



# 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	<b>TUGAS AKHIR TEKNIK ELEKTRO:</b> Sistem Detektor Gempa dan Peringatan Dini Tsunami <i>Decision Support System</i>
Jenis Dokumen	<b>IMPLEMENTASI</b>
Catatan: Dokumen ini dikendalikan penyebarannya oleh Prodi Teknik Elektro ITB	
Nomor Dokumen	<b>B400-01-TA1617.01. 069</b>
Nomor Revisi	<b>02</b>
Nama File	<b>B400 .docx</b>
Tanggal Penerbitan	<b>1 Juni 2017</b>
Unit Penerbit	<b>Prodi Teknik Elektro - ITB</b>
Jumlah Halaman	<b>101</b> (termasuk lembar sampul ini)

Data Pemeriksaan dan Persetujuan				
Ditulis oleh	Nama	Kevin Shidqi	Jabatan	Anggota
	Tanggal	1 June 2017	Tanda Tangan	
	Nama	Bramantio Yuwono	Jabatan	Anggota
	Tanggal	1 June 2017	Tanda Tangan	
	Nama	Christoporus Deo	Jabatan	Anggota
	Tanggal	Putratama 1 June 2017	Tanda Tangan	
Diperiksa oleh	Nama	Dr. techn. Ary Setijadi	Jabatan	Dosen Pembimbing
	Tanggal	Prihatmanto 1 Juni 2017	Tanda Tangan	
Disetujui Oleh	Nama	Dr. techn. Ary Setijadi	Jabatan	<b>Dosen Pembimbing</b>
	Tanggal	Prihatmanto 1 Juni 2017	Tanda Tangan	

# DAFTAR ISI

<b>DAFTAR ISI.....</b>	<b>2</b>
<b>CATATAN SEJARAH PERBAIKAN DOKUMEN.....</b>	<b>3</b>
<b>PROPOSAL PROYEK PENGEMBANGAN “ RANCANG BANGUN <i>FLAPPING WINGS MICRO AERIAL VEHICLE</i>: SISTEM KENDALI, SENSOR, DAN TELEMETRI ” .....</b>	<b>ERROR! BOOKMARK NOT DEFINED.</b>
<b>1    PENGANTAR .....</b>	<b>4</b>
1.1    RINGKASAN ISI DOKUMEN .....	4
1.2    TUJUAN PENULISAN DAN APLIKASI/KEGUNAAN DOKUMEN.....	4
1.3    REFERENSI.....	4
1.4    DAFTAR SINGKATAN .....	4
<b>2    IMPLEMENTASI.....</b>	<b>6</b>
2.1    IMPLEMENTASI PENGIRIMAN PESAN.....	<b>ERROR! BOOKMARK NOT DEFINED.</b>
2.1.1 <i>Pendahuluan</i> .....	<i>Error! Bookmark not defined.</i>
2.1.2 <i>Struktur</i> .....	<i>Error! Bookmark not defined.</i>
2.1.3 <i>Environment</i> .....	<i>Error! Bookmark not defined.</i>
2.1.4 <i>Prosedur Implementasi RabbitMQ</i> .....	<i>Error! Bookmark not defined.</i>
2.1.5 <i>Prosedur Implementasi pengiriman</i> .....	<i>Error! Bookmark not defined.</i>
2.2    SENSOR EARTHQUAKE CATCHER NETWORK ...	<b>ERROR! BOOKMARK NOT DEFINED.</b>
2.2.1 <i>Main Module</i> .....	<i>Error! Bookmark not defined.</i>
2.2.2 <i>Komponen Elektrik</i> .....	<i>Error! Bookmark not defined.</i>
2.2.3 <i>Algoritma</i> .....	<i>Error! Bookmark not defined.</i>
2.3    PENGOLAH DATA DAN GUI.....	<b>ERROR! BOOKMARK NOT DEFINED.</b>
2.3.1 <i>Pengolah Data</i> .....	<i>Error! Bookmark not defined.</i>
2.3.2 <i>Antarmuka Grafis</i> .....	<i>Error! Bookmark not defined.</i>

## Catatan Sejarah Perbaikan Dokumen

REVISI	TANGGAL	OLEH	PERBAIKAN

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

## 1 Pengantar

### 1.1 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

### 1.2 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.

### 1.3 Referensi

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

### 1.4 Daftar Singkatan

SINGKATAN	ARTI

SINGKATAN	ARTI

## 2 Implementasi Proyek

### 2.1 Implementasi Messaging Server dan Database

#### 2.1.1 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. Kemudian pesan yang sudah tersimpan dalam RabbitMQ akan di consume menggunakan console C# untuk kemudian dimasukkan ke dalam database MySQL.

#### 2.1.2 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:

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

Kemudian setelah itu, untuk dapat memasukkan data dari RabbitMQ menuju MySQL, maka dilakukan beberapa tahap implementasi, yaitu:

- a. membuat program yang dapat melakukan consume pada server RabbitMQ yang berkesesuaian (dalam bahasa C#, maupun Python);
- b. pengaturan kecepatan consume di dalam program hingga 1 pesan / detik supaya memiliki time stamp yang teratur;
- c. pengiriman data dari program menuju MySQL untuk disimpan di dalam database.

#### 2.1.3 Environment

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

- a. perangkat keras Laptop Asus X450JE sebagai komputer untuk menjalankan lokal RabbitMQ dan juga sebagai tempat Microsoft Visual Studio 2013 dan MySQL dijalankan.
- b. perangkat lunak Microsoft Visual Studio 2013 sebagai perangkat lunak IDE untuk mengembangkan GUI dan program yang berfungsi sebagai consumer RabbitMQ.
- c. perangkat keras Raspberry Pi 3 sebagai mini komputer untuk menjalankan lokal RabbitMQ

- d. perangkat lunak Python 3.5 sebagai perangkat lunak IDE yang digunakan pada Raspberry Pi 3 yang berfungsi sebagai pengirim pesan maupun consumer pada RabbitMQ
- e. perangkat keras Node MCU Amica sebagai mikrokontroler sensor gempa untuk mengirimkan pesan geospasial menuju server RabbitMQ
- f. perangkat lunak Arduino IDE 1.8.1 sebagai perangkat lunak IDE untuk mengembangkan algoritma pengiriman pesan geospasial menuju RabbitMQ
- g. JSON sebagai library dan syntax pengiriman pesan antara sensor gempa dengan RabbitMQ dan juga antara RabbitMQ dengan consumer

#### 2.1.4 Implementasi Messaging Server RabbitMQ

Penggunaan RabbitMQ pada sebuah devais bisa dilakukan 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:

##### 2.1.4.1 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>

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



**Little Lemur** For development

Max 1M messages per month

Max 20 concurrent connections

Max 100 queues

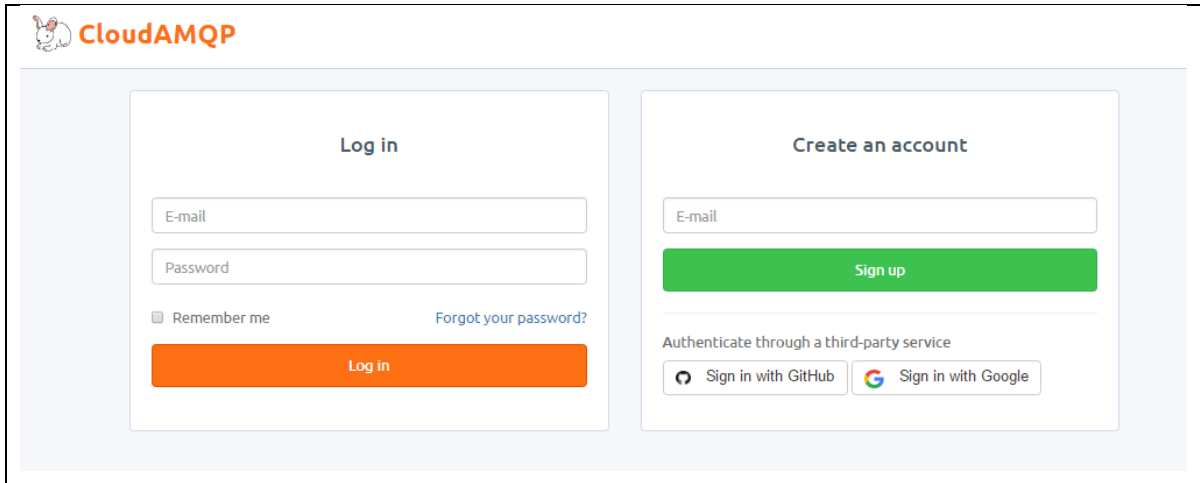
Max 10 000 queued messages

Max idle queue time 28 days

**FREE**

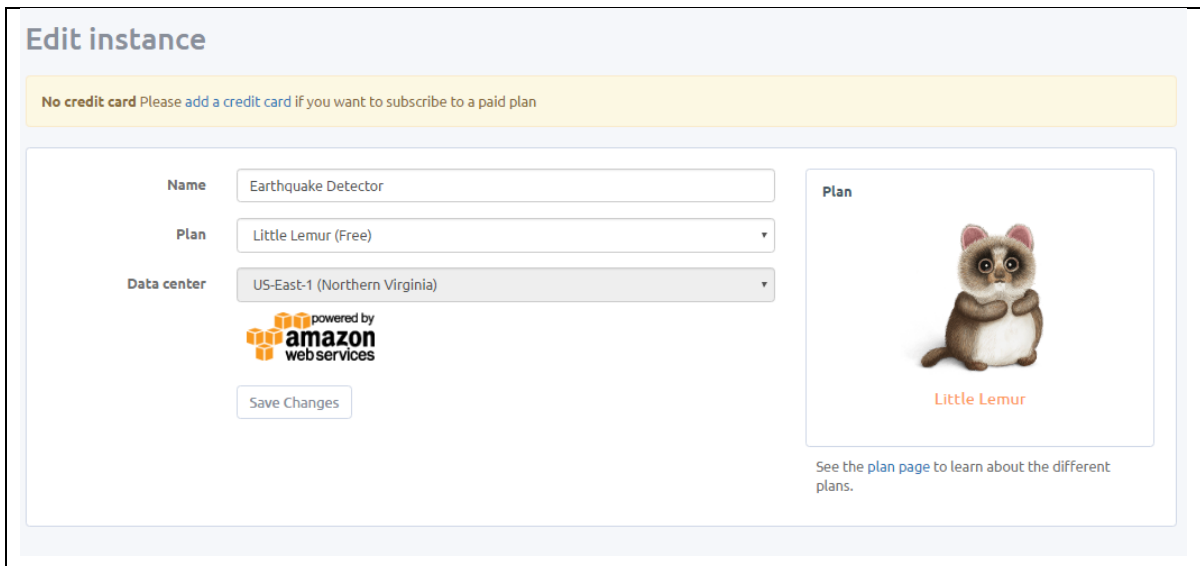
[Try a Little Lemur now](#)

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.



The image shows the CloudAMQP login and registration interface. On the left, the 'Log in' section includes fields for 'E-mail' and 'Password', a 'Remember me' checkbox, a 'Forgot your password?' link, and an orange 'Log in' button. On the right, the 'Create an account' section includes an 'E-mail' field, a green 'Sign up' button, and a section for third-party authentication with 'Sign in with GitHub' and 'Sign in with Google' buttons.

Setelah melakukan pendaftaran, maka akan terbuka halaman web untuk membuat sebuah instance sesuai dengan plan yang dipilih. Masukkan 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.



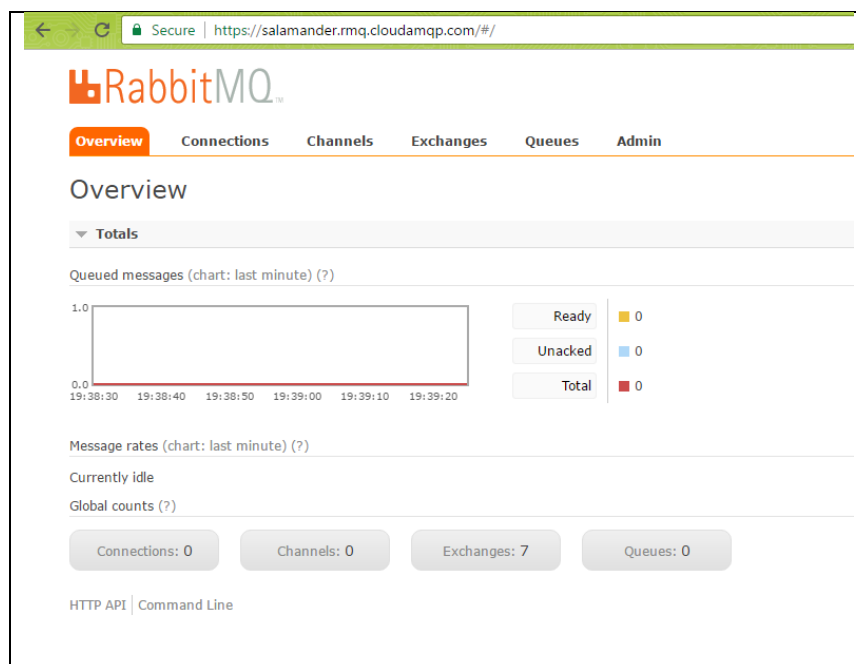
The image shows the 'Edit instance' page. At the top, a yellow banner reads: 'No credit card Please add a credit card if you want to subscribe to a paid plan'. Below this, the 'Name' field contains 'Earthquake Detector'. The 'Plan' dropdown is set to 'Little Lemur (Free)'. The 'Data center' dropdown is set to 'US-East-1 (Northern Virginia)'. Below these fields is the 'powered by amazon web services' logo and a 'Save Changes' button. On the right, the 'Plan' section features a cute lemur illustration and the text 'Little Lemur'. At the bottom right, a link says 'See the plan page to learn about the different plans.'

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.

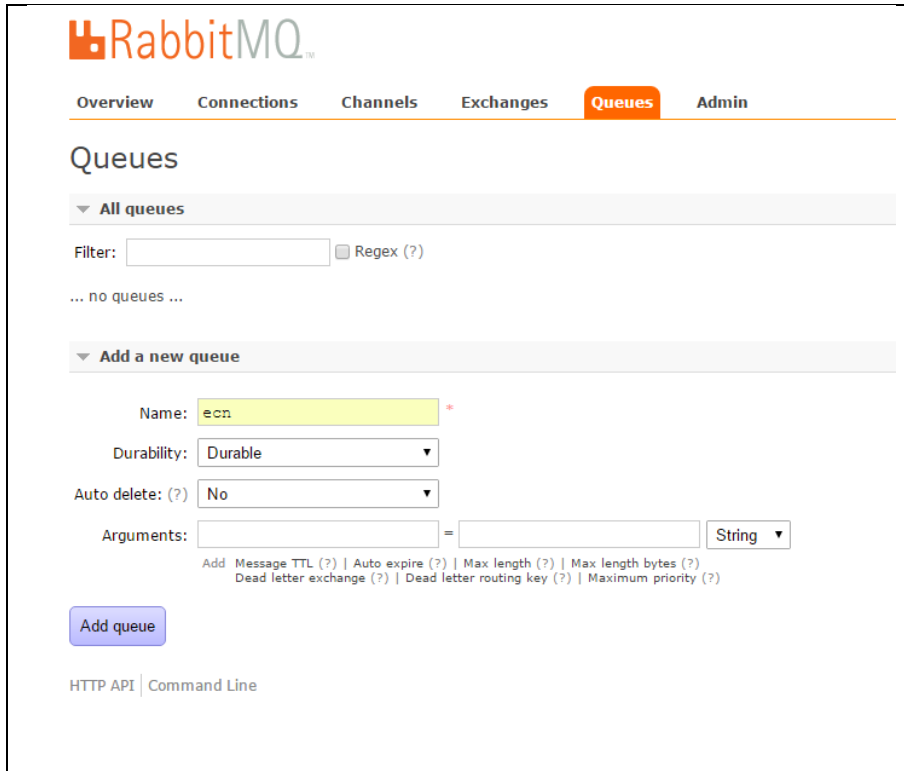


Instances			<a href="#">+ Create New Instance</a>	
Name	Plan	Datacenter	Actions	
Earthquake Detector	Lemur	Amazon Web Services US-East-1 (Northern Virginia)	<a href="#">Edit</a>	<a href="#">RabbitMQ Manager</a>
PythonTest2	Lemur	Amazon Web Services US-East-1 (Northern Virginia)	<a href="#">Edit</a>	<a href="#">RabbitMQ Manager</a>
PythonTest	Lemur	Google Compute Engine Europe West	<a href="#">Edit</a>	<a href="#">RabbitMQ Manager</a>

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:

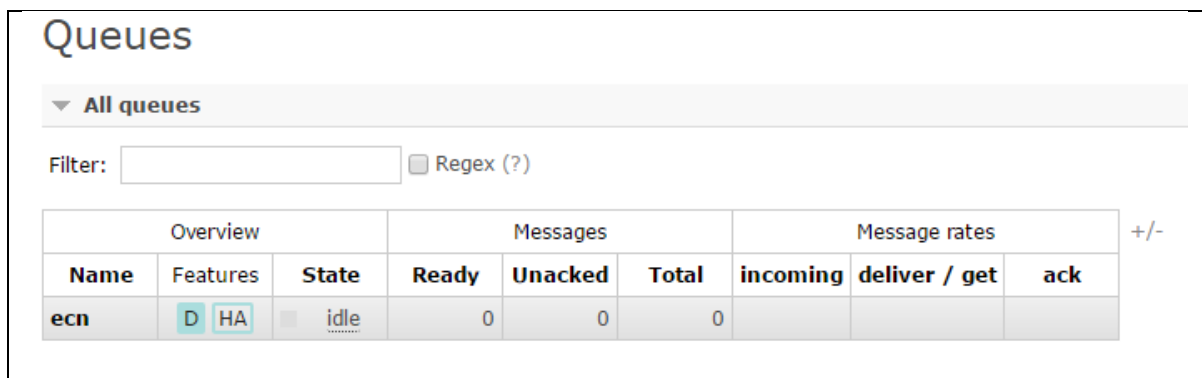


Hal yang dilakukan pertama kali adalah dengan mengklik tab Queues pada bagian atas layar.



The screenshot shows the RabbitMQ Admin interface. At the top, there's a navigation bar with tabs: Overview, Connections, Channels, Exchanges, **Queues**, and Admin. The 'Queues' tab is active. Below the navigation bar, the title 'Queues' is displayed. There's a section 'All queues' with a filter input and a 'Regex (?)' checkbox. Below this, it says '... no queues ...'. Then, there's a section 'Add a new queue' with form fields: 'Name' (set to 'ecn'), 'Durability' (set to 'Durable'), 'Auto delete: (?)' (set to 'No'), and 'Arguments' (empty). Below the arguments field, there are links for 'Add', 'Message TTL (?)', 'Auto expire (?)', 'Max length (?)', 'Max length bytes (?)', 'Dead letter exchange (?)', 'Dead letter routing key (?)', and 'Maximum priority (?)'. At the bottom of the form is an 'Add queue' button. At the very bottom, there are links for 'HTTP API' and 'Command Line'.

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”.



The screenshot shows the RabbitMQ Admin interface with the 'Queues' tab active. Below the 'All queues' section, there's a table with columns: Overview, Messages, and Message rates. The 'Overview' column has sub-columns: Name, Features, and State. The 'Messages' column has sub-columns: Ready, Unacked, and Total. The 'Message rates' column has sub-columns: incoming, deliver / get, and ack. The table shows one queue named 'ecn' with a state of 'idle' and 0 messages. The 'Features' column for 'ecn' shows 'D' and 'HA'.

Overview			Messages			Message rates			+/-
Name	Features	State	Ready	Unacked	Total	incoming	deliver / get	ack	
ecn	D HA	idle	0	0	0				

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

## Exchanges

▼ All exchanges

Filter:  ☐ Regex (?)

Name	Type	Features	Message rate in	Message rate out	+/
(AMQP default)	direct	D			
amq.direct	direct	D			
amq.fanout	fanout	D			
amq.headers	headers	D			
amq.match	headers	D			
amq.rabbitmq.trace	topic	D I			
amq.topic	topic	D			

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

Add binding from this exchange

To queue ▼ :  \*

Routing key:

Arguments:  =  String ▼

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

## Bindings

This exchange

⇓

To	Routing key	Arguments	
ecn	earthquake		<input type="button" value="Unbind"/>

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.

#### 2.1.4.2 Implementasi RabbitMQ dengan menggunakan server RabbitMQ PPTIK

Pada proyek ini, produk akhir harus menggunakan server dari PPTIK ITB. Oleh karena itu dilakukan beberapa pengaturan yang akan dijabarkan berikut ini.

RabbitMQ diimplementasikan dengan menggunakan protokol AMQP seperti dibawah ini:

```
amqp://sensor_gempa:12345@167.205.7.226/%2fdisaster
```

URL diatas merupakan merupakan parameter yang harus ditambahkan pada proses koneksi pengiriman dan pengambilan data dat

Keterangannya adalah sebagai berikut:

- a. amqp → Protokol komunikasi
- b. sensor\_gempa → username server
- c. 12345 → password
- d. 167.205.7.226 → host server
- e. /%2fdisaster → /disaster → virtual host

Penggunaan %2f disini dimaksudkan untuk mengkoreksi kesalahan pembacaan /disaster karena karakter / berpengaruh terhadap protokol komunikasi dalam bentuk URL. Kemudian selain mengatur alamat server seperti yang dijelaskan diatas, pengaturan berikut-berikutnya akan dijelaskan sebagai berikut:

- a. queue → ecn + nomer sensor  
penamaan queue dibebaskan namun harus unik dan khusus dialokasikan untuk sebuah sensor.
- b. exchange → amq.topic  
amq.topic merupakan sebuah exchange default dengan type = topic yang dapat dipakai dalam proses implementasi
- c. routing key → exchange + queue  
routing key dimaksudkan untuk menghubungkan exchange menuju queue tertentu. Hal ini bertujuan untuk memilah pesan sesuai dengan jenis dan nama sensornya.
- d. durable → true  
durable bertujuan untuk menahan pesan supaya tidak hilang ketika server mati.

Sehingga implementasi untuk parameter-parameter diatas adalah sebagai berikut:

```
factory.Uri = "amqp://sensor_gempa:12345@167.205.7.226/%2fdisaster";  
channel.QueueDeclare(queue: "ecn",  
    durable: true,  
    exclusive: false,  
    autoDelete: false,
```

```

        arguments: null);
channel.QueueBind(queue: "ecn",
    exchange: "amq.topic",
    routingKey: "amq.topic.ecn"
);

```

Penggunaan kode diatas adalah implementasi dalam Bahasa C#.

### 2.1.5 Implementasi Database MySQL

Supaya dapat memasukkan pesan kedalam MySQL, maka terlebih dahulu data di consume dengan menggunakan sebuah program. Program yang akan dibuat adalah program C# yang bertujuan untuk melakukan consume pada RabbitMQ dan mendapatkan pesan. Lalu pesan ini dipilah-pilah sesuai dengan kolom yang ada pada MySQL. Algoritma dari program tersebut adalah sebagai berikut:

```

public static void Main(string[] args)
{
    connect_database();
    Timer t = new Timer(TimerCallback, null, 0, 1000);
    Console.ReadLine();
}

```

Program diatas adalah program utama yang menjalankan fungsi koneksi database MySQL dan kemudian membuat Thread baru untuk menjalankan Timer dimana fungsi tersebut akan dipanggil setiap 1 detik.

```

public static void connect_msgserver()
{
    ConnectionFactory factory;
    using (StreamReader r = new StreamReader("config1.json"))
    {
        string json = r.ReadToEnd();
        Config config = JsonConvert.DeserializeObject<Config>(json);

        factory = new ConnectionFactory();
        factory.Uri = "amqp://sensor_gempa:12345@167.205.7.226/%2fdisaster";
    }

    factory.Protocol = Protocols.DefaultProtocol;
    factory.Port = AmqpTcpEndpoint.UseDefaultPort;

    //SENSOR ecn
    using (var connection = factory.CreateConnection())
    using (var channel = connection.CreateModel())
    {
        channel.BasicQos(0, 1, false);
        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);
            try
            {
                data accelReport = JsonConvert.DeserializeObject<data>(message);
            }
            catch (Exception ex)

```

```

        {
            Console.WriteLine("Failed");
        }

    };
    channel.BasicConsume(queue: "ecn", //"emergency_gui",
        noAck: true,
        consumer: consumer);

    Console.WriteLine("Already BasicConsume");
}
}

```

Program diatas adalah sebuah fungsi untuk melakukan consume pada RabbitMQ.

```

public static void connect_database()
{
    string server_host = "localhost";
    string server_password = "MySQLRoot";
    string server_name = "root";

    try
    {
        con.ConnectionString = "server=" + server_host + ";user id=" + server_name +
";password=" + server_password + ";database=earthquake";
        con.Open();
        //MessageBox.Show("Connected to " + server);
    }
    catch (Exception e1)
    {
        Console.WriteLine("Connection failed due to " + e1.ToString());
    }
}

private static void write_database(string[] data)
{
    command_add = con.CreateCommand();
    try
    {
        command_add.CommandText = "INSERT INTO store_id (point_time, time_zone_id,
interval_id, client_id, lattitude_id, longitude_id) VALUES('" + data[0] + "', '" + data[1] + "',
'" + data[2] + "', '" + data[3] + "', '" + data[4] + "', '" + data[5] + "')";
        command_add.ExecuteNonQuery();
    }
    catch (Exception e1)
    {
        try
        {
            command_add.CommandText = "INSERT INTO store_id (point_time, time_zone_id,
interval_id, client_id, lattitude_id, longitude_id) VALUES('" + timestamp + "', '" + empty +
"', '" + empty + "', '" + empty + "', '" + empty + "', '" + empty + "')";
            command_add.ExecuteNonQuery();
        }
        catch
        {
        }
    }

    try
    {
        con.Close();
    }
    catch
    {
        Thread.Sleep(2000);
        con.Close();
    }
    Thread.Sleep(1000);
}
}

```

Program diatas adalah proses untuk melakukan proses koneksi pada database MySQL dan kemudian menuliskan data pada kolom-kolom di MySQL.

### 2.1.6 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 menambahkan 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;
    };
}

```

## Lampiran Program Database.cs

```

using MySql.Data.MySqlClient;
using Newtonsoft.Json;
using RabbitMQ.Client;
using RabbitMQ.Client.Events;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading;
using System.Threading.Tasks;

namespace Database
{
    public static class Program
    {
        public static string timestamp = "";
        public static string empty = "-";
        public static string[] words1 = new string[2];
        public static string[] words = new string[10];
        public static string[] messages;
        public static int count = 0;
        public static int count1 = 0;
        public static MySqlCommand command_add;
        public static MySqlConnection con = new MySqlConnection();

        public static void Main(string[] args)
        {
            connect_database();
            Timer t = new Timer(TimerCallback, null, 0, 1000);
            Console.ReadLine();
        }
    }
}

```

```

public static void connect_msgserver()
{
    ConnectionFactory factory;
    using (StreamReader r = new StreamReader("config1.json"))
    {
        string json = r.ReadToEnd();
        Config config = JsonConvert.DeserializeObject<Config>(json);

        factory = new ConnectionFactory();
        factory.Uri =
"amqp://sensor_gempa:12345@167.205.7.226/%2fdisaster";
    }

    factory.Protocol = Protocols.DefaultProtocol;
    factory.Port = AmqpTcpEndpoint.UseDefaultPort;

    //SENSOR ecn
    using (var connection = factory.CreateConnection())
    using (var channel = connection.CreateModel())
    {

        channel.BasicQos(0, 1, false);
        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);
            try
            {
                data accelReport =
JsonConvert.DeserializeObject<data>(message);
            }
            catch (Exception ex)
            {
                Console.WriteLine("Failed");
            }

        };
        channel.BasicConsume(queue: "ecn", //"emergency_gui",
                               noAck: true,
                               consumer: consumer);

        Console.WriteLine("Already BasicConsume");
    }
}

public static void connect_msgserver2()
{
    //consume_data();
    ConnectionFactory factory;
    factory = new ConnectionFactory();
    factory.Uri =
"amqp://sensor_gempa:12345@167.205.7.226/%2fdisaster";
    factory.Protocol = Protocols.DefaultProtocol;
    factory.Port = AmqpTcpEndpoint.UseDefaultPort;
}

```

```

//SENSOR ecn
using (var connection = factory.CreateConnection())
using (var channel = connection.CreateModel())
{

    channel.BasicQos(0, 1, false);
    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);
        try
        {
            words1[1] = message;
            Console.WriteLine(message);
        }
        catch (Exception ex)
        {
            Console.WriteLine("Failed");
        }
    };

    channel.BasicConsume(queue: "ecn", //"emergency_gui",
        noAck: true,
        consumer: consumer);

    Console.WriteLine("Already BasicConsume");
    //Console.ReadLine();
}

}

public static void connect_database()
{
    string server_host = "localhost";
    string server_password = "MySQLRoot";
    string server_name = "root";

    try
    {
        con.ConnectionString = "server=" + server_host + ";user id=" +
server_name + ";password=" + server_password + ";database=earthquake";
        con.Open();
        //MessageBox.Show("Connected to " + server);
    }
    catch (Exception e1)
    {
        Console.WriteLine("Connection failed due to " +
e1.ToString());
    }
}

private static void write_database(string[] data)
{
    command_add = con.CreateCommand();
    try

```

```

        {
            command_add.CommandText = "INSERT INTO store_id (point_time,
time_zone_id, interval_id, client_id, latitude_id, longitude_id) VALUES('" +
data[0] + "', '" + data[1] + "', '" + data[2] + "', '" + data[3] + "', '" +
data[4] + "', '" + data[5] + "')";
            command_add.ExecuteNonQuery();
        }
        catch (Exception e1)
        {
            try
            {
                command_add.CommandText = "INSERT INTO store_id
(point_time, time_zone_id, interval_id, client_id, latitude_id, longitude_id)
VALUES('" + timestamp + "', '" + empty + "', '" + empty + "', '" + empty + "', '"
+ empty + "', '" + empty + "')";
                command_add.ExecuteNonQuery();
            }
            catch
            {
            }
        }

        try
        {
            con.Close();
        }
        catch
        {
            Thread.Sleep(2000);
            con.Close();
        }
        Thread.Sleep(1000);
    }

    private static void write_database_test(string[] data)
    {
        command_add = con.CreateCommand();
        try
        {
            command_add.CommandText = "INSERT INTO test_id
(point_time_console, point_time_message) VALUES('" + data[0] + "', '" +
data[1] + "')";
            command_add.ExecuteNonQuery();
        }
        catch (Exception e1)
        {
            try
            {
                command_add.CommandText = "INSERT INTO test_id
(point_time_console, point_time_message) VALUES('" + timestamp + "', '" + empty
+ "')";
                command_add.ExecuteNonQuery();
            }
            catch
            {
            }
        }

        //Reset connection to avoid exception
        try
        {
            con.Close();
        }
        catch
        {
            Thread.Sleep(2000);

```

```

        con.Close();
    }

    try
    {
        con.Open();
    }
    catch
    {
        Thread.Sleep(2000);
        con.Close();
    }

    Thread.Sleep(1000);
}

private static void TimerCallback(Object o)
{
    words[0] = "";
    words1[0] = "";
    // Display the date/time when this method got called.
    //Console.WriteLine("In TimerCallback: " +
    DateTime.Now.ToString("yyyy-MM-dd HH:mm:ss"));
    //connect_msgserver();
    timestamp = DateTime.Now.ToString("yyyy-MM-dd HH:mm:ss");
    words[0] = timestamp;
    words1[0] = timestamp;
    connect_msgserver2();
    //connect_msgserver2();
    Console.WriteLine(timestamp);
    write_database_test(words1);
    //write_database_test(words1);
    //count++;
    // Force a garbage collection to occur for this demo.
    GC.Collect();
}

private static void TimerCallback1(Object o)
{
    GC.Collect();
}

public class Config
{
    public string host;
    public string user;
    public string vhost;
    //public string port;
    public string password;
}

}

public class data
{
    public string pointTime;
    public string timeZone;
    public string interval;
    public string clientID;
    public geojson geojson;
    public acc[] accelerations;
};

public class geojson

```

```

    {
        public geometry geometry;
        public prop property;
    };
    public class geometry
    {
        public string type;
        public string[] coordinates;
    };
    public class prop
    {
        public string name;
    };
    public class acc
    {
        public string x;
        public string y;
        public string z;
    };
}
using MySql.Data.MySqlClient;
using Newtonsoft.Json;
using RabbitMQ.Client;
using RabbitMQ.Client.Events;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading;
using System.Threading.Tasks;

namespace Database
{
    public static class Program
    {
        public static string timestamp = "";
        public static string empty = "-";
        public static string[] words1 = new string[2];
        public static string[] words = new string[10];
        public static string[] messages;
        public static int count = 0;
        public static int count1 = 0;
        public static MySqlCommand command_add;
        public static MySqlConnection con = new MySqlConnection();

        public static void Main(string[] args)
        {
            connect_database();
            Timer t = new Timer(TimerCallback, null, 0, 1000);
            Console.ReadLine();
        }

        public static void connect_msgserver()
        {
            ConnectionFactory factory;
            using (StreamReader r = new StreamReader("config1.json"))
            {
                string json = r.ReadToEnd();
                Config config = JsonConvert.DeserializeObject<Config>(json);

                factory = new ConnectionFactory();
                factory.Uri =
"amqp://sensor_gempa:12345@167.205.7.226/%2fdisaster";
            }

            factory.Protocol = Protocols.DefaultProtocol;
            factory.Port = AmqpTcpEndpoint.UseDefaultPort;

```

```

//SENSOR ecn
using (var connection = factory.CreateConnection())
using (var channel = connection.CreateModel())
{

    channel.BasicQos(0, 1, false);
    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);
        try
        {
            data accelReport =
JsonConvert.DeserializeObject<data>(message);
        }
        catch (Exception ex)
        {
            Console.WriteLine("Failed");
        }

    };

    channel.BasicConsume(queue: "ecn", //"emergency_gui",
                        noAck: true,
                        consumer: consumer);

    Console.WriteLine("Already BasicConsume");
}

}

public static void connect_msgserver2()
{
    //consume_data();
    ConnectionFactory factory;
    factory = new ConnectionFactory();
    factory.Uri =
"amqp://sensor_gempa:12345@167.205.7.226/%2fdisaster";
    factory.Protocol = Protocols.DefaultProtocol;
    factory.Port = AmqpTcpEndpoint.UseDefaultPort;

    //SENSOR ecn
    using (var connection = factory.CreateConnection())
    using (var channel = connection.CreateModel())
    {

        channel.BasicQos(0, 1, false);
        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);
            try
            {
                words1[1] = message;
                Console.WriteLine(message);
            }
            catch (Exception ex)
            {
                Console.WriteLine("Failed");
            }
        };
        channel.BasicConsume(queue: "ecn", //"emergency_gui",
                               noAck: true,
                               consumer: consumer);

        Console.WriteLine("Already BasicConsume");
        //Console.ReadLine();
    }

    public static void connect_database()
    {
        string server_host = "localhost";
        string server_password = "MySQLRoot";
        string server_name = "root";

        try
        {
            con.ConnectionString = "server=" + server_host + ";user id=" +
server_name + ";password=" + server_password + ";database=earthquake";
            con.Open();
            //MessageBox.Show("Connected to " + server);
        }
        catch (Exception e1)
        {
            Console.WriteLine("Connection failed due to " +
e1.ToString());
        }
    }

    private static void write_database(string[] data)
    {
        command_add = con.CreateCommand();
        try
        {
            command_add.CommandText = "INSERT INTO store_id (point_time,
time_zone_id, interval_id, client_id, latitude_id, longitude_id) VALUES('" +
data[0] + "', '" + data[1] + "', '" + data[2] + "', '" + data[3] + "', '" +
data[4] + "', '" + data[5] + "')";
            command_add.ExecuteNonQuery();
        }
        catch (Exception e1)
        {
            try
            {
                command_add.CommandText = "INSERT INTO store_id
(point_time, time_zone_id, interval_id, client_id, latitude_id, longitude_id)
VALUES('" + timestamp + "', '" + empty + "', '" + empty + "', '" + empty + "', '"
+ empty + "', '" + empty + "')";
            }
            catch (Exception e2)
            {
                Console.WriteLine("Insert failed due to " +
e2.ToString());
            }
        }
    }
}

```



```

        command_add.ExecuteNonQuery();
    }
    catch
    {
    }

}

try
{
    con.Close();
}
catch
{
    Thread.Sleep(2000);
    con.Close();
}
Thread.Sleep(1000);
}

private static void write_database_test(string[] data)
{
    command_add = con.CreateCommand();
    try
    {
        command_add.CommandText = "INSERT INTO test_id
(point_time_console, point_time_message) VALUES('" + data[0] + "', '" +
data[1] + "')";
        command_add.ExecuteNonQuery();
    }
    catch (Exception e1)
    {
        try
        {
            command_add.CommandText = "INSERT INTO test_id
(point_time_console, point_time_message) VALUES('" + timestamp + "', '" + empty
+ "')";

            command_add.ExecuteNonQuery();
        }
        catch
        {
        }

    }

    //Reset connection to avoid exception
    try
    {
        con.Close();
    }
    catch
    {
        Thread.Sleep(2000);
        con.Close();
    }

    try
    {
        con.Open();
    }
    catch
    {
        Thread.Sleep(2000);
        con.Close();
    }

    Thread.Sleep(1000);
}

```

```

    }

    private static void TimerCallback(Object o)
    {
        words[0] = "";
        words1[0] = "";
        // Display the date/time when this method got called.
        //Console.WriteLine("In TimerCallback: " +
        DateTime.Now.ToString("yyyy-MM-dd HH:mm:ss"));
        //connect_msgserver();
        timestamp = DateTime.Now.ToString("yyyy-MM-dd HH:mm:ss");
        words[0] = timestamp;
        words1[0] = timestamp;
        connect_msgserver2();
        //connect_msgserver2();
        Console.WriteLine(timestamp);
        write_database_test(words1);
        //write_database_test(words1);
        //count++;
        // Force a garbage collection to occur for this demo.
        GC.Collect();

    }

    private static void TimerCallback1(Object o)
    {
        GC.Collect();
    }

    public class Config
    {
        public string host;
        public string user;
        public string vhost;
        //public string port;
        public string password;
    }

}

public class data
{
    public string pointTime;
    public string timeZone;
    public string interval;
    public string clientID;
    public geojson geojson;
    public acc[] accelerations;
};

public class geojson
{
    public geometry geometry;
    public prop property;
};

public class geometry
{
    public string type;
    public string[] coordinates;
};

public class prop
{
    public string name;
};

public class acc
{

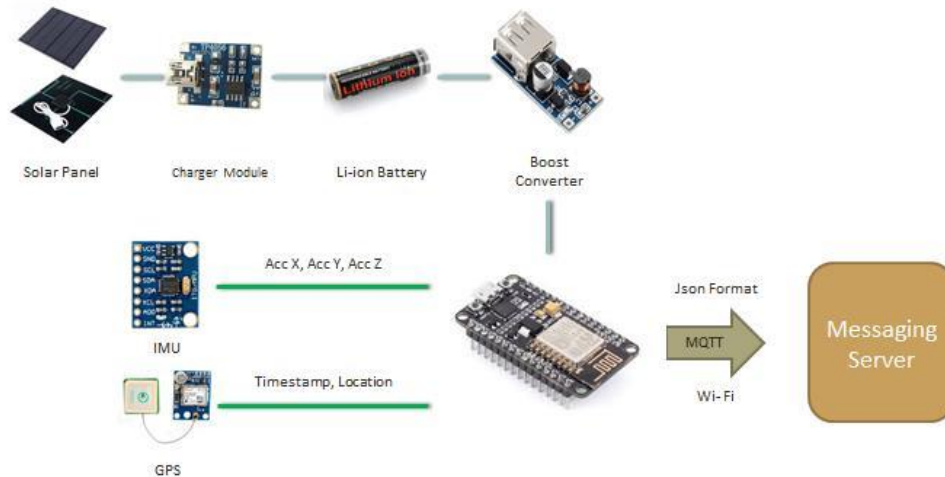
```

```

    public string x;
    public string y;
    public string z;
};
}

```

## 2.2 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 (total harga 3 juta), performa 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 16 bit).

Desain sensor ECN terdiri dari tiga bagian, yaitu modul utama dan modul charge controller. 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.

### 2.1.1 Main Module

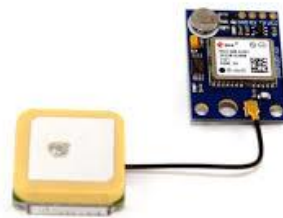
#### 2.1.1.1 Elektrikal

Terdapat tiga komponen utama pada main module, yaitu NodeMCU v2, MPU9255, dan GPS Ublox Neo 6M.



*Gambar NodeMCU v2*

NodeMCU merupakan mikrokontroler 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 40 kali selama 1 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*

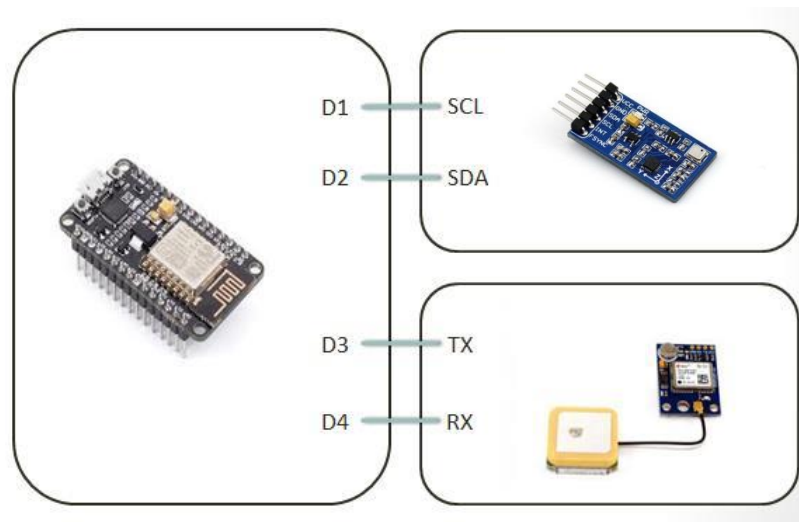
Untuk mendapatkan lokasi sensor dan data waktu yang dapat sinkron dengan sensor-sensor yang lain, digunakan modul GPS Ublox Neo 6M. Modul GPS ini terus menerus mendapat informasi dari satelit dan informasi-informasi tersebut dapat diperoleh NodeMCU dengan menggunakan komunikasi serial. Gelombang yang diterima oleh antenna 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 timestamp yang tersinkronisasi seperti detik, menit, jam, tanggal, bulan, dan tahun



*Gambar Sensor IMU*

Modul yang digunakan untuk mendapatkan data getaran adalah modul MPU9255. Pada modul MPU9255 ini terdapat accelerometer sehingga dapat diperoleh data percepatan dari getaran. Komunikasi yang digunakan modul MPU9255 ini adalah I<sup>2</sup>C. Inisiasi pembacaan data dari MPU9255 dilakukan dengan cara wake-up MPU9255 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.

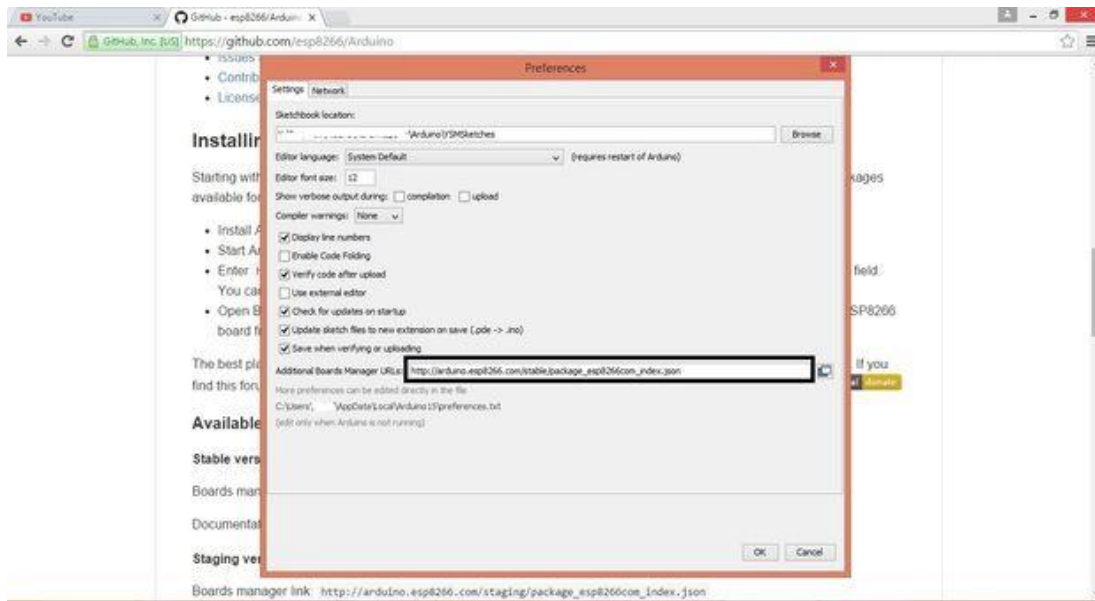


*Gambar Skematik Modul Utama Sensor ECN*

Untuk interkoneksi antar komponen dapat dilihat pada gambar diatas. Pin SCL pada MPU9255 dihubungkan dengan pin D1 pada NodeMCU dan pin SDA pada MPU9255 dihubungkan dengan pin D2 pada NodeMCU agar komunikasi I<sup>2</sup>C 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](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.





```
//=====//
//-----Library & Constant-----//
//=====//

#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include <SoftwareSerial.h>
#include <Wire.h>
#include <TinyGPS++.h>
#include <ArduinoJson.h>

#define SDA_PIN D1
#define SCL_PIN D2
#define PWR_MGMT 0x6B
#define RXPin D3
#define TXPin D4
#define GPSPBaud 9600
#define NData 40 // Amount of Data per one second

const char* TIMEZONE = "Asia/Jakarta";
const char* PROP = "ITB";
const char* TYPE = "Point";
const String SensorID = "1";
const int SendPeriod = 1000; //in ms
const int WaitGPS = 10;
const int N = NData * (SendPeriod/1000); // Amount of Sample
//Initial Coordinate
static const double init lat = -6.889916, init lon = 107.61133;
```

*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 I<sup>2</sup>C yang digunakan untuk memperoleh data dari MPU9255, dan Library TinyGPS++ untuk pembacaan data yang diperoleh dari GPS. Selain itu dilakukan juga deklarasi pin yang digunakan untuk komunikasi I<sup>2</sup>C dan Serial serta deklarasi register-register pada MPU9255. Konstanta GPSPBaud merupakan baudrate yang digunakan untuk berkomunikasi serial dengan modul GPS. Konstanta SendPeriod merupakan periode pengiriman message. Konstanta Ndata adalah jumlah data yang disample dalam waktu satu detik, diisi dengan nilai 40 yang menandakan bahwa frekuensi sampling adalah 40 Hz. Konstanta char TIMEZONE, PROP, TYPE merupakan inisialisasi nilai suatu objek Json. Konstanta SensorID mengimplikasikan bahwa sensor tersebut memiliki ID 1, SensorID ini diperlukan untuk pembeda identitas antar sensor ketika data berada pada server. SendPeriod menandakan bahwa sensor akan mengirimkan data ke server setiap 1 detik. WaitGPS untuk menentukan seberapa lama sensor akan menunggu GPS sampai GPS dapat mengunci komunikasi dengan satelit. Konstanta N mengindikasikan jumlah sampling data yang dikirimkan ke server.

*Jumlah data sampling dalam sekali kirim*

$$= \text{Jumlah sample data dalam satu detik} * \left( \frac{\text{Periode pengiriman data ke server (ms)}}{1000 \text{ ms}} \right)$$



```
//=====//
//=====Connection & Database Variables=====//
//=====//

const char* ssid = "LSKK Basement";    // network SSID (name)
const char* pass = "noiznocon";        // network password
const char* mqtt_server = "167.205.7.226";
const char* server_topic = "amq.topic.ecn"; //MQTT server topic
String mqtt_clientID = "ECN-" + SensorID;
String mqtt_user = "/disaster:sensor_gempa";
String mqtt_password = "12345";
int status = WL_IDLE_STATUS;
WiFiClient espClient;
PubSubClient client(espClient);
```

*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.

```
//=====IMU & GPS Initiation=====//
//=====//

// I2C address of the MPU-9255
#define MPU9250_ADDRESS 0x68
#define MAG_ADDRESS 0x0C

#define ACC_FULL_SCALE_2_G 0x00
#define ACC_FULL_SCALE_4_G 0x08
#define ACC_FULL_SCALE_8_G 0x10
#define ACC_FULL_SCALE_16_G 0x18

#define ACC_FILTER_OFF 0x08

const float ACC_RES = 6.10388817677e-05;
const float MAG_RES = 0.149540696432;
const float G = 9.8;

float axg = 0.0;
float ayg = 0.0;
float azg = 0.0;

float axgf = 0.0;
float aygf = 0.0;
float azgf = 0.0;

float mxg = 0.0;
float myg = 0.0;

float temperaturG = 0.0;

struct MPU9255 {
    float x;
    float y;
    float z;
    float temp;
    float magx;
    float magy;
    float magz;
};

struct times{
    long now;
    long imu;
    long mqtt;
};

MPU9255 data;
times t;

//TinyGPS++ Object
TinyGPSPlus gps;

// The serial connection to the GPS device
SoftwareSerial ss(RXPin, TXPin);
```

*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 I<sup>2</sup>C dari MPU9255. Sedangkan struct MPU9255 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. Struct times digunakan untuk menyimpan data millis() yang digunakan untuk kalkulasi lama proses dari suatu proses. Dapat dilihat juga didefinisikan beberapa register untuk memudahkan jika ingin mengganti konfigurasi penggunaan MPU9255. Konstanta float pada ACC\_RES dan MAG\_RES yang mengindikasikan resolusi pembacaan data percepatan dan magnetometer. Angka tersebut diperoleh dari persamaan sebagai berikut.

$$ACC\_RES = \frac{2}{2^{16-1} - 1} = 6.10388817677e - 05$$

$$MAG\_RES = \frac{4800}{2^{16-1} - 1} = 0.149540696432$$

Persamaan diatas merupakan persamaan untuk menghitung resolusi pembacaan accelerometer dan magnetometer. Nilai 2 & 4800 merupakan nilai maksimum dari pembacaan accelerometer dan magnetometer. Nilai tersebut dapat diubah-ubah dengan mengganti register pada MPU9255.

```
//=====//
//=====JSON OBJECT=====//
//=====//

struct DataIMU {
    String x;
    String y;
    String z;
};

struct MessageData {
    String PointTime;
    String coordinates[2];
    DataIMU acc[N];
};

struct SensorSetting {
    String ClientID;
    String TimeZone;
    String Interval;
    String Properties;
};

struct MessageData payload_data;
struct SensorSetting payload_setting;

int i = 0, j = 0;
bool checkgps = false;
```

*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 dibawah ini.

```
{
  "pointTime": "",
  "timeZone": "Asia/Jakarta",
  "interval": 1000,
  "clientID": "ECN-1",
  "geojson": {
    "geometry": {
      "type": "Point",
      "coordinates": [125.6, 10.1]
    },
    "properties": {
      "name": "ITB"
    }
  },
  "accelerations": [
    {
      "x": -0.1243,
      "y": 14.6464725,
      "z": -12.5433
    }
  ]
}
```

```

void setup()
{
    MPU9255_Init();
    payload_setting = InitJsonObject(payload_setting);
    WiFiConnect();
    client.setServer(mqtt_server, 1883);
    client.setCallback(callback);

    Serial.begin(9600);
    Serial.println();
    ss.begin(GPSBaud);

    while (!checkgps)
    {
        while (ss.available() > 0)
        {
            if (gps.encode(ss.read()))
            {
                displayInfo();
                checkgps = true;
            }
        }
        if (millis() > WaitGPS*SendPeriod && gps.charsProcessed() < 10)
        {
            payload_data.coordinates[0] = String(init_lat);
            payload_data.coordinates[1] = String(init_lon);
            checkgps = true;
        }
    }
}

```

*Gambar Potongan Source Code Setup*

Pada bagian setup Arduino, dilakukan beberapa hal, yaitu inisiasi MPU9255 dan nilai awal objek Json. Setelah itu dilakukan percobaan untuk koneksi dengan access point Wi-Fi. Setelah itu, diatur juga server dan callback untuk protokol MQTT. Pada sensor ini tidak digunakan callback krn tidak menggunakan fitur subscribe MQTT. Setelah itu diinisiasi protokol pembacaan data dari GPS lalu mencoba untuk membaca data longitude dan latitude sampai batas waktu tertentu/konstanta WaitGPS. Ketika sudah berhasil diperoleh data longitude dan latitude, data tersebut disimpan pada objek Json.

```

struct SensorSetting InitJsonObject(struct SensorSetting msg)
{
    msg.ClientID = mqtt_clientID;
    msg.TimeZone = TIMEZONE;
    msg.Interval = String(SendPeriod);
    msg.Properties = PROP;
    return msg;
}

```

*Gambar Potongan Source Code Fungsi InitJsonObject*

Dapat dilihat dari gambar diatas, fungsi InitJsonObject memiliki input objek SensorSetting dan output objek SensorSetting. Pada fungsi ini dilakukan deklarasi awal beberapa variabel dari objek tersebut. Yang dideklarasikan antara lain zona waktu yang digunakan, interval pengiriman, dan ClientID.

Untuk melakukan pembacaan data accelerometer dan magnetometer dari MPU9255 diperlukan beberapa fungsi dasar yaitu, Filter, I2Cread dan I2CwriteByte. Fungsi Filter untuk mengimplementasikan filter lowpass terhadap data accelerometer, fungsi I2Cread untuk membaca data dr suatu device, dan I2CwriteByte digunakan untuk menuliskan data 8 bit thdp suatu device I2C.

```

double Filter(double In, double OutLPS, int FZK)
{
    if(FZK > 0)
    {
        double schritt1 = OutLPS - In;
        double schritt2 = 1.0/FZK;
        double schritt3 = schritt1 * schritt2;
        double wert = OutLPS - schritt3;

        return wert;
    }
    else
    {
        return (In);
    }
}

// This function read Nbytes bytes from I2C device at address Address.
// Put read bytes starting at register Register in the Data array.
void I2Cread(uint8_t Address, uint8_t Register, uint8_t Nbytes, uint8_t* Data)
{
    // Set register address
    Wire.beginTransmission(Address);
    Wire.write(Register);
    Wire.endTransmission();

    // Read Nbytes
    Wire.requestFrom(Address, Nbytes);
    uint8_t index=0;
    while (Wire.available())
        Data[index++]=Wire.read();
}

// Write a byte (Data) in device (Address) at register (Register)
void I2CwriteByte(uint8_t Address, uint8_t Register, uint8_t Data)
{
    // Set register address
    Wire.beginTransmission(Address);
    Wire.write(Register);
    Wire.write(Data);
    Wire.endTransmission();
}

void MPU9255_Init()
{
    Wire.begin(SDA_PIN,SCL_PIN);

    // Set accelerometers low pass filter at 5Hz
    I2CwriteByte(MPU9250_ADDRESS,29,0x06);
}

```

```

// Configure accelerometers range
I2CwriteByte(MPU9250_ADDRESS,28,ACC_FULL_SCALE_2_G);
// accelerometers low pass filter off
I2CwriteByte(MPU9250_ADDRESS,29,ACC_FILTER_OFF);
// Set by pass mode for the magnetometers
I2CwriteByte(MPU9250_ADDRESS,0x37,0x02);

// Request continuous magnetometer measurements in 16 bits
I2CwriteByte(MAG_ADDRESS,0x0A,0x16);

}

```

Dapat dilihat pada fungsi MPU9255\_Init(), hal pertama yang dilakukan adalah menginisiasi penggunaan port untuk I2C. Setelah itu dilakukan untuk mengeset low pass filter 5 Hz, mengatur range pembacaan data accelerometer menjadi antara -2G sampai 2G, mematikan fitur lowpass filter pada MPU9255, mengatur magnetometer menjadi mode pass, dan mengatur untuk pembacaan magnetometer secara kontinu dalam 16 bits.

```

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());
}

```

*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.

```

void loop()
{
    if(!client.loop()) client.connect(mqtt_clientID.c_str(), mqtt_user.c_str(), mqtt_password.c_str());

    if (!client.connected()) {
        reconnect_server();
        i = 0;
    }
    else
    {
        t.now = millis();
        data = Acc_Read();
        t.imu = millis() - t.now ;
        yield();
        payload_data.acc[i].x = char_repr(data.x);
        payload_data.acc[i].y = char_repr(data.y);
        payload_data.acc[i].z = char_repr(data.z);
        if(i==0)
        {
            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());
            payload_data.PointTime = YEAR + "-" + MONTH + "-" + DATE + "T" + HOUR + ":" + MINUTE + ":" + SECOND + "Z";
        }
        i++;

        if (i==N)
        {
            i = 0;
            String message = JsonToString(payload_data,payload_setting);
            char message_t[MQTT_MAX_PACKET_SIZE];
            message.toCharArray(message_t, MQTT_MAX_PACKET_SIZE);

            t.now = millis();
            bool test = client.publish(server_topic, message_t);
            t.mqtt = millis()-t.now;

            if(test){
                Serial.print("publish success ");
                Serial.print(String(t.imu));
                Serial.print(" ");
                Serial.println(String(t.mqtt));
            }

        }

        delay((1000/NData)-t.imu);
    }
}

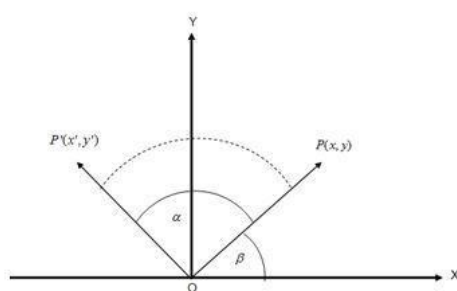
```

*Gambar Potongan Source Code Loop*

Gambar diatas merupakan implementasi algoritma ECN pada saat loop(). Algoritma akan memeriksa apakah sensor terkoneksi dengan server. Jika tidak maka sensor akan terus berusaha untuk melakukan koneksi ulang ke server sampai berhasil. Saat berhasil maka sensor akan mulai untuk melakukan sampling sebanyak N. Pada saat sampling pertama, sensor akan melakukan pembacaan timestamp dan disimpan pada objek Json. Hasil sampling akan diubah representasinya dari float menjadi char. Hal ini bertujuan untuk mengurangi lebar message yang akan dikirim ke server. Ketika sampling selesai dilakukan, Objek Json akan diubah menjadi string dengan format json. String tersebut diubah menjadi char array yang akan dikirimkan ke server. Publish message ke server dilakukan dengan fungsi publish. Setelah itu akan dilakukan fungsi delay yang lamanya tergantung dari lama proses sampling dalam pembacaan sensor IMU MPU9255.



Pada fungsi Acc\_Read dilakukan beberapa proses pembacaan, yaitu pembacaan data accelerometer, temperatur, dan magnetometer. Data temperatur diperlukan untuk memonitor kondisi sensor, krn sensitivitas sensor akan terganggu ketika suhu naik ke tingkat yang seharusnya. Data magnetometer diperlukan untuk menyesuaikan pembacaan data accelerometer terhadap peletakkan sensor. Idealnya sumbu x IMU harus searah dengan kutub utara, ketika terjadi penyimpangan maka pembacaan data accelerometer menjadi berbeda antara beberapa sensor. Sehingga pembacaan accelerometer yang seharusnya memenuhi persamaan sebagai berikut.



$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos \alpha & -\sin \alpha \\ \sin \alpha & \cos \alpha \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

Dengan nilai  $x'$  dan  $y'$  merupakan nilai percepatan dalam sumbu  $x$  dan  $y$  dengan  $x$  searah dengan kutub utara,  $x$  dan  $y$  adalah nilai percepatan dalam sumbu  $x$  dan  $y$  yang belum disesuaikan, dan  $\alpha$  merupakan simpangan antara sumbu  $x$  IMU dengan kutub utara. Untuk metode pembacaan accelerometer dan magnetometer dapat dilakukan dengan membaca dari register 0x3B dan 8 address setelahnya untuk accelerometer dan dari register 0x03 dan 7 address setelahnya untuk magnetometer. Tetapi setelah melakukan pembacaan accelerometer, register data ready harus ditunggu terlebih dahulu sampai bernilai 1, setelah itu pembacaan data magnetometer dapat dilakukan. Untuk lebih jelasnya dapat dilihat source code lengkapnya pada lampiran.



```

void reconnect_server() {
  // Loop until we're reconnected
  while (!client.connected())
  {
    // Serial.print("Attempting MQTT connection...");
    // 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.print(" try again in" );
      Serial.print(String(SendPeriod));
      // Wait 1 Sending Period before retrying
      delay(SendPeriod);
    }
  }
} //end reconnect()

```

*Gambar Potongan Source Code Fungsi reconnect\_server*

Pada fungsi reconnect\_server dilakukan proses inisiasi koneksi ulang lagi ke messaging server CloudAMQP / Server PPTIK. Proses ini terus berlangsung sampai NodeMCU dapat terkoneksi ke server lagi. Status dari proses reconnect server ini dapat dilihat 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.

- -4 : MQTT\_CONNECTION\_TIMEOUT - the server didn't respond within the keepalive time
- -3 : MQTT\_CONNECTION\_LOST - the network connection was broken
- -2 : MQTT\_CONNECT\_FAILED - the network connection failed
- -1 : MQTT\_DISCONNECTED - the client is disconnected cleanly
- 0 : MQTT\_CONNECTED - the client is connected
- 1 : MQTT\_CONNECT\_BAD\_PROTOCOL - the server doesn't support the requested version of MQTT
- 2 : MQTT\_CONNECT\_BAD\_CLIENT\_ID - the server rejected the client identifier
- 3 : MQTT\_CONNECT\_UNAVAILABLE - the server was unable to accept the connection
- 4 : MQTT\_CONNECT\_BAD\_CREDENTIALS - the username/password were rejected
- 5 : MQTT\_CONNECT\_UNAUTHORIZED - the client was not authorized to connect

*Gambar Informasi Mengenai Nilai RC dan Penjelasannya*

```

String JsonToString(struct MessageData msg, struct SensorSetting set)
{
    String a = "";

    a = a + "{" + "\"pointTime\": " + "\"" + msg.PointTime + "\"" + ",";
    a = a + "\"timeZone\": " + "\"" + set.TimeZone + "\"" + ",";
    a = a + "\"interval\": " + "\"" + set.Interval + "\"" + ",";
    a = a + "\"clientID\": " + "\"" + set.ClientID + "\"" + ",";
    a = a + "\"geojson\" : " + "{";

    //a = a + "\"type\": " + "\"" + msg.geometry.type + "\"" + ",";
    a = a + "\"geometry\": " + "{";

    a = a + "\"type\": " + "\"" + TYPE + "\"" + ",";
    a = a + "\"coordinates\": [ " + msg.coordinates[0] + ",";
    a = a + msg.coordinates[1] + "]" + ",";
    a = a + "},";

    a = a + "\"properties\": " + "{";
    a = a + "\"name\": " + "\"" + set.Properties + "\"" + ",";
    a = a + "},";

    a = a + "},";

    a = a + "\"accelerations\": [";

    for (int i = 0; i < N; i++)
    {
        if (i != (N-1))
        {
            a = a + "{";
            a = a + "\"x\": " + msg.acc[i].x + ",";
            a = a + "\"y\": " + msg.acc[i].y + ",";
            a = a + "\"z\": " + msg.acc[i].z + ",";
            a = a + "},";
        }
        else
        {
            a = a + "{";
            a = a + "\"x\": " + msg.acc[i].x + ",";
            a = a + "\"y\": " + msg.acc[i].y + ",";
            a = a + "\"z\": " + msg.acc[i].z + ",";
            a = a + "},";
        }
    }

    a = a + "];

    a = a + "}";

    //Serial.println(a);
    return a;
}

```

*Gambar Potongan Source Code Fungsi JsonToString*

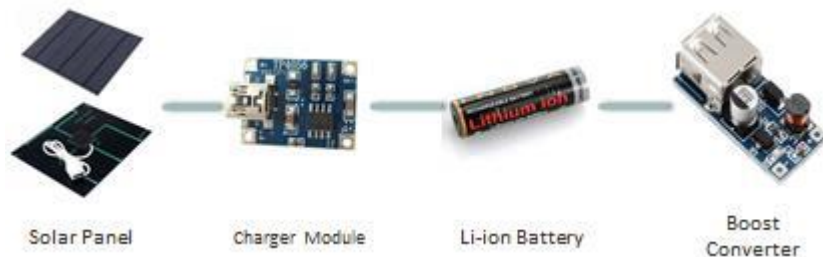
Fungsi JsonToString memiliki input objek Json MessageData dan SensorSetting yang sudah dideklarasikan sebelumnya dengan output dengan tipe data string. Tidak digunakannya library untuk encode dan decode string format Json yang tersedia dikarenakan tingkat kerumitan format Json yang kita gunakan tidak 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.

```
String char_repr(float x)
{
    uint8_t *temp;
    char z[2];
    int16_t y = (int16_t) (x*10000.00);
    temp = (uint8_t *) &y;
    z[0] = (char) temp[0];
    z[1] = (char) temp[1];
    String t = String(z[1]) + String(z[0]);

    return t;
}
```

Untuk mengurangi panjang message saat publish ke messaging server, digunakan teknik penggantian representasi float menjadi representasi char. Input float dikalikan dengan 10000 dan diubah tipe datanya menjadi integer 2 byte. Setelah itu pointer temp menunjuk ke alamat variabel integer 2 byte tsb. Karena setiap address pada arduino berisi 8 byte, maka 8 byte MSB dan 8 byte LSB dapat diakses dengan menggunakan pointer ini. Tiap 8 byte diubah representasinya menjadi char. Lalu kedua buah variabel char itu disatukan. Dengan teknik representasi char ini, panjang data yang di-publish ke server bisa berkurang sebesar 60%-70%.

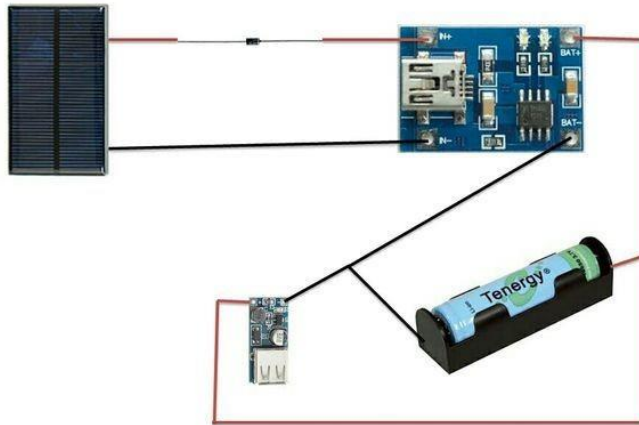
### 2.1.1 Charge Controller Module



*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.

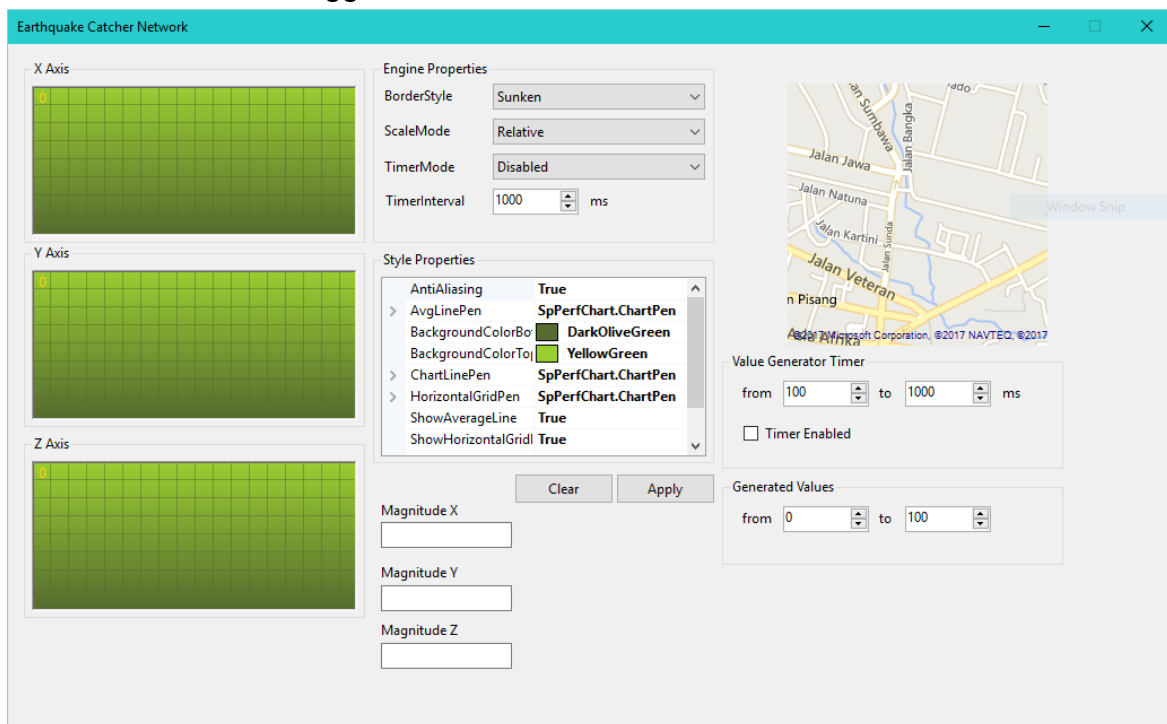
Digunakan solar panel dengan keluaran 5 Volt dan daya 4W sehingga solar panel memiliki kemampuan untuk menyuplai arus yang diperlukan NodeMCU (0,2A) dan charging baterai (0,6A) agar baterai dapat menyuplai sensor pada malam hari. Baterai yang digunakan untuk menyimpan daya adalah 2 baterai Li-Ion dengan 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*

## 2.3 Perangkat Lunak Pengolah Data dan Antarmuka Grafis

### 2.3.1 Antarmuka Pengguna Grafis



Tampilan 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.rm.qcloudamqp.com/lsowqccg";
            factory.Uri =
            "amqp://lsowqccg:kbLv9YbzjQwxz20NH7Rfy98TTV2eK17j@black-
            boar.rm.qcloudamqp.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 = (BorderStyle)Enum.Parse(
            typeof(BorderStyle), 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;
    };
}

```

## 2.3.2 Pengolahan Data

### 2.3.2.1 Akuisisi

Source Code untuk Akuisisi adalah sebagai berikut

```

package gov.usgs.volcanoes.swarm;

import java.awt.BorderLayout;
import java.awt.Dimension;
import java.awt.event.ActionEvent;

```

```

import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;
import java.io.File;
import java.util.ArrayList;
import java.util.Collections;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Set;

```

```

import javax.swing.BorderFactory;
import javax.swing.DefaultListModel;
import javax.swing.JCheckBoxMenuItem;
import javax.swing.JFileChooser;
import javax.swing.JInternalFrame;
import javax.swing.JLabel;
import javax.swing.JList;
import javax.swing.JMenu;
import javax.swing.JMenuBar;
import javax.swing.JMenuItem;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.KeyStroke;
import javax.swing.event.MenuEvent;
import javax.swing.event.MenuListener;

```

```

import gov.usgs.plot.data.file.FileType;
import gov.usgs.volcanoes.core.ui.ExtensionFileFilter;
import gov.usgs.volcanoes.core.util.StringUtils;
import gov.usgs.volcanoes.swarm.data.CachedDataSource;
import gov.usgs.volcanoes.swarm.data.FileDataSource;
import gov.usgs.volcanoes.swarm.internalFrame.InternalFrameListener;
import gov.usgs.volcanoes.swarm.internalFrame.SwarmInternalFrames;
import gov.usgs.volcanoes.swarm.map.MapFrame;
import gov.usgs.volcanoes.swarm.wave.WaveClipboardFrame;

```

```
/**
```

```
*
```

```
* @author Dan Cervelli
```

```
*/
```

```

public class SwarmMenu extends JMenuBar implements InternalFrameListener {
    private static final long serialVersionUID = 1L;
    private JMenu fileMenu;
    private JMenuItem openFile;
    private JMenuItem closeFiles;
    private JMenuItem clearCache;
    private JMenuItem exit;

    private JMenuItem options;

```

```

private String lastLayoutName;
private JMenu layoutMenu;
private JMenuItem saveLayout;
private JMenuItem saveLastLayout;
private JMenuItem removeLayouts;

private JMenu windowMenu;
private JMenuItem tileWaves;
private JMenuItem tileHelicorders;
private JMenuItem fullScreen;
private JCheckBoxMenuItem clipboard;
private JCheckBoxMenuItem chooser;
private JCheckBoxMenuItem map;
private JMenuItem mapToFront;
private JMenuItem closeAll;

private JMenu helpMenu;
private JMenuItem about;

private AboutDialog aboutDialog;

private Map<JInternalFrame, InternalFrameMenuItem> windows;
private Map<SwarmLayout, JMenuItem> layouts;

public SwarmMenu() {
    super();
    windows = new HashMap<JInternalFrame, InternalFrameMenuItem>();
    layouts = new HashMap<SwarmLayout, JMenuItem>();
    createFileMenu();
    createLayoutMenu();
    createWindowMenu();
    createHelpMenu();
    SwarmInternalFrames.addInternalFrameListener(this);
}

private void createFileMenu() {
    fileMenu = new JMenu("File");
    fileMenu.setMnemonic('F');

    openFile = new JMenuItem("Open File...");
    openFile.setMnemonic('O');
    openFile.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            JFileChooser chooser = FileChooser.getFileChooser();
            chooser.resetChoosableFileFilters();
            for (FileType ft : FileType.getKnownTypes()) {
                ExtensionFileFilter f = new ExtensionFileFilter(ft.extension, ft.description);
                chooser.addChoosableFileFilter(f);
            }
        }
    });
}

```

```

        chooser.setFileFilter(chooser.getAcceptAllFileFilter());
        File lastPath = new File(SwarmConfig.getInstance().lastPath);
        chooser.setCurrentDirectory(lastPath);
        chooser.setFileSelectionMode(JFileChooser.FILES_ONLY);
        chooser.setMultiSelectionEnabled(true);
        chooser.setDialogTitle("Open Wave as Data Source");
        int result = chooser.showOpenDialog(Swarm.getApplicationFrame());
        if (result == JFileChooser.APPROVE_OPTION) {
            File[] fs = chooser.getSelectedFiles();
            FileDataSource.getInstance().openFiles(fs);
        }
    }
});
openFile.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_O,
KeyEvent.CTRL_DOWN_MASK));
fileMenu.add(openFile);

closeFiles = new JMenuItem("Close Files");
closeFiles.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        FileDataSource.getInstance().flush();
    }
});
closeFiles.setMnemonic('l');
fileMenu.add(closeFiles);

clearCache = new JMenuItem("Clear Cache");
clearCache.setMnemonic('C');
clearCache.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        CachedDataSource cache = CachedDataSource.getInstance();
        if (cache != null)
            cache.flush();
    }
});
clearCache.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_F12,
KeyEvent.CTRL_DOWN_MASK));
fileMenu.add(clearCache);
fileMenu.addSeparator();

options = new JMenuItem("Options...");
options.setMnemonic('O');
options.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        OptionsDialog od = new OptionsDialog();
        od.setVisible(true);
    }
});
fileMenu.add(options);

```

```

fileMenu.addSeparator();

exit = new JMenuItem("Exit");
exit.setMnemonic('x');
exit.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        Swarm.getApplication().closeApp();
    }
});
fileMenu.add(exit);
add(fileMenu);

fileMenu.addMenuListener(new MenuListener() {
    public void menuSelected(MenuEvent e) {
        CachedDataSource cache = CachedDataSource.getInstance();
        clearCache.setEnabled(!cache.isEmpty());
    }

    public void menuDeselected(MenuEvent e) {
    }

    public void menuCanceled(MenuEvent e) {
    }
});
}

private void createLayoutMenu() {
    layoutMenu = new JMenu("Layout");
    layoutMenu.setMnemonic('L');

    saveLayout = new JMenuItem("Save Layout...");
    saveLayout.setMnemonic('S');
    saveLayout.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            Swarm.getApplication().saveLayout(null);
        }
    });
    saveLayout.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_L,
    KeyEvent.CTRL_DOWN_MASK));
    layoutMenu.add(saveLayout);

    saveLastLayout = new JMenuItem("Overwrite Last Layout...");
    saveLastLayout.setMnemonic('L');
    saveLastLayout.setEnabled(false);
    saveLastLayout.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            if (lastLayoutName != null)
                Swarm.getApplication().saveLayout(lastLayoutName);
        }
    });
}

```

```

        saveLastLayout.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_L,
KeyEvent.CTRL_DOWN_MASK
        | KeyEvent.SHIFT_DOWN_MASK));
        layoutMenu.add(saveLastLayout);

        removeLayouts = new JMenuItem("Remove Layout...");
        removeLayouts.setMnemonic('R');
        removeLayouts.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                RemoveLayoutDialog d = new RemoveLayoutDialog();
                d.setVisible(true);
            }
        });
        layoutMenu.add(removeLayouts);
        layoutMenu.addSeparator();
        add(layoutMenu);
    }

    public String getLastLayoutName() {
        return lastLayoutName;
    }

    public void setLastLayoutName(String ln) {
        lastLayoutName = ln;
        saveLastLayout.setEnabled(true);
        saveLastLayout.setText("Overwrite Last Layout (" + ln + ")");
    }

    public void addLayout(final SwarmLayout sl) {
        JMenuItem mi = new JMenuItem(sl.getName());
        mi.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                sl.process();
                setLastLayoutName(sl.getName());
            }
        });
        int i;
        for (i = 4; i < layoutMenu.getItemCount(); i++) {
            JMenuItem m = layoutMenu.getItem(i);
            if (m.getText().compareToIgnoreCase(sl.getName()) >= 0) {
                layoutMenu.add(mi, i);
                break;
            }
        }
        if (i == layoutMenu.getItemCount())
            layoutMenu.add(mi, i);
        layouts.put(sl, mi);
    }

    public void removeLayout(SwarmLayout sl) {

```



```

JMenuItem mi = layouts.get(sl);
layoutMenu.remove(mi);
layouts.remove(sl);
}

private void createWindowMenu() {
    windowMenu = new JMenu("Window");
    windowMenu.setMnemonic('W');

    chooser = new JCheckBoxMenuItem("Data Chooser");
    chooser.setMnemonic('D');
    chooser.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {

Swarm.getApplication().setChooserVisible(!Swarm.getApplication().isChooserVisible());
        }
    });
    chooser.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_D,
KeyEvent.CTRL_DOWN_MASK));
    windowMenu.add(chooser);

    clipboard = new JCheckBoxMenuItem("Wave Clipboard");
    clipboard.setMnemonic('W');
    clipboard.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            WaveClipboardFrame waveClipboard = WaveClipboardFrame.getInstance();
            waveClipboard.setVisible(!waveClipboard.isVisible());
        }
    });
    clipboard.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_W,
KeyEvent.CTRL_DOWN_MASK));
    windowMenu.add(clipboard);

    final MapFrame mapFrame = MapFrame.getInstance();
    map = new JCheckBoxMenuItem("Map");
    map.setMnemonic('M');
    map.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            mapFrame.setVisible(!mapFrame.isVisible());
        }
    });
    map.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_M,
KeyEvent.CTRL_DOWN_MASK));
    windowMenu.add(map);

    mapToFront = new JMenuItem("Bring Map to Front");
    mapToFront.setMnemonic('F');
    mapToFront.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            mapFrame.setVisible(true);

```

```

    }
});
mapToFront.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_M, 0));
windowMenu.add(mapToFront);
windowMenu.addSeparator();

tileHelicorders = new JMenuItem("Tile Helicorders");
tileHelicorders.setMnemonic('H');
tileHelicorders.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        Swarm.getApplication().tileHelicorders();
    }
});
windowMenu.add(tileHelicorders);

tileWaves = new JMenuItem("Tile Waves");
tileWaves.setMnemonic('v');
tileWaves.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        Swarm.getApplication().tileWaves();
    }
});
windowMenu.add(tileWaves);

windowMenu.addSeparator();

fullScreen = new JMenuItem("Kiosk Mode");
fullScreen.setMnemonic('K');
fullScreen.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        Swarm.getApplication().toggleFullScreenMode();
    }
});
fullScreen.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_F11, 0));
windowMenu.add(fullScreen);

windowMenu.addMenuListener(new MenuListener() {
    public void menuSelected(MenuEvent e) {
        clipboard.setSelected(WaveClipboardFrame.getInstance().isVisible());
        chooser.setSelected(Swarm.getApplication().isChooserVisible());
        map.setSelected(MapFrame.getInstance().isVisible());
    }

    public void menuDeselected(MenuEvent e) {
    }

    public void menuCanceled(MenuEvent e) {
    }
});

```

```

windowMenu.addSeparator();

closeAll = new JMenuItem("Close All");
closeAll.setMnemonic('C');
closeAll.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        SwarmInternalFrames.removeAllFrames();
    }
});
windowMenu.add(closeAll);

add(windowMenu);
}

private void createHelpMenu() {
    helpMenu = new JMenu("Help");
    helpMenu.setMnemonic('H');
    about = new JMenuItem("About...");
    about.setMnemonic('A');
    aboutDialog = new AboutDialog();
    about.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            aboutDialog.update();
            aboutDialog.setVisible(true);
        }
    });

    helpMenu.add(about);
    add(helpMenu);
}

private class InternalFrameMenuItem extends JMenuItem {
    private static final long serialVersionUID = 1L;
    private JInternalFrame frame;

    public InternalFrameMenuItem(JInternalFrame f) {
        frame = f;
        setText(f.getTitle());
        setIcon(f.getFrameIcon());
        addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                try {
                    if (frame.isIcon())
                        frame.setIcon(false);
                    frame.toFront();
                    frame.setSelected(true);
                } catch (Exception ex) {
                }
            }
        });
    }
}

```

```

    }
}

private class RemoveLayoutDialog extends SwarmModalDialog {
    private static final long serialVersionUID = 1L;
    private JList layoutList;
    private DefaultListModel model;

    protected RemoveLayoutDialog() {
        super(Swarm.getApplicationFrame(), "Remove Layouts");
        setSizeAndLocation();
    }

    protected void createUI() {
        super.createUI();
        Set<String> keys = swarmConfig.layouts.keySet();
        List<String> sls = new ArrayList<String>();
        sls.addAll(keys);
        Collections.sort(sls, StringUtils.getCaseInsensitiveStringComparator());
        model = new DefaultListModel();
        for (String sl : sls)
            model.addElement(sl);
        layoutList = new JList(model);
        JPanel panel = new JPanel(new BorderLayout());
        panel.setBorder(BorderFactory.createEmptyBorder(5, 9, 5, 9));
        int h = Math.max(200, Math.min(350, sls.size() * 19));
        panel.setPreferredSize(new Dimension(200, h));
        panel.add(new JLabel("Select layouts to remove:"), BorderLayout.NORTH);
        panel.add(new JScrollPane(layoutList), BorderLayout.CENTER);
        mainPanel.add(panel, BorderLayout.CENTER);
    }

    public void wasOK() {
        Object[] toRemove = layoutList.getSelectedValues();

        for (Object key : toRemove) {
            SwarmLayout layout = swarmConfig.layouts.get((String)key);
            if (layout != null) {
                JMenuItem mi = layouts.get(layout);
                layoutMenu.remove(mi);
                swarmConfig.removeLayout(layout);
            }
        }
    }

    public void internalFrameAdded(JInternalFrame f) {
        InternalFrameMenuItem mi = new InternalFrameMenuItem(f);
        windows.put(f, mi);
        windowMenu.add(mi);
    }
}

```

```

    }

    public void internalFrameRemoved(JInternalFrame f) {
        InternalFrameMenuItem mi = windows.get(f);
        windows.remove(f);
        windowMenu.remove(mi);
    }
}

```

### 2.3.2.2 Pemrosesan

#### 2.3.2.2.1 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.

```

package gov.usgs.volcanoes.swarm;

import com.jgoodies.looks.plastic.Plastic3DLookAndFeel;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import java.awt.Dimension;
import java.awt.Frame;
import java.awt.KeyboardFocusManager;
import java.awt.Point;
import java.awt.event.ActionEvent;
import java.awt.event.FocusEvent;
import java.awt.event.FocusListener;
import java.awt.event.InputEvent;
import java.awt.event.KeyEvent;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.util.ArrayList;

import javax.swing.AbstractAction;

```

```
import javax.swing.BorderFactory;
import javax.swing.JDesktopPane;
import javax.swing.JFrame;
import javax.swing.JInternalFrame;
import javax.swing.JOptionPane;
import javax.swing.JSplitPane;
import javax.swing.KeyStroke;
import javax.swing.SwingUtilities;
import javax.swing.UIManager;
```

```
import gov.usgs.plot.data.Wave;
import gov.usgs.volcanoes.core.configfile.ConfigFile;
import gov.usgs.volcanoes.core.quakeml.Event;
import gov.usgs.volcanoes.core.time.CurrentTime;
import gov.usgs.volcanoes.core.time.J2kSec;
import gov.usgs.volcanoes.core.ui.GlobalKeyManager;
import gov.usgs.volcanoes.core.util.StringUtils;
import gov.usgs.volcanoes.swarm.chooser.DataChooser;
import gov.usgs.volcanoes.swarm.data.CachedDataSource;
import gov.usgs.volcanoes.swarm.data.SeismicDataSource;
import gov.usgs.volcanoes.swarm.event.EventFrame;
import gov.usgs.volcanoes.swarm.heli.HelicorderViewerFrame;
import gov.usgs.volcanoes.swarm.internalFrame.InternalFrameListener;
import gov.usgs.volcanoes.swarm.internalFrame.SwarmInternalFrames;
import gov.usgs.volcanoes.swarm.map.MapFrame;
import gov.usgs.volcanoes.swarm.rsam.RsamViewerFrame;
import gov.usgs.volcanoes.swarm.wave.MultiMonitor;
import gov.usgs.volcanoes.swarm.wave.WaveClipboardFrame;
import gov.usgs.volcanoes.swarm.wave.WaveViewPanel;
import gov.usgs.volcanoes.swarm.wave.WaveViewerFrame;
```

```
/**
```

```
 * The main UI and application class for Swarm. Only functions directly pertaining to the
UI and
```

```
 * overall application operation belong here.
```

```
 *
```

```
 * TODO: resize listener TODO: chooser visibility TODO: name worker thread for better
debugging
```

```
 *
```

```
 * @author Dan Cervelli, Peter Cervelli, and Thomas Parker.
```

```
 */
```

```
public class Swarm extends JFrame implements InternalFrameListener {
```

```
    private static final Logger LOGGER = LoggerFactory.getLogger(Swarm.class);
```

```
    private static final long serialVersionUID = -1;
```

```
    private static Swarm application;
```

```
    private static JFrame applicationFrame;
```

```
    private static JDesktopPane desktop;
```

```
    private JSplitPane split;
```

```

private SwarmMenu swarmMenu;
private final CachedDataSource cache;

private static final String TITLE = "Swarm";

private static final int LEFT = 1;
private static final int RIGHT = 2;
private static final int TOP = 3;
private static final int BOTTOM = 4;
private static final int BOTTOM_LEFT = 5;
private static final int BOTTOM_RIGHT = 6;
private static final int TOP_LEFT = 7;
private static final int TOP_RIGHT = 8;

private static boolean fullScreen = false;
private int oldState = 0;
private Dimension oldSize;
private Point oldLocation;
private AbstractAction toggleFullScreenAction;
private static SwarmConfig config;
private String lastLayout = "";

public Swarm(final String[] args) {
    super(TITLE + " [" + Version.POM_VERSION + "]");
    LOGGER.info("Swarm version/date: " + Version.VERSION_STRING);
    application = this;
    applicationFrame = this;
    setIconImage(Icons.swarm.getImage());

    config = SwarmConfig.getInstance();
    config.createConfig(args);

    cache = CachedDataSource.getInstance();

    SwarmInternalFrames.addInternalFrameListener(this);
    checkJavaVersion();
    setupGlobalKeys();
    createUI();
}

private void checkJavaVersion() {
    final String version = System.getProperty("java.version");
    LOGGER.info("java.version: " + version);
    if (version.startsWith("1.1") || version.startsWith("1.2") || version.startsWith("1.3")
        || version.startsWith("1.4")) {
        JOptionPane.showMessageDialog(this, String
            .format("%s %s requires at least Java version 1.5 or above.", TITLE,
                Version.POM_VERSION),
            "Error", JOptionPane.ERROR_MESSAGE);
        System.exit(-1);
    }
}

```

```

    }

    final Runtime rt = Runtime.getRuntime();
    LOGGER.info("maximum heap size: " +
StringUtils.numBytesToString(rt.maxMemory()));
}

private void setupGlobalKeys() {
    // clean this up a bit and decide if I really want to use this ghkm
    // thingy
    final GlobalKeyManager m = GlobalKeyManager.getInstance();
    m.getInputMap().put(KeyStroke.getKeyStroke("F12"), "focus");
    m.getActionMap().put("focus", new AbstractAction() {
        private static final long serialVersionUID = -1;

        public void actionPerformed(final ActionEvent e) {
            final KeyboardFocusManager kfm =
KeyboardFocusManager.getCurrentKeyboardFocusManager();
            System.out.println("Focus check: \n" + "Current window: " +
kfm.getFocusedWindow() + "\n\n"
+ "Current component: " + kfm.getFocusOwner() + "\n");
        }
    });

    m.getInputMap().put(KeyStroke.getKeyStroke("alt F12"), "outputcache");
    m.getActionMap().put("outputcache", new AbstractAction() {
        private static final long serialVersionUID = -1;

        public void actionPerformed(final ActionEvent e) {
            if (cache != null)
                cache.output();
        }
    });

    m.getInputMap().put(KeyStroke.getKeyStroke("ctrl L"), "savelayouth");
    m.getActionMap().put("savelayouth", new AbstractAction() {
        private static final long serialVersionUID = -1;

        public void actionPerformed(final ActionEvent e) {
            saveLayout(null);
        }
    });

    m.getInputMap().put(KeyStroke.getKeyStroke("ctrl shift L"), "savelayouthlast");
    m.getActionMap().put("savelayouthlast", new AbstractAction() {
        private static final long serialVersionUID = -1;

        public void actionPerformed(final ActionEvent e) {
            final String ll = swarmMenu.getLastLayoutName();
            if (ll != null)

```



```

        saveLayout(ll);
    }
});

m.getInputMap().put(KeyStroke.getKeyStroke(KeyEvent.VK_F11,
"fullScreenToggle");
m.getInputMap().put(KeyStroke.getKeyStroke(KeyEvent.VK_F11,
InputEvent.CTRL_DOWN_MASK),
"fullScreenToggle");
m.getInputMap().put(KeyStroke.getKeyStroke(KeyEvent.VK_BACK_SLASH,
InputEvent.CTRL_DOWN_MASK),
"fullScreenToggle");
toggleFullScreenAction = new AbstractAction() {
    private static final long serialVersionUID = -1;

    public void actionPerformed(final ActionEvent e) {
        toggleFullScreenMode();
    }
};
m.getActionMap().put("fullScreenToggle", toggleFullScreenAction);

m.getInputMap().put(KeyStroke.getKeyStroke(KeyEvent.VK_RIGHT,
InputEvent.CTRL_DOWN_MASK),
"flushRight");
m.getActionMap().put("flushRight", new AbstractAction() {
    private static final long serialVersionUID = -1;

    public void actionPerformed(final ActionEvent e) {
        flushRight();
    }
});

m.getInputMap().put(KeyStroke.getKeyStroke(KeyEvent.VK_LEFT,
InputEvent.CTRL_DOWN_MASK),
"flushLeft");
m.getActionMap().put("flushLeft", new AbstractAction() {
    private static final long serialVersionUID = -1;

    public void actionPerformed(final ActionEvent e) {
        flushLeft();
    }
});

m.getInputMap().put(KeyStroke.getKeyStroke(KeyEvent.VK_UP,
InputEvent.CTRL_DOWN_MASK),
"flushTop");
m.getActionMap().put("flushTop", new AbstractAction() {
    private static final long serialVersionUID = -1;

    public void actionPerformed(final ActionEvent e) {

```

```

        flushTop();
    }
});

m.getInputMap().put(KeyStroke.getKeyStroke(KeyEvent.VK_DOWN,
InputEvent.CTRL_DOWN_MASK),
    "flushBottom");
m.getActionMap().put("flushBottom", new AbstractAction() {
    private static final long serialVersionUID = -1;

    public void actionPerformed(final ActionEvent e) {
        flushBottom();
    }
});

m.getInputMap().put(KeyStroke.getKeyStroke(KeyEvent.VK_PAGE_DOWN,
InputEvent.CTRL_DOWN_MASK),
    "flushBottomRight");
m.getActionMap().put("flushBottomRight", new AbstractAction() {
    private static final long serialVersionUID = -1;

    public void actionPerformed(final ActionEvent e) {
        flushBottomRight();
    }
});

m.getInputMap().put(KeyStroke.getKeyStroke(KeyEvent.VK_END,
InputEvent.CTRL_DOWN_MASK),
    "flushBottomLeft");
m.getActionMap().put("flushBottomLeft", new AbstractAction() {
    private static final long serialVersionUID = -1;

    public void actionPerformed(final ActionEvent e) {
        flushBottomLeft();
    }
});

m.getInputMap().put(KeyStroke.getKeyStroke(KeyEvent.VK_PAGE_UP,
InputEvent.CTRL_DOWN_MASK),
    "flushTopRight");
m.getActionMap().put("flushTopRight", new AbstractAction() {
    private static final long serialVersionUID = -1;

    public void actionPerformed(final ActionEvent e) {
        flushTopRight();
    }
});

m.getInputMap().put(KeyStroke.getKeyStroke(KeyEvent.VK_HOME,
InputEvent.CTRL_DOWN_MASK),

```

```

        "flushTopLeft");
m.getActionMap().put("flushTopLeft", new AbstractAction() {
    private static final long serialVersionUID = -1;

    public void actionPerformed(final ActionEvent e) {
        flushTopLeft();
    }
});

}

@Deprecated
public static Swarm getApplication() {
    return application;
}

public static JFrame getApplicationFrame() {
    return applicationFrame;
}

private void createUI() {
    this.addWindowListener(new WindowAdapter() {
        @Override
        public void windowClosing(final WindowEvent e) {
            closeApp();
        }
    });
    this.addFocusListener(new FocusListener() {
        public void focusGained(final FocusEvent e) {
            // The main Swarm window has no need for the focus. If it gets
            // it
            // then it attempts to pass it on to the first helicorder,
            // failing
            // that it gives it to the first wave.
            if (SwarmInternalFrames.frameCount() > 0) {
                JInternalFrame jf = null;
                for (int i = 0; i < SwarmInternalFrames.frameCount(); i++) {
                    final JInternalFrame f = SwarmInternalFrames.getFrames().get(i);
                    if (f instanceof HelicorderViewerFrame) {
                        jf = f;
                        break;
                    }
                }
                if (jf == null)
                    jf = SwarmInternalFrames.getFrames().get(0);
                jf.requestFocus();
            }
        }
    });

    public void focusLost(final FocusEvent e) {}
}

```

```

});

desktop = new JDesktopPane();
desktop.setBorder(BorderFactory.createLineBorder(DataChooser.LINE_COLOR));
desktop.setDragMode(JDesktopPane.OUTLINE_DRAG_MODE);
// disable dragging in fullscreen mode
/*
 * desktop.setDesktopManager(new DefaultDesktopManager() { private static final long
 * serialVersionUID = -1; public void beginDraggingFrame(JComponent f) { if
(fullScreen) return;
 * else super.beginDraggingFrame(f); }
 *
 * public void dragFrame(JComponent f, int x, int y) { if (fullScreen) return; else
 * super.dragFrame(f, x, y); } });
 */
this.setSize(config.windowWidth, config.windowHeight);
this.setLocation(config.windowX, config.windowY);
if (config.windowMaximized)
    this.setExtendedState(Frame.MAXIMIZED_BOTH);

this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

final DataChooser chooser = DataChooser.getInstance();
split = SwarmUtil.createStrippedSplitPane(JSplitPane.HORIZONTAL_SPLIT, chooser,
desktop);
split.setDividerLocation(config.chooserDividerLocation);
split.setDividerSize(4);
setChooserVisible(config.chooserVisible);
chooser.setDividerLocation(config.nearestDividerLocation);

final WaveClipboardFrame waveClipboard = WaveClipboardFrame.getInstance();
desktop.add(waveClipboard);
waveClipboard.setVisible(config.clipboardVisible);
if (Swarm.config.clipboardMaximized) {
    try {
        waveClipboard.setMaximum(true);
    } catch (final Exception e) {
    }
}

final MapFrame mapFrame = MapFrame.getInstance();
SwarmInternalFrames.add(mapFrame);
mapFrame.setVisible(config.mapVisible);
if (Swarm.config.mapMaximized) {
    try {
        mapFrame.setMaximum(true);
    } catch (final Exception e) {
    }
}

```

```

mapFrameToFront();

swarmMenu = new SwarmMenu();
this.setJMenuBar(swarmMenu);

for (final SwarmLayout sl : config.layouts.values())
    swarmMenu.addLayout(sl);

this.setVisible(true);

final long offset = CurrentTime.getInstance().getOffset();
if (Math.abs(offset) > 10 * 60 * 1000)
    JOptionPane.showMessageDialog(this,
        "You're system clock is off by more than 10 minutes.\n"
        + "This is just for your information, Swarm will not be affected by this.",
        "System Clock", JOptionPane.INFORMATION_MESSAGE);
}

public void setChooserVisible(final boolean vis) {
    if (vis) {
        split.setRightComponent(desktop);
        split.setDividerLocation(config.chooserDividerLocation);
        setContentPane(split);
    } else {
        if (isChooserVisible())
            config.chooserDividerLocation = split.getDividerLocation();
        setContentPane(desktop);
    }
    if (SwingUtilities.isEventDispatchThread())
        validate();
}

public boolean isChooserVisible() {
    return getContentPane() == split;
}

public static boolean isFullScreenMode() {
    return fullScreen;
}

public void toggleFullScreenMode() {
    setFullScreenMode(!fullScreen);
}

public void setFullScreenMode(final boolean full) {
    if (fullScreen == full)
        return;

    requestFocus();
    fullScreen = full;
}

```

```

this.dispose();
this.setUndecorated(full);
this.setResizable(!full);

final WaveClipboardFrame waveClipboard = WaveClipboardFrame.getInstance();
waveClipboard.setVisible(!full);
waveClipboard.toBack();

// Does this do anything?
this.setVisible(true);

if (full) {
    this.setJMenuBar(null);
    config.chooserDividerLocation = split.getDividerLocation();
    oldState = this.getExtendedState();
    oldSize = this.getSize();
    oldLocation = this.getLocation();
    this.setContentPane(desktop);
    this.setVisible(true);
    this.setExtendedState(Frame.MAXIMIZED_BOTH);
    desktop.setSize(this.getSize());
    desktop.setPreferredSize(this.getSize());
} else {
    this.setJMenuBar(swarmMenu);
    this.setExtendedState(oldState);
    this.setSize(oldSize);
    this.setLocation(oldLocation);
    this.setVisible(true);
    split.setRightComponent(desktop);
    split.setDividerLocation(config.chooserDividerLocation);
    this.setContentPane(split);
}

for (final JInternalFrame frame : SwarmInternalFrames.getFrames()) {
    // Why did this check isVisible()?
    // if (frame.isVisible() && frame instanceof Kioskable) {
    if (frame instanceof Kioskable) {
        final Kioskable f = (Kioskable) frame;
        f.setKioskMode(full);
    } else {
        frame.setVisible(!full);
    }
}
validate();

tileKioskFrames();
}

public void closeApp() {
    final Point p = this.getLocation();

```

```

if (this.getExtendedState() == Frame.MAXIMIZED_BOTH)
    config.windowMaximized = true;
else {
    final Dimension d = this.getSize();
    config.windowX = p.x;
    config.windowY = p.y;
    config.windowWidth = d.width;
    config.windowHeight = d.height;
    config.windowMaximized = false;
}

final WaveClipboardFrame waveClipboard = WaveClipboardFrame.getInstance();
if (waveClipboard.isMaximum())
    config.clipboardMaximized = true;
else {
    config.clipboardX = waveClipboard.getX();
    config.clipboardY = waveClipboard.getY();
    config.clipboardWidth = waveClipboard.getWidth();
    config.clipboardHeight = waveClipboard.getHeight();
    config.clipboardMaximized = false;
}
config.clipboardVisible = WaveClipboardFrame.getInstance().isVisible();

final MapFrame mapFrame = MapFrame.getInstance();
if (mapFrame.isMaximum())
    config.mapMaximized = true;
else {
    config.mapX = mapFrame.getX();
    config.mapY = mapFrame.getY();
    config.mapWidth = mapFrame.getWidth();
    config.mapHeight = mapFrame.getHeight();
    config.mapMaximized = false;
}
config.mapVisible = mapFrame.isVisible();

config.chooserDividerLocation = split.getDividerLocation();
config.chooserVisible = isChooserVisible();

DataChooser.getInstance().updateConfig(config);

config.kiosk = Boolean.toString(fullScreen);

if (config.saveConfig) {
    final ConfigFile configFile = config.toConfigFile();
    configFile.remove("configFile");
    configFile.writeToFile(config.configFilename);
}

```

```

waveClipboard.removeWaves();
try {
    for (final JInternalFrame frame : SwarmInternalFrames.getFrames())
        frame.setClosed(true);
} catch (final Exception e) {
} // doesn't matter at this point
System.exit(0);
}

```

```

public static void loadClipboardWave(final SeismicDataSource source, final String
channel) {

```

```

    final WaveViewPanel wvp = new WaveViewPanel();
    wvp.setChannel(channel);
    wvp.setDataSource(source);
    final WaveViewPanel cwvp = WaveClipboardFrame.getInstance().getSelected();
    double st = 0;
    double et = 0;
    if (cwvp == null) {
        final double now = J2kSec.now();
        st = now - 180;
        et = now;
    } else {
        st = cwvp.getStartTime();
        et = cwvp.getEndTime();
    }
    final double fst = st;
    final double fet = et;

```

```

final SwingWorker worker = new SwingWorker() {
    WaveClipboardFrame waveClipboard = WaveClipboardFrame.getInstance();

```

@Override

```

public Object construct() {
    // double now = CurrentTime.nowJ2K();
    waveClipboard.getThrobber().increment();
    final Wave sw = source.getWave(channel, fst, fet);
    wvp.setWave(sw, fst, fet);
    return null;
}

```

@Override

```

public void finished() {
    waveClipboard.getThrobber().decrement();
    waveClipboard.setVisible(true);
    waveClipboard.toFront();
    try {
        waveClipboard.setSelected(true);
    } catch (final Exception e) {
    }
    waveClipboard.addWave(wvp);
}

```



```

    }
};
worker.start();
}

public static WaveViewerFrame openRealtimeWave(final SeismicDataSource source,
    final String channel) {
    final WaveViewerFrame frame = new WaveViewerFrame(source, channel);
    SwarmInternalFrames.add(frame);
    return frame;
}

public static HelicorderViewerFrame openHelicorder(final SeismicDataSource source,
    final String channel, final double time) {
    source.establish();
    final HelicorderViewerFrame frame = new HelicorderViewerFrame(source, channel,
time);
    frame.addLinkListeners();
    SwarmInternalFrames.add(frame);
    return frame;
}

public static RsamViewerFrame openRsam(final SeismicDataSource source, final String
channel) {
    final RsamViewerFrame frame = new RsamViewerFrame(source, channel);
    SwarmInternalFrames.add(frame);
    return frame;
}

// public static PickerFrame openPicker(final WaveViewPanel insetWavePanel) {
//     PickerWavePanel p = new PickerWavePanel(insetWavePanel);
//     p.setDataSource(insetWavePanel.getDataSource().getCopy());
//     PickerFrame pickerFrame = new PickerFrame();
//     pickerFrame.setVisible(true);
//     pickerFrame.requestFocus();
//     SwarmInternalFrames.add(pickerFrame);
//     pickerFrame.setBaseWave(p);
//     return pickerFrame;
// }

// public static PickerFrame openPicker(Event event) {
//     PickerFrame pickerFrame = new PickerFrame(event);
//     pickerFrame.setVisible(true);
//     pickerFrame.requestFocus();
//     SwarmInternalFrames.add(pickerFrame);
//     return pickerFrame;
// }

public void saveLayout(String name) {

```

```

final boolean fixedName = (name != null);
final SwarmLayout sl = getCurrentLayout();
boolean done = false;
while (!done) {
    if (name == null) {
        name = (String) JOptionPane.showInputDialog(this, "Enter a name for this layout:",
            "Save Layout", JOptionPane.INFORMATION_MESSAGE, null, null, lastLayout);
        name = name.trim();
    }
    if (name != null && !"".equals(name)) {
        if (Swarm.config.layouts.containsKey(name)) {
            boolean overwrite = false;
            if (!fixedName) {
                final int opt = JOptionPane.showConfirmDialog(this,
                    "A layout by that name already exists. Overwrite?", "Warning",
                    JOptionPane.YES_NO_OPTION);
                overwrite = (opt == JOptionPane.YES_OPTION);
            } else
                overwrite = true;

            if (overwrite) {
                if (fixedName) {
                    JOptionPane.showMessageDialog(this, "Layout overwritten.");
                }
                swarmMenu.removeLayout(Swarm.config.layouts.get(name));
                Swarm.config.removeLayout(Swarm.config.layouts.get(name));
            }
        }

        if (!Swarm.config.layouts.containsKey(name)) {
            swarmMenu.setLastLayoutName(name);
            sl.setName(name);
            sl.save();
            swarmMenu.addLayout(sl);
            Swarm.config.addLayout(sl);
            done = true;
            lastLayout = name;
        }
    } else
        done = true; // cancelled
}
}

```

```

public SwarmLayout getCurrentLayout() {
    final ConfigFile cf = new ConfigFile();
    cf.put("name", "Current Layout");
    cf.put("kiosk", Boolean.toString(isFullScreenMode()));
    cf.put("kioskX", Integer.toString(getX()));
    cf.put("kioskY", Integer.toString(getY()));
}

```

```

DataChooser.getInstance().saveLayout(cf, "chooser");

final SwarmLayout sl = new SwarmLayout(cf);
int i = 0;
for (final JInternalFrame frame : SwarmInternalFrames.getFrames()) {
    if (frame instanceof HelicorderViewerFrame) {
        final HelicorderViewerFrame hvf = (HelicorderViewerFrame) frame;
        hvf.saveLayout(cf, "helicorder-" + i++);
    } else if (frame instanceof MultiMonitor) {
        final MultiMonitor mm = (MultiMonitor) frame;
        mm.saveLayout(cf, "monitor-" + i++);
    }
}

final MapFrame mapFrame = MapFrame.getInstance();
if (mapFrame.isVisible())
    mapFrame.saveLayout(cf, "map");

return sl;
}

public void flush(final int position) {

    final Dimension ds = desktop.getSize();
    JInternalFrame bingo = null;

    final WaveClipboardFrame waveClipboard = WaveClipboardFrame.getInstance();
    if (waveClipboard.isSelected()) {
        bingo = waveClipboard;
    } else
        for (final JInternalFrame frame : SwarmInternalFrames.getFrames())
            if (frame.isSelected())
                bingo = frame;

    if (bingo != null)
        switch (position) {
            case LEFT:
                bingo.setSize(ds.width / 2, ds.height);
                bingo.setLocation(0, 0);
                break;

            case RIGHT:
                bingo.setSize(ds.width / 2, ds.height);
                bingo.setLocation(ds.width / 2, 0);
                break;

            case TOP:
                bingo.setSize(ds.width, ds.height / 2);

```

```

        bingo.setLocation(0, 0);
        break;

    case BOTTOM:
        bingo.setSize(ds.width, ds.height / 2);
        bingo.setLocation(0, ds.height / 2);
        break;

    case BOTTOM_LEFT:
        bingo.setSize(ds.width / 2, ds.height / 2);
        bingo.setLocation(0, ds.height / 2);
        break;

    case BOTTOM_RIGHT:
        bingo.setSize(ds.width / 2, ds.height / 2);
        bingo.setLocation(ds.width / 2, ds.height / 2);
        break;

    case TOP_LEFT:
        bingo.setSize(ds.width / 2, ds.height / 2);
        bingo.setLocation(0, 0);
        break;

    case TOP_RIGHT:
        bingo.setSize(ds.width / 2, ds.height / 2);
        bingo.setLocation(ds.width / 2, 0);
        break;
    }
}

public void flushLeft() {
    flush(LEFT);
}

public void flushRight() {
    flush(RIGHT);
}

public void flushTop() {
    flush(TOP);
}

public void flushBottom() {
    flush(BOTTOM);
}

public void flushBottomRight() {
    flush(BOTTOM_RIGHT);
}
}

```

```

public void flushBottomLeft() {
    flush(BOTTOM_LEFT);
}

public void flushTopRight() {
    flush(TOP_RIGHT);
}

public void flushTopLeft() {
    flush(TOP_LEFT);
}

public void tileKioskFrames() {
    final Dimension ds = desktop.getSize();

    final ArrayList<JInternalFrame> ks = new ArrayList<JInternalFrame>();
    for (final JInternalFrame frame : SwarmInternalFrames.getFrames()) {
        if (frame.isVisible() && frame instanceof Kioskable)
            ks.add(frame);
    }

    if (ks.size() == 0)
        return;

    int mapCount = 0;
    int heliCount = 0;
    int monitorCount = 0;
    for (final JInternalFrame frame : ks) {
        if (frame instanceof MapFrame)
            mapCount++;
        else if (frame instanceof HelicorderViewerFrame)
            heliCount++;
        else if (frame instanceof MultiMonitor)
            monitorCount++;
    }

    if (ks.size() == 4) {
        final int w = ds.width / 2;
        final int h = ds.height / 2;
        final JInternalFrame hvf0 = ks.get(0);
        final JInternalFrame hvf1 = ks.get(1);
        final JInternalFrame hvf2 = ks.get(2);
        final JInternalFrame hvf3 = ks.get(3);
        hvf0.setSize(w, h);
        hvf0.setLocation(0, 0);
        hvf1.setSize(w, h);
        hvf1.setLocation(w, 0);
        hvf2.setSize(w, h);
        hvf2.setLocation(0, h);
        hvf3.setSize(w, h);
    }
}

```

```

        hvf3.setLocation(w, h);
    } else if (ks.size() == 3 && mapCount == 1 && heliCount == 1 && monitorCount ==
1) {
        final int w = ds.width / 2;
        final int h = ds.height / 2;
        for (final JInternalFrame frame : ks) {
            if (frame instanceof MapFrame) {
                frame.setLocation(w, h);
                frame.setSize(w, h);
            } else if (frame instanceof HelicorderViewerFrame) {
                frame.setLocation(0, 0);
                frame.setSize(w, h * 2);
            } else if (frame instanceof MultiMonitor) {
                frame.setLocation(w, 0);
                frame.setSize(w, h);
            }
        }
    } else {
        final int w = ds.width / ks.size();
        int cx = 0;
        for (int i = 0; i < ks.size(); i++) {
            final JInternalFrame hvf = ks.get(i);
            try {
                hvf.setIcon(false);
                hvf.setMaximum(false);
            } catch (final Exception e) {
            }
            hvf.setSize(w, ds.height);
            hvf.setLocation(cx, 0);
            cx += w;
        }
    }
}

```

```

public void tileHelicorders() {
    final Dimension ds = desktop.getSize();

    final      ArrayList<HelicorderViewerFrame>      hcs      =      new
ArrayList<HelicorderViewerFrame>(10);
    for (final JInternalFrame frame : SwarmInternalFrames.getFrames()) {
        if (frame instanceof HelicorderViewerFrame)
            hcs.add((HelicorderViewerFrame) frame);
    }

    if (hcs.size() == 0)
        return;

    if (hcs.size() == 4) {
        final int w = ds.width / 2;
        final int h = ds.height / 2;

```

```

final HelicorderViewerFrame hvf0 = hcs.get(0);
final HelicorderViewerFrame hvf1 = hcs.get(1);
final HelicorderViewerFrame hvf2 = hcs.get(2);
final HelicorderViewerFrame hvf3 = hcs.get(3);
hvf0.setSize(w, h);
hvf0.setLocation(0, 0);
hvf1.setSize(w, h);
hvf1.setLocation(w, 0);
hvf2.setSize(w, h);
hvf2.setLocation(0, h);
hvf3.setSize(w, h);
hvf3.setLocation(w, h);
} else {
    final int w = ds.width / hcs.size();
    int cx = 0;
    for (int i = 0; i < hcs.size(); i++) {
        final HelicorderViewerFrame hvf = hcs.get(i);
        try {
            hvf.setIcon(false);
            hvf.setMaximum(false);
        } catch (final Exception e) {
        }
        hvf.setSize(w, ds.height);
        hvf.setLocation(cx, 0);
        cx += w;
    }
}
}

public void tileWaves() {
    final Dimension ds = desktop.getSize();

    int wc = 0;
    for (final JInternalFrame frame : SwarmInternalFrames.getFrames()) {
        if (frame instanceof WaveViewerFrame)
            wc++;
    }

    if (wc == 0)
        return;

    final int h = ds.height / wc;
    int cy = 0;
    for (final JInternalFrame frame : SwarmInternalFrames.getFrames()) {
        if (frame instanceof WaveViewerFrame) {
            final WaveViewerFrame wvf = (WaveViewerFrame) frame;
            try {
                wvf.setIcon(false);
                wvf.setMaximum(false);
            } catch (final Exception e) {
            }
        }
    }
}

```

```

    }
    wvf.setSize(ds.width, h);
    wvf.setLocation(0, cy);
    cy += h;
    }
}
}

public static void setFrameLayer(final JInternalFrame c, final int layer) {
    desktop.setLayer(c, layer, 0);
}

private void parseKiosk() {
    final String[] kiosks = config.kiosk.split(",");
    if (config.kiosk.startsWith("layout:")) {
        final String layout = config.kiosk.substring(7);
        final SwarmLayout sl = config.layouts.get(layout);
        if (sl != null) {
            lastLayout = layout;
            sl.process();
        } else
            LOGGER.warn("could not start with layout: " + layout);
    }
    boolean set = false;
    for (int i = 0; i < kiosks.length; i++) {
        final String[] ch = kiosks[i].split(";");
        final SeismicDataSource sds = config.getSource(ch[0]);
        if (sds == null)
            continue;
        openHelicorder(sds, ch[1], Double.NaN);
        set = true;
    }
    if (config.kiosk.equals("true"))
        set = true;

    if (set)
        toggleFullScreenMode();
    else
        LOGGER.warn("no helicorders, skipping kiosk mode.");
}

public void internalFrameAdded(final JInternalFrame f) {
    desktop.add(f);
    SwingUtilities.invokeLater(new Runnable() {
        public void run() {
            f.toFront();
            try {
                f.setSelected(true);
            } catch (final Exception e) {
            }
        }
    });
}

```



```

    }
    });
}

public void internalFrameRemoved(final JInternalFrame f) {
    // do nothing
}

public static void main(final String[] args) {
    try {
        UIManager.setLookAndFeel(new Plastic3DLookAndFeel());
        UIManager.put("InternalFrame.border", SwarmUtil.getInternalFrameBorder());
    } catch (final Exception e) {
    }

    final Swarm swarm = new Swarm(args);

    if (Swarm.config.isKiosk())
        swarm.parseKiosk();
}

public static EventFrame openEvent(final Event event) {
    final EventFrame eventFrame = new EventFrame(event);
    eventFrame.setVisible(true);
    eventFrame.requestFocus();
    SwarmInternalFrames.add(eventFrame);

    return eventFrame;
}

}

```

#### 2.3.2.2.2 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 source code berikut:

```
package gov.usgs.volcanoes.swarm.data;
```

```

import gov.usgs.plot.data.HelicorderData;
import gov.usgs.plot.data.RSAMData;
import gov.usgs.plot.data.Wave;

import java.util.List;

import javax.swing.event.EventListenerList;

/**
 * Base class for seismic data sources.
 *
 * @author Dan Cervelli
 */
abstract public class SeismicDataSource {
    protected String name = "Unnamed Data Source";
    protected boolean storeInUserConfig = true;
    protected boolean useCache = true;
    protected int minimumRefreshInterval = 1;

    protected EventListenerList listeners = new EventListenerList();

    public Gulper createGulper(GulperList gl, String k, String ch, double t1, double t2,
int size, int delay) {
        return new Gulper(gl, k, this, ch, t1, t2, size, delay);
    }

    abstract public List<String> getChannels();

    abstract public void parse(String params);

    /**
     * Either returns the wave successfully or null if the data source could
     * not get the wave.
     *
     * @param station
     * @param t1
     * @param t2
     * @return wave if possible
     */
    abstract public Wave getWave(String station, double t1, double t2);

    abstract public HelicorderData getHelicorder(String station, double t1, double t2,
GulperListener gl);

    abstract public String toConfigString();

    protected SeismicDataSource() {
        // explicit default constructor needed for reflection
    }

```

```

public void addListener(SeismicDataSourceListener l) {
    listeners.add(SeismicDataSourceListener.class, l);
}

public void removeListener(SeismicDataSourceListener l) {
    listeners.remove(SeismicDataSourceListener.class, l);
}

public void fireChannelsUpdated() {
    Object[] ls = listeners.getListenerList();
    for (int i = ls.length - 2; i >= 0; i -= 2)
        if (ls[i] == SeismicDataSourceListener.class)
            ((SeismicDataSourceListener) ls[i + 1]).channelsUpdated();
}

public void fireChannelsProgress(String id, double p) {
    Object[] ls = listeners.getListenerList();
    for (int i = ls.length - 2; i >= 0; i -= 2)
        if (ls[i] == SeismicDataSourceListener.class)
            ((SeismicDataSourceListener) ls[i + 1]).channelsProgress(id,
p);
}

public void fireHelicorderProgress(String id, double p) {
    Object[] ls = listeners.getListenerList();
    for (int i = ls.length - 2; i >= 0; i -= 2)
        if (ls[i] == SeismicDataSourceListener.class)
            ((SeismicDataSourceListener) ls[i + 1]).helicorderProgress(id, p);
}

public void notifyDataNotNeeded(String station, double t1, double t2,
GulperListener gl) {
}

public void setStoreInUserConfig(boolean b) {
    storeInUserConfig = b;
}

public boolean isStoreInUserConfig() {
    return storeInUserConfig;
}

public void setUseCache(boolean b) {
    useCache = b;
}

public boolean isUseCache() {
    return useCache;
}

```

```

/**
 * Is this data source active; that is, is new data being added in real-time
 * to this data source?
 *
 * @return whether or not this is an active data source
 */
public boolean isActiveSource() {
    return false;
}

/**
 * Close the data source.
 */
public void close() {
}

public void remove() {
}

/**
 * Get a copy of the data source. The default implementation returns an
 * identical copy, that is, <code>this</code>.
 *
 * This is confusing. Why return a reference to an object the caller already has?
Should this really return a
 * clone? If so, why not call it clone()? --tjp
 *
 * @return the identical data source (this)
 */
public SeismicDataSource getCopy() {
    return this;
}

/**
 * Get a string representation of this data source. The default implementation
 * return the name of the data source.
 *
 * @return the string representation of this data source
 */
public String toString() {
    return name;
}

/**
 * Sets the data source name.
 *
 * @param s
 *         the new name
 */

```

```

    public void setName(String s) {
        name = s;
    }

    /**
     * Gets the data source name.
     *
     * @return the name
     */
    public String getName() {
        return name;
    }

    public void establish() {
    }

    public int getMinimumRefreshInterval() {
        return minimumRefreshInterval;
    }
}

```

### 2.3.2.2.3 *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 ini diimplementasikan dengan source code berikut:

```

package gov.usgs.volcanoes.swarm.wave;

import java.awt.Color;
import java.awt.Graphics2D;
import java.awt.event.MouseEvent;
import java.awt.geom.GeneralPath;
import java.awt.geom.Line2D;

import gov.usgs.plot.Plot;
import gov.usgs.plot.data.SliceWave;

```

```

import gov.usgs.plot.data.Wave;
import gov.usgs.plot.render.TextRenderer;
import gov.usgs.plot.render.wave.SliceWaveRenderer;
import gov.usgs.volcanoes.swarm.Metadata;
import gov.usgs.volcanoes.swarm.SwarmConfig;

/**
 * A component that renders a wave in either a standard wave view, a frequency spectra,
or
 * spectrogram. Relies heavily on the Valve plotting package.
 *
 * TODO: move filter method
 *
 *
 * @author Dan Cervelli
 */
public class WaveViewPanel extends AbstractWavePanel {
    /**
     * Constructs a WaveViewPanel with default settings.
     */
    public WaveViewPanel() {
        this(new WaveViewSettings());
    }

    /**
     * Constructs a WaveViewPanel with specified settings.
     *
     * @param s the settings
     */
    public WaveViewPanel(WaveViewSettings s) {
        super(s);
    }

    /**
     * Constructs a WaveViewPanel set up the same as a source WaveViewPanel. Used when
copying a
     * waveform to the clipboard.
     *
     * @param p the source WaveViewPanel
     */
    public WaveViewPanel(WaveViewPanel p) {
        super(p);
    }

    @Override
    protected void processRightMousePress(MouseEvent e) {
        settings.cycleType();
    }

    private void paintMark(Graphics2D g2, double j2k) {

```

```

if (Double.isNaN(j2k) || j2k < startTime || j2k > endTime)
    return;

double[] t = getTranslation();
if (t == null)
    return;

double x = (j2k - t[1]) / t[0];
g2.setColor(DARK_GREEN);
g2.draw(new Line2D.Double(x, yOffset, x, getHeight() - bottomHeight - 1));

GeneralPath gp = new GeneralPath();
gp.moveTo((float) x, yOffset);
gp.lineTo((float) x - 5, yOffset - 7);
gp.lineTo((float) x + 5, yOffset - 7);
gp.closePath();
g2.setPaint(Color.GREEN);
g2.fill(gp);
g2.setColor(DARK_GREEN);
g2.draw(gp);
}

@Override
protected void annotateImage(Graphics2D g2) {
    if (!Double.isNaN(mark1))
        paintMark(g2, mark1);

    if (!Double.isNaN(mark2))
        paintMark(g2, mark2);
}

@Override
protected void processRightMouseRelease(MouseEvent e) {
    // do nothing
}
}

```

#### 2.3.2.2.4 *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:

```
package gov.usgs.volcanoes.swarm.heli;

import gov.usgs.volcanoes.core.configfile.ConfigFile;
import gov.usgs.volcanoes.swarm.SwarmConfig;

/**
 * Settings for a helicorder.
 *
 * TODO: eliminate this class in favor of vdx.HelicorderSettings
 *
 * @author Dan Cervelli
 */
public class HelicorderViewerSettings
{
    public String channel;
    public int timeChunk; // seconds
    public int span; // minutes
    public int waveZoomOffset; // seconds
    private double bottomTime; // Double.NaN for now
    public int refreshInterval;
    public int scrollSize;
    public boolean forceCenter;
    public int clipBars;
    private long lastBottomTimeSet;

    public boolean autoScale;
    public boolean showClip;
    public boolean alertClip;
    public int alertClipTimeout;
    public int clipValue;
    public int barRange;
    public double barMult;
    public HelicorderViewPanel view;

    private static SwarmConfig swarmConfig;

    public HelicorderViewerSettings(String ch)
    {
        swarmConfig = SwarmConfig.getInstance();

        channel = ch;
        timeChunk = swarmConfig.timeChunk * 60;
        span = swarmConfig.span * 60;
        waveZoomOffset = 30;
        bottomTime = Double.NaN;
        refreshInterval = 15;
        scrollSize = 24;
    }
}
```



```

        forceCenter = false;
        clipBars = 21;

        clipValue = 2999;
        showClip = swarmConfig.showClip;
        alertClip = swarmConfig.alertClip;
        alertClipTimeout = swarmConfig.alertClipTimeout * 60;
        barRange = 1500;
        barMult = 3;
        autoScale = true;
    }

    public long getLastBottomTimeSet()
    {
        return System.currentTimeMillis() - lastBottomTimeSet;
    }

    public void setBottomTime(double bt)
    {
        lastBottomTimeSet = System.currentTimeMillis();
        bottomTime = bt;
    }

    public double getBottomTime()
    {
        return bottomTime;
    }

    public void set(ConfigFile cf)
    {
        timeChunk = Integer.parseInt(cf.getString("timeChunk"));
        span = Integer.parseInt(cf.getString("span"));
        waveZoomOffset = Integer.parseInt(cf.getString("waveZoomOffset"));
        refreshInterval = Integer.parseInt(cf.getString("refreshInterval"));
        scrollSize = Integer.parseInt(cf.getString("scrollSize"));
        clipValue = Integer.parseInt(cf.getString("clipValue"));
        clipBars = Integer.parseInt(cf.getString("clipBars"));
        barRange = Integer.parseInt(cf.getString("barRange"));
        alertClipTimeout = Integer.parseInt(cf.getString("alertClipTimeout"));
        setBottomTime(Double.parseDouble(cf.getString("bottomTime")));
        barMult = Double.parseDouble(cf.getString("barMult"));
        forceCenter = Boolean.parseBoolean(cf.getString("forceCenter"));
        autoScale = Boolean.parseBoolean(cf.getString("autoScale"));
        showClip = Boolean.parseBoolean(cf.getString("showClip"));
        alertClip = Boolean.parseBoolean(cf.getString("alertClip"));
    }

    public void save(ConfigFile cf, String prefix)
    {
        cf.put(prefix + ".channel", channel);
    }

```

```

        cf.put(prefix + ".timeChunk", Integer.toString(timeChunk));
        cf.put(prefix + ".span", Integer.toString(span));
        cf.put(prefix + ".waveZoomOffset", Integer.toString(waveZoomOffset));
        cf.put(prefix + ".refreshInterval", Integer.toString(refreshInterval));
        cf.put(prefix + ".scrollSize", Integer.toString(scrollSize));
        cf.put(prefix + ".clipValue", Integer.toString(clipValue));
        cf.put(prefix + ".clipBars", Integer.toString(clipBars));
        cf.put(prefix + ".barRange", Integer.toString(barRange));
        cf.put(prefix + ".alertClipTimeout", Integer.toString(alertClipTimeout));
        cf.put(prefix + ".bottomTime", Double.toString(bottomTime));
        cf.put(prefix + ".barMult", Double.toString(barMult));
        cf.put(prefix + ".forceCenter", Boolean.toString(forceCenter));
        cf.put(prefix + ".autoScale", Boolean.toString(autoScale));
        cf.put(prefix + ".showClip", Boolean.toString(showClip));
        cf.put(prefix + ".alertClip", Boolean.toString(alertClip));
    }

    public void parseSettingsString(String o)
    {
        //      String[] opts = Util.splitString(o, ",");
        String[] opts = o.split(",");
        for (int i = 0; i < opts.length; i++)
        {
            try
            {
                String key = opts[i].substring(0, opts[i].indexOf('='));
                String value = opts[i].substring(opts[i].indexOf('=') + 1);

                if (key.equals("x")) // minutes
                    timeChunk = Integer.parseInt(value) * 60;
                else if (key.equals("y")) // hours
                    span = Integer.parseInt(value) * 60;
            }
            catch (Exception e)
            {
                System.err.println("Could not parse setting: " + opts[i]);
            }
        }
    }

    public void notifyView()
    {
        if (view != null)
            view.settingsChanged();
    }
}

```

### 2.3.2.3 Fungsi tambahan

```

/*
 * 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.rmcloudamqp.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 MPU9255{
    uint8_t x;
    uint8_t y;
    uint8_t z;
};

MPU9255 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
MPU9255_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))
        {
            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 active low on the ESP-01)
    } else {
        digitalWrite(BUILTIN_LED, HIGH); // Turn the LED off by making the
voltage HIGH
    }
}

```

```

}

//=====//
//=====IMU & GPS Function Procedure=====//
//=====//

void MPU9255_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);
}

MPU9255 Acc_Read()
{
    MPU9255 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;
}
```