**INSTITUT TEKNOLOGI BANDUNG**

**PROGRAM STUDI TEKNIK ELEKTRO**

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

Bandung 40132

**Dokumentasi Produk Tugas Akhir**

Lembar Sampul Dokumen

|  |  |  |
| --- | --- | --- |
| Judul Dokumen | TUGAS AKHIR TEKNIK ELEKTRO:  Sistem Detektor Gempa dan Tsunami Decision Support System | |
|  |  | |
| Jenis Dokumen | PENGUJIAN | |
|  | Catatan: Dokumen ini dikendalikan penyebarannya oleh Prodi Teknik Elektro ITB | |
| Nomor Dokumen | **B500-02-TA1617.01. 069** | |
|  |  | |
| Nomor Revisi | **01** | |
|  |  | |
| Nama File |  | |
|  |  | |
| Tanggal Penerbitan | 4 May 2017 | |
|  |  | |
| Unit Penerbit | Prodi Teknik Elektro - ITB | |
|  |  | |
| Jumlah Halaman | 21 | (termasuk lembar sampul ini) |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Data Pemeriksaan dan Persetujuan | | | | |
| Ditulis | Nama | Christoporus Deo Putratama | Jabatan | Anggota |
| oleh | Tanggal | 4 May 2017 | Tanda Tangan |  |
|  | Nama | Bramantio Yuwono | Jabatan | Anggota |
| Tanggal | 4 May 2017 | Tanda Tangan |  |
| Nama | Kevin Shidqi Prakoso | Jabatan | Anggota |
| Tanggal | 4 May 2017 | Tanda Tangan |  |
| Disetujui | Nama | Dr. techn. Ary Setijadi  Prihatmanto | Jabatan | Dosen Pembimbing |
| oleh | Tanggal | 4 May 2017 | Tanda Tangan |  |
|  |  |  |  |  |

# DAFTAR ISI

[DAFTAR ISI 2](#_Toc481088055)

[Catatan Sejarah Perbaikan Dokumen 3](#_Toc481088056)

[Proposal Perancangan Sistem Detektor Gempa dan Tsunami Decision Support System 4](#_Toc481088057)

[1 Pengantar 4](#_Toc481088058)

[1.1 Ringkasan Isi Dokumen 4](#_Toc481088059)

[1.2 Tujuan Penulisan dan Aplikasi/Kegunaan Dokumen 4](#_Toc481088060)

[1.3 Referensi 4](#_Toc481088061)

[1.4 Daftar Singkatan 4](#_Toc481088062)

[2 Pengujian dan Verifikasi Proyek 5](#_Toc481088063)

[2.1 Pengujian Sensor Seismik ECN (Earthquake Catcher Network) 5](#_Toc481088064)

[2.1.2 Pengujian Algoritma Sensor ECN 6](#_Toc481088065)

[2.1.3 Pengujian Daya Tahan Sistem Daya 7](#_Toc481088066)

[2.2 Pengujian Messaging Server dan Database 8](#_Toc481088067)

[2.2.1 Pengujian Messaging Server RabbitMQ 8](#_Toc481088068)

[2.2.2 Pengujian Database MySQL 10](#_Toc481088069)

[2.2.3 Pengujian Latency Jaringan Sistem 11](#_Toc481088070)

[2.3 Pengujian Algoritma Analisis Gempa 12](#_Toc481088071)

[2.3.1 Pengujian Algoritma Umum Program 12](#_Toc481088072)

[2.3.2 Pengujian Algoritma Counting 14](#_Toc481088073)

[2.3.3 Pengujian Algoritma Magnitude 17](#_Toc481088074)

[2.3.4 Pengujian Algoritma Spektrum Frekuensi 19](#_Toc481088075)

# Catatan Sejarah Perbaikan Dokumen

|  |  |  |  |
| --- | --- | --- | --- |
| Versi | Tanggal | Oleh | Perbaikan |
| 1 | 27 April 2017 | C. Deo P. Bramantio Y. Kevin S.P. | Versi pertama dokumen B500 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Proposal Perancangan Sistem Detektor Gempa dan Tsunami Decision Support System

# Pengantar

## Ringkasan Isi Dokumen

Dokumen ini berisi tentang perancangan dari Tugas Akhir yang berjudul Sistem Detektor Gempa dan Tsunami Decision Support System Bab 1 berisi pengantar isi dokumen yang terdiri dari ringkasan isi dokumen, tujuan penulisan, referensi, serta daftar singkatan. Pada Bab 2, berisi pengujian modul-modul yang telah dibuat pada B400. Bab ini terdiri dari pengujian sensor gempa, pengujian messaging server dan database, dan pengujian sistem pengolahan data gempa.

## Tujuan Penulisan dan Aplikasi/Kegunaan Dokumen

Tujuan penulisan dokumen ini adalah sebagai bahan dokumentasi tahapan pengujian proyek perancangan Sistem Detektor Gempa dan Tsunami Decision Support System, sebagai acuan untuk pihak lain yang ingin mempelajari proyek ini.

## Referensi

1. Rudloff, Alexander, *German-Indonesian Tsunami Early Warning System (GITEWS) Decision Support System (DSS),* Deutschen Zentrums für Luft- und Raumfahrt (DLR): Köln (2010)
2. <http://www.codeproject.com/Articles/1117162/Detecting-Seismic-Waves-with-Android-and-IOT>, diakses pada 18 September 2016.

## Daftar Singkatan

| Singkatan | Arti |
| --- | --- |
| LIPI | *Lembaga Ilmu Pengetahuan Indonesia* |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# 

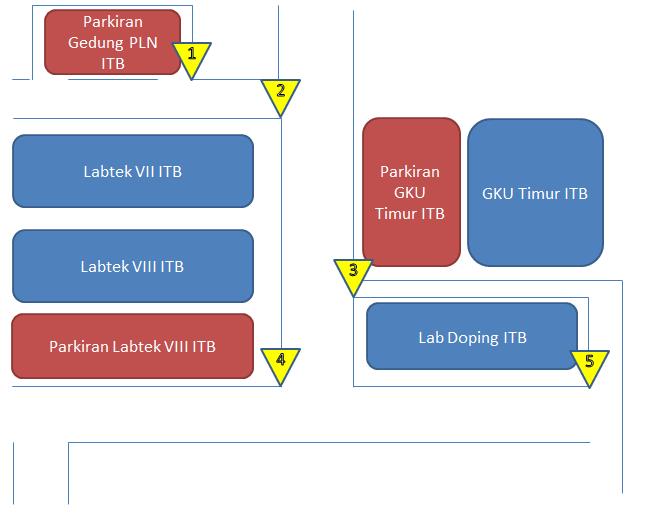
# Pengujian dan Verifikasi Proyek

## Pengujian Sensor Seismik ECN (Earthquake Catcher Network)

Pengujian Sensor Seismik ECN berfungsi untuk mengetahui peforma sensor yang sudah didesain. Pengujian diperlukan untuk melihat peforma sensor ECN dari beberapa aspek, yaitu Akurasi GPS, Akurasi IMU, Algoritma Sensor ECN, dan Daya Tahan Sistem Daya.

* + 1. Pengujian GPS

Pengujian akurasi GPS dilakukan dengan cara mengukur pembacaan GPS pada suatu titik tertentu dengan menggunakan modul GPS Ublox Neo-6m dan dibandingkan dengan pembacaan data longitude dan latitude pada Google Maps di titik yang sama. Pengujian GPS ini dilakukan di beberapa titik di area labtek VIII ITB, Labtek VII ITB, GKU Timur ITB, dan Laboratorium Doping ITB. Berikut adalah titik pengujian GPS yang digunakan.



**Gambar 2.1 Denah Titik Tempat Pengujian GPS**

Titik pengujian sengaja dilakukan di sudut jalan untuk memudahkan proses pengujian dan mengurangi error yang dilakukan ketika menetapkan titik tersebut pada Google Maps. Berikut adalah perbandingan antara data GPS Google Maps dan modul GPS Ublox Neo 6M.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Titik** | **Google Maps** | | **Pengujian GPS** | | **Error** | |
| **Lat** | **Lon** | **Lat** | **Lon** | **Lat** | **Lon** |
| **1** | **-6,889866** | **107,610984** | **-6,889791** | **107,610939** | **7,5E-05** | **4,5E-05** |
| **2** | **-6,889926** | **107,611537** | **-6,88995** | **107,611649** | **2,4E-05** | **0,000112** |
| **3** | **-6,890599** | **107,611615** | **-6,890493** | **107,611603** | **0,000106** | **1,2E-05** |
| **4** | **-6,890971** | **107,611534** | **-6,890993** | **107,61158** | **2,2E-05** | **4,6E-05** |
| **5** | **-6,890958** | **107,612103** | **-6,890976** | **107,612076** | **1,8E-05** | **2,7E-05** |
| **Error** | | | | | **4,9E-05** | **4,8E-05** |

**Tabel 2.1 Pengujian GPS**

Dapat dilihat pada tabel diatas bahwa error dari pembacaan latitude pada modul GPS Neo 6M adalah 0,000049 dan error dari pembacaan longitude padamodul GPS Neo 6M adalah 0,000048. Pada jalur khatulistiwa, 1 derajat latitude sebanding dengan 110.57 km dan 1 derajat longitude sebanding dengan 111.32 km. Sehingga dapat disimpulkan bahwa error dari pembacaan modul GPS Neo 6M adalah 5,41793 m pada latitude dan 5,34336 m.

### Pengujian Sensor IMU 9255

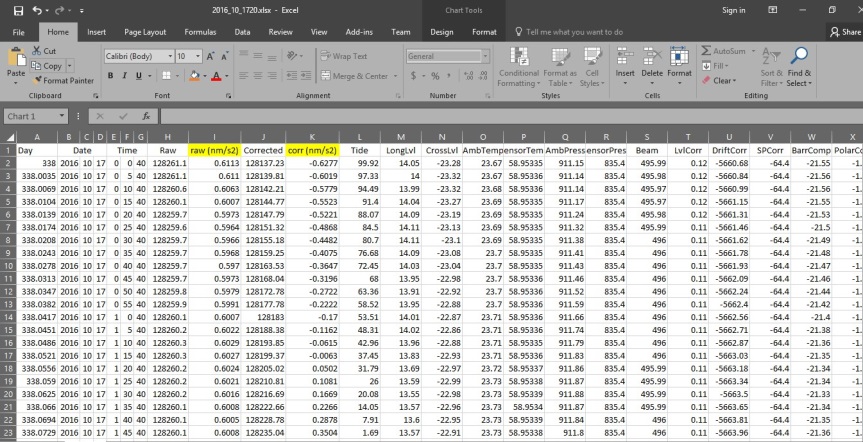
#### Metode Pengujian

Pengujian terhadap sensor IMU 9255 dilakukan dengan cara membandingkan nilai yang terbaca pada sensor dengan pembacaan sensor yang telah terkalibrasi. Pada awalnya, dilakukan perbandingan dengan menggunakan sensor seismik yang dimiliki oleh Divisi Geoteknologi LIPI. Berikut adalah gambar sensor seismik yang dimiliki oleh Divisi Geoteknologi LIPI.



**Gambar Sensor Seismik Divisi Geoteknologi LIPI**

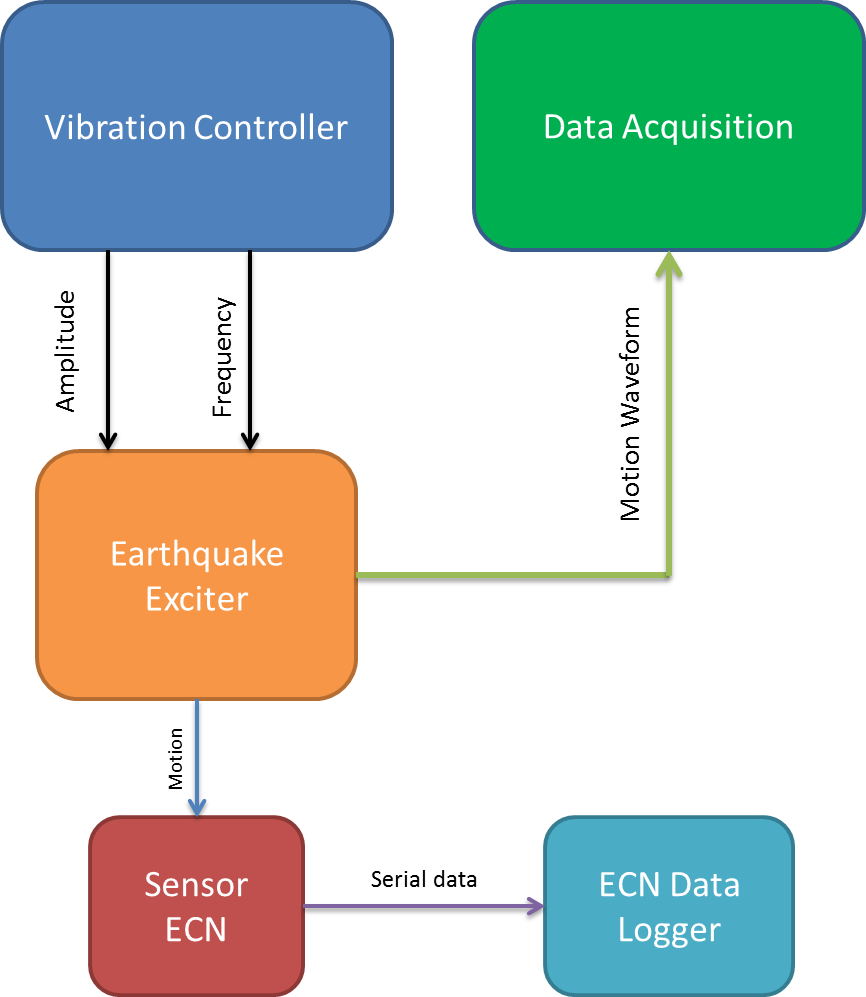
Tetapi dikarenakan tidak sesuainya data yang dicatat oleh sensor seismik LIPI dengan pembacaan sensor ECN, serta tidak diizinkannya untuk melakukan goyangan terhadap sensor karena ditakutkan rusak maka pengujian tidak dapat dilakukan. Berikut data yang cuplikan data yang dibaca oleh sensor seismik divisi geoteknologi LIPI.



**Gambar Cuplikan Data yang Terbaca pada Sensor Seismik LIPI**

Dapat dilihat pada gambar diatas, bahwa data yang terbaca merupakan data yang sudah mengalami pemrosesan, sedangkan sensor ECN hanya dapat memperoleh data mentah percepatan dan pemrosesan dilakukan pada server sehingga tidak dapat dilakukan karena latensi jaringan yang tidak dapat diprediksi sehingga data dari kedua buah sensor sulit untuk dibandingkan.

Sebagai pengganti sensor seismik Divisi Geoteknologi LIPI, digunakan Earthquake Exciter Lab Dinamika PAU ITB sebagai sensor pembanding yang sudah terkalibrasi. Untuk skema pengujian dapat dilihat pada gambar berikut.



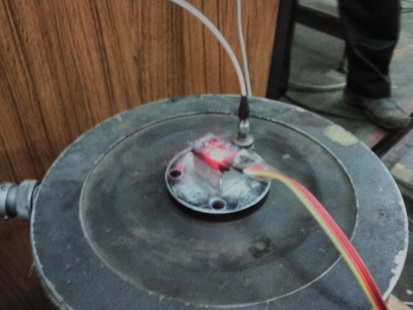
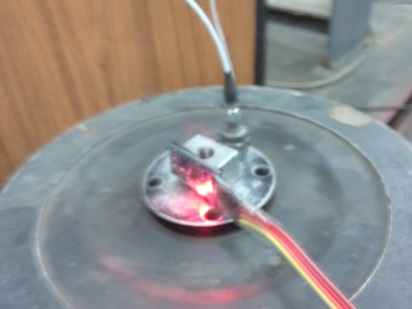
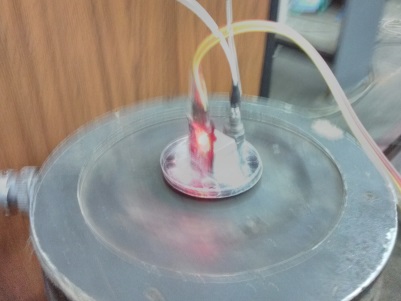
**Gambar Skema Pengujian Sensor IMU MPU9255 menggunakan Earthquake Exciter**

Earthquake exciter adalah sebuah alat yang dapat menghasilkan getaran sesuai dengan input magnitude dan frekuensi yang diberikan. Pada pengujian ini sensor IMU MPU9255 diletakkan dan ditempelkan pada Earthquake Exciter sehingga IMU akan bergetar bersamaan dengan Earthquake Exciter. Input frekuensi dan amplitude diberikan dengan menggunakan Vibration Controller yang terhubung dengan Earthquake Exciter. Data acquisition digunakan untuk memvalidasi besar getaran yang dihasilkan oleh Earthquake Exciter. Berikut adalah gambar beberapa alat yang digunakan dalam melakukan pengujian sensor IMU MPU9255.



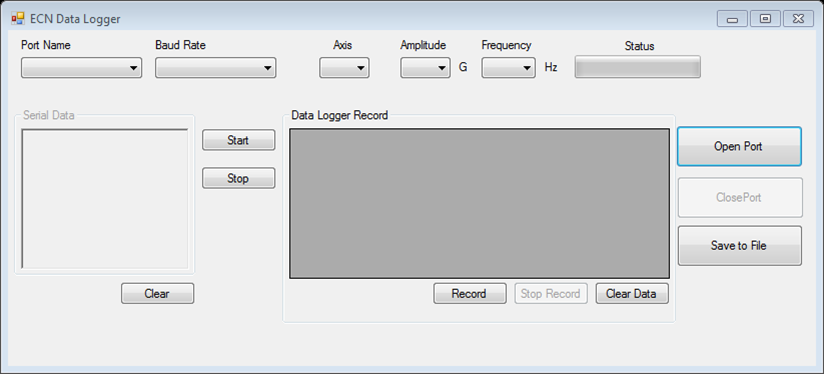
**Gambar Earthquake Exciter (kiri), Vibration Controller (tengah), Data Aqcuisition (kanan)**

Pengujian dilakukan pada amplitude 0.5 G, 1.0 G, dan 1.5 G untuk setiap frekuensi 10 Hz, 15 Hz, 20 Hz dan sumbu x, y, dan z. Sehingga dilakukan pengujian sebanyak 27 kali. Frekuensi dan Amplitude dapat diubah menggunakan vibration controller, sedangkan pengujian untuk sumbu x,y, dan z dilakukan dengan mengubah posisi peletakkan/penempelan sensor IMU MPU9255 pada Earthquake Exciter. Gambar peletakkan sensor IMU MPU9255 pada sumbu x,y,z dapat dilihat pada gambar berikut.



**Gambar Peletakkan Sensor IMU MPU9255 untuk Pengujian Sumbu x (tengah), y (kiri), z (kanan)**

Untuk memudahkan proses pembacaan data sensor ECN, sensor MPU9255 terhubung ke sebuah NodeMCU yang selalu membaca percepatan dengan frekuensi 40 Hz dan mengirimkan hasil pembacaan tersebut ke GUI ECN Data Logger dengan menggunakan komunikasi serial.



**Gambar Tampilan GUI ECN Data Logger**

Langkah-langkah untuk menggunakan GUI ini adalah sebagai berikut.

1. Pilih Port Name dan BaudRate sesuai dengan yang digunakan, lalu klik Open Port
2. Setelah port serial berhasil dibuka, tekan tombol Start untuk memulai mengambil data yang dikirimkan dari NodeMCU, tombol stop untuk menghentikan proses mengambil data yang dikirimkan dari NodeMCU
3. Tombol Record digunakan untuk memulai mencatat pembacaan data dari String yang dikirimkan oleh NodeMCU ke dalam tabel, Clear Data untuk menghapus tabel yang sudah direkam, dan Stop Record untuk menghentikan proses penyimpanan data serial dari NodeMCU ke tabel.
4. Save to File digunakan untuk menyimpan tabel yang sudah dicatat ke dalam file excel

Open Port dapat dilakukan dengan menggunakan fungsi SerialPort.Open() dan Close Port dapat diimplementasikan dengan menggunakan fungsi SerialPort.Close(). Proses pembacaan string dilakukan dengan menggunakan timer setiap 1ms dengan menggunakan SerialPort.ReadLine(). Ketika tombol Record ditekan, proses penyimpanan data pada tabel akan mulai dilakukan dengan assign variabel boolean dengan nilai true. Sedangkan untuk perintah Save to File dapat dilihat dari potongan source code sebagai berikut.

|  |
| --- |
| SaveFileDialog sfd = new SaveFileDialog();  sfd.Filter = "Excel Documents (\*.xlsx)|\*.xlsx";  String name = comboBox1.SelectedItem.ToString() + " "+ AmplitudeBox.SelectedItem.ToString() + "G " + FrequencyBox.SelectedItem.ToString()+"Hz";  sfd.FileName = "Test MPU9255 " + name +".xlsx";  if (sfd.ShowDialog() == DialogResult.OK)  {  var workbook = new XLWorkbook();  workbook.Worksheets.Add(table, "Test MPU9255");  workbook.SaveAs(sfd.FileName);  table.Clear();  MessageBox.Show("Save File Success !", "Information", MessageBoxButtons.OK, MessageBoxIcon.Information);  } |

#### Hasil Pengujian

### Pengujian Algoritma Sensor ECN

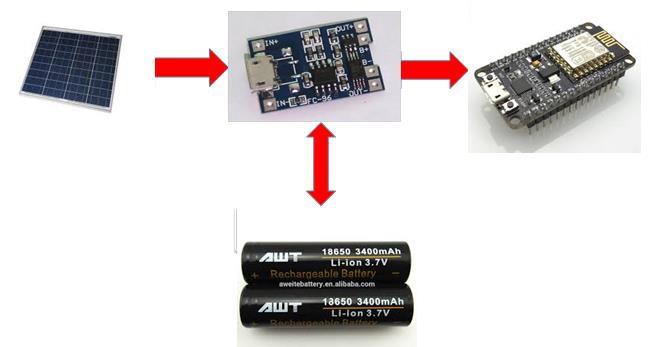
Pada pengujian algoritma sensor ECN, dilakukan dua jenis pengujian, yaitu pengujian waktu yang dibutuhkan dari beberapa proses penting pada algoritma yang didesain dan pengujian fungsional dari algoritma sensor ECN yang sudah didesain. Pada pengujian waktu proses, diamati tiga proses yaitu, proses pembacaan data IMU, pembacaan GPS, dan proses publish MQTT. Metodologi pengujian waktu proses ini dilakukan dengan memanfaatkan fungsi millis(). Sehingga pada algoritma sensor, Cuma ditambahkan fungsi millis sebelum dan setelah proses pembacaan data IMU, pembacaan GPS, dan publish MQTT.

Pengamatan ini dilakukan sebanyak 60 kali proses pengiriman data ke server setiap 1 detik dengan 40 data sampling IMU. Hasil pengujian waktu proses pada algoritma sensor dengan kondisi tanpa menunggu queue RabbitMQ dapat dilihat pada lampiran dan grafik dibawah ini.

Dapat dilihat bahwa proses pembacaan GPS memakan waktu proses yang paling sebentar dengan range antara 0-4 ms, sehingga karena proses pembacaan GPS ini hanya dilakukan satu kali tiap satu detik maka lama proses pembacaan GPS tidak mengganggu sampling IMU. Sedangkan lama proses pembacaan IMU mengambil waktu antara 1-18 ms. Karena sensor harus dapat melakukan sampling dengan frekuensi 40 Hz atau setiap 25 ms maka lama proses pembacaan IMU yang dilakukan oleh sensor masih membuat sensor dapat melakukan sampling data IMU dengan periode 25 ms. Dapat dilihat bahwa dalam kondisi tidak menunggu queue RabbitMQ, proses pengiriman memakan waktu proses cukup lama yaitu diantara 206–322 ms. Hal tersebut cukup mengganggu proses pembacaan IMU, karena pada saat sensor publish message ke RabbitMQ, proses sampling data akan berhenti selama proses publish message tersebut. Hal tersebut dapat diatasi dengan mengimplementasikan RTOS(Real Time Operating System) pada board NodeMCU, tetapi proses implementasi terhambat karena library yang sangat terbatas jika menggunakan RTOS sehingga harus membuat library untuk pembacaan IMU, pembacaan GPS, dan MQTT yang memakan waktu lama.

### Pengujian Daya Tahan Sistem Daya

Pengujian Daya Tahan Sistem Daya dilakukan dengan mencoba rangkaian charging solar panel yang terhubung dengan beban lalu diamati tegangan solar panel dalam beberapa waktu tertentu. Setelah itu, saat malam hari diuji apakah sistem daya yang didesain dapat menyuplai daya ke sensor dari malam hari sampai malam hari. Rangkaian sistem daya dapat dilihat pada gambar berikut.



Pada pengujian ini, diamati tegangan pada 4 kondisi yang didefinisikan, yaitu cerah pagi untuk jam 07.00 pagi sampai jam 11.00 saat kondisi cerah, cerah siang untuk jam 11.00 sampai jam 16.00 saat kondisi cerah, gelap untuk jam 16.00 sampai jam 07.00 dan mendung ketika sinar matahari terhalang oleh awan saat siang hari. Hasil pengamatan menunjukkan bahwa tegangan rata-rata yang dihasilkan solar panel saat cerah pagi adalah 4,8 Volt, saat cerah siang tegangan rata-rata yang dihasilkan adalah 5,6 Volt, saat gelap tegangan rata-rata yang dihasilkan adalah 0,2 Volt, dan saat mendung tegangan rata-rata yang dihasilkan adalah 4,5 Volt. Berdasarkan datasheet modul charging baterai li-ion, input tegangan yang harus disuplai solar panel harus berada di range 4,5-5,5 Volt.Dapat disimpulkan dari data diatas bahwa modul dapat di-charge pada kondisi cerah pagi, cerah siang, dan mendung. Sehingga NodeMCU harus disuplai oleh baterai dari jam 16.00-07.00 pagi besok harinya.

Tetapi saat proses pengujian, tegangan baterai Li-ion menurun ke angka 2,8 Volt dari kondisi normal (3,7 Volt) karena terjadi short pada modul charging sehingga pengujian kapasitas baterai setelah di-charge pada siang hari tidak dapat dilakukan.

## Pengujian Messaging Server dan Database

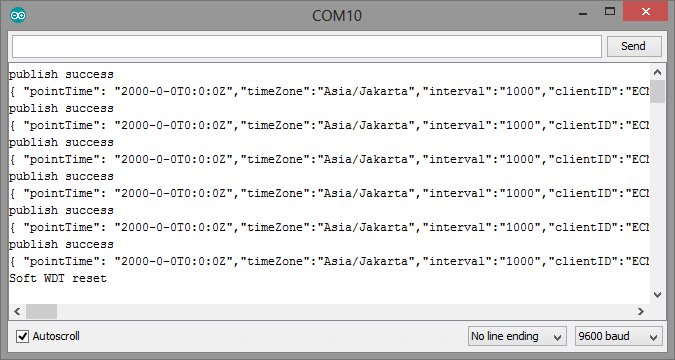
Aspek yang ditinjau pada pengujian Messaging Server adalah pesan dapat diterima oleh Messaging Server dengan baik kemudian dapat meneruskan pesan tersebut ke consumer yang sesuai. Harus dilakukan sebuah pengaturan pada pengirim sehingga sebuah pesan tidak akan bisa dikonsumsi oleh consumer yang tidak sesuai. Setelah pesan diterima oleh consumer, pesan tersebut akan dikirimkan menuju sebuah database. Data yang telah disimpan dapat digunakan untuk diolah datanya.

Pengujian dilakukan dengan mengirimkan pesan dari sensor maupun dari program penguji. Pesan kemudian akan dikirimkan menuju Messaging Server. Supaya mengetahui bahwa pessan dapat terkirim, maka harus ada consumer yang menjalankan perintah consume Messaging Server. Apabila pesan dapat di consume, maka proses pengiriman dari sensor menuju Messaging Server berjalan dengan lancar.

### Pengujian Messaging Server RabbitMQ

Sensor mengirimkan sebuah pesan setiap 1 detik menuju Messaging Server dengan jaringan WiFi yang sudah di set. Pesan ini merupakan pesan informasi geospasial dengan format JSON. Penguji melakukan debugging pada sensor dengan membaca komunikasi serial sehingga mengetahui apa yang dikirimkan oleh sensor.

Berikut ini data yang dihasilkan oleh sensor saat berhasil mengirimkan sebuah pesan beserta tampilan debugging dari pesan yang dihasilkan:



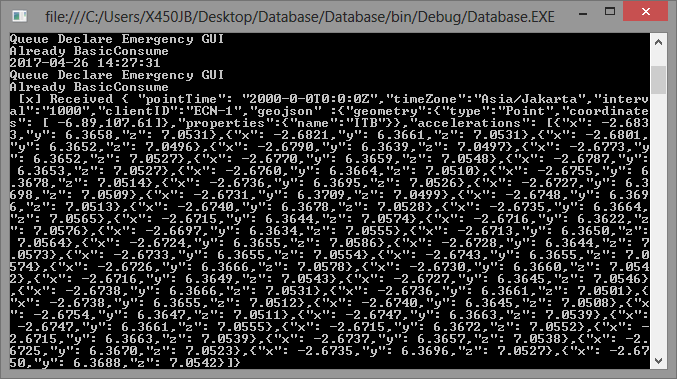
Kemudian isi pesan yang dikirimkan adalah sebagai berikut:

|  |
| --- |
| { "pointTime": "2000-0-0T0:0:0Z","timeZone":"Asia/Jakarta","interval":"1000","clientID":"ECN-1","geojson" :{"geometry":{"type":"Point","coordinates": [ -6.89,107.61]},"properties":{"name":"ITB"}},"accelerations": [{"x": -2.0661,"y": 4.8328,"z": 5.2528},{"x": -2.0859,"y": 4.8904,"z": 5.3096},{"x": -2.1068,"y": 4.9404,"z": 5.3676},{"x": -2.1281,"y": 4.9897,"z": 5.4209},{"x": -2.1516,"y": 5.0375,"z": 5.4726},{"x": -2.1710,"y": 5.0862,"z": 5.5196},{"x": -2.1899,"y": 5.1320,"z": 5.5701},{"x": -2.2104,"y": 5.1750,"z": 5.6138},{"x": -2.2282,"y": 5.2156,"z": 5.6570},{"x": -2.2458,"y": 5.2601,"z": 5.7015},{"x": -2.2648,"y": 5.3003,"z": 5.7426},{"x": -2.2820,"y": 5.3374,"z": 5.7857},{"x": -2.2963,"y": 5.3774,"z": 5.8231},{"x": -2.3130,"y": 5.4122,"z": 5.8600},{"x": -2.3297,"y": 5.4444,"z": 5.8985},{"x": -2.3458,"y": 5.4768,"z": 5.9321},{"x": -2.3593,"y": 5.5074,"z": 5.9641},{"x": -2.3726,"y": 5.5376,"z": 5.9999},{"x": -2.3879,"y": 5.5690,"z": 6.0290},{"x": -2.3978,"y": 5.5987,"z": 6.0641},{"x": -2.4110,"y": 5.6320,"z": 6.0913},{"x": -2.4203,"y": 5.6601,"z": 6.1175},{"x": -2.4339,"y": 5.6857,"z": 6.1457},{"x": -2.4411,"y": 5.7115,"z": 6.1718},{"x": -2.4504,"y": 5.7361,"z": 6.1995},{"x": -2.4614,"y": 5.7576,"z": 6.2212},{"x": -2.4707,"y": 5.7819,"z": 6.2485},{"x": -2.4815,"y": 5.8033,"z": 6.2715},{"x": -2.4923,"y": 5.8249,"z": 6.2971},{"x": -2.5006,"y": 5.8451,"z": 6.3182},{"x": -2.5088,"y": 5.8637,"z": 6.3422},{"x": -2.5156,"y": 5.8826,"z": 6.3636},{"x": -2.5221,"y": 5.9009,"z": 6.3845},{"x": -2.5288,"y": 5.9186,"z": 6.4063},{"x": -2.5368,"y": 5.9352,"z": 6.4273},{"x": -2.5464,"y": 5.9518,"z": 6.4467},{"x": -2.5534,"y": 5.9692,"z": 6.4634},{"x": -2.5586,"y": 5.9802,"z": 6.4793},{"x": -2.5656,"y": 5.9951,"z": 6.4893},{"x": -2.5717,"y": 6.0085,"z": 6.5053}]} |

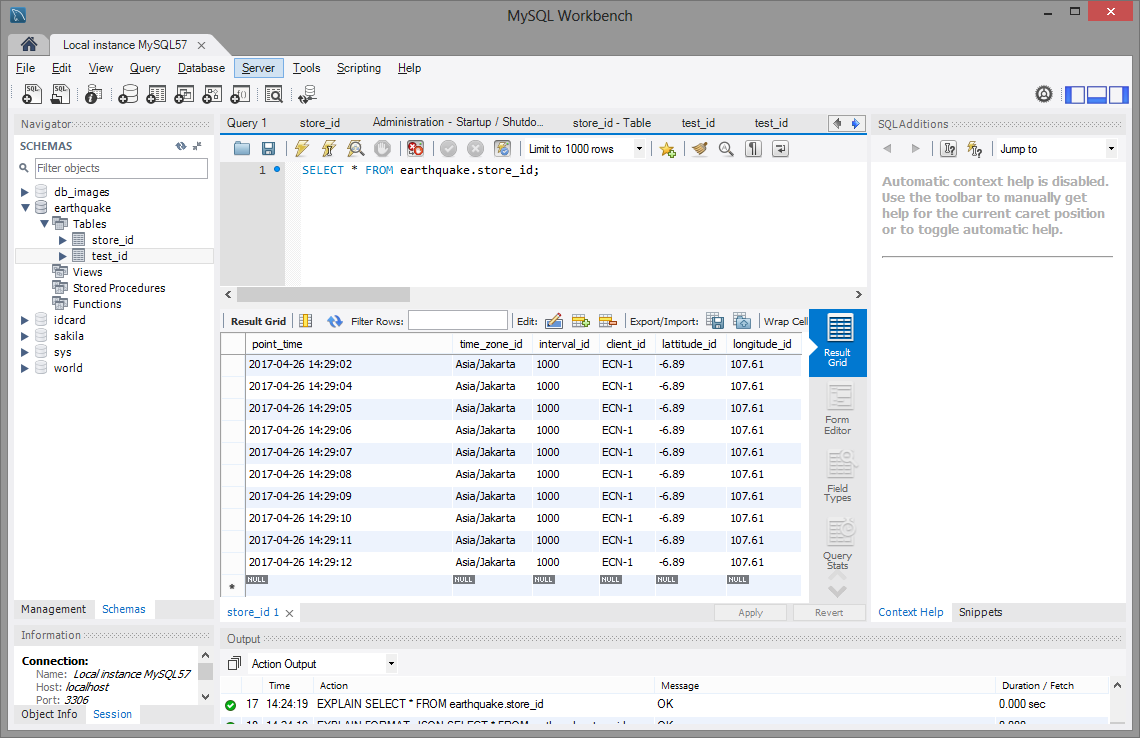
Pesan yang terkirim oleh sensor memiliki format yang sesuai dan data yang valid. Validitas pesan ini akan dicek kemudian oleh consumer.

### Pengujian Database MySQL

Langkah selanjutnya, akan dicek apabila pesan tersebut berhasil diunduh dari Messaging Server. Ketika program consume oleh console C# dijalankan, maka hasilnya adalah sebagai berikut:



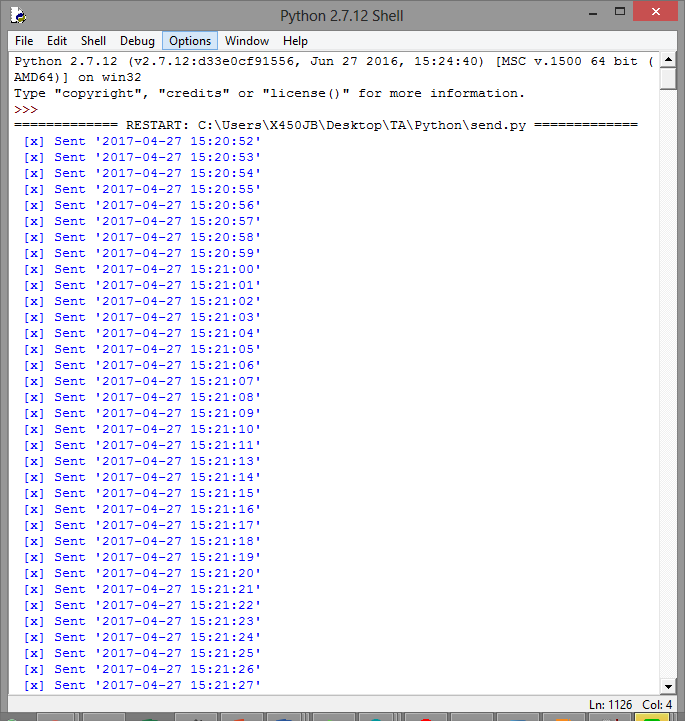
Setelah melakukan consume, maka program consumer akan mengirimkan pesan menuju MySQL. Hasilnya adalah sebagai berikut:



Data yang tersimpan ini akan digunakan untuk pengolahan data gempa.

### Pengujian Latency Jaringan Sistem

Namun selain kemampuan untuk dapat mengirimkan pesan, sistem ini juga harus handal untuk dapat berjalan selama mungkin. Oleh karena itu, keaslian data secara aktual juga harus diperhitungkan. Pengujian dilakukan dengan mengirimkan sebuah pesan Timestamp dari sebuah producer kemudian mencocokkan Timestamp tsb dengan Timestamp consumer. Hasil pengiriman Timestamp producer dengan menggunakan Python adalah sebagai berikut:



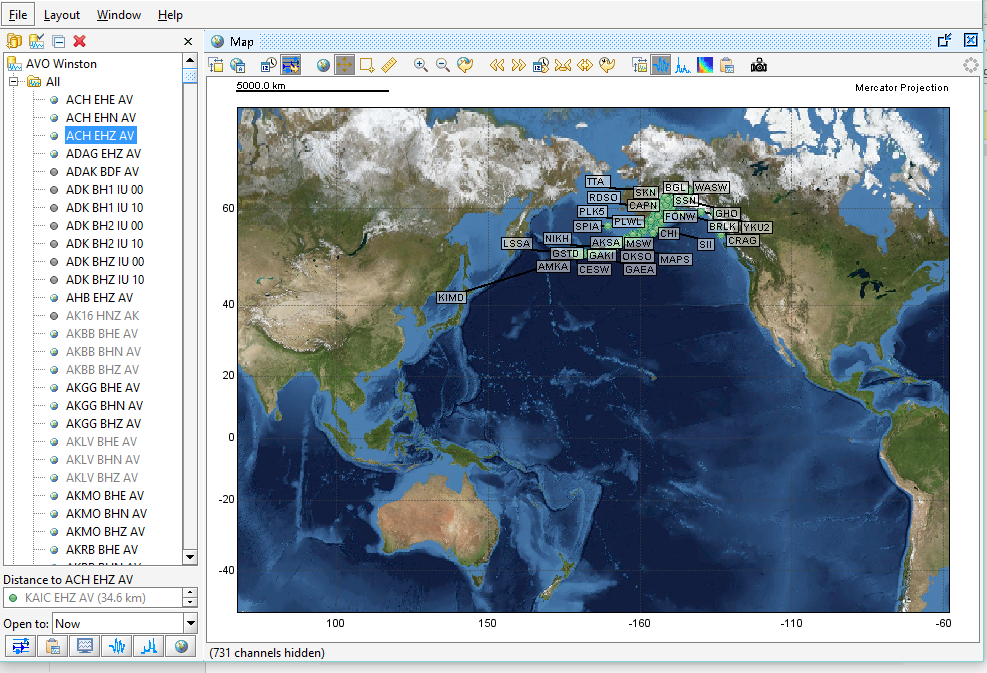
Kemudian pesan di dalam Messaging Server di consume oleh console C# dan digabungkan dengan Timestamp consumer. Kedua time stamp ini kemudian disimpan ke dalam Database MySQL dan di export datanya untuk pengolahan lebih lanjut. Hasil pengolahannya adalah sebagai berikut:

Artinya terdapat beberapa buah data yang tidak real time karena terlambat untuk sampai ke pengolahan server. Setelah diolah datanya, didapatkan hasil sebagai berikut:

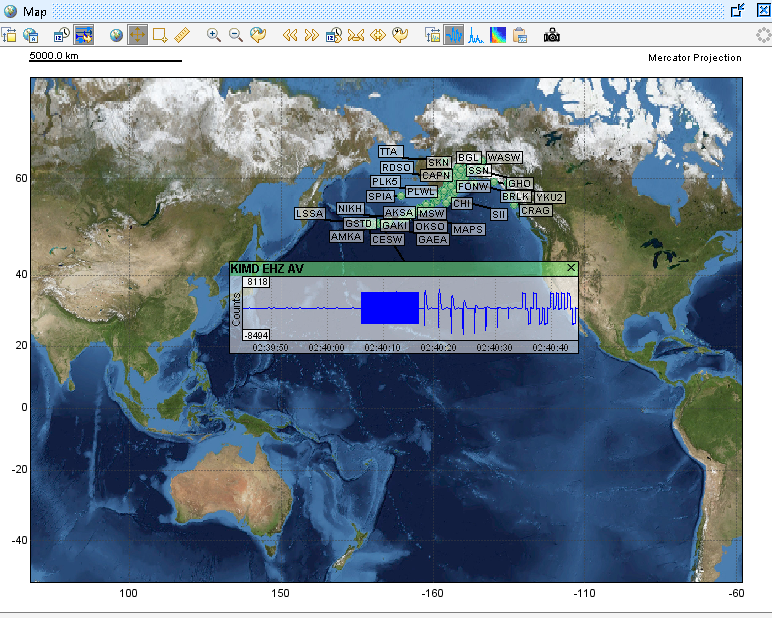
Terdapat peluang sebesar 61% bahwa data yang dikirimkan adalah data real-time ketika melalui sistem jaringan.

## Pengujian Algoritma Analisis Gempa

### Pengujian Algoritma Umum Program



Algoritma umum yang dimaksud disini adalah fungsionalitas dasar dari program analisis yang dibuat, yaitu seperti menampilkan peta yang disediakan oleh NASA, koneksi ke jaringan sensor gempa yang dimiliki USGS, serta berfungsinya semua menu dan toolbar yang disediakan pada tampilan. Dari gambar diatas dapat dilihat beberapa fungsi yang sudah berjalan dengan cukup baik. Pada sebelah kiri terdapat daftar sensor yang bisa diakses oleh program. Sensor dengan lingkaran hijau adalah sensor yang bisa diakses saat itu juga, sedangkan sensor dengan lingkaran hijau pucat adalah sensor yang aktif namun sedang mengalami kesulitan untuk mengirim data. Sensor yang tidak aktif ditandai dengan lingkaran berwarna abu-abu. Pada sebelah kanan terdapat peta dengan penanda GPS untuk masing-masing sensor yang aktif. Beberapa tombol penting yang bisa digunakan untuk pengaturan tampilan antara lain yaitu pembesaran/pengecilan, koneksi ke server baru, tampilan *waveform,* serta mode kiosk.

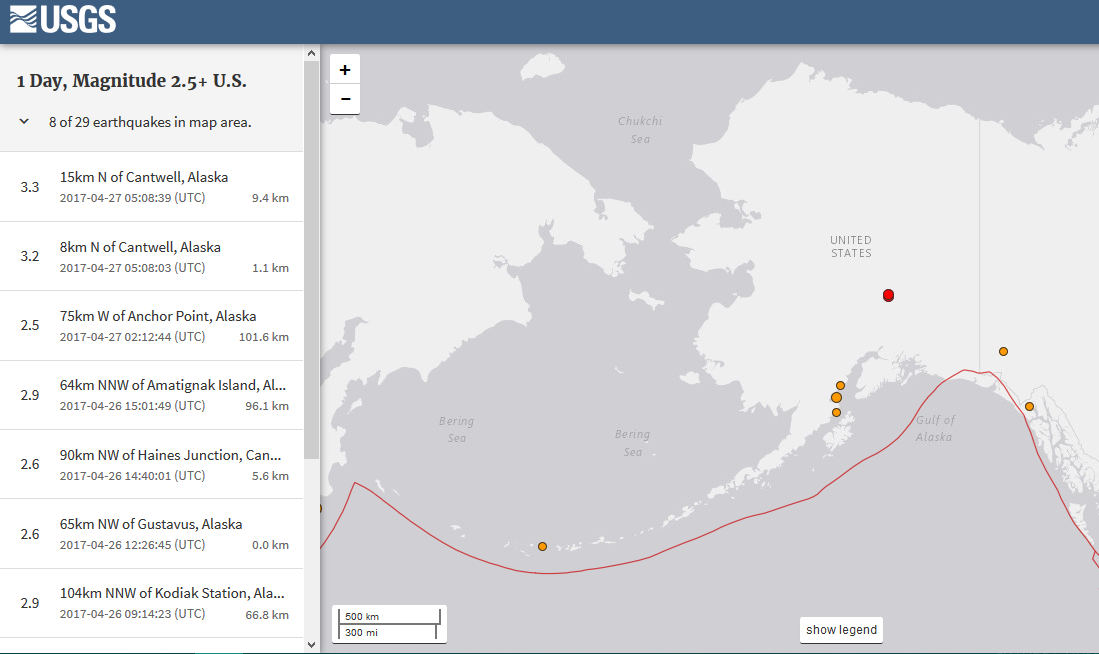


Tampilan diatas menunjukkan pembacaan sensor yang didapat ketika salah satu penanda GPS pada peta diklik. Pembacaan tersebut adalah data yang didapat sejak satu menit terakhir, dihitung dari data terakhir yang didapat dari sensor. Pembacaan data dari semua sensor belum berlangsung secara *real-time*, karena ada *latency* dari pengiriman data oleh sensor hingga data tiba di database dan bisa diproses. Data *latency* dari masing-masing sensor dapat dilihat pada tabel berikut.

|  |  |  |
| --- | --- | --- |
| Lokasi Sensor | Jarak Sensor ke Pemancar GSM | Latency (Rata-rata) |
| KAKN BHV | 3.2 km | 1.2s |
| GSTD EHN | 5.0 km | 1.1s |
| KABU BHE | 7.2 km | 1.9s |
| KCE EHZ | 9.1 km | 1.7s |
| ANCK EHZ | 9.9 km | 4.3s |
| KCG EHE | 16.5 km | 3.2s |
| KAIC EHZ | 34.6 km | 3.6s |

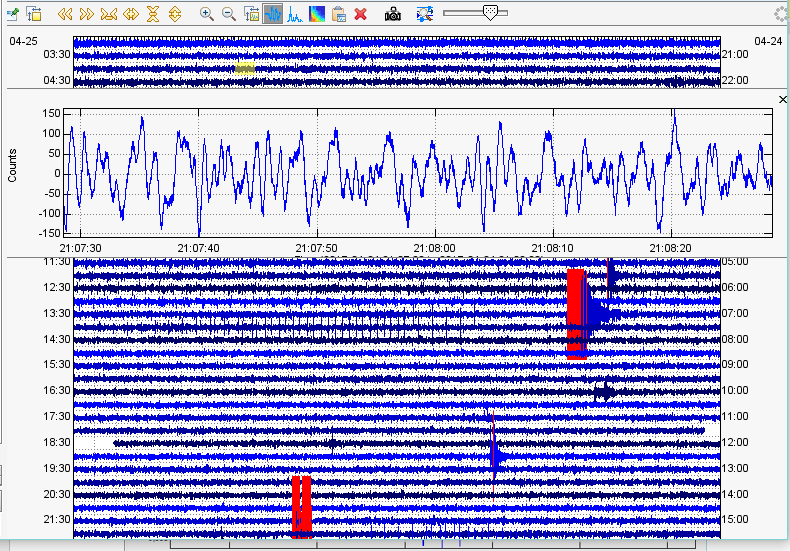
Data tersebut diambil dari jaringan sensor yang dimiliki oleh USGS. Namun, kami berekspektasi bahwa *latency* pada jaringan kami hampir sama karena desainnya serupa. Jaringan yang digunakan adalah GSM. Dengan data seperti diatas, GSM dirasa sudah cukup cepat untuk mentransmisikan data dari sensor ke server.

### Pengujian Algoritma *Counting*



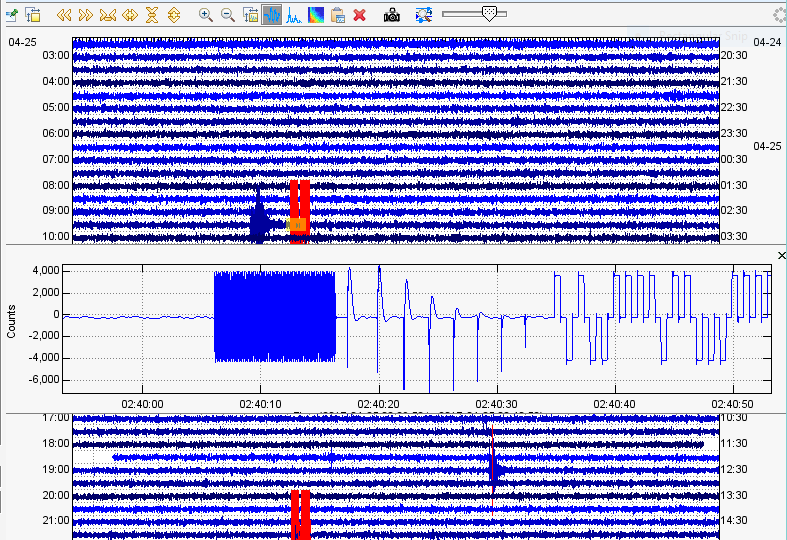
Untuk melakukan pengujian algoritma analisis data seismik kami, digunakan cara sebagai berikut. Kami melakukan analisis dengan menerapkan algoritma kami pada data dari jaringan sensor gempa yang dimiliki USGS. Jaringan sensor tersebut terletak di Alaska, Amerika Serikat. Untuk membuktikan apakah kesimpulan dari hasil analisis kami benar atau tidak, kami membandingkannya dengan hasil analisis yang dilakukan oleh USGS sendiri. Hasil tersebut dapat dilihat di *website* USGS, dimana dapat dilihat gempa terakhir yang terjadi di suatu negara bagian di Amerika Serikat. Data yang ditampilkan seperti gambar diatas. Data tersebut berupa *magnitude*, jarak pusat gempa dari tempat terdekat yang berpenghuni, kedalaman gempa dari permukaan laut, serta waktu terjadinya gempa tersebut. Kami memilih untuk membandingkan dengan jaringan gempa di Alaska, karena di daerah tersebut cukup sering terjadi gempa.

Implementasi algoritma kami menghitung *magnitude* dari gempa yang terjadi, frekuensi getaran, *count* dari getaran, serta spektrum frekuensi dari getaran, yang nantinya akan digunakan untuk menentukan lokasi gempa. Namun, hingga dokumen ini ditulis, implementasi dari algoritma untuk menentukan lokasi gempa belum selesai diimplementasikan. Maka itu, kami hanya membandingkan *magnitude* dan frekuensi dari gempa hasil analisis kami sendiri dengan hasil yang diperoleh dari website USGS. Mengenai algoritma *count*, fungsi dari algoritma ini adalah menghitung jumlah perubahan arah *jerk*, atau turunan dari akselerasi per jam. Nilai *count* yang kurang dari 100 bisa dianggap sebagai keadaan normal. Apabila nilai *count* melebihi3600, atau 1 Hz, maka bisa dianggap bahwa sensor tersebut mendeteksi ada gempa yang terjadi.

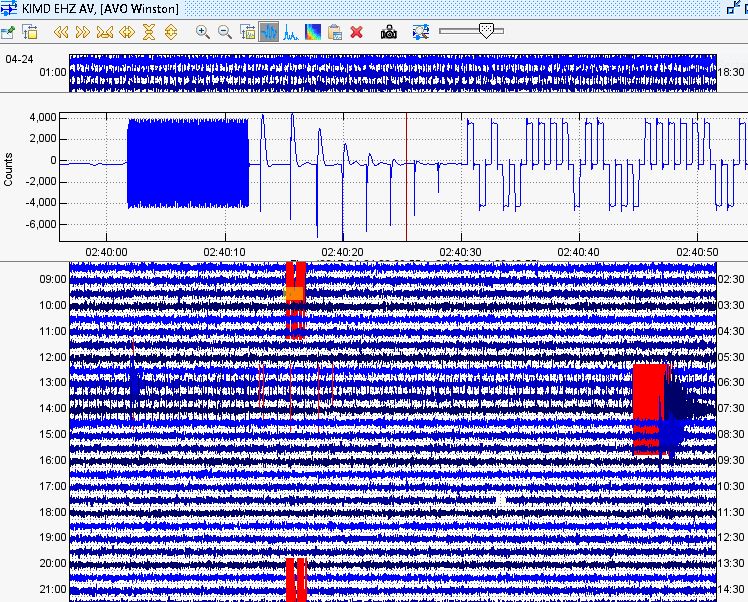


Gambar diatas adalah penerapan algoritma *count* pada data ketika keaaan normal. Bisa dilihat bahwa nilai *count* tidak ada yang melebihi 200. Dengan kata lain, ada getaran namun dengan frekuensi yang sangat rendah. Getaran ini dianggap sebagai *noise*, karena sensor tidak didesain untuk mendeteksi getaran dengan frekuensi dibawah 1 Hz. Bagian yang di-*highlight* dengan warna kuning adalah bagian dari rekaman sensor gempa yang menjadi fokus. Bagian yang bisa menjadi fokus dalam satu waktu sepanjang sekitar 1 menit.

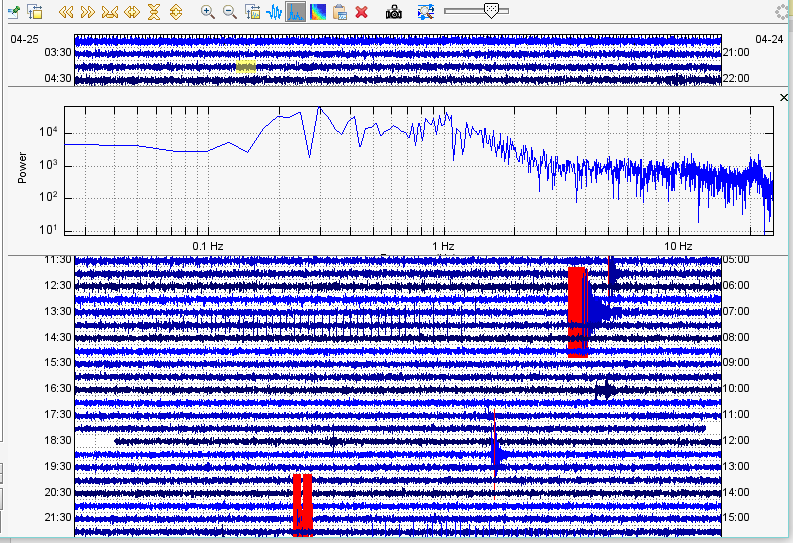
Warna merah tersebut menunjukkan adanya getaran yang dideteksi diatas 1 Hz. Getaran seperti ini mengindikasikan adanya gempa. Namun, bisa juga ada getaran lokal yang dialami oleh salah satu sensor, seperti kendaraan yang lewat atau semacamnya. Maka itu, apabila data gempa diperoleh pada satu sensor saja, belum dianggap terjadi gempa. Dibutuhkan minimal tiga sensor yang mendeteksi gempa pada waktu yang berdekatan untuk menganggap bahwa gempa telah terjadi. Angka tiga ini dipilih karena dibutuhkan minimal tiga sensor untuk melakukan analisis mengenai lokasi gempa. Contoh dari getaran yang tidak dinaggap gempa adalah warna merah yang terjadi pada jam 07.00 di gambar diatas, serta warna merah saat jam 15.00



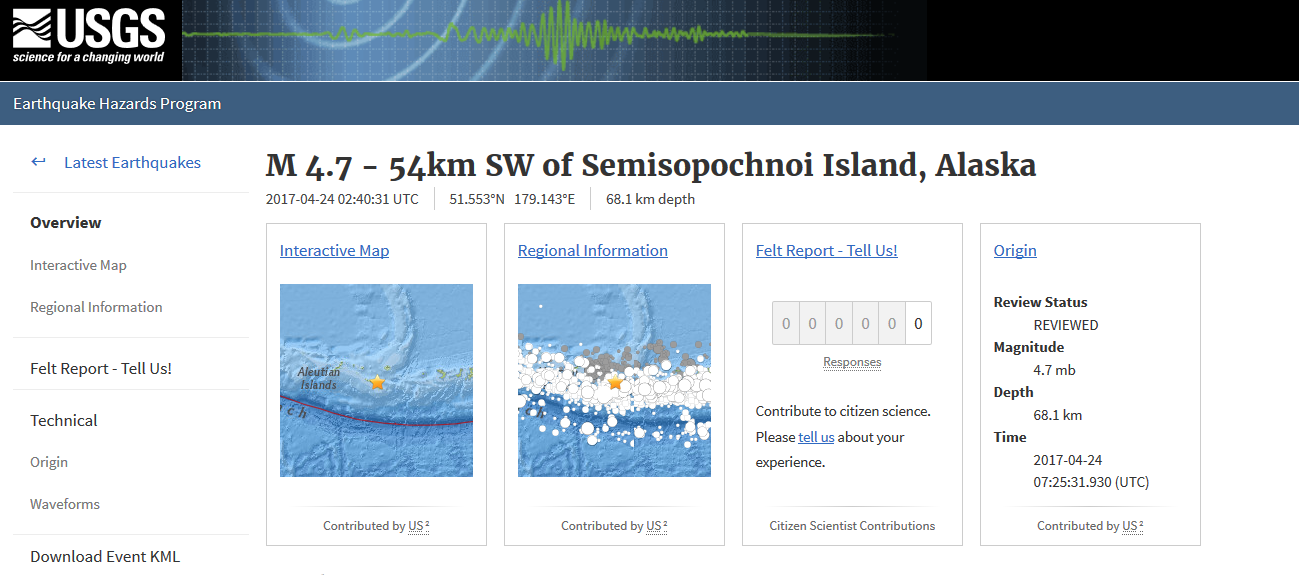
Gambar diatas adalah contoh dari getaran yang diambil saat gempa sungguhan terjadi. Data getaran tersebut juga dialami oleh sensor-sensor lain, dengan adanya perbedaan pada waktu terjadinya getaran tersebut. Perbedaan waktu ini terjadi karena getaran gempa perlu waktu untuk merambat ke sensor yang letaknya cukup jauh. Bila bagian yang berwarna merah difokuskan, maka bisa dilihat bahwa saat terjadi gempa, nilai *count* berubah dengan sangat cepat dan signifikan. Hal ini artinya terjadi getaran dengan frekuensi yang dianggap cukup tinggi untuk sebuah gempa tektonik. Pada gambar dibawah, getaran yang sama terkam di sensor lain, dengan perbedaan waktu yang kecil.



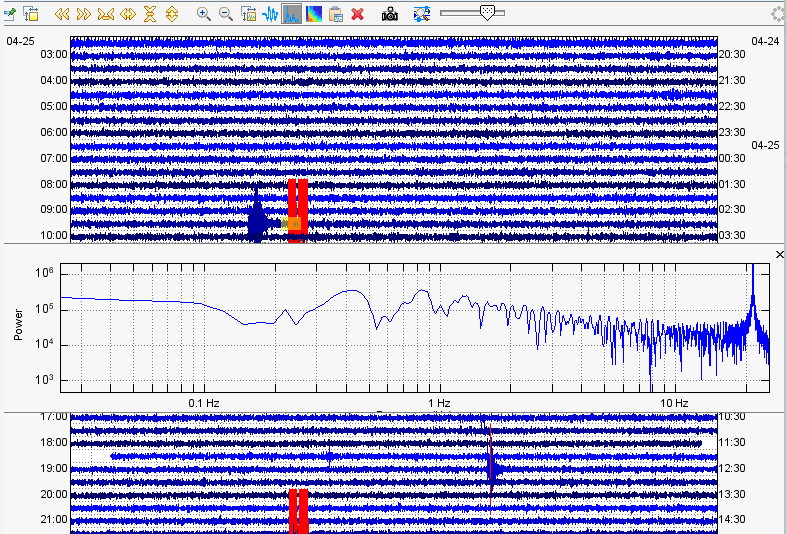
### Pengujian Algoritma Magnitude



Bagian ini akan menjelaskan pengujian algoritma untuk menghitung *magnitude* dari gempa. *Magnitude* disini didasarkan pada *moment magnitude*, atau kekuatan gempa yang diukur berdasarkan jarak perpindahan lempeng tektonik, dan gaya yang dibutuhkan untuk melakukan perpindahan tersebut. Pengujian dengan metode ini dirasa lebih baik dibandingkan dengan skala *Richter,* karena skala momen lebih akurat untuk gempa dengan kekuatan besar. Pada keadaan normal, analisis menghasilkan data seperti gambar diatas. Diatas frekuensi 1-2 Hz, *magnitude* yang terukur selalu kurang dari 3, yaitu batas ketelitian sensor. Karena itu, data dengan *magnitude* dibawah 3 dianggap bukan gempa.



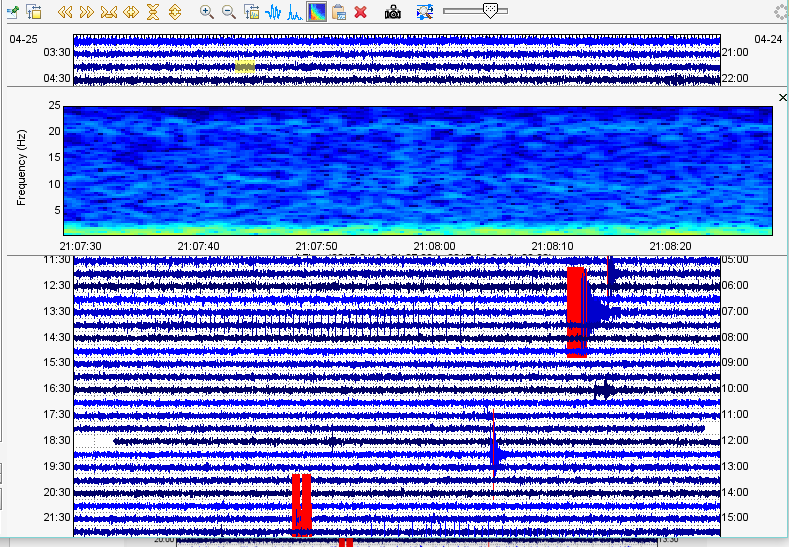
Pada gambar diatas, terdapat data gempa yang terjadi di sekitar Alaska, pada jam 02.40 dengan kekuatan 4,7. Gempa yang sama terdeteksi pada sensor seperti yang terlihat pada gambar dibawah. Pada analisis ini, terlihat ada lonjakan pada *magnitude* gempa dengan frekuensi sekitar 11 Hz. Puncaknya sekitar *magnitude* 6*,* namun rata-rata dari lonjakan tersebut sekitar 5, atau tepatnya 4,93. Hal ini hampir sama dengan kesimpulan yang diperoleh USGS.



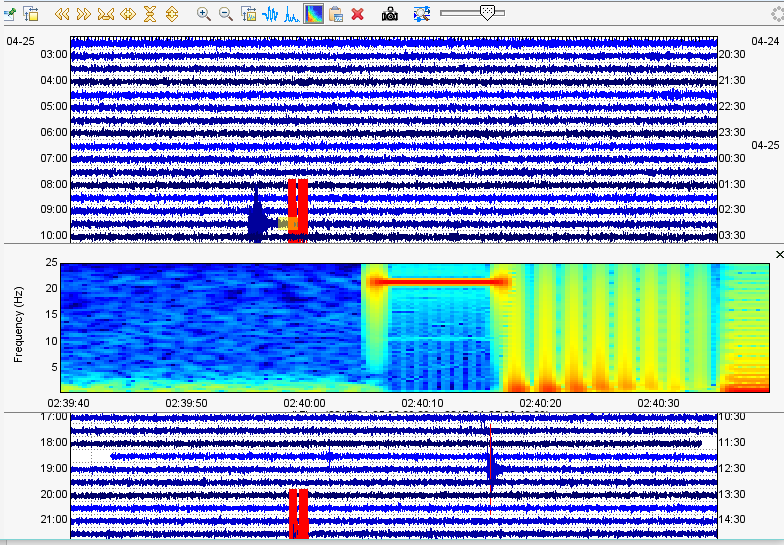
Tabel dibawah menunjukkan beberapa gempa yang terjadi sebelum dokumen ini ditulis, data yang berkaitan, dan hasil analisis menggunakan algoritma kami. Dari tabel tersebut dapat disimpulkan bahwa algoritma sudah cukup akurat untuk gempa diatas skala 4, namun dibawah itu hasil analisis tidak bisa diandalkan.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No. | Waktu | Lokasi | Kedalaman | *Magnitude* USGS | Magnitude Algoritma |
| 1 | 2017-04-26 07:02:28 (UTC) | 53.828°N 160.645°E | 39.8 km | 4.9 | 4.8765 |
| 2 | 2017-04-26 15:01:49 (UTC) | 51.817°N 179.367°W | 96.1 km | 2.9 | 3.5638 |
| 3 | 2017-04-26 09:14:23 (UTC) | 58.548°N 153.567°W | 66.8 km | 2.9 | 3.3653 |
| 4 | 2017-04-27 02:12:44 (UTC) | 59.750°N 153.179°W | 101.6 km | 2.5 | 3.3673 |
| 5 | 2017-04-27 05:08:39 (UTC) | 63.532°N 148.976°W | 9.4 km | 3.3 | 3.6563 |

### Pengujian Algoritma Spektrum Frekuensi



Algoritma spektrum frekuensi menggunakan transformasi Fourier dengan metode Fast Fourier Transform (FFT) untuk menentukan spektrum frekuensi dari gempa pada waktu tertentu. Spektrum frekuensi ini nantinya akan digunakan untuk menentukan lokasi dari gempa. Namun hingga dokumen ini ditulis, algoritma lokasi belum selesai diimplementasikan sehingga belum bisa diuji.



## LAMPIRAN

**Tabel perhitungan lama proses algoritma dengan kondisi tanpa menunggu queue RabbitMQ**

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Process Time (ms)** | | |
| **IMU Processing** | **GPS Processing** | **MQTT Publish Processing** |
| **1** | **1** | **0** | **256** |
| **2** | **9** | **0** | **212** |
| **3** | **1** | **1** | **212** |
| **4** | **18** | **3** | **228** |
| **5** | **1** | **0** | **210** |
| **6** | **17** | **3** | **236** |
| **7** | **2** | **0** | **212** |
| **8** | **1** | **0** | **221** |
| **9** | **17** | **3** | **211** |
| **10** | **16** | **0** | **248** |
| **11** | **1** | **0** | **217** |
| **12** | **16** | **4** | **229** |
| **13** | **1** | **0** | **208** |
| **14** | **17** | **2** | **220** |
| **15** | **1** | **0** | **208** |
| **16** | **2** | **0** | **221** |
| **17** | **17** | **3** | **208** |
| **18** | **1** | **0** | **209** |
| **19** | **1** | **0** | **246** |
| **20** | **2** | **0** | **214** |
| **21** | **17** | **2** | **218** |
| **22** | **2** | **0** | **208** |
| **23** | **1** | **0** | **252** |
| **24** | **18** | **3** | **206** |
| **25** | **1** | **1** | **209** |
| **26** | **17** | **2** | **231** |
| **27** | **1** | **0** | **245** |
| **28** | **1** | **0** | **215** |
| **29** | **18** | **3** | **207** |
| **30** | **1** | **0** | **234** |
| **31** | **17** | **2** | **223** |
| **32** | **1** | **0** | **217** |
| **33** | **16** | **2** | **236** |
| **34** | **1** | **0** | **217** |
| **35** | **16** | **2** | **236** |
| **36** | **1** | **0** | **224** |
| **37** | **18** | **2** | **233** |
| **38** | **1** | **1** | **224** |
| **39** | **17** | **2** | **228** |
| **40** | **18** | **2** | **275** |
| **41** | **1** | **0** | **269** |
| **42** | **1** | **0** | **254** |
| **43** | **1** | **0** | **231** |
| **44** | **1** | **0** | **227** |
| **45** | **18** | **2** | **235** |
| **46** | **2** | **0** | **214** |
| **47** | **17** | **3** | **235** |
| **48** | **1** | **0** | **215** |
| **49** | **1** | **0** | **271** |
| **50** | **16** | **2** | **251** |
| **51** | **1** | **0** | **214** |
| **52** | **17** | **2** | **274** |
| **53** | **1** | **1** | **221** |
| **54** | **18** | **2** | **243** |
| **55** | **2** | **0** | **238** |
| **56** | **1** | **0** | **276** |
| **57** | **17** | **3** | **227** |
| **58** | **1** | **0** | **276** |
| **59** | **17** | **3** | **274** |
| **60** | **1** | **0** | **322** |
| **Average** | **7,666666667** | **1,016666667** | **232,1833333** |
| **Maximum** | **18** | **4** | **322** |
| **Minimum** | **1** | **0** | **206** |

**Source Code GUI ECN Data Logger**

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.ComponentModel;  using System.Data;  using System.Drawing;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Windows.Forms;  using System.IO.Ports;  using ClosedXML.Excel;  namespace ECN\_Data\_Logger  {    public partial class FormECNDataLogger : Form  {  static string DataXYZ;  public float[] Data = new float[3];  public bool record = false;  DataTable table = new DataTable();  public float[,] Sample = new float[120,3];  int i = 0, N=0;  public FormECNDataLogger()  {  InitializeComponent();  GetPortName();  table.Columns.Add("Acc X", typeof(string));  table.Columns.Add("Acc Y", typeof(string));  table.Columns.Add("Acc Z", typeof(string));  }  void GetPortName()  {  String[] portname = SerialPort.GetPortNames();  PortNameBox.Items.AddRange(portname);  }  private void OpenPortButton\_Click(object sender, EventArgs e)  {  try  {  if(PortNameBox.Text == "" || BaudRateBox.Text == "")  {  MessageBox.Show("Please select port settings !!!","Error");  }  else  {  SerialPort.PortName = PortNameBox.Text;  SerialPort.BaudRate = Convert.ToInt32(BaudRateBox.Text);  //SerialPort.DataReceived += new SerialDataReceivedEventHandler(DataReceivedHandler);  SerialPort.Open();  StatusBar.Value = 100;  OpenPortButton.Enabled = true;  ClosePortButton.Enabled = true;  Box.Enabled = true;  StartButton.Enabled = true;  StopButton.Enabled = false;  }  }  catch(UnauthorizedAccessException)  {  MessageBox.Show("Unauthorized Access !!!", "Error");  }  }  private void ClosePortButton\_Click(object sender, EventArgs e)  {  timer1.Stop();  SerialPort.Close();  StatusBar.Value = 0;  Box.Enabled = false;  ClosePortButton.Enabled = false;  StopButton.Enabled = false;  StartButton.Enabled = true;  }  /\*  private static void DataReceivedHandler( object sender, SerialDataReceivedEventArgs e)  {  SerialPort sp = (SerialPort)sender;  string indata = sp.ReadExisting();  DataXYZ = indata;  }  \*/    private void timer1\_Tick(object sender, EventArgs e)  {  /\*  try  {\*/  DataXYZ = SerialPort.ReadLine();  SerialDataBox.AppendText(DataXYZ);  SerialDataBox.AppendText("\n");  char[] delimiterChars = {'\t'};  String[] XYZ = DataXYZ.Split(delimiterChars);  //Data[0] = Convert.ToSingle(XYZ[0]);  //Data[1] = Convert.ToSingle(XYZ[1]);  //Data[2] = Convert.ToSingle(XYZ[2]);  XYZ[0] = XYZ[0].Replace(".",",");  XYZ[1] = XYZ[1].Replace(".",",");  XYZ[2] = XYZ[2].Replace(".",",");  if (record)  {  table.Rows.Add(XYZ[0],XYZ[1],XYZ[2]);  /\*  N++;  Sample[i,0] = Data[0];  Sample[i,1] = Data[1];  Sample[i,2] = Data[2];  i++;  if (i == 120)  i = 0;  \*/  }  /\*  }  catch (Exception)  {  MessageBox.Show("Exception !!!","Error");  }  \*/  }  private void StartButton\_Click(object sender, EventArgs e)  {  timer1.Start();  StartButton.Enabled = false;  StopButton.Enabled = true;  }  private void StopButton\_Click(object sender, EventArgs e)  {  timer1.Stop();  StopButton.Enabled = false;  StartButton.Enabled = true;  }  private void ActivityBox\_TextChanged(object sender, EventArgs e)  {  }  private void ClearButton\_Click(object sender, EventArgs e)  {  SerialDataBox.Clear();  }  private void RecordButton\_Click(object sender, EventArgs e)  {  StopRecordButton.Enabled = true;  RecordButton.Enabled = false;  record = true;  i = 0;  //table.Rows.Add("Amplitude", AmplitudeBox.Text,"G");  //table.Rows.Add("Frequency", FrequencyBox.Text,"Hz");  }  private void StopRecordButton\_Click(object sender, EventArgs e)  {  StopRecordButton.Enabled = false;  RecordButton.Enabled = true;  record = false;  /\*for(i=0;i<120;i++)  {  table.Rows.Add(Sample[i,0], Sample[i,1], Sample[i,2]);  }\*/  DataGridView.DataSource = table;  }  private void ClearDataButton\_Click(object sender, EventArgs e)  {  table.Clear();  DataGridView.DataSource = table;  i = 0;  }  private void button1\_Click(object sender, EventArgs e)  {  try  {  SaveFileDialog sfd = new SaveFileDialog();  sfd.Filter = "Excel Documents (\*.xlsx)|\*.xlsx";  String name = comboBox1.SelectedItem.ToString() + " "+ AmplitudeBox.SelectedItem.ToString() + "G " + FrequencyBox.SelectedItem.ToString() + "Hz";  sfd.FileName = "Test MPU9255 " + name +".xlsx";  if (sfd.ShowDialog() == DialogResult.OK)  {  var workbook = new XLWorkbook();  workbook.Worksheets.Add(table, "Test MPU9255");  workbook.SaveAs(sfd.FileName);  table.Clear();  MessageBox.Show("Save File Success !", "Information", MessageBoxButtons.OK, MessageBoxIcon.Information);  }  }  catch  {  MessageBox.Show("Save File Failed", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);  }  }  }  } |