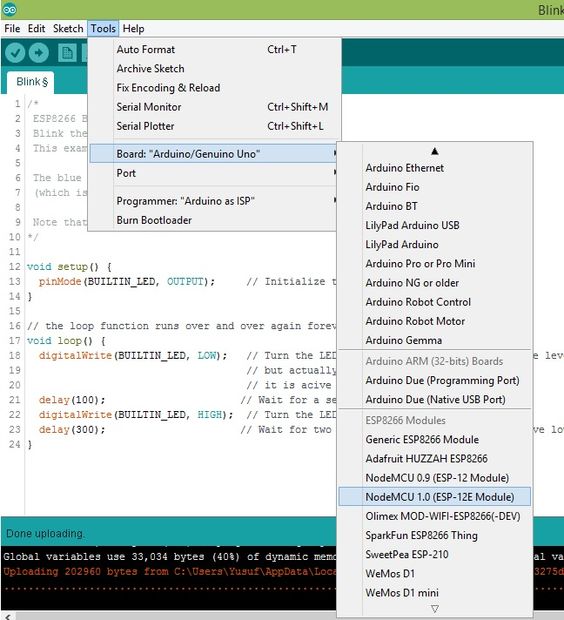
2.1.1.2 Algoritma

Digunakan IDE Arduino untuk mempermudah proses debugging dan upload program ke NodeMCU. Langkah pertama yang dilakukan adalah pada bagian preferences pada IDE Arduino, kolom Additional Board Manager URLs diisi dengan <http://arduino.esp8266.com/stable/package_esp8266com_index.json>



Gambar Tampilan untuk Menambahkan Board pada IDE Arduino

Setelah itu IDE Arduino harus di-restart. Setelah IDE Arduino dibuka kembali, board NodeMCU sudah terinstall pada IDE Arduino. Langkah selanjutnya yang dilakukan adalah memilih board pada bagian Tools -> Board.



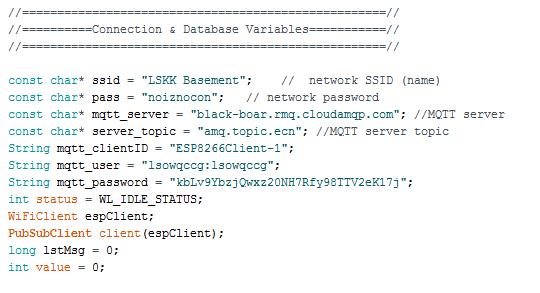
Gambar Tampilan untuk Setting Board dan Port

Setelah itu pilih Port pada bagian Tools -> Port. Setelah langkah-langkah tersebut dilakukan, proses kompilasi dan upload program ke NodeMCU dapat dilakukan pada IDE Arduino.



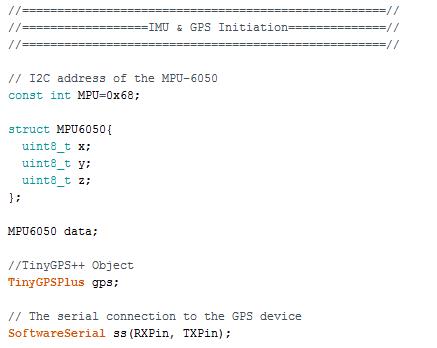
Gambar Potongan Source Code Include Library & Deklarasi Konstanta

Pada algoritma awal, dilakukan beberapa inisiasi seperti include library dan pendefinisian konstanta. Pada sensor ECN ini digunakan 5 library yaitu library ESP8266Wifi untuk koneksi Wi-Fi pada NodeMCU, PubSubClient untuk protokol MQTT, SoftwareSerial untuk deklarasi pin GPIO sebagai pin Serial tambahan yang digunakan untuk modul GPS, Wire untuk komunikasi I2C yang digunakan untuk memperoleh data dari MPU6050, dan Library TinyGPS++ untuk pembacaan data yang diperoleh dari GPS. Selain itu dilakukan juga deklarasi pin yang digunakan untuk komunikasi I2C dan Serial serta deklarasi register-register pada MPU6050. Konstanta IMU\_RES merupakan resolusi data percepatan pada sensor IMU, diberi nilai 1 karena data yang ingin dikirimkan memiliki tipe data integer pembacaan ADC pada IMU. Konstanta GPSBaud merupakan baudrate yang digunakan untuk berkomunikasi serial dengan modul GPS. Konstanta SendPeriod merupakan periode pengiriman message.



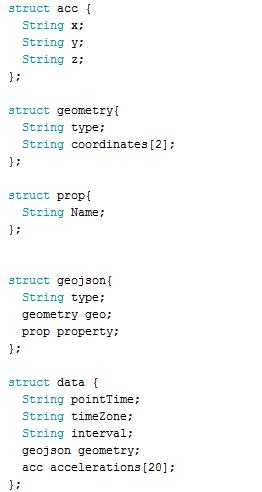
Gambar Potongan Source Code Deklarasi Variabel Koneksi dan MQTT

Untuk melakukan pengiriman diperlukan beberapa variabel yang harus digunakan. Untuk melakukan koneksi Wi-Fi dengan menggunakan NodeMCU hanya diperlukan dua variabel yaitu ssid dan pass yang bertipe array of constant character. Dua variabel ini menandakan access point mana yang ingin digunakan oleh NodeMCU. Sedangkan untuk pengaturan server MQTT digunakan variabel mqtt\_server, server\_topic, mqtt\_clientID, mqtt\_user, dan mqtt\_password. Variabel mqtt\_server digunakan untuk mendeklarasikan server manayang ingin digunakan pada CloudAMQP. Variabel server\_topic digunakan untuk mendeklarasikan topic mana yang ingin dikirimkan message pada server. Format dari variabel ini adalah “x.t” dimana “x” merupakan nama exchange yang digunakan dan “t” merupakan nama topic yang digunakan pada CloudAMQP. Sedangkan variabel mqtt\_clientID digunakan untuk mendeklarasikan nama client ketika melakukan koneksi ke server ClousAMQP. Nama yang digunakan boleh bebas asal tidak sama antara satu sensor dengan sensor yang lainnya. Variabel mqtt\_user dan mqtt\_password digunakan untuk mendeklarasikan username dari sensor beserta passwordnya. Format username yang digunakan adalah “v.u”, dimana “v” merupakan nama vhost yang digunakan dan “u” merupakan nama username yang digunakan. Dalam kasus dengan menggunakan CloudAMQP ini nama vhost dan username yang digunakan sama. Sedangkan espClient merupakan objek yang digunakan pada fungsi-fungsi pada library ESP8266WiFi dan objek PubSubClient merupakan hasil dari fungsi client pada library PubSubClient dengan input objek WiFiClient espClient.



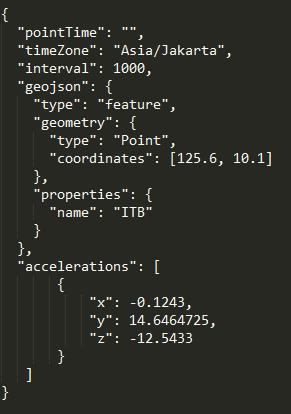
Gambar Potongan Source Code Deklarasi Variabel IMU

Pada bagian dari potongan source code yang digunakan diatas, dilakukan deklarasi beberapa variabel yang berhubungan dengan proses pembacaan data dari sensor IMU dan GPS. Konstanta MPU mengindikasikan alamat I2C dari MPU6050. Sedangkan struct MPU6050 dibuat untuk mempermudah parsing data ketika melakukan pembacaan dari sensor IMU. Lalu dilakukan juga inisiasi objek TinyGPSPlus dan SoftwareSerial yang digunakan pada library TinyGPS++ untuk proses pembacaan data yang diterima dari GPS.

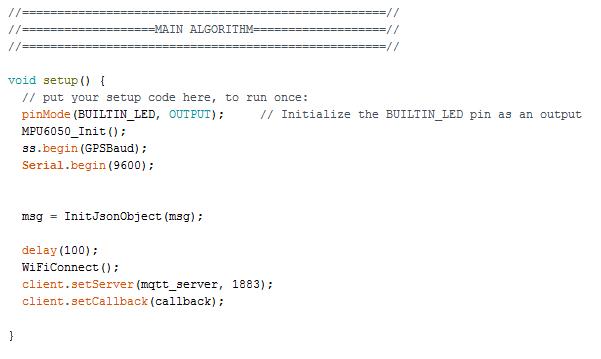


Gambar Potongan Source Code Deklarasi Objek Json

Dilakukan juga deklarasi objek yang berkolerasi dengan format message json yang ingin dikirimkan ke server. Format Json yang digunakan dapat dilihat pada gambar dibawah ini.

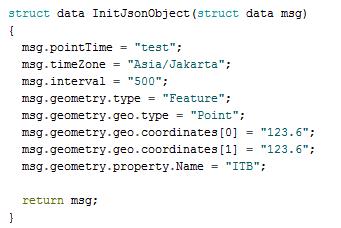


Gambar Format Message Json yang Digunakan



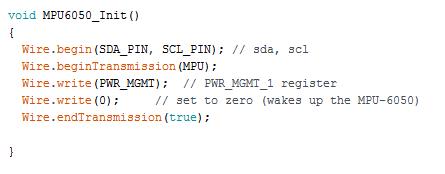
Gambar Potongan Source Code Setup

Pada bagian setup Arduino, dilakukan beberapa hal. Dilakukan deklarasi port built in LED sebagai port output. Setelah itu dilakukan inisiasi proses pembacaan data dari MPU6050 dan deklarasi baudrate pada port serial tambahan dan serial pada NodeMCU, dan deklarasi nilai dari setiap objek Json. Setelah itu dilakukan proses inisiasi koneksi Wi-Fi dan pengesetan server dan port pada protokol MQTT. Fungsi setCallback digunakan ketika device men-subscribe message dari server, karena sensor ECN tidak men-subscribe apapun dari server sehingga fungsi callback dapat dihiraukan.



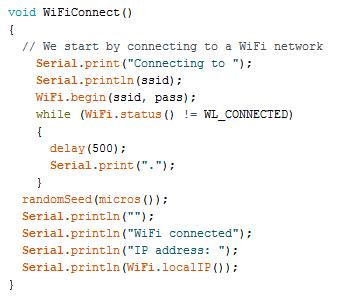
Gambar Potongan Source Code Fungsi InitJsonObject

Dapat dilihat dari gambar diatas, fungsi InitJsonObject memiliki input objek Json dan output objek Json. Pada fungsi ini dilakukan deklarasi awal beberapa variabel dari objek tersebut. Yang dideklarasikan antara lain zona waktu yang digunakan, interval pengiriman, dan lokasi dari sensor. Latitude dan Longitude perlu dideklarasikan asal terlebih dahulu untuk menghindari error pada fungsi pembentukan string dengan format Json.



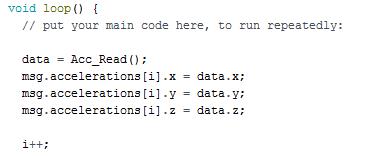
Gambar Potongan Source Code Fungsi MPU6050\_Init

Pada fungsi MPU6050\_Init dilakukan deklarasi port SDA dan SCL yang digunakan pada NodeMCU. Setelah itu dimulai transmisi pada alamat register I2C MPU6050. Lalu dilakukan penulisan bit 0 pada register 0x6B. Lalu koneksi I2C antara NodeMCU dengan MPU6050 dihentikan.



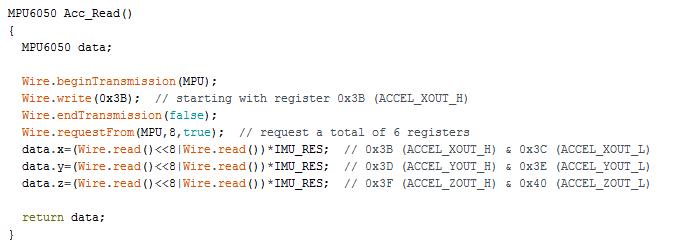
Gambar Potongan Source Code Fungsi WiFiConnect

Pada fungsi WiFiConnect dilakukan proses inisiasi koneksi antara NodeMCU dengan Access Point. Fungsi ini akan terus menerus berjalan sampai ada Access Point yang terhubung. Status dari proses ini dapat dilihat pada serial monitor pada IDE Arduino.



Gambar Potongan Source Code Loop

Gambar diatas merupakan implementasi algoritma loop dari sensor ECN. Hal yang pertama kali dilakukan pada bagian loop adalah pembacaan data percepatan dari IMU dengan menggunakan fungsi Acc\_Read. Jumlah sampling yang diambil berjumlah 20 dengan delay selama 25 ms setiap proses sampling dilakukan. Ketika proses sampling dilakukan, selalu dilakukan pengecekan apakah server masih terhubung dengan sensor ECN. Ketika sudah tidak terhubung maka proses sampling akan berhenti dan akan dilanjutkan untuk melakukan koneksi ulang ke server dengan menggunakan fungsi reconnect\_server. Ketika proses sampling telah selesai atau i==20, sensorECN akan mengirimkan message ke server dengan menggunakan fungsi publish pada library PubSubClient. Sebelum dilakukan pengiriman, dilakukan dulu pembentukan format Json dari objek Json yang telah dibuat pada sebelumnya (msg) dengan menggunakan fungsi JsonToString. Setelah itu string hasil pembentukan format Json tersebut diubah menjadi array of character dengan menggunakan fungsi bawaan dari Arduino toCharArray.



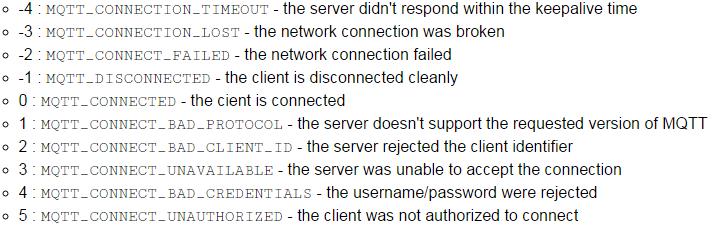
Gambar Potongan Source Code Fungsi Acc\_Read

Pada fungsi Acc\_Read dilakukan beberapa proses dalam pembacaan data percepatan dengan menggunakan MPU6050. Hal pertama yang dilakukan adalah memulai transmisi komunikasi I2C ke alamat I2C dari MPU6050. Setelah itu dilakukan pembacaaan dari register 0x3B sampai 0x40 dan disimpan pada objek data. Dapat dilihat karena satu register hanya dapat menyimpan 8 bit sedangkan pembacaan dari MPU6050 menggunakan 16 bit ADC sehingga untuk setiap salah satu axis percepatan diwakili dengan dua register pada MPU6050. Sehingga diperlukan proses shift left sebanyak 8 kali pada register yang menyimpan pembacaan MSB dari ADC MPU6050.

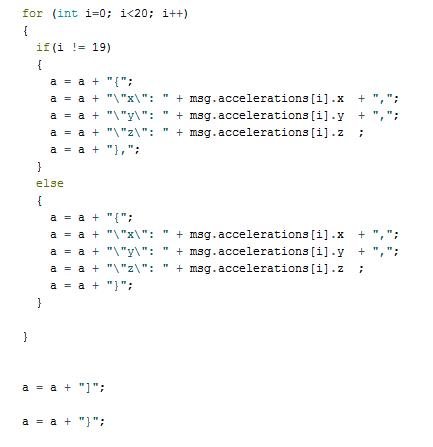
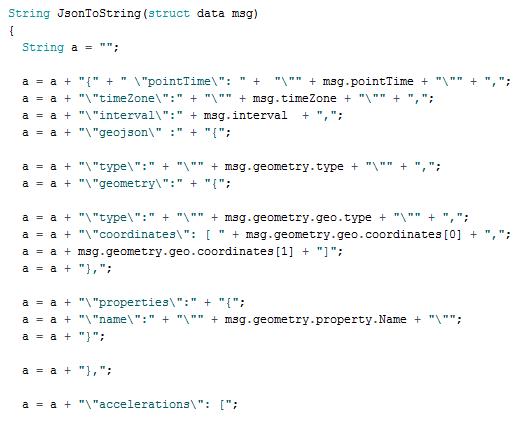


Gambar Potongan Source Code Fungsi reconnect\_server

Pada fungsi reconnect\_server dilakukan proses inisasi koneksi ulang lagi ke messaging server (CloudAMQP). Proses ini terus berlangsung sampai NodeMCU dapat terkoneksi ke server lagi. Status dari proses reconnect server ini dapat dilihar pada serial monitor. Penyebab error dalam percobaan koneksi ke server dapat dilihat dari nilai rc-nya. Informasi mengenai alasan error-nya sangat berguna untuk melakukan proses debugging. Keterangan mengenai nilai rc dapat dilihat pada gambar berikut.



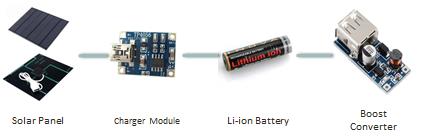
Gambar Informasi Mengenai Nilai RC dan Penjelasannya



Gambar Potongan Source Code Fungsi JsonToString

Fungsi JsonToString memiliki input objek Json yang sudah dideklarasikan sebelumnya dengan output dengan tipe data string. Tidak digunakannya library untuk encode dan decode string format Json yang tersedia dikarenakan tingkat kerumitan format Json yang kita gunakan tidak dapat di-handle oleh fungsi-fungsi tersebut, oleh karena itu dibuat fungsi encode format Json ini. Langkah yang dilakukan sangat sederhana, yaitu hanya menyusun string sesuai dengan format Json sehingga terbentuk string yang bersesuaian dengan objek Json yang kita gunakan. Dilakukan proses penambahan string a sesuai dengan format Json yang diinginkan. Perhatikan bahwa “\”” digunakan untuk mencetak string petik awal atau akhir.

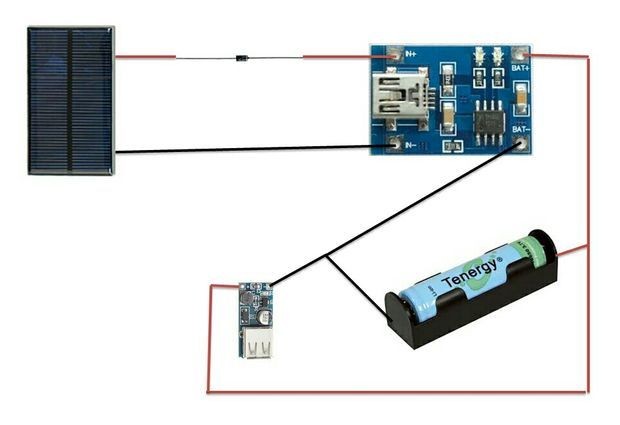
2.1.1 Charge Controller Module



Gambar Diagram Blok Charge Controller Module

Untuk membuat sensor ECN dapat berjalan secara terus menerus diperlukannya sumber daya yang dapat diperbaharui untuk menyuplai sensor ECN. Digunakan solar panel karena mudah untuk mengaksesnya serta cukup sederhana dalam proses implementasinya serta harga yang tidak terlalu mahal.

Digunakan solar panel dengan keluaran 5 Volt dan daya 4W sehingga solar panel memiliki kemampuan untuk menyuplai arus yang diperlukan NodeMCU (0,2A) dan charging baterai (0,6A) agar baterai dapat menyuplai sensor pada malam hari. Baterai yang digunakan untuk menyimpan daya adalah 2 baterai Li-Ion dengan kapasitas masing-masing adalah 2200 mAh. Pemilihan Li-Ion sendiri karena lebih tidak mudah rusaknya baterai ketika digunakan jika dibandingkan dengan baterai-baterai yang dapat di-charge lainnya seperti NiMH dan NiCD. Untuk mengatur tingkat charging baterai, digunakan modul Li-Ion battery charge module TP4056. Agar stabilnya tegangan yang diberikan oleh charge module ini, diperlukan modul boost converter. Digunakan modul boost converter dengan range input dari 0,9V – 5V dan arus keluaran antara 0,2 – 0,3 A. Untuk koneksi antar modulnya dapat dilihat pada gambar dibawah ini.



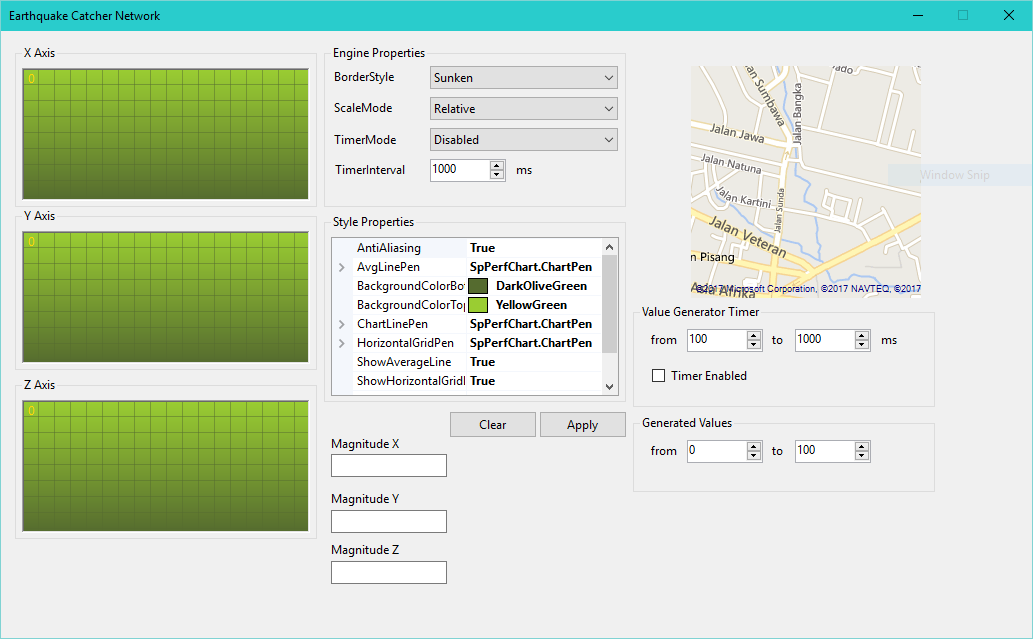
Gambar Skematik Modul Charge Controller

**LAMPIRAN**

|  |
| --- |
| /\*  \* Earthquake Catcher Network v1.0  \* Capstone Design by  \* Christoporus Deo Putratama  \* Kevin Shidqi  \* Bramantio Yuwono  \*  \* Read seismic waves using IMU sensor  \* Read location and exact time using GPS  \* Sending those data using Wi-Fi using MQTT  \*  \*/  //====================================================//  //=================Library & Constant=================//  //====================================================//  #include <ESP8266WiFi.h>  #include <PubSubClient.h>  #include <SoftwareSerial.h>  #include <Wire.h>  #include <TinyGPS++.h>  #define SDA\_PIN D2  #define SCL\_PIN D1  #define PWR\_MGMT 0x6B  #define IMU\_RES 1  #define RXPin D3  #define TXPin D4  #define GPSBaud 9600  #define SendPeriod 500 //in ms  //====================================================//  //==========Connection & Database Variables===========//  //====================================================//  const char**\*** ssid **=** "LSKK Basement"**;** // network SSID (name)  const char**\*** pass **=** "noiznocon"**;** // network password  const char**\*** mqtt\_server **=** "black-boar.rmq.cloudamqp.com"**;** //MQTT server  const char**\*** server\_topic **=** "amq.topic.ecn"**;** //MQTT server topic  String mqtt\_clientID **=** "ESP8266Client-1"**;**  String mqtt\_user **=** "lsowqccg:lsowqccg"**;**  String mqtt\_password **=** "kbLv9YbzjQwxz20NH7Rfy98TTV2eK17j"**;**  int status **=** WL\_IDLE\_STATUS**;**  WiFiClient espClient**;**  PubSubClient client**(**espClient**);**  long lstMsg **=** 0**;**  int value **=** 0**;**  //====================================================//  //==================IMU & GPS Initiation==============//  //====================================================//  // I2C address of the MPU-6050  const int MPU**=**0x68**;**  struct MPU6050**{**  uint8\_t x**;**  uint8\_t y**;**  uint8\_t z**;**  **};**  MPU6050 data**;**  //TinyGPS++ Object  TinyGPSPlus gps**;**  // The serial connection to the GPS device  SoftwareSerial ss**(**RXPin**,** TXPin**);**  //===================================================//  //===================JSON OBJECT=====================//  //===================================================//  struct acc **{**  String x**;**  String y**;**  String z**;**  **};**  struct geometry**{**  String type**;**  String coordinates**[**2**];**  **};**  struct prop**{**  String Name**;**  **};**  struct geojson**{**  String type**;**  geometry geo**;**  prop property**;**  **};**  struct data **{**  String pointTime**;**  String timeZone**;**  String interval**;**  geojson geometry**;**  acc accelerations**[**20**];**  **};**  struct data msg**;**  struct data InitJsonObject**(**struct data msg**);**  String JsonToString**(**struct data msg**);**  int i **=** 0**;**  //====================================================//  //===================MAIN ALGORITHM===================//  //====================================================//  void setup**()** **{**  // put your setup code here, to run once:  pinMode**(**BUILTIN\_LED**,** OUTPUT**);** // Initialize the BUILTIN\_LED pin as an output  MPU6050\_Init**();**  ss**.**begin**(**GPSBaud**);**  Serial**.**begin**(**9600**);**  msg **=** InitJsonObject**(**msg**);**    delay**(**100**);**  WiFiConnect**();**  client**.**setServer**(**mqtt\_server**,** 1883**);**  client**.**setCallback**(**callback**);**    **}**  void loop**()** **{**  // put your main code here, to run repeatedly:    data **=** Acc\_Read**();**  msg**.**accelerations**[**i**].**x **=** data**.**x**;**  msg**.**accelerations**[**i**].**y **=** data**.**y**;**  msg**.**accelerations**[**i**].**z **=** data**.**z**;**    i**++;**  **if** **(!**client**.**connected**())** **{**  reconnect\_server**();**  **}**  client**.**loop**();**  long now **=** millis**();**  //if (now - lstMsg > SendPeriod)  **if(**i**==**20**)**  **{**  i **=** 0**;**  **while** **(**ss**.**available**()** **>** 0**)**  **if** **(**gps**.**encode**(**ss**.**read**()))**  displayInfo**();**  **else**  Serial**.**println**(**"INVALID"**);**    //lstMsg = now;  String message **=** JsonToString**(**msg**);**  //msg.printTo(message);  char message\_t**[**800**];**  message**.**toCharArray**(**message\_t**,**800**);**  //publish sensor data to MQTT broker  bool test **=** client**.**publish**(**server\_topic**,** message\_t**);**  **if(**test**)**  Serial**.**println**(**"publish success"**);**  **}**  delay**(**25**);**  **}**  //====================================================//  //======Wi-Fi Connection & MQTT Function Procedure====//  //====================================================//  void WiFiConnect**()**  **{**  // We start by connecting to a WiFi network  Serial**.**print**(**"Connecting to "**);**  Serial**.**println**(**ssid**);**  WiFi**.**begin**(**ssid**,** pass**);**  **while** **(**WiFi**.**status**()** **!=** WL\_CONNECTED**)**  **{**  delay**(**500**);**  Serial**.**print**(**"."**);**  **}**  randomSeed**(**micros**());**  Serial**.**println**(**""**);**  Serial**.**println**(**"WiFi connected"**);**  Serial**.**println**(**"IP address: "**);**  Serial**.**println**(**WiFi**.**localIP**());**  **}**  void reconnect\_server**()** **{**  // Loop until we're reconnected  **while** **(!**client**.**connected**())**  **{**  Serial**.**print**(**"Attempting MQTT connection..."**);**  // Create a random client ID  String clientId **=** "ESP8266Client-"**;**  clientId **+=** String**(**random**(**0xffff**),** HEX**);**  // Attempt to connect  //if you MQTT broker has clientID,username and password  //please change following line to if (client.connect(clientId,userName,passWord))  **if** **(**client**.**connect**(**mqtt\_clientID**.**c\_str**(),** mqtt\_user**.**c\_str**(),**mqtt\_password**.**c\_str**()))**  **{**  Serial**.**println**(**"connected"**);**  **}** **else** **{**  Serial**.**print**(**"failed, rc="**);**  Serial**.**print**(**client**.**state**());**  Serial**.**println**(**" try again in 5 seconds"**);**  // Wait 6 seconds before retrying  delay**(**6000**);**  **}**  **}**  **}** //end reconnect()  void callback**(**char**\*** topic**,** byte**\*** payload**,** unsigned int length**)**  **{**  Serial**.**print**(**"Message arrived ["**);**  Serial**.**print**(**topic**);**  Serial**.**print**(**"] "**);**  **for** **(**int i **=** 0**;** i **<** length**;** i**++)** **{**  Serial**.**print**((**char**)**payload**[**i**]);**  **}**  Serial**.**println**();**  // Switch on the LED if an 1 was received as first character  **if** **((**char**)**payload**[**0**]** **==** '1'**)** **{**  digitalWrite**(**BUILTIN\_LED**,** LOW**);** // Turn the LED on (Note that LOW is the voltage level  // but actually the LED is on; this is because  // it is acive low on the ESP-01)  **}** **else** **{**  digitalWrite**(**BUILTIN\_LED**,** HIGH**);** // Turn the LED off by making the voltage HIGH  **}**  **}**  //====================================================//  //==============IMU & GPS Function Procedure==========//  //====================================================//  void MPU6050\_Init**()**  **{**  Wire**.**begin**(**SDA\_PIN**,** SCL\_PIN**);** // sda, scl  Wire**.**beginTransmission**(**MPU**);**  Wire**.**write**(**PWR\_MGMT**);** // PWR\_MGMT\_1 register  Wire**.**write**(**0**);** // set to zero (wakes up the MPU-6050)  Wire**.**endTransmission**(true);**  **}**  MPU6050 Acc\_Read**()**  **{**  MPU6050 data**;**    Wire**.**beginTransmission**(**MPU**);**  Wire**.**write**(**0x3B**);** // starting with register 0x3B (ACCEL\_XOUT\_H)  Wire**.**endTransmission**(false);**  Wire**.**requestFrom**(**MPU**,**8**,true);** // request a total of 6 registers  data**.**x**=(**Wire**.**read**()<<**8**|**Wire**.**read**())\***IMU\_RES**;** // 0x3B (ACCEL\_XOUT\_H) & 0x3C (ACCEL\_XOUT\_L)  data**.**y**=(**Wire**.**read**()<<**8**|**Wire**.**read**())\***IMU\_RES**;** // 0x3D (ACCEL\_YOUT\_H) & 0x3E (ACCEL\_YOUT\_L)  data**.**z**=(**Wire**.**read**()<<**8**|**Wire**.**read**())\***IMU\_RES**;** // 0x3F (ACCEL\_ZOUT\_H) & 0x40 (ACCEL\_ZOUT\_L)    **return** data**;**  **}**  void displayInfo**()**  **{**  **if(**gps**.**location**.**isValid**())**  **{**  msg**.**geometry**.**geo**.**coordinates**[**0**]** **=** String**(**gps**.**location**.**lat**(),**3**);**  msg**.**geometry**.**geo**.**coordinates**[**1**]** **=** String**(**gps**.**location**.**lng**(),**3**);**  Serial**.**println**(**gps**.**location**.**lat**(),**3**);**  Serial**.**println**(**gps**.**location**.**lng**(),**3**);**  **}**  **else**  **{**  Serial**.**println**(**"INVALID"**);**  **}**  **if(**gps**.**date**.**isValid**()** **&&** gps**.**time**.**isValid**())**  **{**  String YEAR **=** String**(**gps**.**date**.**year**());**  String MONTH **=** String**(**gps**.**date**.**month**());**  String DATE **=** String**(**gps**.**date**.**day**());**  String HOUR **=** String**(**gps**.**time**.**hour**());**  String MINUTE **=** String**(**gps**.**time**.**minute**());**  String SECOND **=** String**(**gps**.**time**.**second**());**  msg**.**pointTime **=** YEAR **+** "-" **+** MONTH **+** "-" **+** DATE **+** "T" **+** HOUR **+** ":" **+** MINUTE **+** ":"**+** SECOND **+** "Z"**;**  **}**  **else**  **{**  Serial**.**println**(**"INVALID"**);**  **}**  **}**  //====================================================//  //==================ENCODE JSON FUNCTION==============//  //====================================================//  struct data InitJsonObject**(**struct data msg**)**  **{**  msg**.**pointTime **=** "test"**;**  msg**.**timeZone **=** "Asia/Jakarta"**;**  msg**.**interval **=** "500"**;**  msg**.**geometry**.**type **=** "Feature"**;**  msg**.**geometry**.**geo**.**type **=** "Point"**;**  msg**.**geometry**.**geo**.**coordinates**[**0**]** **=** "123.6"**;**  msg**.**geometry**.**geo**.**coordinates**[**1**]** **=** "123.6"**;**  msg**.**geometry**.**property**.**Name **=** "ITB"**;**  **return** msg**;**  **}**  String JsonToString**(**struct data msg**)**  **{**  String a **=** ""**;**  a **=** a **+** "{" **+** " \"pointTime\": " **+** "\"" **+** msg**.**pointTime **+** "\"" **+** ","**;**  a **=** a **+** "\"timeZone\":" **+** "\"" **+** msg**.**timeZone **+** "\"" **+** ","**;**  a **=** a **+** "\"interval\":" **+** msg**.**interval **+** ","**;**  a **=** a **+** "\"geojson\" :" **+** "{"**;**  a **=** a **+** "\"type\":" **+** "\"" **+** msg**.**geometry**.**type **+** "\"" **+** ","**;**  a **=** a **+** "\"geometry\":" **+** "{"**;**  a **=** a **+** "\"type\":" **+** "\"" **+** msg**.**geometry**.**geo**.**type **+** "\"" **+** ","**;**  a **=** a **+** "\"coordinates\": [ " **+** msg**.**geometry**.**geo**.**coordinates**[**0**]** **+** ","**;**  a **=** a **+** msg**.**geometry**.**geo**.**coordinates**[**1**]** **+** "]"**;**  a **=** a **+** "},"**;**  a **=** a **+** "\"properties\":" **+** "{"**;**  a **=** a **+** "\"name\":" **+** "\"" **+** msg**.**geometry**.**property**.**Name **+** "\""**;**  a **=** a **+** "}"**;**  a **=** a **+** "},"**;**  a **=** a **+** "\"accelerations\": ["**;**  **for** **(**int i**=**0**;** i**<**20**;** i**++)**  **{**  **if(**i **!=** 19**)**  **{**  a **=** a **+** "{"**;**  a **=** a **+** "\"x\": " **+** msg**.**accelerations**[**i**].**x **+** ","**;**  a **=** a **+** "\"y\": " **+** msg**.**accelerations**[**i**].**y **+** ","**;**  a **=** a **+** "\"z\": " **+** msg**.**accelerations**[**i**].**z **;**  a **=** a **+** "},"**;**  **}**  **else**  **{**  a **=** a **+** "{"**;**  a **=** a **+** "\"x\": " **+** msg**.**accelerations**[**i**].**x **+** ","**;**  a **=** a **+** "\"y\": " **+** msg**.**accelerations**[**i**].**y **+** ","**;**  a **=** a **+** "\"z\": " **+** msg**.**accelerations**[**i**].**z **;**  a **=** a **+** "}"**;**  **}**    **}**      a **=** a **+** "]"**;**  a **=** a **+** "}"**;**      **return** a**;**  **}** |

## Perangkat Lunak Pengolah Data dan Antarmuka Grafis

### Antarmuka Pengguna Grafis



Tamplian ini adalah tampilan awal dari antarmuka pengguna grafis sistem detektor gempa ini. Hanya data dari satu sensor yang dimasukkan ke dalam proses yang ada dibalik tampilan ini. Data tersebut berupa percepatan ke arah x,y,dan z, serta data posisi yang ditampilkan melalui peta di pojok kanan atas. Sensor akan ditandai dengan pin berwarna biru di peta tersebut. Bila ada lebih dari satu sensor yang terhubung, maka akan ada banyak pin biru di peta. Tombol dan bar lainnya digunakan untuk melakukan pengaturan terhadap grafik data percepatan yang ditampilkan pada kotak hijau di bagian kiri. Selain itu, magnituda dari getaran pada masing-masing sumbu juga ditampilkan secara kuantitatif di bagian bawah.

Peta tersebut disediakan oleh Bing secara gratis, melalui proyek gMap.Net, yaitu paket peta dan gps yang khusus dibuat untuk platform .NET, termasuk didalamnya C#. Program akan menerima data lokasi dari masing-masing sensor berupa lintang dan bujur, dan program lalu akan membuat sebuah pin biru di peta yang lokasinya sesuai dengan lintang dan bujur yang diterima. Pin dapat dibuat pada peta dengan jumlah yang tidak terbatas. Pengguna juga dapat menggeser peta sehingga bagian lain dari bumi yang terlihat pada layar, dan juga melakukan pembesaran atau pengecilan.

Tampilan grafik yang merepresentasikan data percepatan yang dikirim sensor bisa diatur sebagai berikut. Tiga pilihan yang terletak pada bagian atas berfungsi untuk mengatur *engine* dari grafik. Sebagai contoh, ada yang mengatur kecepatan grafik bergeser ke kiri, ada yang mengatur pembesaran/pengecilan skala grafik secara otomatis, dan juga mengatur jenis dari perbatasan grafik, apakah absolut atau bergerak mengikuti nilai grafik yang sekarang.

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

using System.Threading;

using RabbitMQ.Client;

using RabbitMQ.Client.Events;

using GMap.NET;

using GMap.NET.WindowsForms;

using GMap.NET.WindowsForms.Markers;

using GMap.NET.MapProviders;

using Newtonsoft.Json;

using System.IO;

namespace SimplePerfChart

{

public partial class FrmTestingForm : Form

{

private object valueGenSync = new object();

private Random randGen = new Random();

//public static decimal data = 0;

private int valueGenFrom = -5;

private int valueGenTo = 5;

private int valueGenTimerFrom = 100;

private int valueGenTimerTo = 1000;

public static string data = "";

public struct accel

{

public decimal accel\_x;

public decimal accel\_y;

public decimal accel\_z;

}

public FrmTestingForm()

{

InitializeComponent();

this.Font = SystemInformation.MenuFont;

propGrid.SelectedObject = perfChart.PerfChartStyle;

// Apply default Properties

perfChart.TimerInterval = 1000;

// Populate DrowDown Boxes

foreach (String item in System.Enum.GetNames(typeof(Border3DStyle)))

{

cmbBxBorder.Items.Add(item);

}

foreach (String item in System.Enum.GetNames(typeof(SpPerfChart.ScaleMode)))

{

cmbBxScaleMode.Items.Add(item);

}

foreach (String item in System.Enum.GetNames(typeof(SpPerfChart.TimerMode)))

{

cmbBxTimerMode.Items.Add(item);

}

// Select default values

cmbBxTimerMode.SelectedItem = perfChart.TimerMode.ToString();

cmbBxScaleMode.SelectedItem = perfChart.ScaleMode.ToString();

cmbBxBorder.SelectedItem = perfChart.BorderStyle.ToString();

}

private void chkBxTimerEnabled\_CheckedChanged(object sender, EventArgs e)

{

if (chkBxTimerEnabled.Checked && !bgWrkTimer.IsBusy)

{

RunTimer();

}

}

private void RunTimer()

{

int waitFor = randGen.Next(valueGenTimerFrom, valueGenTimerTo);

bgWrkTimer.RunWorkerAsync(waitFor);

}

private void bgWrkTimer\_DoWork(object sender, DoWorkEventArgs e)

{

Thread.Sleep(Convert.ToInt32(e.Argument));

}

public class Config

{

public string host;

public string user;

public string vhost;

//public string port;

public string password;

}

private static void consume\_data()

{

//accel accel\_data;

//AccelerationReport accelerations = new AccelerationReport();

//int genValue = randGen.Next(valueGenFrom, valueGenTo);

}

private void bgWrkTimer\_RunWorkerCompleted(object sender, RunWorkerCompletedEventArgs e)

{

//consume\_data();

ConnectionFactory factory;

using (StreamReader r = new StreamReader("config1.json"))

{

string json = r.ReadToEnd();

Config config = JsonConvert.DeserializeObject<Config>(json);

factory = new ConnectionFactory();// { HostName = config.host, UserName = config.user, VirtualHost = config.vhost, Password = config.password };

//factory.Uri = "amqp://lsowqccg:kbLv9YbzjQwxz20NH7Rfy98TTV2eK17j@black-boar.rmq.cloudamqp.com/lsowqccg";

factory.Uri = "amqp://lsowqccg:kbLv9YbzjQwxz20NH7Rfy98TTV2eK17j@black-boar.rmq.cloudamqp.com/lsowqccg";

}

factory.Protocol = Protocols.DefaultProtocol;

factory.Port = AmqpTcpEndpoint.UseDefaultPort;

using (var connection = factory.CreateConnection())

using (var channel = connection.CreateModel())

{

channel.QueueDeclare(queue: "ecn", //"emergency\_gui",

durable: true,

exclusive: false,

autoDelete: false,

arguments: null);

channel.QueueBind(queue: "ecn", //"emergency\_gui",

exchange: "amq.topic",

routingKey: "amq.topic.ecn" //emergency"

);

Console.WriteLine("Queue Declare Emergency GUI");

var consumer = new EventingBasicConsumer(channel);

consumer.Received += (model, ea) =>

{

var body = ea.Body;

var message = Encoding.UTF8.GetString(body);

Console.WriteLine(" [x] Received {0}", message);

Console.WriteLine(" ///////////////////////////////////////////////////////////////////////////////////////////////////");

data = message;

//accel\_data = ParsingMessage(data);

try

{

//AccelerationReport accelReport = JsonConvert.DeserializeObject<AccelerationReport>(message);

data accelReport = JsonConvert.DeserializeObject<data>(message);

//Console.WriteLine("{0} {1} {2}", accel\_data.accel\_x, accel\_data.accel\_y, accel\_data.accel\_z);

//console.writeline("{0} {1} {2} {3}", accelreport.geometry.geometry.coordinates, accelreport.accelerations[0].x, accelreport.accelerations[0].y, accelreport.accelerations[0].z);

//perfchart.addvalue((decimal)accelreport.accelerations[0].x \* 1000);

//perfchart1.addvalue((decimal)accelreport.accelerations[0].y \* 1000);

//perfchart2.addvalue((decimal)accelreport.accelerations[0].z \* 1000);

//Console.WriteLine("{0} {1} {2} {3} {4}", accelReport.geometry.geometryData.coordinate.lon, accelReport.geometry.geometryData.coordinate.lng, accelReport.accelerations.x, accelReport.accelerations.y, accelReport.accelerations.z);

Console.WriteLine("{0} {1} {2} {3} {4}", accelReport.geojson.geometry.coordinates[0], accelReport.geojson.geometry.coordinates[1], accelReport.accelerations[0].x, accelReport.accelerations[0].y, accelReport.accelerations[0].z);

for (int i = 0 ; i < 20; i++)

{

perfChart.AddValue((decimal)int.Parse(accelReport.accelerations[i].x));

perfChart1.AddValue((decimal)int.Parse(accelReport.accelerations[i].y));

perfChart2.AddValue((decimal)int.Parse(accelReport.accelerations[i].z));

}

gMapControl1.Position = new GMap.NET.PointLatLng(double.Parse(accelReport.geojson.geometry.coordinates[0]), double.Parse(accelReport.geojson.geometry.coordinates[1]));

}

catch (Exception ex)

{

Console.WriteLine("ERROR: {0}", ex);

}

};

channel.BasicConsume(queue: "ecn", //"emergency\_gui",

noAck: true,

consumer: consumer);

Console.WriteLine("Already BasicConsume");

//Console.ReadLine();

}

if (chkBxTimerEnabled.Checked)

{

RunTimer();

}

}

private void cmbBxBorder\_SelectedIndexChanged(object sender, EventArgs e)

{

perfChart.BorderStyle = (Border3DStyle)Enum.Parse(

typeof(Border3DStyle), cmbBxBorder.SelectedItem.ToString()

);

}

private void cmbBxScaleMode\_SelectedIndexChanged(object sender, EventArgs e)

{

perfChart.ScaleMode = (SpPerfChart.ScaleMode)Enum.Parse(

typeof(SpPerfChart.ScaleMode), cmbBxScaleMode.SelectedItem.ToString()

);

}

private void cmbBxTimerMode\_SelectedIndexChanged(object sender, EventArgs e)

{

perfChart.TimerMode = (SpPerfChart.TimerMode)Enum.Parse(

typeof(SpPerfChart.TimerMode), cmbBxTimerMode.SelectedItem.ToString()

);

}

private void numUpDnTimerInterval\_ValueChanged(object sender, EventArgs e)

{

perfChart.TimerInterval = Convert.ToInt32(numUpDnTimerInterval.Value);

}

private void btnApply\_Click(object sender, EventArgs e)

{

valueGenFrom = Convert.ToInt32(numUpDnValFrom.Value);

valueGenTo = Convert.ToInt32(numUpDnValTo.Value);

if (valueGenTo < valueGenFrom)

{

valueGenTo = valueGenFrom;

numUpDnValTo.Value = valueGenTo;

}

valueGenTimerFrom = Convert.ToInt32(numUpDnFromInterval.Value);

valueGenTimerTo = Convert.ToInt32(numUpDnToInterval.Value);

if (valueGenTimerTo < valueGenTimerFrom)

{

valueGenTimerTo = valueGenTimerFrom;

numUpDnToInterval.Value = valueGenTimerTo;

}

}

private void btnClear\_Click(object sender, EventArgs e)

{

perfChart.Clear();

}

//public static void receive()

//{

// var factory = new ConnectionFactory() { HostName = "localhost" };

// using (var connection = factory.CreateConnection())

// using (var channel = connection.CreateModel())

// {

// //channel.ExchangeDeclare(exchange: "data", type: "fanout");

// channel.QueueDeclare(queue: "emergency\_gui",

// durable: false,

// exclusive: false,

// autoDelete: false,

// arguments: null);

// channel.QueueBind(queue: "emergency\_gui",

// exchange: "amq.topic",

// routingKey: "emergency");

// var consumer = new EventingBasicConsumer(channel);

// consumer.Received += (model, ea) =>

// {

// var body = ea.Body;

// var message = Encoding.UTF8.GetString(body);

// Console.WriteLine(" [x] Received {0}", message);

// //data = Convert.ToDecimal(message);

// };

// channel.BasicConsume(queue: "emergency\_gui",

// noAck: true,

// consumer: consumer);

// Console.WriteLine(" Press [enter] to exit.");

// Console.ReadLine();

// }

//}

private void FrmTestingForm\_Load(object sender, EventArgs e)

{

}

private void gMapControl1\_Load(object sender, EventArgs e)

{

gMapControl1.MapProvider = GMap.NET.MapProviders.BingMapProvider.Instance;

GMap.NET.GMaps.Instance.Mode = GMap.NET.AccessMode.ServerOnly;

gMapControl1.Position = new GMap.NET.PointLatLng(-6.9175, 107.6191);

gMapControl1.ShowCenter = false;

GMapOverlay markers = new GMapOverlay("markers");

GMapMarker marker = new GMarkerGoogle(

new PointLatLng(-6.890903, 107.610378),

GMarkerGoogleType.blue\_pushpin);

markers.Markers.Add(marker);

gMapControl1.Overlays.Add(markers);

}

private accel ParsingMessage(string messages)

{

accel accel\_data;

string data\_x, data\_y, data\_z;

char[] delimiters = { '<', '>' };

string[] parseMessage = messages.Split(delimiters);

data\_x = parseMessage[1];

data\_y = parseMessage[2];

data\_z = parseMessage[3];

accel\_data.accel\_x = (int)Math.Ceiling(float.Parse(data\_x) \* 1000);

accel\_data.accel\_y = (int)Math.Ceiling(float.Parse(data\_y) \* 1000);

accel\_data.accel\_z = (int)Math.Ceiling(float.Parse(data\_z) \* 1000);

return accel\_data;

}

private void perfChart\_Load(object sender, EventArgs e)

{

//receive();

}

}

class data

{

public string pointTime;

public string timeZone;

public string interval;

public geojson geojson;

public acc[] accelerations;

};

class geojson

{

public string type;

public geometry geometry;

public prop property;

};

class geometry

{

public string type;

public string[] coordinates;

};

class prop

{

public string name;

};

class acc

{

public string x;

public string y;

public string z;

};

}

### Pengolahan Data

#### Akuisisi

#### Pemrosesan

##### Master

SCmaster dirancang sebagai semacam microkernel atau mediator yang delegasi permintaan klien. Oleh karena itu adalah aplikasi utama yang bertanggung jawab untuk orkestrasi dari sistem terdistribusi. Untuk berpartisipasi dalam sistem terdistribusi klien perlu mengirim permintaan connect ke scmaster tersebut. Pada gilirannya master mengembalikan pesan pengakuan yang baik menginformasikan klien masuk atau penolakan. Jika permintaan connect berhasil pesan pengakuan akan memberikan klien dengan kelompok pesan yang tersedia dapat berlangganan. Selain itu, semua klien yang terkoneksi akan diberitahu tentang anggota baru bergabung. Dalam kasus master dikonfigurasi dengan database klien juga akan menerima langsung menindaklanjuti pesan yang memegang alamat database ini. Alamat dapat digunakan untuk mengambil data arsip nanti. Setelah sambungan sudah didirikan setiap pesan akan melewati master pertama di mana itu diproses sesuai dan kemudian diteruskan ke kelompok sasaran. Setelah klien dilakukan dengan memproses pesan putuskan akan dikirim ke master yang pada gilirannya memberitahu semua klien yang tersisa tentang meninggalkannya.

scmaster dapat dikonfigurasi dengan database untuk memastikan integritas sistem. Sebelum pesan didistribusikan oleh scmaster pesan ditulis ke database yang ditentukan. Dengan cara ini setiap pesan disimpan sebelum memasuki sistem. Dalam kasus kecelakaan semua informasi yang diperlukan dapat dipulihkan dari database. Saat ini, sopir ada untuk MySQL, PostgreSQL dan sqlite. Perhatikan bahwa scmaster dapat berjalan tanpa database tetapi kehilangan integritas data dalam melakukannya.

##### Autoloc

scautoloc adalah program SeisComP3 bertanggung jawab untuk secara otomatis mencari peristiwa seismik di dekat-real time. Ini biasanya berjalan sebagai daemon, terus membaca picks dan amplitudo dan memproses mereka secara real time. Sebuah modus offline juga tersedia. scautoloc membaca picks otomatis dan beberapa amplitudo terkait. Atas dasar itu mencoba untuk mengidentifikasi kombinasi picks yang sesuai dengan peristiwa seismik umum. Jika lokasi yang diproduksi memenuhi kriteria konsistensi tertentu, dilaporkan, yaitu diteruskan ke program lain yang mengambil asal sebagai masukan.

prosedur lokasi

Prosedur scautoloc untuk mengidentifikasi dan menemukan kejadian gempa pada dasarnya terdiri dari langkah-langkah berikut:

Pilih persiapan

Dalam scautoloc setiap pick masuk harus disertai dengan serangkaian tertentu amplitudo. Sejak di SeisComP3 model data amplitudo dan picks adalah objek yang independen, amplitudo ditambahkan sebagai atribut untuk picks sesuai mereka setelah diterima oleh scautoloc.

Pilih filtering

Setiap pick masuk disaring, yaitu akan diperiksa jika pick sudah usang dan jika set lengkap amplitudo terkait hadir sudah. Jika stasiun menghasilkan picks sangat sering, ini dianggap gangguan lebih mungkin dan mengakibatkan ambang batas SNR meningkat.

Asosiasi

Hal ini pertama kali mencoba untuk mengasosiasikan memilih masuk dengan asal-usul dikenal. Terutama untuk acara besar dengan lokasi yang stabil berdasarkan banyak picks sudah terkait, ini adalah cara yang lebih disukai untuk menangani memilih. Jika asosiasi berhasil, proses nukleasi dapat dilewati. Dalam keadaan tertentu picks keduanya berhubungan dan dimasukkan ke nucleator tersebut.

nukleasi

Jika hubungan langsung gagal, scautoloc mencoba untuk membuat asal baru dari unassociated, picks diterima sebelumnya dan lainnya. Proses ini disebut "nukleasi". scautoloc melakukan pencarian jaringan lebih dari ruang dan waktu, yang merupakan prosedur agak mahal karena membutuhkan banyak sumber daya baik dari segi CPU dan RAM. algoritma nukleasi tambahan akan tersedia di masa depan. grid adalah satu set diskrit -in poin sewenang-wenang principle- bahwa sampel bidang minat yang cukup padat. Dalam pencarian grid, masing-masing titik grid diambil sebagai hiposenter hipotetis untuk semua picks masuk. Setiap pick yang masuk kembali diproyeksikan dalam waktu untuk masing-masing titik grid, dengan asumsi bahwa itu adalah pertama-tiba "P" onset. Jika pick memang sesuai dengan "P" kedatangan peristiwa seismik, dan jika acara ini tercatat cukup banyak stasiun, kembali diproyeksikan pick baru akan mengelompok dengan picks sebelumnya dari acara yang sama. cluster akan terpadat sekitar waktu asal pada titik grid terdekat hiposenter tersebut. Pada prinsipnya, grid bisa begitu padat sehingga lokasi yang diperoleh dari pencarian grid dapat digunakan secara langsung. Namun, seperti RAM memori serta kecepatan CPU terbatas, hal ini tidak mungkin. Oleh karena itu, jika cluster diidentifikasi sebagai asal potensial, itu tidak berarti bahwa semua picks kontribusi sebenarnya sesuai dengan "P" kedatangan. Ini mungkin juga menjadi pertandingan kebetulan disebabkan oleh kekasaran grid atau kemungkinan kontaminasi oleh kebisingan dijemput. Oleh karena itu, program lokasi (LocSAT) dijalankan dalam rangka untuk mencoba lokasi dan menguji apakah set picks memang membentuk hiposenter konsisten. Jika memilih RMS residual terlalu besar, perbaikan diupayakan oleh tidak termasuk setiap picks kontribusi sekali untuk menguji apakah pengurangan RMS dapat dicapai. Jika asal baru memenuhi semua persyaratan, itu diterima sebagai lokasi acara seismik baru.

Titik-titik grid ditentukan dalam file teks "grid.txt". Default file dikirimkan dengan scautoloc mendefinisikan grid dengan poin secara global bahkan didistribusikan di permukaan, dan titik kedalaman terbatas wilayah yang dikenal kegempaan yang mendalam. Ini dapat dimodifikasi, tetapi tidak harus terdiri terlalu banyak titik-titik grid (> 3000, tergantung pada kecepatan CPU dan RAM). Lihat di bawah untuk rincian lebih lanjut tentang file jaringan.

Origin Fixing

Asal diproduksi atau diperbarui melalui asosiasi dan / atau nukleasi masih dapat terkontaminasi oleh fase keliru ditafsirkan sebagai "P" kedatangan. scautoloc mencoba untuk meningkatkan asal usulnya berdasarkan mis memilih SNR dan amplitudo. Pada langkah pengolahan ini, juga berusaha untuk mengaitkan fase yang menyelinap melalui selama upaya asosiasi pertama, misalnya karena lokasi awal adalah salah. Jika asal berisi sejumlah cukup kedatangan menganggap hasil lokasi cukup baik, scautoloc tambahan mencoba untuk mengasosiasikan picks sebagai fase sekunder seperti "pP". fase sekunder seperti hanya "lemah terkait", yaitu fase ini tidak digunakan untuk lokasi. Untuk analis, namun, hal ini berguna untuk memiliki kemungkinan "pP" fase telah ditetapkan.

Origin filtering

Proses ini melibatkan pemeriksaan konsistensi akhir / asal diperbarui baru dll Selama prosedur ini, asal-usul tidak diubah lagi.

##### Amplitude

SCamp mengukur beberapa jenis amplitudo dari data gelombang. Hal mendengarkan usul dan langkah-langkah amplitudo dalam waktu jendela ditentukan dari asal. Benda-benda amplitudo yang dihasilkan dikirim ke "AMPLITUDO" kelompok messaging. banyol adalah mitra dari scmag. Biasanya, semua amplitudo dihitung sekaligus dengan banyol dan kemudian dipublikasikan. Hanya sangat jarang amplitudo perlu menghitung ulang jika lokasi asal berubah secara signifikan. amplitudo dapat digunakan kembali oleh scmag, membuat besarnya perhitungan dan pembaruan efisien. Saat ini, pemilih otomatis di SeisComP 3, scautopick, juga mengukur satu set kecil amplitudo (yaitu "SNR" dan "mb", rasio signal-to-noise dan amplitudo yang digunakan dalam mb perhitungan besarnya, masing-masing) untuk setiap pick otomatis di jendela waktu yang tetap. Jika ada sudah ada amplitudo, misalnya sebelumnya ditentukan satu per scautopick, perampok tidak akan mengukur lagi untuk aliran masing-masing. Amplitudo juga diperlukan, namun, untuk picks manual. banyol melakukan hal ini juga. Pilihan dengan berat lebih kecil dari 0,5 di sesuai Asal dibuang.

Amplitudo untuk besaran berikut saat ini dihitung:

MLV

besarnya lokal dihitung dari komponen vertikal menggunakan istilah koreksi sesuai dengan ML standar.

MLsed

amplitudo lokal dihitung pada horizontals. Lihat MLsed.

mb

Gelombang Band tubuh besarnya sempit menggunakan urutan ketiga Butterworth filter dengan frekuensi sudut 0,7 dan 2,0 Hz. Perhatikan bahwa amplitudo ini juga dihitung dengan scautopick untuk semua picks otomatis.

mB

Gelombang Band tubuh besarnya luas.

Mw (mB)

Estimasi besarnya saat Mw berdasarkan mB.

Perhatikan bahwa untuk digunakan oleh scmag, nama-nama dari berbagai jenis besarnya harus sama persis.

##### Magnitude

Menghitung besaran.

Tujuan dari scmag adalah untuk menghitung besaran. Dibutuhkan amplitudo dan asal-usul sebagai masukan dan menghasilkan StationMagnitudes dan NetworkMagnitudes sebagai output. besaran yang dihasilkan dikirim ke kelompok "BESARAN". scmag tidak mengakses bentuk gelombang apapun. Hanya menggunakan amplitudo sebelumnya dihitung, misalnya oleh perampok. Tujuan dari scmag adalah decoupling besarnya perhitungan dari pengukuran amplitudo. Hal ini memungkinkan beberapa modul untuk menghasilkan amplitudo secara bersamaan, seperti scautopick dan perampok. Begitu asal masuk, amplitudo berkaitan dengan picks diambil baik dari buffer memori atau database untuk menghitung besaran. Saat ini jenis besaran berikut ini diimplementasikan:

MLV

besarnya lokal dihitung dari komponen vertikal menggunakan istilah koreksi sesuai dengan ML standar

MLH

besarnya lokal dihitung dari komponen horizontal untuk spesifikasi SED. Lihat MLH

mb

Gelombang Band tubuh besarnya sempit menggunakan urutan ketiga Butterworth filter dengan frekuensi sudut 0,7 dan 2,0 Hz.

mB

Gelombang Band tubuh besarnya luas.

mWP

Gelombang tubuh besarnya Tsuboi et al. (1995)

Selain itu, scmag menghitung besaran turunan berikut:

Mw (mB)

Estimasi besarnya saat Mw berdasarkan mB menggunakan Mw vs mB regresi Bormann dan Saul (2008)

Mw (MWP)

Estimasi saat ini besarnya Mw berdasarkan MWp menggunakan Mw vs MWp regresi Whitmore et al. (2002).

M (ringkasan)

Ringkasan besarnya, yang terdiri dari rata-rata tertimbang dari besaran individu dan mencoba untuk menjadi mungkin kompromi terbaik antara semua besaran. Lihat di bawah untuk konfigurasi dan juga scevent untuk cara menambahkan besarnya ringkasan ke daftar kemungkinan besaran disukai atau bagaimana membuatnya selalu disukai.

Mw (avg)

Estimasi saat ini besarnya Mw berdasarkan rata-rata tertimbang dari besaran lain, saat MLV, mb dan Mw (mB), di masa depan mungkin besaran lain juga, terutama yang cocok untuk acara yang sangat besar. Tujuan Mw (avg) adalah memiliki, pada setiap tahap selama pengolahan, sebuah "kemungkinan terbaik" estimasi besarnya dengan menggabungkan semua besaran yang tersedia menjadi satu, rata-rata tertimbang. Awalnya rata-rata akan terdiri dari hanya MLV dan / atau pengukuran mb, tapi begitu Mw (mB) pengukuran menjadi tersedia, ini (dan di masa depan lainnya besaran besar-event) menjadi semakin lebih berat dalam rata-rata.

Konfigurasi

besaran = [MLV, mb, mB, MWp]

Mendefinisikan jenis besaran untuk menghitung. Sebuah besarnya jenis tertentu akan dihitung hanya ketika amplitudo yang sesuai ada. Periksa amplitudo dihitung dengan scautopick dan perampok. Juga, mb dan mB hanya dihitung untuk stasiun antara 5 dan 105 derajat sehingga jika autoloc.maxStationDistance diatur ke 5 atau kurang di scautoloc maka tidak ada mb, mB, atau besaran mereka berasal akan tersedia. Mw (mB) berasal dari mB dan secara otomatis dihitung saat mB ada di daftar.

magnitudes.average = [default] | berarti | trimmedMean | median | trimmedMedian

Mendefinisikan metode rata-rata untuk digunakan saat menghitung besarnya jaringan. konfigurasi lebih lanjut adalah mungkin. Lihat di bawah.

summaryMagnitude.enabled = [benar] | palsu

Mengaktifkan / menonaktifkan perhitungan besarnya ringkasan

summaryMagnitude.type = [M]

Tentukan nama besarnya ringkasan.

summaryMagnitude.coefficients.a = [0, Mw (mB): 0,4, Mw (MWP): 0,4]

summaryMagnitude.coefficients.b = [1, MLV: 2, Mw (mB): - 1, Mw (MWP): - 1]

Tentukan koefisien untuk menghitung berat kontribusi besarnya diberikan kepada besarnya ringkasan. Nama sebelum usus besar harus sesuai jenis besarnya berkontribusi. Berat = a \* NetworkMagnitude.stationCount + b. Ringkasan besarnya kemudian dihitung sebagai M = sum (NetworkMagnitude \* Berat (NetworkMagnitude)) / sum (berat (NetworkMagnitude)).

summaryMagnitude.whitelist = []

summaryMagnitude.blacklist = [mB, MWp]

Tentukan besaran untuk memasukkan ke dalam besarnya ringkasan. Jika daftar putih didefinisikan maka hanya besaran dari jenis yang termasuk dalam mangnitude ringkasan. Jika daftar putih kosong dan daftar hitam memiliki entri maka semua besarnya disertakan kecuali dalam daftar hitam. Jika kedua daftar yang kosong maka semua besaran jaringan termasuk dalam besarnya ringkasan.

besaran Averaging

metode rata-rata besarnya dapat dikonfigurasi dengan magnitudes.average variabel. Lihat di atas.

Default rata algoritma untuk semua besaran adalah algoritma scmag saat ini (berarti kurang dari 4 besaran stasiun, trimmedMean (25). Bila menggunakan pakai ketik nilai persen harus ditugaskan untuk menentukan pemangkasan (mis trimmedMean (25)

##### event

Associates Origins dengan Acara atau bentuk Acara baru jika tidak ada pasangan yang cocok ditemukan. Memilih besarnya disukai.

Sebagai konsekuensi dari sistem real-time sistem SeisComP3 menciptakan beberapa asal-usul (hasil dari proses lokalisasi) untuk satu gempa karena seperti berjalannya waktu lebih fase seismik yang tersedia. scevent menerima usul ini dan mengaitkan asal-usul peristiwa. Hal ini juga memungkinkan untuk mengimpor Origins dari instansi lain.

asal Matching

Scevent rekan Origins dengan Acara dengan mencari pertandingan terbaik dari baru (masuk) Asal ke Origins lain untuk Acara yang ada. Jika pertandingan tidak ditemukan Event baru dapat dibentuk. The Origin baru cocok untuk Asal ada dengan membandingkan perbedaan lokasi (horizontal saja), Asal perbedaan waktu, dan pencocokan Pilihan. Pertandingan terbaik pertama lebih disukai di mana pilihan untuk pertandingan Asal adalah (terendah ke tertinggi);

(1) Lokasi dan Waktu (terendah)

Perbedaan lokasi horisontal kurang dari eventAssociation.maximumDistance (derajat) dan perbedaan waktu Asal kurang dari eventAssociation.maximumTimeSpan

(2) Pilihan

dua Origins memiliki lebih dari eventAssociation.minimumMatchingArrivals cocok Pilihan. Catatan (memeriksa ini dengan Jan) tampak bahwa Pilihlah kesetaraan didasarkan hanya pada publicID. Ini secara efektif berarti bahwa untuk Pilihan untuk menjadi sama mereka akan perlu dibuat oleh konfigurasi pemilih yang sama.

(3) Pilihan dan Lokasi dan Waktu (tertinggi)

ini adalah pertandingan terbaik, baik dari Lokasi-dan-Waktu dan Picks kriteria di atas puas.

Catatan: untuk efisiensi Acara di cache di-scan pertama dan jika tidak ada pertandingan yang ditemukan maka database dipindai untuk jendela waktu eventAssociation.eventTimeBefore eventAssociation.eventTimeAfter sekitar waktu Asal masuk.

Urutan objek dalam cache akan mempengaruhi pertandingan terbaik pertama - adalah waktu atau penyisipan memerintahkan?

Kemungkinan Perbaikan (Geoff dan Kaya berdiskusi dengan Jan) Lokasi dan waktu perbandingan dapat ditingkatkan dengan mencari Asal dengan waktu minium delta dan perbedaan. Perbandingan Pilihlah dapat ditingkatkan dengan pencocokan fuzzy pada waktu Pick Nama, dan Channel.

Tidak ada Pertandingan Asal

Jika tidak ada acara dengan Asal yang cocok dengan Asal masuk ditemukan maka acara baru terbentuk dan Asal dikaitkan ke acara itu. Kriteria berikut diterapkan untuk memungkinkan penciptaan Event baru:

Badan Origin tidak dalam daftar hitam (processing.blacklist.agencies).

dan

Jika Asal merupakan Automatic maka ia memiliki lebih dari eventAssociation.minimumDefiningPhases Pilihan.

Gambar 5-1: Asosiasi asal oleh matching\_picks

Preferred Asal

Seperti telah disebutkan, salah satu gempa dapat diwakili oleh beberapa asal. Jadi, satu asal harus dipilih antara asal-usul terkait mewakili lokasi dan waktu terbaik. Hal ini dilakukan dengan aturan berikut:

Asal pengguna atau dikonfirmasi terbaru memiliki prioritas tertinggi dan selalu disukai

asal Manual atau dikonfirmasi selalu prioritas yang lebih tinggi daripada asal otomatis

Dalam hal asal disukai sebenarnya otomatis, asal otomatis masuk akan disukai, jika mengandung lebih kedatangan / fase dan memiliki besaran jaringan menjadi magnitude yang lebih disukai mungkin

Besaran Preferred

Tugas ketiga scevent adalah memilih besarnya disukai. Untuk tujuan ini aturan berikut digunakan:

besaran jaringan hanya berlaku jika lebih dari jumlah yang ditentukan besaran stasiun ada (default = 3). Mw (mB) adalah besarnya kuat gerak menghasilkan kriteria khusus: Mw (mB) adalah sah jika sama atau lebih dari jumlah yang ditentukan besaran stasiun ada (default = 8).

Prioritas besaran adalah Mw (mB), MLV, mb (dari tinggi ke rendah). Untuk Mw (mB) kriteria prioritas tambahan didefinisikan.

Jika lebih dari jumlah yang ditentukan besaran stasiun ada untuk Mw (mB) (default = 30), Mw (mB) akan selalu disukai

Jika kurang dari jumlah yang ditentukan dari besaran stasiun ada untuk Mw (mB) (default = 30), Mw (mB) akan lebih disukai saat

Jumlah semua stasiun besaran untuk Mw (mB) dan untuk mb dibagi 2 adalah lebih dari nilai yang ditetapkan (default = 6)

Jika jumlah besaran stasiun untuk Mw (mB) lebih besar atau sama dari setengah dari jumlah besarnya stasiun untuk mb

##### quality parameter

scqc menentukan parameter kualitas seismik data stream. Parameter output rata-rata kontrol kualitas (QC) parameter waktu dalam hal pesan kualitas gelombang. Dalam interval reguler pesan laporan yang dikirim mengandung rata-rata representasi jangka pendek dari parameter QC khusus untuk rentang waktu tertentu. pesan alarm yang dihasilkan jika rata-rata jangka pendek (misalnya 90-an) dari parameter QC berbeda dari rata-rata jangka panjang (mis 3600s) lebih dari batas yang ditentukan.

Untuk menghindari beban yang berlebihan, pesan QC dikirim didistribusikan dari waktu ke waktu. Parameter berikut ditentukan:

Delay [s]

Perbedaan waktu antara waktu kedatangan dan terakhir kali akhir record ditambah setengah record panjang (berarti data yang latency, berlaku untuk semua sampel dalam catatan)

Latency [s]

Perbedaan waktu antara waktu saat ini dan catatan waktu kedatangan (pakan latency)

Offset [jumlah]

Rata-rata nilai semua sampel dari rekor

RMS [jumlah]

Offset akar dikoreksi mean square (RMS) nilai rekor

Spike (interval [s], amplitudo [jumlah])

Dalam kasus terjadinya lonjakan rekor parameter ini memberikan interval waktu antara paku yang berdekatan dan amplitudo rata-rata spike; internal daftar paku disimpan (lonjakan waktu, lonjakan amplitudo); algoritma lonjakan finder masih awal

Gap (interval [s], panjang [s])

Dalam kasus kesenjangan data antara dua catatan berturut-turut parameter ini memberikan waktu jeda interval dan panjang rata-rata kesenjangan

Timing [%]

kualitas miniseed catatan waktu (0 - 100%)

#### Fungsi tambahan