

**PERANCANGAN DAN IMPLEMENTASI SENSOR SEISMIK  
ECN PADA SISTEM JARINGAN DETEKTOR GEMPA DAN  
TSUNAMI DECISION SUPPORT SYSTEM**

**TUGAS AKHIR**

Oleh

**BRAMANTIO YUWONO**

**NIM : 13213126**

**Program Studi Teknik Elektro**



**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA  
INSTITUT TEKNOLOGI BANDUNG  
2017**

**PERANCANGAN DAN IMPLEMENTASI SENSOR SEISMIK  
PADA SISTEM JARINGAN DETEKTOR GEMPA DAN  
TSUNAMI DECISION SUPPORT SYSTEM**

**Oleh :**

**Bramantio Yuwono**

Tugas Akhir ini telah diterima dan disahkan  
sebagai persyaratan untuk memperoleh gelar

**SARJANA TEKNIK**

di

**PROGRAM STUDI TEKNIK ELEKTRO  
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA  
INSTITUT TEKNOLOGI BANDUNG**

Bandung, 12 Mei 2017

Disetujui oleh :

Pembimbing I,

Pembimbing II,

Dr. Ary Setijadi Prihatmanto, ST, MT  
NIP. 197208271997021003

Dr. Ir. Aciek Ida Wuryandari MT.  
NIP. 195410051982112001

# **ABSTRAK**

## **PERANCANGAN DAN IMPLEMENTASI SENSOR SEISMIK PADA SISTEM JARINGAN DETEKTOR GEMPA DAN TSUNAMI DECISION SUPPORT SYSTEM**

**Oleh**

**Bramantio Yuwono**

**NIM: 13213126**

**PROGRAM STUDI TEKNIK ELEKTRO**

Sensor Seismik ECN (Earthquake Catcher Network) merupakan sensor yang dapat melakukan pembacaan percepatan getaran tanah, *timestamp*, dan data lokasi sensor lalu dikirimkan data hasil pembacaan tersebut ke server dengan protokol MQTT dengan menggunakan koneksi *Wi-Fi*. Sensor seismik terdiri dari sensor IMU (*Inertial Measurement Unit*), modul GPS (*Global Positioning System*) dan sistem daya. Sensor IMU digunakan untuk melakukan *sampling* percepatan getaran tanah dengan frekuensi *sampling* 40 Hz dan digunakan juga untuk melakukan pembacaan data magnetometer sebagai pengaturan orientasi sensor. Modul GPS digunakan untuk memperoleh pembacaan data *latitude* dan *longitude* letak sensor dan *timestamp* yang terdiri dari data detik, menit, jam, hari, bulan, dan tahun. Sistem daya yang terdiri dari *solar panel*, *charging module*, dan baterai li-ion berfungsi untuk menyuplai daya sensor mandiri yang memastikan sensor tersuplai daya selama 24 jam setiap harinya. Hasil dari implementasi sensor ini adalah *sampling* data percepatan getaran tanah dengan frekuensi *sampling* 40 Hz tidak dapat dilakukan dengan sempurna dikarenakan waktu pengiriman data ke server membutuhkan waktu yang lebih lama dari waktu *sampling* dan sensor ECN dapat bertahan selama 24 jam dengan menggunakan sistem daya yang telah didesain.

Kata kunci : Gelombang Seismik , Modul GPS, Sensor IMU , Sensor Seismik ECN, Solar Panel

## **ABSTRACT**

### **SEISMIC SENSOR DESIGN AND IMPLEMENTATION ON EARTHQUAKE AND TSUNAMI DETECTOR NETWORK AND DECISION SUPPORT SYSTEM**

**By**

**Bramantio Yuwono**

**NIM : 13213126**

**ELECTRICAL ENGINEERING**

The ECN (Earthquake Catcher Network) Seismic Sensor is a sensor that can read the acceleration of ground vibration, timestamp, and sensor location data and then send the reading data to the server with MQTT protocol using Wi-Fi connection. The seismic sensor consists of IMU (Inertial Measurement Unit) sensors, GPS module (Global Positioning System) and power system. The IMU sensor is used for sampling of ground vibration acceleration with 40 Hz sampling frequency and is also used to perform magnetometer data reading as sensor orientation adjustment. The GPS module is used to obtain the latitude and longitude readings of sensor and timestamp which consist of data of seconds, minutes, hours, days, months, and years. The power system consisting of solar panels, charging modules, and li-ion batteries serves to supply independent sensor power that ensures the sensor is supplied for 24 hours per day. The result of this sensor implementation is the sampling of ground vibration acceleration data with 40 Hz sampling frequency can not be done perfectly because the time of data delivery to the server takes longer than sampling time and ECN sensor can last for 24 hours with using power system which have been designed .

Kata kunci : Seismic Wave , GPS Module, IMU Sensor , ECN Seismic Sensor, Solar Panel

## KATA PENGANTAR

Puji syukur penulis panjatkan ke hadirat Allah ta'ala, yang atas rahmat dan karuniaNya penulis dapat menyelesaikan penulisan dan pengerjaan tugas akhir ini. Shalawat dan salam tercurah kepada Rasulullah Muhammad shallallahu 'alyhi wasallam beserta keluarganya, shahabatnya, dan para pengikutnya hingga akhir zaman.

Selama mengerjakan tugas akhir ini, penulis mendapat bantuan dan dukungan dari berbagai pihak. Untuk itu, penulis ingin mengucapkan terima kasih kepada:

- ✓ Ayah dan Ibuku, Prabowo Tri Irianto dan Yulia Zein, yang selalu membimbing, mendidik, mengajarkan kedisiplinan tanpa kenal lelah dan mendoakan penulis hingga bisa seperti sekarang ini, serta adikku tercinta, Adityo Gea Devaro, yang selalu mendukung dan membantu segala keputusanku sampai saat ini;
- ✓ Bapak Dr. Ary Setijadi Prihatmanto, ST, MT dan Ibu Dr. Ir. Aciek Ida Wuryandari MT, selaku Dosen Pembimbing yang telah membimbing penulisan serta memberikan arahan dalam pengerjaan tugas akhir ini;
- ✓ Ibu Lina Handayani dan Bapak Ridwan, selaku pegawai pusat penelitian geoteknologi LIPI dan pegawai balai informasi LIPI yang sudah dengan baik hati telah menjadi tempat untuk berkonsultasi dan bertanya dalam proses pengerjaan tugas akhir ini;
- ✓ Bapak Lutfi selaku teknisi lab. Dinamika yang selalu semangat dan untuk membimbing kami dalam menggunakan fasilitas lab tersebut;
- ✓ Christoporus Deo Putratama dan Kevin Shidqi Prakoso selaku 1 tim seperjuangan bersama dengan penulis yang tidak kenal lelah menyelesaikan setiap tantangan yang hadir dalam pengerjaan tugas akhir ini maupun lomba LSI Design Contest di Okinawa, selalu memberikan motivasi dan semangat satu sama lain, semoga bisa bertemu lagi dengan cerita kesuksesan masing – masing;
- ✓ Teman-teman seperjuangan di ruang riset mandiri tempat penulis mengerjakan tugas akhir ini yang selalu memberikan keceriaan, dukungan, dan motivasi untuk pantang menyerah;

- ✓ Brian Benyamin R, Yosua Sepri Andasto, Muhammad Aldo Aditya Nugroho, Fiana Fauzia, Larissa Arindini dan Reyhan Fariz Taqwantara yang sudah membantu penulis dalam menyusun video dokumentasi tugas akhir elektro ini;
- ✓ Seluruh teman-teman Teknik Elektro ITB angkatan 2013, serta seluruh teman-teman HME ITB;
- ✓ dan semua pihak yang membantu, yang tidak dapat penulis sebutkan satu persatu.

Penulis menyadari bahwa pengerjaan tugas akhir ini tidaklah sempurna. Karena itu kritik dan saran sangat diharapkan oleh penulis.

Akhir kata, semoga buku tugas akhir ini dapat bermanfaat bagi para pembaca.

Bandung, 12 Mei 2017

Penulis

Bramantio Yuwono, 13213126

## DAFTAR ISI

<b>ABSTRAK .....</b>	<b>III</b>
<b>ABSTRACT .....</b>	<b>IV</b>
<b>KATA PENGANTAR .....</b>	<b>V</b>
<b>DAFTAR ISI.....</b>	<b>VII</b>
<b>DAFTAR GAMBAR .....</b>	<b>IX</b>
<b>DAFTAR SINGKATAN DAN LAMBANG .....</b>	<b>X</b>
<b>BAB I PENDAHULUAN .....</b>	<b>1</b>
I.1    Latar Belakang .....	1
I.2    Perumusan Masalah.....	2
I.3    Tujuan dan Manfaat Penelitian.....	2
I.4    Batasan Masalah.....	3
I.5    Metodologi .....	3
I.6    Sistematika Penulisan.....	4
<b>BAB II TINJAUAN PUSTAKA .....</b>	<b>5</b>
II.1    NodeMCU 1.0 (Modul ESP-12E ) .....	5
II.2    MPU9255 .....	5
II.3    GPS Ublox NEO-6M.....	6
II.4    MQTT.....	6
II.5    Baterai Li-ion .....	7
II.6    Charging Module TP4056 .....	7
<b>BAB III PERANCANGAN .....</b>	<b>8</b>
III.1    Algoritma Sensor.....	9
III.1.1 <i>Algoritma Pembacaan Sensor IMU MPU9255</i> .....	10

III.1.2	<i>Algoritma Pembacaan GPS</i> .....	13
III.1.3	<i>Algoritma Format Json</i> .....	14
III.1.4	<i>Algoritma Representasi Char</i> .....	17
III.1.5	<i>Algoritma Publish MQTT</i> .....	18
III.2	<i>Sistem Daya</i> .....	19
III.3	<i>Skema Instalasi Sensor</i> .....	21
<b>BAB IV IMPLEMENTASI DAN PENGUJIAN</b> .....		<b>24</b>
IV.1	Prosedur Pengujian .....	24
IV.1.1	Pengujian Ketelitian Pembacaan Latitude dan Longitude pada modul GPS	24
IV.1.2	Kalibrasi Pembacaan Accelerometer MPU9255.....	25
IV.1.3	Pengujian Lama Proses Algoritma.....	29
IV.1.4	Pengujian Sistem Daya .....	30
IV.2	Hasil Pengujian.....	31
IV.2.1	Pengujian Ketelitian Pembacaan Latitude dan Longitude pada modul GPS	31
IV.2.2	Kalibrasi Pembacaan Accelerometer MPU9255.....	32
IV.2.3	Pengujian Lama Proses Algoritma.....	32
IV.2.4	Pengujian Sistem Daya .....	34
IV.2.5	Evaluasi Harga Sensor Seismik ECN .....	36
<b>BAB V KESIMPULAN DAN SARAN</b> .....		<b>37</b>
V.1	Kesimpulan.....	37
V.2	Saran .....	37
<b>DAFTAR PUSTAKA</b> .....		<b>38</b>
<b>LAMPIRAN</b> .....		<b>39</b>



## DAFTAR GAMBAR

Gambar III.1 Blok Diagram Sensor Seismik ECN .....	8
Gambar III.2 Flowchart umum sensor seismik ECN .....	9
Gambar III.3 Transformasi Linear (Rotasi) .....	12
Gambar III.4 Skema Algoritma Representasi Char .....	18
Gambar III.5 Rangkaian Sistem Daya Sensor Seismik ECN.....	20
Gambar III.6 Desain PCB .....	21
Gambar III.7 Desain Casing Sensor Seismik ECN .....	22
Gambar III.8 Gambar Skema Instalasi .....	23
Gambar IV.1 Denah Titik Tempat Pengujian GPS .....	24
Gambar IV.2 Sensor Seismik Divisi Geoteknologi LIPI .....	25
Gambar IV.3 Gambar Cuplikan Data yang Terbaca pada Sensor Seismik LIPI .....	26
Gambar IV.4 Skema Pengujian Sensor IMU MPU9255 menggunakan Earthquake Exciter .....	26
Gambar IV.5 Earthquake Exciter, Vibration Controller, Data Acquisition .....	27
Gambar IV.6 Peletakkan Sensor IMU MPU9255 untuk Pengujian Sumbu y, x, dan z .....	27
Gambar IV.7 Tampilan GUI ECN Data Logger .....	28
Gambar IV.8 Grafik Lama Proses Algoritma Sensor Seismik ECN .....	33
Gambar IV.9 Rangkaian Sistem Daya yang dapat Menyuplai Sensor Seismik selama 24 Jam .....	35
Gambar IV.10 Sistem Daya dengan Modem Mifi .....	36

## DAFTAR SINGKATAN DAN LAMBANG

ECN	Earthquake Catcher Network
IMU	Inertial Measurement Unit
GPS	Global Positioning System
MQTT	Message Queueing Telemetry Transport
QCN	Quake Catcher Network
EWS	Early Warning System
NMEA	National Marine Electronics Association
GPIO	General Purpose Input/Output
PCB	Printed Circuit Board
PVC	Polyvinylchloride
LIPI	Lembaga Ilmu Pengetahuan Indonesia
PAU ITB	Pusat Antar Universitas Institut Teknologi Bandung

# **BAB I**

## **PENDAHULUAN**

### **I.1 Latar Belakang**

Secara geografis, Indonesia terletak pada pertemuan 3 lempeng tektonik utama dunia yang bergerak relatif saling mendesak satu dengan lainnya. Hal ini menyebabkan Indonesia menjadi negara yang di dalamnya rawan terjadi bencana, khususnya bencana gempa bumi. Berdasarkan data dari Badan Meteorologi Klimatologi dan Geofisika (BMKG), di wilayah Indonesia dapat dideteksi sekitar 4000 gempa bumi terjadi pertahun, sedangkan gempa bumi berkekuatan di atas 5,5 SR dan gempa bumi yang dapat dirasakan oleh manusia, terjadi rata-rata sekitar 70–100 kali per tahun, dan gempa bumi tektonik yang menimbulkan kerusakan terjadi antara 1–2 kali per tahun. Sejak tahun 1991 sampai dengan 2011 tercatat telah terjadi 186 kali gempabumi tektonik yang merusak. Gempa bumi dengan magnitudo besar (7 SR atau lebih) dengan kedalaman yang dangkal di bawah laut, dapat menimbulkan tsunami karena adanya perubahan ketinggian kolom air dalam waktu singkat. Oleh karena itu berdasarkan fakta-fakta di atas, kesiapan dan ketanggapan terhadap kondisi darurat dan bencana khususnya bencana gempa bumi dan tsunami menjadi sesuatu yang mendesak bagi negara dan masyarakat Indonesia.

Salah satu solusi yang dapat diupayakan dalam rangka meningkatkan kemampuan negara dan masyarakat Indonesia dalam menghadapi kondisi darurat kebencanaan adalah dengan membangun Sistem Peringatan Dini (Early Warning System – EWS) khususnya untuk bencana gempa bumi dan tsunami. Supaya EWS dapat terlaksana, pembangunan infrastruktur deteksi gempa yang efektif dan responsif menjadi urgensi yang utama. Hal ini dikarenakan jumlah seismometer yang ada di Indonesia tergolong sedikit untuk mencakup luas wilayah Indonesia. Selain itu, pengambilan keputusan untuk melakukan diseminasi juga harus didukung data dengan tingkat kepercayaan yang tinggi, dan itu bisa dilakukan ketika semakin banyak detektor gempa yang terpasang di Indonesia.

Earthquake Catcher Network (ECN) merupakan pengembangan teknologi yang dibuat dari teknologi yang sebelumnya dibuat oleh Stanford University dengan judul Quake Catcher Network(QCN). Teknologi detektor gempa yang murah dan diintegrasikan dengan jaringan internet memungkinkan peneliti untuk memperoleh data seismik dengan lebih akurat dan juga dapat meningkatkan performa diseminasi dan EWS itu sendiri. Dalam tulisan ini, penulis akan berfokus pada sensor seismik yang berfungsi untuk melakukan pembacaan data seismik/getaran tanah, timestamp, dan lokasi sensor dan mengirimkan data tersebut ke server dengan protokol MQTT pada koneksi Wi-Fi.

## **I.2 Perumusan Masalah**

Berdasarkan latar belakang di atas, maka masalah yang akan diangkat oleh penulis dalam tugas akhir ini adalah sebagai berikut:

- Bagaimana desain dan algoritma yang digunakan agar sensor ECN dapat memenuhi spesifikasi ?
- Bagaimana cara untuk mengurangi *missed sampling time rate* dalam proses pengiriman data ke server?
- Bagaimana desain sistem daya agar sensor dapat bekerja selama 24 jam pada setiap harinya ?

## **I.3 Tujuan dan Manfaat Penelitian**

Tugas akhir ini memiliki tujuan sebagai berikut.

1. Mendesain algoritma agar sensor ECN dapat berfungsi sesuai spesifikasi.
2. Memperoleh cara untuk mengurangi missed sampling time rate dalam proses pengiriman data ke server.
3. Mendesain sistem daya agar sensor dapat bekerja selama 24 jam pada setiap harinya.

## **I.4 Batasan Masalah**

Batasan masalah yang penulis rumuskan dalam tugas akhir ini adalah sebagai berikut:

- Algoritma yang diimplementasikan terbatas pada single tasking algorithm
- Sistem daya yang didesain bekerja dalam kondisi cuaca yang normal atau tidak hujan dan berawan
- Magnitude gempa yang dapat dideteksi sensor diatas 4 dalam radius 100 km

## **I.5 Metodologi**

Metode penelitian yang dilakukan dalam mengerjakan tugas akhir ini adalah sebagai berikut.

1. Penentuan topik
2. Studi pustaka
3. Penentuan spesifikasi
4. Perancangan sistem
5. Implementasi sistem rancangan
6. Pengujian dan perbaikan sistem

## **I.6 Sistematika Penulisan**

Penulis membagi laporan menjadi lima bab dengan urutan sebagai berikut :

### **BAB I PENDAHULUAN**

Pada bab ini dibahas mengenai latar belakang, perumusan masalah, tujuan dan manfaat penelitian, batasan masalah, metodologi pengerjaan, dan sistematika penulisan.

### **BAB II TINJAUAN PUSTAKA**

Pada bab ini dijabarkan dasar-dasar teori yang menjadi acuan dasar penulis dalam merancang dan mengimplementasikan sensor seismik ECN

### **BAB III PERANCANGAN**

Dalam bab ini dijelaskan mengenai perancangan sensor seismik ECN menggunakan NodeMCU, GPS, Sensor IMU, serta sistem daya dan skema instalasi sensor.

### **BAB IV PENGUJIAN**

Bab ini menjelaskan hasil pengujian lama proses setiap fungsi pada algoritma, error pembacaan data lokasi sensor, kalibrasi dan penyesuaian hasil pembacaan percepatan getaran, dan tahan lama sistem daya yang didesain.

### **BAB V KESIMPULAN DAN SARAN**

Pada bab ini dijabarkan mengenai kesimpulan yang didapatkan pada tugas akhir ini, dan saran untuk pengembangan sistem selanjutnya.

## BAB II

### TINJAUAN PUSTAKA

#### II.1 NodeMCU 1.0 (Modul ESP-12E )

NodeMCU adalah *firmware* berbasis eLua untuk modul Espressif ESP8266 *Wi-Fi SOC*. *Firmware* NodeMCU adalah proyek pendamping untuk *development kit* NodeMCU, *development kit* yang *open source* dan dipakai dengan *chip* ESP8266-12E. Spesifikasi dari NodeMCU adalah sebagai berikut.

- Termasuk *plug & play* USB-TTL
- 10 GPIO
- Modul *Wi-Fi* yang tersertifikasi FCC dengan antena pada PCB
- *Wi-Fi* 2.4 GHz 802.11 b/g/n, didukung WPA/WPA2
- Tegangan Operasi 3.0~3.6 Volt
- Arus saat beroperasi rata-rata 80mA

#### II.2 MPU9255

MPU-9255 adalah modul yang terdiri dari dua *dies* yang diintegrasikan dalam single QFN. Satu *die* menempatkan 3-Axis *gyroscope* dan 3-Axis *accelerometer*, sedangkan satu *die* lainnya menempatkan AK8963 3-Axis Magnetometer dari Asahi Kasei Microdevices Corporation. MPU-9255 menyediakan tiga 16-bit *analog-to-digital converter* untuk menkuantisasi output *gyroscope*, tiga 16-bit *analog-to-digital converter* untuk menkuantisasi output *accelerometer*, tiga 16-bit *analog-to-digital converter* untuk mengkuantisasi output magnetometer. Untuk mendeteksi gerakan yang cepat dan pelan secara presisi, terdapat beberapa *full-scale range*, yaitu  $\pm 250$ ,  $\pm 500$ ,  $\pm 1000$ , and  $\pm 2000^\circ/\text{sec}$  (dps) untuk *gyroscope*,  $\pm 2\text{g}$ ,  $\pm 4\text{g}$ ,  $\pm 8\text{g}$ , and  $\pm 16\text{g}$  untuk *accelerometer*, dan  $\pm 4800\mu\text{T}$  pada magnetometer.

### II.3 GPS Ublox NEO-6M

Seri Modul Neo-6 adalah *family* dari *receiver* GPS yang dilengkapi u-blox 6 *positioning engine* yang berperforma tinggi. *Positioning Engine* yang memiliki 50-channel menawarkan *Time-To-First-Fix* kurang dari satu detik. Mesin akuisisi khusus dengan 2 juta korelator mampu melakukan pencarian frekuensi/waktu sejajar secara paralel memungkinkan menemukan satelit secara langsung. Desain dan teknologi yang inovatif menekan sumber *jamming* dan mengurangi efek *multipath*, memberikan navigasi GPS NEO-6 kinerja navigasi yang baik bahkan di lingkungan yang tidak biasa. Modul GPS seri 6 menggunakan protokol NMEA dalam berkomunikasi.

### II.4 MQTT

MQTT adalah protokol *client server publish/subscribe messaging transport*. Protokol MQTT merupakan protokol yang ringan, *open source*, sederhana dan didesain agar mudah untuk diimplementasikan. Karakteristik ini membuat protokol ini ideal untuk digunakan pada berbagai situasi, termasuk keadaan terbatas seperti *Machine to Machine (M2M)* dan *Internet Of Things (IoT)* dimana *small-code-footprint* dibutuhkan dan *bandwidth* jaringan terbatas.

Protokol berjalan pada protokol TCP/IP atau melalui protokol jaringan lain yang menyediakan koneksi yang *ordered*, *lossless*, dan *bidirectional*. Fitur-fitur protokol MQTT adalah sebagai berikut.

- Penggunaan pola pesan *publish/subscribe* sehingga disediakan fitur pengiriman pesan dengan distribusi *one-to-many*
- Pengiriman pesan yang bersifat agnostik terhadap konten dari payload
- Terdapat 3 jenis quality of service dalam pengiriman message:
  1. “*At most once*”, dimana message yang dikirim berdasarkan *best effort* dari keadaan operasi. Kehilangan *message* dapat terjadi.
  2. “*At least once*”, dimana *message* dapat dipastikan terkirim sampai ke server namun duplikat *message* pada server mungkin terjadi.



3. “*Exactly once*”, dimana pesan dijamin akan sampai terkirim ke server tepat satu kali.
- Pengiriman *small overhead* dan pertukaran protokol diminimalisasi untuk mengurangi *network traffic*.
  - Terdapat mekanisme untuk memberi tahu pihak yang berkepentingan saat terjadi pemutusan koneksi yang abnormal

## II.5 Baterai Li-ion

Baterai Li-ion adalah salah satu baterai yang dapat diisi ulang (*rechargeable battery*). Pada baterai Li-ion, ion litium bergerak dari elektroda negatif ke elektroda positif saat dilepaskan, dan kembali saat diisi ulang. Baterai Li-ion memakai senyawa litium interkalasi sebagai bahan elektrodanya, berbeda dengan litium metalik yang dipakai pada baterai li-ion yang tidak dapat diisi ulang. Baterai Li-ion merupakan baterai isi ulang yang lebih awet jika dibandingkan dengan baterai isi ulang lainnya seperti NiMH dan NiCd karena baterai Li-ion merupakan baterai dengan tingkat self discharge yang paling rendah jika dibandingkan dengan baterai rechargeable lainnya.

Tabel II-1 Perbandingan Tingkat Self Discharge Baterai NiMH, NiCD, dan Li-ion

CELL TYPE	NI-MH	NI-CD	LI-ION
SELF-DISCHARGE @ 20°C (%/MONTH)	20-30	15-20	5-10

## II.6 Charging Module TP4056

TP4056 merupakan *linear charger* dengan arus dan tegangan konstan untuk satu sel baterai Li-ion. Tidak dibutuhkan *blocking diode* karena terdapat arsitektur PMOSFET internal yang mencegah mengalirnya arus charging negatif. Tegangan *charging* konstan pada 4.2 V dan arus *charging* dapat diatur dengan menggunakan resistor eksternal. TP4056 akan memutuskan siklus charging setelah arus charging turun ke 10% dari nilai setelah *float voltage* tercapai.

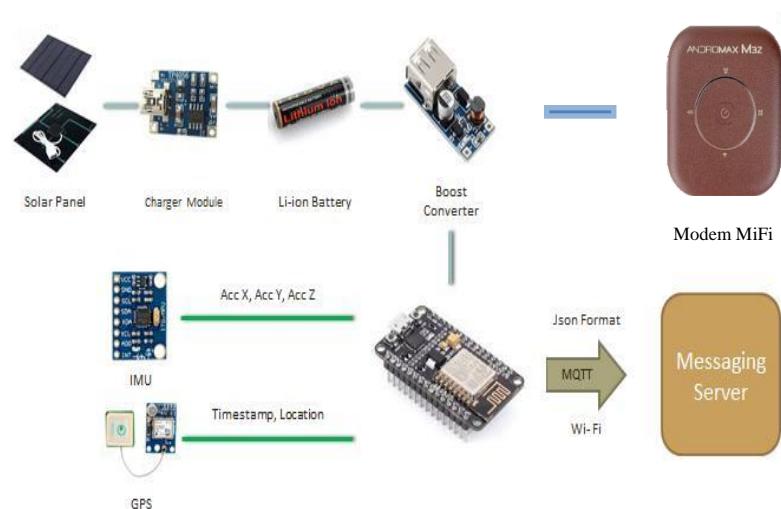
## BAB III

### PERANCANGAN

Sensor Seismik ECN yang didesain harus memenuhi beberapa spesifikasi sebagai berikut.

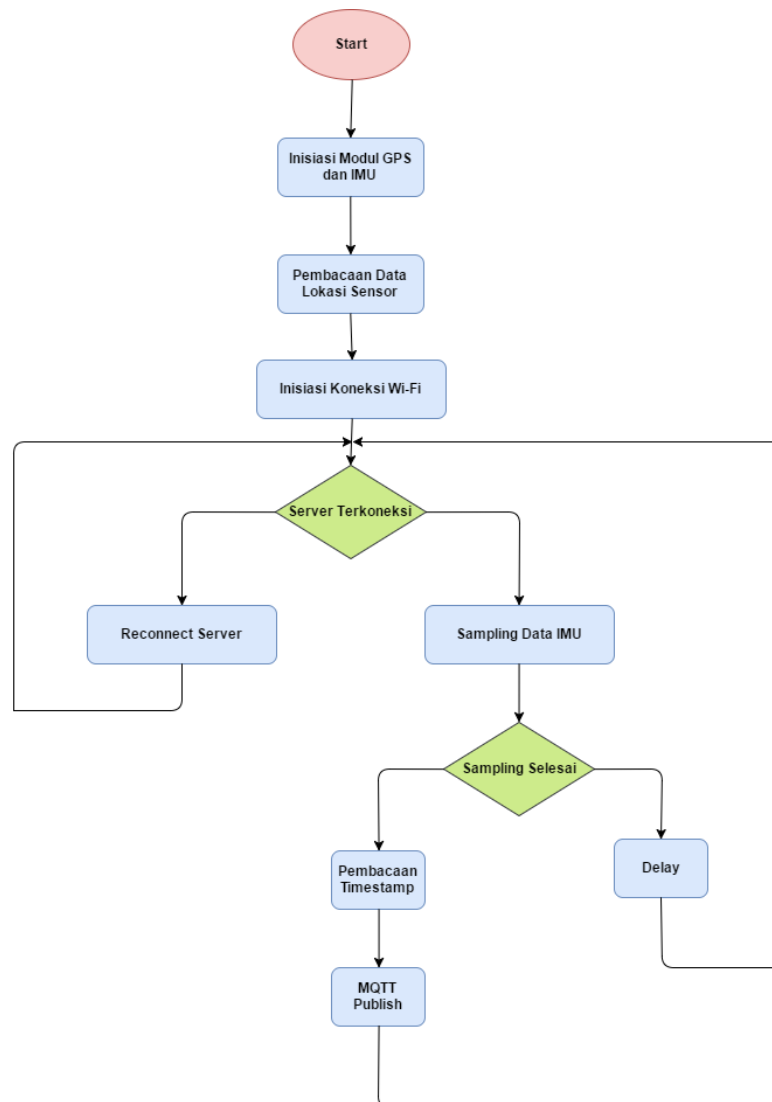
- Dapat mengirimkan data secara *real time* (frekuensi *sampling* pembacaan data 40 Hz)
- Sensor harus dapat bekerja selama 24 jam pada setiap harinya
- Harga sensor seismik lebih murah daripada harga seismometer yaitu 1/100 dari harga seismometer pada umumnya atau seharga 3 juta Rupiah
- Sensor dapat mendeteksi gempa dengan *magnitude* diatas 4 pada radius 100 km

Sensor Seismik ECN terdiri dari sistem daya yang berfungsi untuk membuat sensor disuplai selama 24 jam pada setiap harinya dan terdiri dari *Solar Panel*, *Charger Module*, Baterai Li-ion, dan *Boost Converter*, sensor IMU yang berfungsi untuk membaca data percepatan getaran tanah/data seismik dan melakukan pengaturan orientasi dengan menggunakan magnetometer, dan modul GPS yang digunakan untuk membaca timestamp dan data lokasi sensor. Digunakan Modem MiFi sebagai penyedia koneksi Wi-Fi kepada sensor seismik ECN Diagram blok dari sensor ECN dapat dilihat pada gambar berikut.



Gambar III.1 Blok Diagram Sensor Seismik ECN

### III.1 Algoritma Sensor



Gambar III.2 Flowchart umum sensor seismik ECN

Gambar diatas merupakan algoritma khusus dari Sensor Seismik ECN. Pada awal mula program dilakukan inisiasi untuk membuat modul GPS, IMU, dan koneksi Wi-Fi dapat digunakan. Dilakukan juga pembacaan *latitude* dan *longitude* sehingga pembacaan data lokasi sensor hanya dilakukan pada awal program saja. Pada bagian loop() sensor akan melakukan pemeriksaan apakah sensor terhubung ke server, jika tidak sensor akan melakukan percobaan koneksi ke server ulang sampai sensor terkoneksi dengan server. Ketika sudah terkoneksi dengan server, sensor akan melakukan *sampling* data percepatan dengan periode tertentu sesuai dengan

frekuensi *sampling*-nya. Ketika telah diperoleh jumlah *sampling* yang diinginkan maka akan dilakukan proses *publish message* ke *messaging server* dengan melalui protokol MQTT. Tetapi sebelum dilakukan *publish*, dilakukan terlebih dahulu pembacaan *Timestamp* dari GPS. Setelah proses *publish* selesai dilakukan, sensor akan mengulangi proses yang sudah dijelaskan dari awal kembali. Jika saat proses *sampling* dilakukan, koneksi server dengan sensor terputus, maka sensor akan menghapus data *sampling* yang sudah dikumpulkan dan akan mengulangi proses *sampling* dari awal kembali dan tidak ada data *sampling* yang hilang ditengah-tengah saat data tersebut dikirimkan ke server sehingga terputusnya koneksi ke server tidak mengurangi validitas data hasil *sampling*.

### III.1.1 Algoritma Pembacaan Sensor IMU MPU9255

Proses pembacaan IMU dilakukan dengan beberapa tahap sebagai berikut.

1. Inisialisasi sensor IMU pada bagian *setup()*, dilakukan beberapa konfigurasi sensor IMU, yaitu pengesetan *range* pembacaan *accelerometer* menjadi  $\pm 2g$  dan pengesetan frekuensi *sampling* pada modul *digital low pass filter* MPU9255 pada frekuensi 20 Hz
2. Pembacaan data *Accelerometer* dilakukan dari *register* 0x3B sampai 6 *register* setelahnya pada *device address* 0x68
3. Pembacaan data *Magnetometer* dilakukan dari *register* 0x03 sampai 6 *register* setelahnya pada *device address* 0x0C
4. Dilakukan pengaturan orientasi sensor yang berdasarkan arah mata angin utara

Pada implementasi algoritma pembacaan sensor IMU MPU9255 dimanfaatkan library *Wire.h* sebagai *library* yang dapat mengimplementasikan komunikasi I<sup>2</sup>C sensor IMU. *Range* pembacaan diatur menjadi  $\pm 2g$  dikarenakan dengan *range* tersebut dapat dideteksi gempa dengan *magnitude* diatas 4 dalam radius 100 km berdasarkan <sup>[9]</sup>. Sehingga resolusi pembacaan *accelerometer* dan *magnetometer* menjadi sebagai berikut.

$$ACC_{RES} = \frac{2}{2^{16-1} - 1} = 6.10388817677e - 05g$$

$$MAG_{RES} = \frac{4800}{2^{16-1} - 1} = 0.149540696432 \mu T$$

Sehingga dapat diperoleh data *accelerometer* dan magnetometer dalam g dan  $\mu T$  dengan persamaan sebagai berikut.

$$acc[x, y, z] = ACC_{RES} * (acc_{highbyte} \ll 8 | acc_{lowbyte})$$

$$mag[x, y, z] = MAG_{RES} * (mag_{highbyte} \ll 8 | mag_{lowbyte})$$

Diaktifkan modul *digital low pass filter* MPU 9255 pada frekuensi 20 Hz disesuaikan dengan frekuensi maksimum gelombang seismik yaitu 20 Hz. Sebelum dilakukan pengaturan orientasi sensor berdasarkan arah mata angin utara, dilakukan *tilt compensation* dari magnetometer terlebih dahulu dikarenakan keterbatasan pembacaan sensor magnetometer ketika sumbu z sensor tidak sejajar dengan arah gravitasi. *Tilt compensation* dilakukan dengan mengimplementasikan persamaan berikut ini.

$$axf = \text{atan}\left(\frac{acc_x}{\sqrt{acc_y^2 + acc_z^2}}\right)$$

$$ayf = \text{atan}\left(\frac{acc_y}{\sqrt{acc_x^2 + acc_z^2}}\right)$$

$$mxh = my * \cos(ayf) + my * \sin(ayf) \sin(axf) - mz * \cos(axf) \sin(ayf)$$

$$myh = my * \cos(axf) + mz * \sin(axf)$$

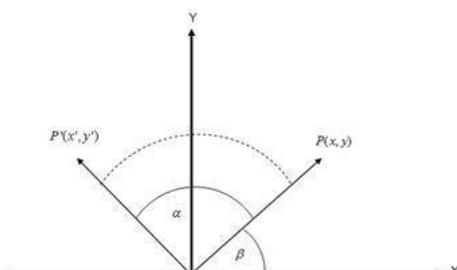
Keterangan :

- Accx merupakan pembacaan percepatan pada sumbu x
- Accy merupakan pembacaan percepatan pada sumbu y
- Accz merupakan pembacaan percepatan pada sumbu z
- Axf merupakan sudut yaw dari sensor IMU
- Ayf merupakan sudut pitch dari sensor IMU
- Mx merupakan pembacaan magnetometer pada sumbu x
- My merupakan pembacaan magnetometer pada sumbu y
- Mz merupakan pembacaan magnetometer pada sumbu z
- Mxh merupakan pembacaan magnetometer pada sumbu x yang sudah mengalami tilt compensation
- Myh merupakan pembacaan magnetometer pada sumbu y yang sudah mengalami tilt compensation

Setelah diberlakukan tilt compensation pada pembacaan magnetometer, dapat diperoleh perbedaan sudut antara sumbu x pada sensor dengan arah mata angin utara dengan persamaan berikut ini.

$$\theta_{compass} = \text{atan}\left(\frac{myh}{mxh}\right)$$

Setelah diperoleh perbedaan sudut antara sumbu x pada sensor dengan arah mata angin utara, dapat diperoleh data percepatan yang orientasinya disesuaikan dengan arah mata angin dengan menggunakan transformasi linier rotasi dengan menggunakan matriks transformasi sebagai berikut.



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

Gambar III.3 Transformasi Linear (Rotasi)

Sehingga dapat diperoleh nilai percepatan dalam sumbu x dan y yang telah disesuaikan dengan menggunakan persamaan sebagai berikut.

$$acc'_x = acc_x * \cos(\theta_{compass}) - acc_y * \sin(\theta_{compass})$$

$$acc'_y = acc_x * \sin(\theta_{compass}) + acc_y * \cos(\theta_{compass})$$

Keterangan

- $Acc'_x$  merupakan data pembacaan accelerometer pada sumbu x yang telah disesuaikan
- $Acc'_y$  merupakan data pembacaan accelerometer pada sumbu y yang telah disesuaikan

Untuk implementasi algoritma dapat dilihat pada *source code* fungsi MPU9255\_Init() dan Acc\_Read() pada lampiran.

### III.1.2 Algoritma Pembacaan GPS

Untuk mengimplementasikan algoritma untuk pembacaan data dari GPS digunakan dua buah *library* yaitu SoftwareSerial.h dan TinyGPS++.h . *Library software serial* digunakan untuk *porting* pin GPIO agar dapat digunakan sebagai pin untuk komunikasi serial. Dibutuhkan *porting* pin GPIO lain sebagai pin komunikasi serial dimaksudkan untuk memudahkan *troubleshooting* dengan pin serial bawaan NodeMCU. Sedangkan *library* TinyGPS++ digunakan untuk membaca data GPS dengan format NMEA melalui pin komunikasi serial yang sudah dideklarasikan. Untuk implementasi algoritma pembacaan *latitude* dan *longitude* dari GPS dapat dilihat pada potongan *source code* bagian setup() berikut ini.

```
ss.begin(GPSBaud);

while (!checkgps)

{

    while (ss.available() > 0)
    {
        if (gps.encode(ss.read()))
        {
            displayInfo();
            checkgps = true;
        }
    }
}
```

```

    }
}
if (millis() > WaitGPS*1000 && gps.charsProcessed() < 10)
{
    payload_data.coordinates[0] = String(init_lat);
    payload_data.coordinates[1] = String(init_lon);
    checkgps = true;
}
}

```

Dapat dilihat dari potongan source code diatas, bahwa pembacaan *latitude* dan *longitude* pada Sensor Seismik ECN dibatasi dalam jangka waktu tertentu yang ditentukan konstanta WaitGPS yang diatur nilainya menjadi 10 detik. Sedangkan implementasi algoritma pembacaan *timestamp* dari GPS dapat dilihat pada potongan *source code* bagian loop() berikut ini.

```

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";

```

Hasil pembacaan *timestamp* modul GPS dibuat menjadi format *timestamp* pada Javascript. Digunakannya format ini dimaksudkan untuk mempermudah pemrosesan data *timestamp* di server.

### III.1.3 Algoritma Format Json

Digunakan format Json pada *message* yang akan dikirim ke server untuk mempermudah pemrosesan data atau *parsing* data pada server. Format Json yang digunakan dengan terdapat 1 data *sampling* percepatan dapat dilihat sebagai berikut.



```

{
  "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
    }
  ]
}

```

Untuk mengimplementasikan format ini pada algoritma sensor, dibuat 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 digunakan 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. Implementasi fungsi `JsonToChar` dapat dilihat pada potongan source code berikut ini.

```

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 + "];

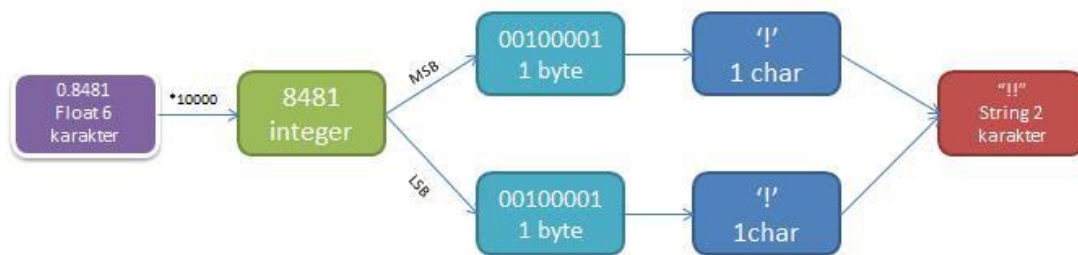
a = a + "}";

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

```

#### III.1.4 Algoritma Representasi Char

Dikarenakan *message* yang dikirimkan ke server pada protokol MQTT memiliki panjang karakter yang panjang (terdapat 40 data float \* 3 sumbu pada *sampling* selama 1 detik dan string dalam format Json lainnya ), diperlukan suatu cara yang digunakan untuk mengurangi panjang *message* yang dikirimkan ke server. Dengan mengurangi panjang *message*, lama proses pengiriman data ke server pun akan semakin berkurang. Untuk mengurangi panjang *message*, digunakan representasi karakter pada kumpulan data percepatan yang bertipe float. Proses representasinya dapat dilihat pada skema berikut ini.



Gambar III.4 Skema Algoritma Representasi Char

Variabel float dengan empat angka dibelakang koma yang jika dijadikan string akan memiliki 6 karakter, dikalikan dengan 10000 dan di-*casting* menjadi variabel integer. Tetapi sebelum diikalikan dengan 10000, dipastikan dulu bahwa *range* nilai float berada diantara -2.000 ~ 2.000. Setelah dijadikan variabel integer, 8 bit MSB dan 8 bit LSB dipisahkan menjadi dua buah variabel berukuran 1 byte yang dapat di-*casting* menjadi variabel char. Dua buah variabel char tersebut digabungkan dan menghasilkan representasi char yang memiliki 2 buah karakter. Dapat dilihat bahwa dengan representasi char ini, dapat mereduksi panjang string yang akan dikirimkan ke server, dari yang memiliki panjang 6 karakter pada satu buah data *sampling* percepatan getaran dapat diubah menjadi hanya 2 karakter atau dapat terbilang mengurangi sebesar 66% dari panjang data sebelumnya. Dengan representasi char ini lama proses pengiriman data dapat diminimalisir. Implementasi representasi char ini dapat dilihat pada fungsi berikut ini.

### III.1.5 Algoritma Publish MQTT

Untuk mengimplementasikan protokol MQTT dalam pengiriman message, digunakan *library* `pubsubclient.h`. Untuk melakukan publish message, hanya diperlukan fungsi *publish* sebagai berikut.

```
String message = JsonToString(payload_data,payload_setting);
char message_t[MQTT_MAX_PACKET_SIZE];
message.toCharArray(message_t, MQTT_MAX_PACKET_SIZE);
bool test = client.publish(server topic, message_t);
```

Sebelum melakukan *publish* diperlukan untuk mendeklarasikan beberapa variabel untuk melakukan pengaturan server, *username*, dan *password* dari *messaging server*.

Implementasi algoritma deklarasi variabel dapat dilihat pada potongan source code berikut ini.

```
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);
```

Dan untuk melakukan inisiasi koneksi ke server dapat menggunakan fungsi `setServer`. Implementasi source code dapat dilihat sebagai berikut.

```
client.setServer(mqtt_server, 1883);
```

### III.2 Sistem Daya

Untuk menyuplai sensor yang harus bekerja selama 24 jam terus-menerus, diperlukan sistem daya. Dalam implementasi digunakan *solar panel* yang akan menyuplai sensor pada siang hari. Karena *solar panel* tidak dapat menyuplai daya pada malam hari maka diperlukan rangkaian *charging* baterai, agar ketika malam sistem daya ini masih dapat menyuplai sensor seismik ECN. Sehingga digunakan modul *charge controller* baterai Li-ion yang berbasis chip TP4056. Modul TP4056 ini dapat mengatur tingkat *charging* baterai Li-ion karena untuk mengisi baterai Li-ion dibutuhkan tegangan konstan pada 4.2 Volt dan arus yang berubah sesuai dengan kapasitas baterai yang sudah terisi. Baterai Li-ion dipilih karena sifatnya yang tidak mudah rusak jika dilakukan proses *charge-discharge* dibandingkan dengan baterai lain.

Untuk menentukan berapa daya solar panel dan kapasitas baterai yang dibutuhkan, dilakukan beberapa asumsi dalam perhitungan. Asumsi pertama adalah solar panel tidak dapat menyuplai daya selama 12 jam sehari yaitu dari jam 6 sore sampai jam 6 pagi. Asumsi kedua adalah solar panel hanya dapat menyuplai sensor selama 7 jam jika cuaca cerah. Asumsi ketiga adalah sensor membutuhkan arus sebesar 1A agar dapat bekerja. Dengan diketahui tegangan kerja sensor dan daya maksimum yang dibutuhkan NodeMCU maka dapat diperoleh spesifikasi daya yang dapat disuplai solar panel dan kapasitas baterai yang dibutuhkan.

Pada malam hari :

$$V_{solar} = V_{battery} = V_{sensor} = 5 V$$

$$I_{battery} = I_{sensor} = 1 A \text{ selama } 12 \text{ jam}$$

$$C_{battery} = 1 A * 12h = 12000 mAh$$

Pada siang hari :

$$I_{solar} = I_{battery} + I_{sensor}$$

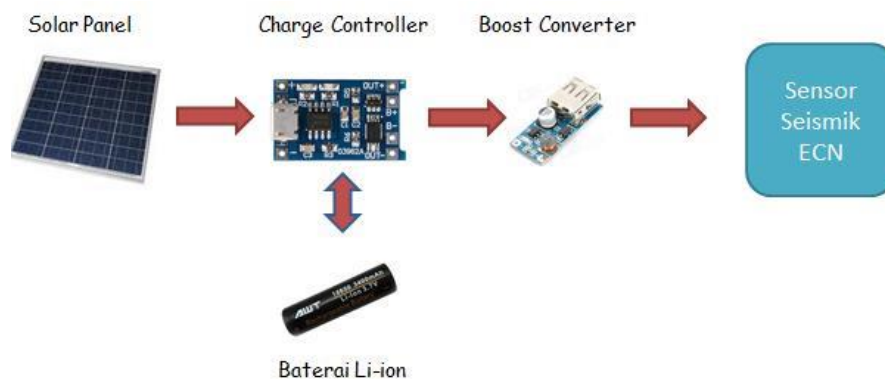
$$C_{battery} = 7 * I_{battery} = 12000 mAh$$

$$I_{battery} = 1.7 A, I_{solar} = 2.7 A$$

$$P_{solar} = 13.5 W$$

Diperoleh bahwa dibutuhkan baterai dengan kapasitas 12000 mAh dan *solar panel* yang memiliki suplai daya sebesar 13.5 Wp. Perhatikan bahwa perhitungan ini tidak memperhatikan fakta bahwa *solar panel* tidak akan menyuplai daya sebesar 13.5 W secara konstan dan *loss* daya pada rangkaian *charge controller*. Sehingga hitungan ini hanya sebagai dasar penentuan sistem daya yang tepat untuk sensor dan diperlukan percobaan lagi untuk mengetahui berapa daya solar panel yang dibutuhkan dan kapasitas baterai yang dibutuhkan.

Untuk Rangkaian Sistem Daya dapat dilihat pada gambar berikut.

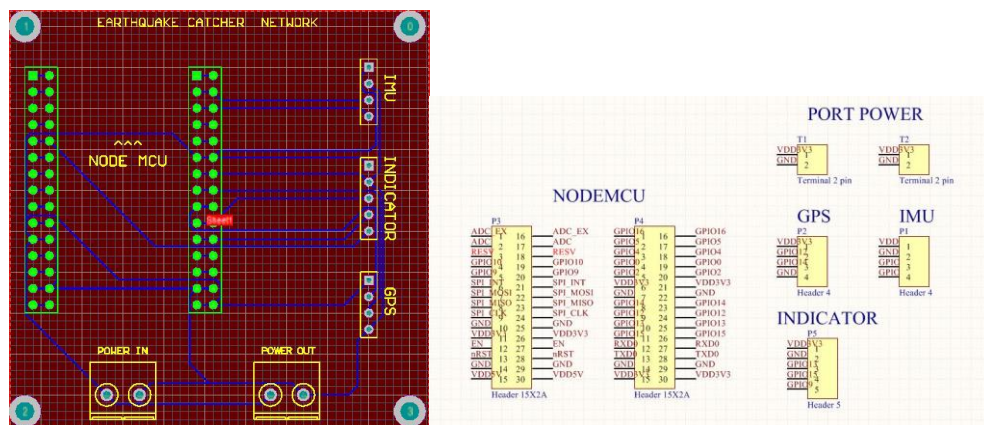


Gambar III.5 Rangkaian Sistem Daya Sensor Seismik ECN

- *Solar Panel* 5 Volt 4 W, digunakan sebagai sumber daya yang dapat mengisi baterai saat siang hari.
- *Charge Controller* TP4056, digunakan untuk mengatur tingkat charging baterai Li-ion. Hanya dapat menerima arus maksimum sebesar 1A dari sumber daya.
- Baterai Li-ion 3.7 Volt 3400 mAh, digunakan untuk menyimpan daya yang diberikan *solar panel* pada siang hari dan menyuplai daya ke sensor seismik ECN pada malam hari
- 0.9~5 Volt *Boost converter*, digunakan untuk menstabilkan tegangan yang diberikan ke sensor seismik ECN

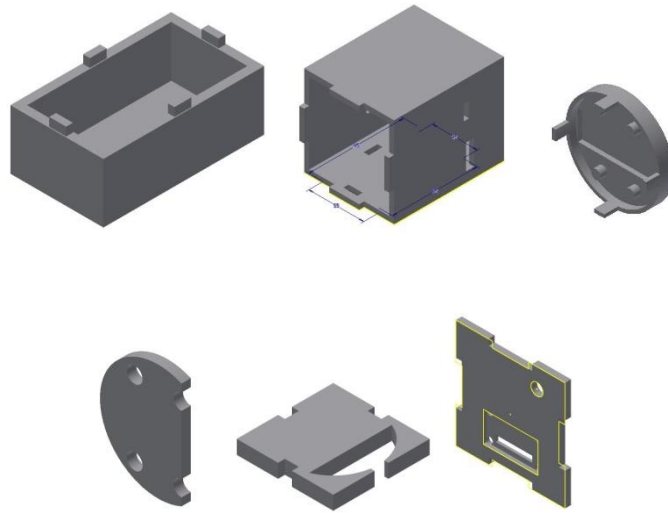
### III.3 Skema Instalasi Sensor

Agar sensor dapat bekerja dengan baik, tidak terganggu dengan lingkungan sekitar dan tahan lama, diperlukan skema instalasi sensor seismik. Oleh karena itu didesain *PCB Board* agar komponen mudah diletakkan dan tidak perlu melakukan wiring saat instalasi sensor dilakukan. Berikut adalah desain *board PCB* yang digunakan.



Gambar III.6 Desain PCB

Dapat dilihat bahwa koneksi dari mikrokontroler NodeMCU ke GPS, IMU, dan modul Indikator digunakan *header* untuk memudah pemasangan dan instalasi. Ukuran board PCB yang didesain adalah 6.5 cm x 6.5 cm menyesuaikan dengan *casing* yang didesain. Untuk koneksi PCB ini ke sistem daya digunakan *terminal* dengan dua *port* sehingga lebih mudah dihubungkan antar sistem.



**Gambar III.7 Desain Casing Sensor Seismik ECN**

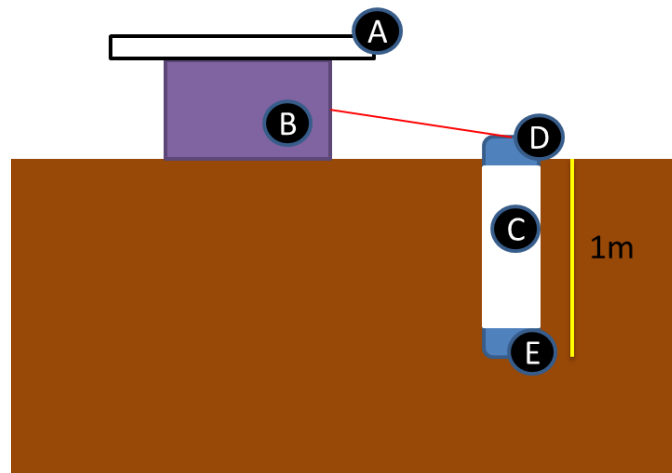
Keterangan:

Casing 1 (Kiri Atas), Casing 2 (Tengah Atas), Casing 3 (Kanan Atas), Casing 4 (Kiri Bawah), Casing 5 (Tengah Bawah), Casing 6 (Kanan Bawah)

Sedangkan dapat dilihat pada gambar, desain model 3D yang digunakan sebagai meletakkan semua komponen sensor seismik ECN. Desain pada gambar pertama berfungsi untuk tempat meletakkan baterai sehingga baterai mudah diganti dan dapat ditempel pada *casing* utama pada gambar kedua yang berfungsi sebagai tempat meletakkan PCB dan sistem daya. Pada *casing* utama tersebut, terdapat lubang untuk meletakkan *casing* baterai dan lubang sebagai tempat meletakkan GPS dan sebagai tempat kabel IMU yang akan digunakan pada casing IMU pada gambar 3,4 dan 5. Gambar 3 merupakan tempat untuk meletakkan modul MPU9255 yang dapat ditutup oleh *casing* pada gambar 4 dan 5. Desain 3D pada gambar 6 berfungsi untuk menutup casing utama pada gambar 2 yang sudah dijelaskan sehingga kabel untuk *solar panel* dapat masuk ke modul sistem mikrokontroler dan *casing* utama. *Casing* pada gambar 6 juga berfungsi untuk meletakkan modul indikator.



Sehingga untuk skema instalasi sensor seismik ECN dapat dilihat pada skema dibawah ini.



Gambar III.8 Gambar Skema Instalasi

#### Keterangan

- A. Solar Panel
- B. Sensor ECN Utama yang terdiri dari *Board PCB*, *Casing 1,2 & 6*
- C. Pipa PVC dengan diameter 3" sepanjang 1m.
- D. Penutup pipa PVC dengan menggunakan *casing 5*
- E. Tempat sensor IMU berada, menggunakan *casing 3 & 4* dan sensor IMU dihubungkan dengan sensor ECN utama dengan menggunakan kabel

## BAB IV

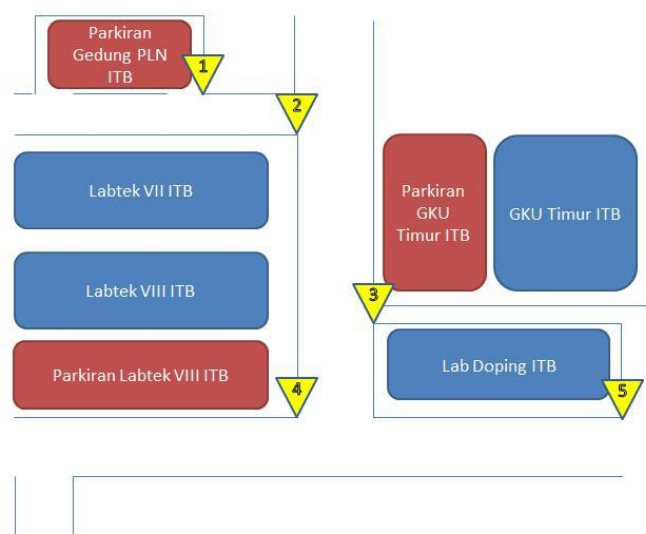
### IMPLEMENTASI DAN PENGUJIAN

#### IV.1 Prosedur Pengujian

Dilakukan 4 pengujian terhadap sensor seismik ECN yang didesain, yaitu pengujian ketelitian pembacaan *latitude* dan *longitude* pada modul GPS, kalibrasi pembacaan data percepatan gelombang seismik pada sensor IMU MPU9255, pengujian lama proses beberapa algoritma pada beberapa proses fungsional (IMU, GPS, Publish MQTT), dan Sistem Daya.

##### IV.1.1 Pengujian Ketelitian Pembacaan Latitude dan Longitude pada modul 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 IV.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*.

#### **IV.1.2 Kalibrasi Pembacaan Accelerometer MPU9255**

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 IV.2 Sensor Seismik Divisi Geoteknologi LIPI**

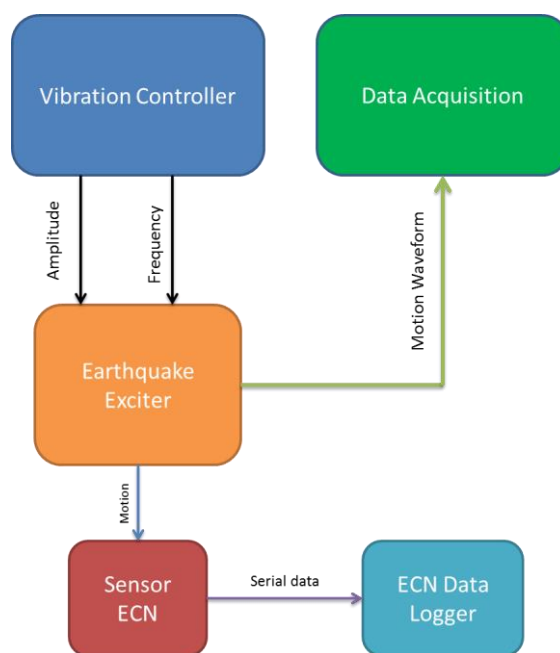
Tetapi dikarenakan tidak sesuainya data yang dicatat oleh sensor seismik LIPI dengan pembacaan sensor ECN, serta tidak diizinkannya untuk melakukan goyangan terhadap sensor karena ditakutkan rusak maka pengujian tidak dapat dilakukan. Berikut data yang cuplikan data yang dibaca oleh sensor seismik divisi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

=LIPI_10_172016 - Excel										Sheet1										File Home Insert Layout Formulas Data Review View Add-ins Tools Developer										Get info and what you want to do										Share									
File Home Insert Layout Formulas Data Review View Add-ins Tools Developer										Get info and what you want to do										Share																													
File Home Insert Layout Formulas Data Review View Add-ins Tools Developer										Get info and what you want to do										Share																													
File Home Insert Layout Formulas Data Review View Add-ins Tools Developer										Get info and what you want to do										Share																													
File Home Insert Layout Formulas Data Review View Add-ins Tools Developer										Get info and what you want to do										Share																													
File Home Insert Layout Formulas Data Review View Add-ins Tools Developer										Get info and what you want to do										Share																													
File Home Insert Layout Formulas Data Review View Add-ins Tools Developer										Get info and what you want to do										Share																													
File Home Insert Layout Formulas Data Review View Add-ins Tools Developer										Get info and what you want to do										Share																													
File Home Insert Layout Formulas Data Review View Add-ins Tools Developer										Get info and what you want to do										Share																													
File Home Insert Layout Formulas Data Review View Add-ins Tools Developer										Get info and what you want to do										Share																													
File Home Insert Layout Formulas Data Review View Add-ins Tools Developer										Get info and what you want to do										Share																													
File Home Insert Layout Formulas Data Review View Add-ins Tools Developer										Get info and what you want to do										Share																													
File Home Insert Layout Formulas Data Review View Add-ins Tools Developer										Get info and what you want to do										Share																													
File Home Insert Layout Formulas Data Review View Add-ins Tools Developer										Get info and what you want to do										Share																													
File Home Insert Layout Formulas Data Review View Add-ins Tools Developer										Get info and what you want to do										Share																													
File Home Insert Layout Formulas Data Review View Add-ins Tools Developer										Get info and what you want to do										Share																													
File Home Insert Layout Formulas Data Review View Add-ins Tools Developer										Get info and what you want to do										Share																													
File Home Insert Layout Formulas Data Review View Add-ins Tools Developer										Get info and what you want to do										Share																													
File Home Insert Layout Formulas Data Review View Add-ins Tools Developer										Get info and what you want to do										Share																													
File Home Insert Layout Formulas Data Review View Add-ins Tools Developer										Get info and what you want to do										Share																													
File Home Insert Layout Formulas Data Review View Add-ins Tools Developer										Get info and what you want to do										Share																													
File Home Insert Layout Formulas Data Review View Add-ins Tools Developer										Get info and what you want to do										Share																													
File Home Insert Layout Formulas Data Review View Add-ins Tools Developer										Get info and what you want to do										Share																													
File Home Insert Layout Formulas Data Review View Add-ins Tools Developer										Get info and what you want to do										Share																													
File Home Insert Layout Formulas Data Review View Add-ins Tools Developer										Get info and what you want to do										Share																													
File Home Insert Layout Formulas Data Review View Add-ins Tools Developer										Get info and what you want to do										Share																													
File Home Insert Layout Formulas Data Review View Add-ins Tools Developer										Get info and what you want to do										Share																													
File Home Insert Layout Formulas Data Review View Add-ins Tools Developer										Get info and what you want to do										Share																													
File Home Insert Layout Formulas Data Review View Add-ins Tools Developer										Get info and what you want to do										Share																													
File Home Insert Layout Formulas Data Review View Add-ins Tools Developer										Get info and what you want to do										Share																													
File Home Insert Layout Formulas Data Review View Add-ins Tools Developer										Get info and what you want to do										Share																													
File Home Insert Layout Formulas Data Review View Add-ins Tools Developer										Get info and what you want to do										Share																													
File Home Insert Layout Formulas Data Review View Add-ins Tools Developer										Get info and what you want to do										Share																													
File Home Insert Layout Formulas Data Review View Add-ins Tools Developer										Get info and what you want to do										Share																													
File Home Insert Layout Formulas Data Review View Add-ins Tools Developer										Get info and what you want to do										Share																													
File Home Insert Layout Formulas Data Review View Add-ins Tools Developer										Get info and what you want to do										Share																													
File Home Insert Layout Formulas Data Review View Add-ins Tools Developer										Get info and what you want to do										Share																													
File Home Insert Layout Formulas Data Review View Add-ins Tools Developer										Get info and what you want to do										Share																													
File Home Insert Layout Formulas Data Review View Add-ins Tools Developer										Get info and what you want to do										Share																													
File Home Insert Layout Formulas Data Review View Add-ins Tools Developer										Get info and what you want to do										Share																													
File Home Insert Layout Formulas Data Review View Add-ins Tools Developer										Get info and what you want to do										Share																													
File Home Insert Layout Formulas Data Review View Add-ins Tools Developer										Get info and what you want to do										Share																													
File Home Insert Layout Formulas Data Review View Add-ins Tools Developer										Get info and what you want to do										Share																													
File Home Insert Layout Formulas Data Review View Add-ins Tools Developer										Get info and what you want to do										Share																													
File Home Insert Layout Formulas Data Review View Add-ins Tools Developer										Get info and what you want to do										Share																													
File Home Insert Layout Formulas Data Review View Add-ins Tools Developer										Get info and what you want to do										Share																													
File Home Insert Layout Formulas Data Review View Add-ins Tools Developer										Get info and what you want to do										Share																													
File Home Insert Layout Formulas Data Review View Add-ins Tools Developer										Get info and what you want to do										Share																													
File Home Insert Layout Formulas Data Review View Add-ins Tools Developer										Get info and what you want to do										Share																													
File Home Insert Layout Formulas Data Review View Add-ins Tools Developer										Get info and what you want to do										Share																													
File Home Insert Layout Formulas Data Review View Add-ins Tools Developer										Get info and what you want to do										Share																													
File Home Insert Layout Formulas Data Review View Add-ins Tools Developer										Get info and what you want to do										Share																													
File Home Insert Layout Formulas Data Review View Add-ins Tools Developer										Get info and what you want to do										Share																													
File Home Insert Layout Formulas Data Review View Add-ins Tools Developer										Get info and what you want to do										Share																													
File Home Insert Layout Formulas Data Review View Add-ins Tools Developer										Get info and what you want to do										Share																													
File Home Insert Layout Formulas Data Review View Add-ins Tools Developer										Get info and what you want to do										Share																													
File Home Insert Layout Formulas Data Review View Add-ins Tools Developer										Get info and what you want to do										Share																													
File Home Insert Layout Formulas Data Review View Add-ins Tools Developer										Get info and what you want to do										Share																													
File Home Insert Layout Formulas Data Review View Add-ins Tools Developer										Get info and what you want to do										Share																													
File Home Insert Layout Formulas Data Review View Add-ins Tools Developer										Get info and what you want to do										Share																													
File Home Insert Layout Formulas Data Review View Add-ins Tools Developer										Get info and what you want to do										Share																													
File Home Insert Layout Formulas Data Review View Add-ins Tools Developer										Get info and what you want to do										Share																													
File Home Insert Layout Formulas Data Review View Add-ins Tools Developer										Get info and what you want to do										Share																													
File Home Insert Layout Formulas Data Review View Add-ins Tools Developer										Get info and what you want to do										Share																													
File Home Insert Layout Formulas Data Review View Add-ins Tools Developer										Get info and what you want to do										Share																													
File Home Insert Layout Formulas Data Review View Add-ins Tools Developer										Get info and what you want to do										Share																													

Gambar IV.3 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 IV.4 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.



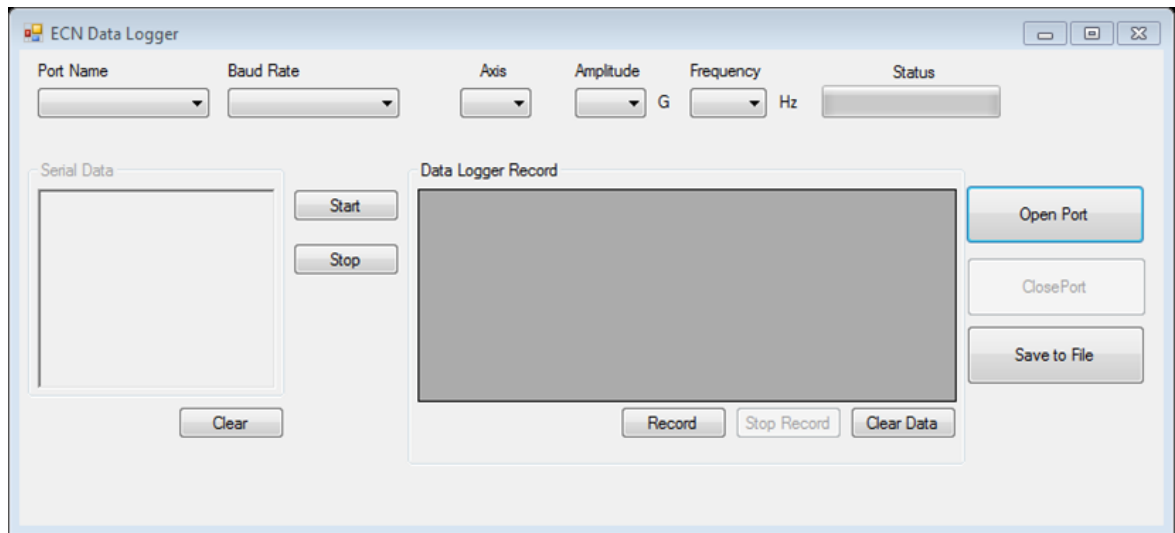
**Gambar IV.5 Earthquake Exciter, Vibration Controller, Data Acquisition**

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 IV.6 Peletakkan Sensor IMU MPU9255 untuk Pengujian Sumbu y, x, dan z**

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 IV.7 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);
}

```

#### IV.1.3 Pengujian Lama Proses Algoritma

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. Pengujian dilakukan sebanyak 60 kali proses pengiriman data ke server setiap 1 detik dengan 40 data sampling IMU.

#### IV.1.4 Pengujian Sistem Daya

Pada pengujian digunakan baterai 2 AWT yang disusun secara paralel dengan kapasitas total 6800 mAh dan digunakan solar panel dengan daya 5 watt peak. Dilakukan beberapa pengujian sistem daya sebagai berikut.

- Pengujian untuk mengetahui nilai tegangan minimum baterai saat sensor seismik ECN tidak dapat bekerja. Dapat dilakukan dengan pengamatan tegangan baterai saat sensor seismik ECN mulai berhenti bekerja.
- Pengujian untuk mengetahui nilai tegangan baterai saat berada pada *full capacity*. Pengujian dilakukan dengan pengamatan terhadap tegangan baterai pada kondisi *full capacity* saat dilakukan pengisian daya oleh *solar panel*, kondisi *full capacity* ditunjukkan dengan nyala indikator LED biru pada modul charge controller.
- Pengujian untuk mengetahui durasi waktu yang dibutuhkan untuk charging baterai dalam kondisi kosong sampai ke kondisi *full capacity*



## IV.2 Hasil Pengujian

### IV.2.1 Pengujian Ketelitian Pembacaan Latitude dan Longitude pada modul GPS

Berikut adalah perbandingan antara data GPS *Google Maps* dan modul GPS.

Tabel IV-1 Perbandingan Data Pembacaan Latitude dan Longitude pada Google Maps dan Modul GPS

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

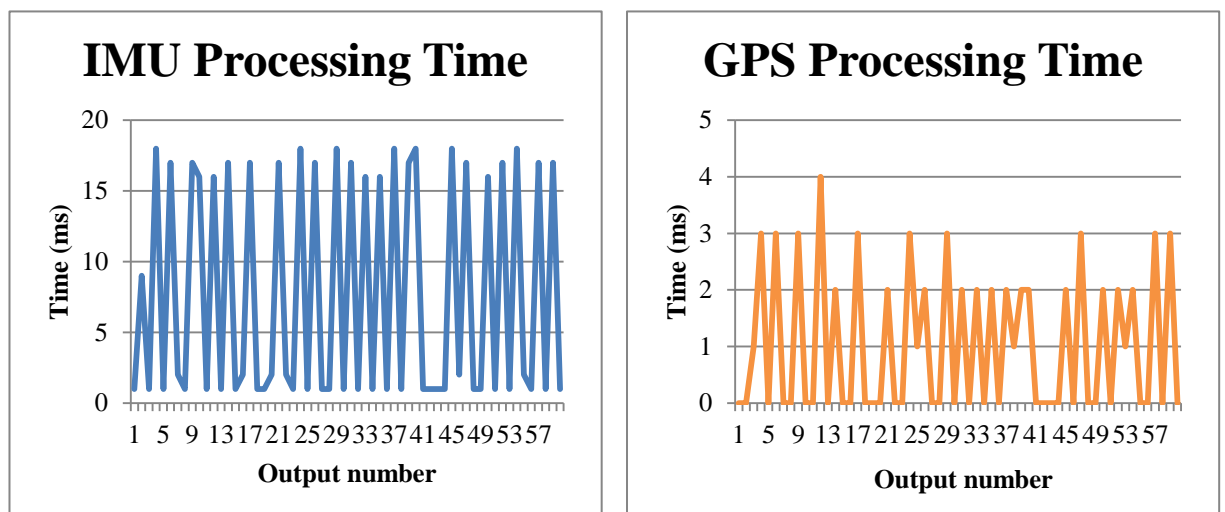
Dapat dilihat pada tabel diatas bahwa error dari pembacaan latitude pada modul GPS Neo 6M adalah 0,000049 dan error dari pembacaan longitude pada modul GPS Neo 6M adalah 0,000048. Pada jalur khatulistiwa, 1 derajat latitude sebanding dengan 110.57 km dan 1 derajat longitude sebanding dengan 111.32 km. Sehingga dapat disimpulkan bahwa error dari pembacaan modul GPS Neo 6M adalah 5,41793 m pada latitude dan 5,34336 m. Dengan error yang diketahui dan data kisaran kecepatan rambat gelombang seismik diantara 2-8 km/h, diperoleh error waktu dalam kisaran 2,385 – 9,72 s. Hasil error waktu ini dimasukkan sebagai input dari fungsi pada aplikasi SeisComp3 untuk menentukan error penentuan lokasi gempa. Diperoleh error sekitar 100m sehingga dapat disimpulkan bahwa error ini dapat dimaklumi karena data lokasi gempa biasanya disampaikan dalam satuan km. (Contoh: pusat gempa berada pada 10 km di tenggara kota Yogyakarta)

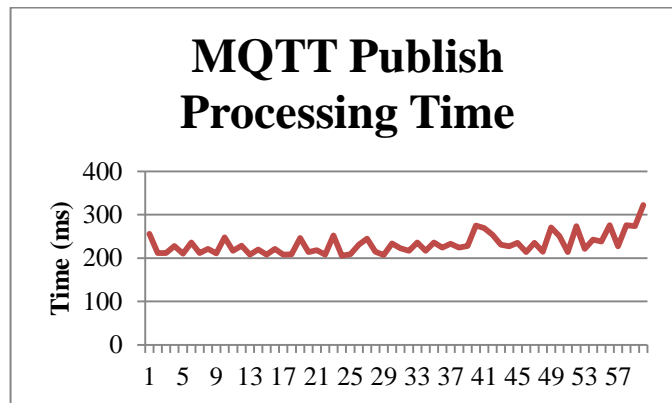
#### IV.2.2 Kalibrasi Pembacaan Accelerometer MPU9255

Hasil pembacaan Accelerometer pada MPU9255 digunakan untuk menghitung sensitifitas dari pembacaan sensor IMU MPU9255. Dari hasil pembacaan tersebut, dilakukan interpolasi gelombang sinusoidal pada data yang sudah didapatkan sehingga diperoleh sensitivitas sensor pada amplitude dan frekuensi tertentu. Data yang sudah diinterpolasi akan dibandingkan untuk amplitude 0.5G, 1.0G, dan 1.5G pada frekuensi 5 Hz, 10 Hz, dan 15 Hz untuk diketahui sensitifitasnya. Setelah dilakukan interpolasi dapat disimpulkan bahwa hubungan pembacaan pada amplitude yang berbeda dan frekuensi yang sama linear, sedangkan pembacaan pada amplitude yang sama pada frekuensi yang berbeda tidak menghasilkan perbedaan hasil pembacaan. Data sensitivitas tersebut diimplementasikan pada sistem pemrosesan dengan menggunakan SeisComp3. Contoh hasil interpolasi pembacaan accelerometer MPU9255 dapat dilihat pada lampiran.

#### IV.2.3 Pengujian Lama Proses Algoritma

Hasil pengujian waktu proses pada algoritma sensor dengan kondisi tanpa menunggu *queue* RabbitMQ dapat dilihat pada lampiran dan grafik dibawah ini.





Gambar IV.8 Grafik Lama Proses Algoritma Sensor Seismik ECN

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. Tetapi, lama proses tersebut dapat diminimalisir dengan cara mengurangi panjang *message* yang dikirimkan ke server. Cara mengurangi panjang *message* sudah dijelaskan pada bab sebelumnya, yaitu dengan mengganti representasi float menjadi representasi char.

#### IV.2.4 Pengujian Sistem Daya

Setelah diamati, dapat diperoleh bahwa tegangan baterai saat *full capacity* adalah 3.55 volt dan tegangan baterai saat sensor tidak dapat mendapatkan suplai agar dapat bekerja adalah 2.8 volt. Dan lama waktu yang dibutuhkan sistem daya untuk *charging baterai* dari kondisi kosong sampai ke kondisi *full-capacity* adalah 2 jam yaitu dari jam 7 pagi sampai 9 pagi. Hasil pengujian tegangan baterai saat dilakukan *charging* dapat dilihat dari tabel berikut.

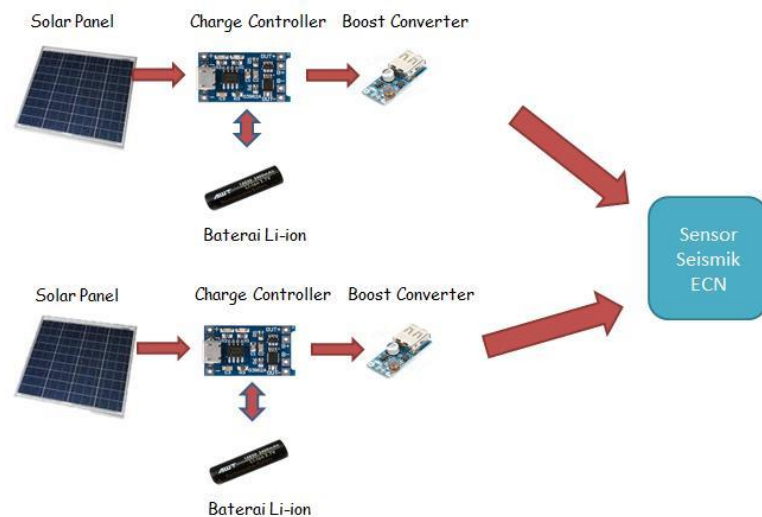
Tabel IV-2 Hasil Pengukuran Tegangan Solar Panel dan Baterai

Jam	Vsolar (V)	Vbatt (V)	Keterangan
12.00	5,27	3,495	Cerah berawan
12.31	3,92	3,559	Cerah terik
13.01	4,3	3,561	Cerah
13.32	3,92	3,551	Berawan

Dapat dilihat, tegangan solar panel saat diisi pada siang hari dan tidak hujan melebihi batas minimum modul TP4056 melakukan *charging* yaitu 2.9 volt sehingga pada proses charging baterai saat siang hari dapat dilakukan. Dikarenakan hujan, pada jam 14.00 proses pengisian baterai diberhentikan dan pengujian dilanjutkan dengan pengujian daya tahan baterai menyuplai sensor seismik ECN dari keadaan *full-capacity*. Pengujian dimulai dari pukul 14.00 dan sensor mati saat pukul 15.02 sehingga dapat disimpulkan bahwa baterai 6800 mAh dapat menyuplai sensor selama 13 jam.

Sehingga untuk memenuhi spesifikasi, dibutuhkan konfigurasi baterai dengan kapasitas sebesar dua kali lipat dari kapasitas baterai pada saat pengujian dilakukan, yaitu 13600 mAh dan konfigurasi *solar panel* yang dapat menyuplai daya sebesar

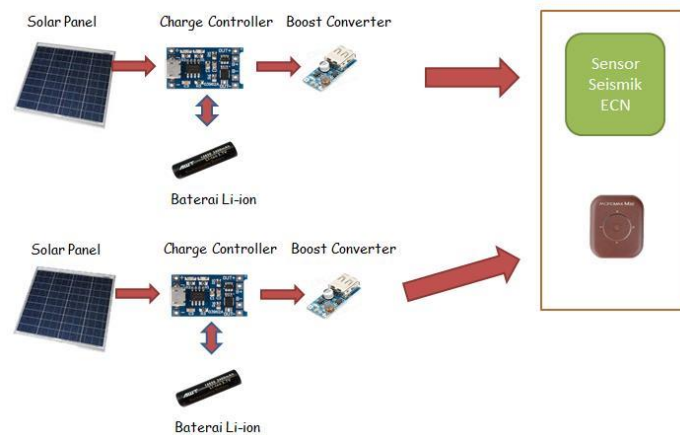
dua kali lipat dari daya yang disuplai pada saat pengujian berlangsung, yaitu sebesar 8 Wp. Desain sistem daya dapat diimplementasikan dengan cara membuat rangkaian sistem daya yang identik dengan sistem daya yang sudah digunakan dan dihubungkan outputnya pada sensor seismik ECN secara paralel. Berikut adalah sistem daya yang dapat menyuplai sensor seismik ECN selama 24 jam.



**Gambar IV.9 Rangkaian Sistem Daya yang dapat Menyuplai Sensor Seismik selama 24 Jam**

Perhatikan bahwa digunakan 2 buah solar panel yang tidak terhubung langsung ke modul *Charge Controller* dikarenakan terbatasnya arus yang dapat diterima oleh *charge controller* yaitu sebesar 1A sedangkan satu *solar panel* yang digunakan tersebut dapat menyuplai daya sebesar 0.8 A pada saat kondisi puncak. Output dari *boost converter* dihubungkan secara paralel ke sensor ECN sehingga dua sistem daya yang paralel tersebut dapat saling menyuplai daya sensor seismik ECN secara bersamaan.

Pada rangkaian diatas, beban yang disuplai oleh sistem daya hanya sensor seismik ECN saja, belum termasuk modem MiFi yang digunakan untuk menyediakan koneksi untuk sensor seismik ECN. Digunakan modem MiFi yang harus disuplai 200mA agar dapat bekerja sehingga dapat disuplai oleh sistem daya dengan desain diatas, sehingga rangkaian sensor berubah menjadi sebagai berikut.



**Gambar IV.10 Sistem Daya dengan Modem Mifi**

#### IV.2.5 Evaluasi Harga Sensor Seismik ECN

Berikut adalah daftar harga komponen yang dibutuhkan untuk membuat sensor seismik ECN.

**Tabel IV-3 Daftar Harga Komponen per Sensor Seismik ECN**

Komponen	Harga Satuan	Kuantitas	Harga Total
NodeMCU	Rp 60.000	1	Rp 60.000
MPU9255	Rp 400.000	1	Rp 400.000
GPS Ublox Neo 6M	Rp 165.000	1	Rp 165.000
PCB, Header, Terminal, Spacer, Kabel AWG	Rp 60.000	1	Rp 60.000
Solar Panel 5V 4W	Rp 210.000	2	Rp 420.000
Charging Module TP4056	Rp 10.000	2	Rp 20.000
Boost Converter	Rp 10.000	2	Rp 20.000
Baterai Li-ion AWT 3400 mAh	Rp 90.000	8	Rp 720.000
modem MiFi Andromax M3Z	Rp 340.000	1	Rp 340.000
Kelengkapan Lain	Rp 75.000	1	Rp 75.000
Total			Rp 2.280.000

Dapat dilihat pada tabel bahwa total harga yang dibutuhkan untuk membuat sensor seismik ECN adalah sebesar Rp2.280.000,00 , sehingga dapat disimpulkan sensor seismik yang didesain sudah memenuhi spesifikasi yang diajukan.

## BAB V

### KESIMPULAN DAN SARAN

#### V.1 Kesimpulan

Setelah dilakukan proses desain dan implementasi sensor seismik ECN, dapat disimpulkan beberapa hal, yaitu.

1. Algoritma dengan menggunakan library Wire, TinyGPS, SoftwareSerial, dan PubSubClient pada sensor seismik ECN dapat diimplementasikan. Tetapi karena implementasi hanya terbatas pada algoritma *single tasking*, maka terdapat waktu *sampling* yang hilang karena digunakan untuk melakukan pengiriman data.
2. Cara untuk mengurangi *missed sampling rate* dapat dilakukan dengan cara merubah representasi float menjadi representasi char.
3. Sistem daya yang dapat membuat sensor bekerja selama 24 jam adalah dengan mengimplementasikan *solar panel* dengan 8 Wp dan baterai dengan kapasitas 13600 mAh

#### V.2 Saran

Dalam implementasi selanjutnya sebaiknya dilakukan implementasi dengan menggunakan algoritma yang dapat melakukan *multi-tasking* sehingga lama proses pengiriman *message* ke server dapat diatasi dengan mengatur *scheduling* task pada program *multi tasking* dengan menggunakan RTOS. Dapat didesain 2 *task*, yaitu *task sampling* data dan *task* pengiriman data, *task* pengiriman data hanya dapat dilakukan ketika *task sampling* data sudah selesai dan akan memulai proses *sampling* data yang baru. Ketika *task sampling* data sudah memulai proses *sampling* data, *task* pengiriman dapat dilakukan bersamaan sehingga proses pengiriman data tidak mengganggu proses *sampling* data.

## DAFTAR PUSTAKA

1. Ai Thinker Team (2015). ESP8266 WROOM WiFi Module Datasheet
2. [http://nodemcu.com/index\\_en.html](http://nodemcu.com/index_en.html), diakses pada tanggal 9 Mei 2017 pada pukul 16.49
3. InvenSense (2014). MPU-9255 Product Specification DS-000007
4. U-blox (2011).U-blox 6 GPS Modules Data Sheet GPS.G6-HW-09005-E
5. Banks, Andrew & Gupta, Rahul (2014). MQTT Version 3.1.1, OASIS Standard
6. Chester Simpson (2011).Charateristics of Rechargeable Batteries SNVA533
7. Winter, M. & Brodd, J. (2004). "What Are Batteries, Fuel Cells, and Supercapacitors?"
8. NanJing Top Power ASIC Corp .TP4056 1A Standalone Linear Li-Ion Battery Charger with Thermal Regulation in SOP-8
9. Oswal, Mehul dkk. (2010). A comparative study of Lithium-Ion Batteries
10. Evans, J.R., Allen, R.M., Chung, A.I., Cochran, E.S., Guy, R., Hellweg, M., Lawrence, J.F., Performance of Several Low-Cost Accelerometers, Seism. Res. Letts., 85, 147-158, 2014.



# LAMPIRAN

## Source code Sensor Seismik ECN

```
/*  
    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>  
#include <ArduinoJson.h>  
  
#define SDA_PIN D1  
#define SCL_PIN D2  
#define PWR_MGMT 0x6B  
#define RXPIN D3  
#define TXPIN D4
```

```

#define GPSBaud 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;

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

//const char* ssid = "L4BD4S4R-TU";    // network SSID (name)
//const char* ssid = "TU Sostek";    // network SSID (name)
//const char* ssid = "marb";    // network SSID (name)
//const char* pass = "kumahaaing";    // network password
//const char* pass = "gatauudahdiganti";    // network password
//const char* pass = "123456!a";    // network password
const char* ssid = "LSKK Basement";    // network SSID (name)
const char* pass = "noiznocon";    // network password
//const char* ssid = "HME ITB";    // network SSID (name)
//const char* pass = "hmehattrick";    // network password
//const char* ssid = "Muntilan-41";    // network SSID (name)
//const char* pass = "12345678";    // network password
//const char* mqtt_server = "black-boar.rmcloudamqp.com"; //MQTT
server
//const char* mqtt_server = "127.0.0.1"; //MQTT server
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 = "lsowqccg:lsowqccg";
//String mqtt_user = "guest";

```

```

String mqtt_user = "/disaster:sensor_gempa";
//String mqtt_password = "kbLv9YbzjQwxz20NH7Rfy98TTV2eK17j";
//String mqtt_password = "guest";
String mqtt_password = "12345";
int status = WL_IDLE_STATUS;
WiFiClient espClient;
PubSubClient client(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
#define ACC_FILTER_20HZ 0x0C

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 mxt = 0.0;

```

```

float myt = 0.0;
float mzt = 0.0;

float mxtf = 0.0;
float mytf = 0.0;
float mztf = 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);

//=====//

```

```

//=====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;

//=====//
//=====MAIN ALGORITHM=====//
//=====//

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*1000 && gps.charsProcessed() < 10)
    {
        payload_data.coordinates[0] = String(init_lat);
        payload_data.coordinates[1] = String(init_lon);
        checkgps = true;
    }
}

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);
}

}

//=====//
//=====ENCODE JSON FUNCTION=====//
//=====//

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

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;
}

//=====//
//=====Wi-Fi Connection & MQTT Function Procedure=====//
//=====//
void WiFiConnect()
{
    // We start by connecting to a WiFi network
    WiFi.begin(ssid, pass);
    while (WiFi.status() != WL_CONNECTED)
    {
        delay(500);
    }
    randomSeed(micros());
}

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()

void callback(char* topic, byte* payload, unsigned int length)
{

}

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

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, ACC_FILTER_20HZ);

    // Configure accelerometers range
    I2CwriteByte(MPU9250_ADDRESS, 28, ACC_FULL_SCALE_2_G);
}

```

```

// Set by pass mode for the magnetometers
I2CwriteByte(MPU9250_ADDRESS,0x37,0x02);

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

}

struct MPU9255 Acc_Read()
{
    struct MPU9255 data;

    // _____
    // ::: accelerometer and gyroscope :::

    // Read accelerometer and gyroscope
    uint8_t Buf[14];
    I2Cread(MPU9250_ADDRESS,0x3B,8,Buf);

    // Create 16 bits values from 8 bits data

    // Accelerometer
    int16_t ax=Buf[0]<<8 | Buf[1];
    int16_t ay=Buf[2]<<8 | Buf[3];
    int16_t az=Buf[4]<<8 | Buf[5];

    // Temperatur
    int16_t temperatur=Buf[6]<<8 | Buf[7];

    //Acc in G
    axg = ax * ACC_RES;
    ayg = ay * ACC_RES;
    azg = az * ACC_RES;

    axgf = axg;//Filter(axg,axgf,30);
    aygf = ayg;//Filter(ayg,aygf,30);
    azgf = azg;//Filter(azg,azgf,30);

```

```

// Temp in Grad

temperaturG = temperatur * 1;
temperaturG = temperaturG / 100;

// _____
// ::: Magnetometer :::

// Read register Status 1 and wait for the DRDY: Data Ready

uint8_t ST1;
do
{
    I2Cread(MAG_ADDRESS, 0x02, 1, &ST1);
}
while (!(ST1 & 0x01));

// Read magnetometer data
uint8_t Mag[7];
I2Cread(MAG_ADDRESS, 0x03, 7, Mag);

// Create 16 bits values from 8 bits data

// Magnetometer
int16_t mx = Mag[3] << 8 | Mag[2];
int16_t my = Mag[1] << 8 | Mag[0];
int16_t mz = Mag[5] << 8 | Mag[4];

//Mag in uT
mxt = mx * MAG_RES;
myt = my * MAG_RES;
mzt = mz * MAG_RES;

mxtf = mxt; //Filter(mxt, mxtf, 300);

```

```

mytf = myt;//Filter(myt,mytf,300);
mztf = mzt;//Filter(mzt,mztf,300);

data.x = axgf * G;
data.y = aygf * G;
data.z = azgf * G;
data.temp = temperaturG;
data.magx = mxtf;
data.magy = mytf;
data.magz = mztf;

float xh,yh,ayf,axf,var_compass;

axf = atan( data.x / (sqrt(sq(data.y) + sq(data.z))));
ayf = atan( data.y / (sqrt(sq(data.x) + sq(data.z))));

axf *= 180.00;    ayf *= 180.00;
axf /= 3.141592; ayf /= 3.141592;

xh=mxtf*cos(ayf)+mytf*sin(ayf)*sin(axf)-mztf*cos(axf)*sin(ayf);
yh=mytf*cos(axf)+mztf*sin(axf);

var_compass=atan2((double)yh,(double)xh) * (180 / PI) -90; //
angle in degrees
if (var_compass>0){var_compass=var_compass-360;}
var_compass=360+var_compass;

data.x = axgf*G*cos(var_compass) - aygf*G*sin(var_compass);
data.y = axgf*G*sin(var_compass) + aygf*G*cos(var_compass);

return data;
}

void displayInfo()
{
    if (gps.location.isValid())
    {

```

```

        payload_data.coordinates[0] = String (gps.location.lat(), 6);
        payload_data.coordinates[1] = String (gps.location.lng(), 6);
    }
    else
    {

        payload_data.coordinates[0] = String(init_lat);
        payload_data.coordinates[1] = String(init_lon);
    }
}

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

    return t;
}

float rev_char_repr(String x)
{
    char y[2];
    byte z[2];
    int16_t yz;
    float t;
    int len,i,j,k;

    len = x.length();
    x.toCharArray(y,len);

    z[1] = (int8_t) y[0];
    z[0] = (int8_t) y[1];

```



```
yz = (int16_t)((z[1]<<8)|(z[0]));  
t = ((float)yz*0.0001);  
  
return t;  
}
```

## 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);
            }
        }
        catch { }
    }
}

```

```

        //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);

        }
    }
}

```

**Tabel Hasil Interpolasi Pembacaan MPU9255 dengan amplitude 1G**

Sampling ke-	Nilai (Normalisasi)
1	0
2	0,286388717
3	0,521984089
4	0,706955164
5	0,841470985
6	0,925700598
7	0,95981305
8	0,943977385
9	0,878362648
10	0,763137886
11	0,598472144
12	0,387664295
13	0,146532522
14	-0,105975164
15	-0,350910751
16	-0,56976717
17	-0,750635163
18	-0,886684474
19	-0,9712155
20	-0,997528635
21	-0,958924275
22	-0,852284664
23	-0,688819439
24	-0,483320086
25	-0,250578091
26	-0,005384939
27	0,237467883
28	0,463188891
29	0,656986599
30	0,806585456
31	0,90977366
32	0,966855343
33	0,978134636
34	0,943915674
35	0,864502588
36	0,74019951
37	0,571310574
38	0,358139911
39	0,100991654
40	-0,199830064