PENGUKURAN GELOMBANG AIR LAUT MENGGUNAKAN *ACCELEROMETER* PADA PSoC UNTUK DETEKSI PARAMETER TSUNAMI

SKRIPSI

Disusun sebagai salah satu syarat untuk memperoleh gelar Sarjana Teknik (S.T).

Disusun oleh:

DWI WAHYUDI

3332150071

JURUSAN TEKNIK ELEKTRO

FAKULTAS TEKNIK

UNIVERSITAS SULTAN AGENG TIRTAYASA

2022

LEMBAR PERNYATAAN KEASLIAN SKRIPSI

Dengan ini saya sebagai penulis Skripsi berikut:

Judul	: Pengukuran Gelombang Air Laut Menggunakan *Accelerometer* Pada Psoc Untuk Deteksi Parameter Tsunami
Nama Mahasiswa	: Dwi Wahyudi
NPM	: 3332150071
Fakultas/Jurusan	: Teknik / Teknik Elektro

Menyatakan dengan sesungguhnya bahwa Skripsi tersebut di atas adalah benar-benar hasil karya asli saya dan tidak memuat hasil karya orang lain, kecuali dinyatakan melalui rujukan yang benar dan dapat dipertanggungjawabkan. Apabila dikemudian hari ditemukan hal-hal yang menunjukkan bahwa sebagian atau seluruh karya ini bukan karya saya, maka saya bersedia dituntut melalui hukum yang berlaku. Saya juga bersedia menanggung segala akibat hukum yang timbul dari pernyataan yang secara sadar dan sengaja saya nyatakan melalui lembar ini.

Cilegon, 30 Agustus 2022

Dwi Wahyudi

3332150071

LEMBAR PENGESAHAN

Dengan ini ditetapkan bahwa Skripsi berikut

Judul	: Pengukuran Gelombang Air Laut Menggunakan *Accelerometer* Pada Psoc Untuk Deteksi Parameter Tsunami
Nama Mahasiswa	: Dwi Wahyudi
NPM	: 3332150071
Fakultas/Jurusan	: Teknik/Teknik Elektro

Telah diuji dan dipertahankan pada tanggal 8 September 2022 melalui Sidang Skripsi di Fakultas Teknik Universitas Sultan Ageng Tirtayasa Cilegon dan dinyatakan LULUS.

Dewan Penguji

		Tanda tangan
Pembimbing I	: Dr. Romi Wiryadinata, S.T., M.Eng.	.....
Penguji I	: Imamul Muttakin, S.T., M.Eng.	.....
Penguji II	: Ceri Ahendyarti, S.T., M.Eng.	.....

Mengetahui,
Ketua Jurusan

Dr. Romi Wiryadinata, S.T., M.Eng.
NIP. 198307032009121006

PRAKATA

Puji syukur kehadirat Allah Subhanahu Wa Ta'ala, yang telah memberikan nikmat dan karunia-Nya, sehingga penulis dapat menyelesaikan skripsi yang berjudul **"Pengukuran Gelombang Air Laut Menggunakan *Accelerometer* Pada Psoc Untuk Deteksi Parameter Tsunami"**. Shalawat serta salam semoga tercurah kepada baginda Nabi Muhammad Shallallahu Alaihi Wasallam. Skripsi ini disusun sebagai salah satu syarat dalam memperoleh gelar Sarjana Teknik (S.T) di Universitas Sultan Ageng Tirtayasa. Penyusunan skripsi ini berbagai bantuan telah banyak penulis terima baik materil dan moril, oleh karena itu dalam kesempatan ini penulis menyampaikan rasa terimakasih yang sebesar-besarnya kepada:

1. Kedua orang tua yang telah memberikan dukungan berupa kritik, saran dan finansial selama skripsi ini.
2. Bapak Dr. Romi Wiryadinata, S.T., M.Eng. selaku dosen pembimbing 1 yang telah memberikan bimbingan, masukkan serta arahan kepada penulis selama skripsi ini.
3. Bapak Dr. Romi Wiryadinata, S.T., M.Eng. selaku Ketua Jurusan Teknik Elektro Fakultas Teknik Universitas Sultan Ageng Tirtayasa.
4. Teman-teman yang telah memberikan dukungan berupa kritik dan saran selama skripsi ini.

Akhir kata, penulis menyadari bahwa dalam penyusunan skripsi ini masih banyak terdapat kekurangan dan kesalahan. Sehingga penulis mengharapkan saran dan kritik yang membangun. Semoga skripsi ini dapat membawa manfaat bagi pengembangan ilmu.

Cilegon, 30 Agustus 2022

Penulis

ABSTRAK

Dwi Wahyudi

Teknik Elektro

Pengukuran Gelombang Air Laut Menggunakan *Accelerometer* Pada PSoC Untuk Deteksi Parameter Tsunami

Tsunami sudah beberapa kali terjadi di Indonesia, namun alat untuk mendeteksi tsunami masih kurang dikarenakan banyak alat yang hilang dan rusak. Salah satu solusi masalah ini dengan menggunakan *wave buoy* untuk pengukur tinggi dan perioda gelombang. Skripsi ini bertujuan untuk membuat *wave buoy* menggunakan IMU dan Madgwick *filter* untuk mendapatkan nilai perpindahan dan algoritma *zero crossing detection* untuk mendeteksi parameter gelombang. Penelitian ini menggunakan sensor MPU9250, mikrokontroler Arduino Mega dan kartu memori. Pengujian dilakukan dengan menggerakan sensor dengan gerakan gelombang menggunakan tangan. Hasil pengujian didapatkan berhasil mendeteksi parameter gelombang dengan nilai RMSE untuk tinggi gelombang sebesar 29,32%.

Kata kunci: MPU9250, Arduino Mega 2560, Madgwick *Filter*, *Trapezoidal Rule*, posisi linear.

ABSTRACT

Dwi Wahyudi

Electrical Engineering

Measurement of Sea Waves Using Accelerometer On Psoc For Tsunami Parameter Detection

Tsunami have occurred several times in Indonesia, but device for detect the tsunami is still less because a lot of devices are lost and damaged. One of the solutions is wave buoy that measure wave height and period. The purpose of this research is to make wave buoy using IMU and madgwick filter to find the displacement value and zero crossing detection algorithm to detect wave parameter. This research use IMU, microcontroller Arduino Mega and memory card. Testing is done with move the sensor by imitating wave movement with hand. Test successfully detect wave parameter with RMSE is 29,32%.

Keywords:
MPU9250, Arduino Mega 2560, Madgwick Filter, Trapezoidial Rule, Linear Position.

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1 Latar Belakang

Akhir tahun 2018 tepatnya Desember 2018 Provinsi Banten dan Lampung khususnya daerah pesisir mulai dari Carita hingga Sumur diterjang tsunami yang diakibatkan oleh longsor dari anak gunung Krakatau [1]. Indonesia juga pernah mengalami tsunami yang sangat besar yaitu 26 Desember 2004 di Propinsi Daerah Istimewa Nanggroe Aceh Darussalam (NAD) [1][2]. Penyebab tsunami di Aceh adalah Gempa Tektonik berskala 9 Skala Richter, dengan pusat gempa berada sekitar 200 km sebelah barat daya Provinsi NAD, di Samudra Hindia[2][3]. Dampak tsunami juga sangat merugikan, negara Jepang adalah salah satu negara dengan daerah pantai berpenduduk padat di dunia dan sejarah panjang mengenai gempa bumi, tsunami telah memporak-porandakan seluruh populasi pantai, ada juga sejarah kehancuran berat akibat tsunami di Alaska, Kepulauan-kepulauan Hawaii, dan Amerika Selatan [3].

Fenomena Tsunami sendiri adalah serangkaian gelombang laut yang memiliki panjang gelombang dan periode yang sangat panjang, dihasilkan oleh suatu gangguan yang memindahkan masa air laut dalam jumlah yang besar dari keadaan setimbangnya [3]. Gelombang tsunami memiliki dampak atau kerugian yang besar sehingga perlu dilakukan pencegahan untuk mengurangi kerugian dan korban jiwa. Deteksi dini tsunami bisa melalui dua cara yaitu dari sumber gempa atau dari gelombang tsunami itu sendiri. Perkiraan dampak tsunami juga dapat dilakukan dengan menggunakan metode numerik yang membutuhkan data karakteristik gelombang tsunami sehingga hasil yang didapat berdasarkan gelombang laut sangatlah efektif.

Gelombang laut atau *sea wave* (SW) adalah pergerakan naik turunnya air dengan arah tegak lurus permukaan air laut yang membentuk kurva sinusoidal, umumnya gelombang laut disebabkan oleh tiupan angin baik secara langsung atau tidak langsung [4]. Penelitian mengenai gelombang permukaan laut sangat penting untuk membantu pengambilan keputusan pada keselamatan di laut, operasi perairan dan juga perencanaan bangunan pertahanan pantai [5]. Data gelombang laut juga

dapat membantu acuan bagi kebutuhan masyarakat dan pemerintah dalam melaksanakan kegiatan pelayaran, perdagangan, perikanan [5].

Upaya pemerintah Indonesia dalam meghadapi tsunami dengan cara membuat *Tsunami Early Warning System*, Tsunami *buoy* yang digunakan di Indonesia sendiri terdiri empat jenis *buoy*, yaitu *Deep Ocean Assessment and Reporting Tsunamis* (DART) Amerika, *German-Indonesian Tsunami Warning System* (GITWS), *Buoy Wavestan* dan INA-*buoy* buatan Indonesia [6][7]. *Buoy* sempat tidak berfungsi dari tahun 2012 sampai 2018 dikarenakan rusak atau hilang, saat ini Indonesia sudah kembali memasang tsunami buoy dibeberapa titik di wilayah Indonesia dan tahun 2022 BPPT berencana untuk memasang 20 tsunami buoy[8][9].

Penelitian mengenai alat pengukur gelombang laut menggunakan sensor *accelerometer* pernah dilakukan. Sensor *accelerometer* yang digunakan terdapat pada modul MPU6050. Metode perhitungan penelitian ini dengan cara mengintergralkan sebanyak dua kali nilai percepatan sumbu z dan tidak menggunakan sensor *gyroscope* yang terdapat pada modul MPU6050. Hasil penelitian ini memiliki akurasi sebesar 94,95% [10].

Penelitian selanjutnya membuat *Wave buoy* untuk mengukur tinggi gelombang di pesisir. Penelitian ini mengukur tinggi gelombang laut menggunakan modul MPU9250 yang di dalamnya terdepat sensor *accelerometer, gyroscope* dan *magnetometer*. Sensor diproses menggunakan mikrokontroler Arduino mega 2560. Ketinggian gelombang didapatkan dengan kecepatan sumbu z diproses dengan *Fourier series*. Penelitian ini memiliki kekurangan yaitu modul yang digunakan tidak efisien karena dari tiga sensor hanya satu sensor yang digunakan yaitu *accelerometer* [11].

Penelitian lain untuk mengukur karakteristik laut. Sensor yang digunakan adalah MPU9250 dengan mikrokontroler Arduino Mega 2560. Penelitian ini memperkirakan sifat-sifat dasar medan gelombang yaitu tinggi gelombang signifikan, arah dan periode gelombang dominan, spektra *directional* permukaan. Data mentah dari sensor diproses setelah alat menghitung gelombang selama durasi tertentu pada laptop. Pengukuran simultan percepatan vertikal dan kemiringan gelombang dalam dua arah ortogonal memberikan dasar untuk estimasi spektra terarah [12].

Penelitian yang lain mengenai pengukuran karakteristik gelombang laut (tinggi gelombang, periode gelombang dan arah gelombang) yaitu dengan menggunakan GPS (*Global Positioning System*). Parameter gelombang laut didapatkan berdasarkan pergerakan *buoy* yang digerakan oleh gelombang laut. Metode yang digunakan adalah *zero down-crossing*, data ketinggian dari GPS juga dilakukan *filtering* menggunakan *high pass filter*. Alat ini memiliki kelemahan apabila harus mengukur gelombang tsunami karena delay yang diperlukan oleh gps serta kondisi cuaca yang sangat berpengaruh pada data dari GPS [13].

Pengukuran gelombang laut, kecepatan arus air dan deteksi tsunami menggunakan sensor MPU9250 dipilih karena lebih efektif dari segi biaya dan digunakan juga oleh *wave buoy* untuk komersil juga telah terbukti [12]. Pengukuran gelombang laut menggunakan *wave buoy* sendiri bisa dengan *accelerometer* atau GPS, kedua metode ini menghasilkan performa yang sama-sama bagus [12][13][14]. Berdasarkan kekurangan dari GPS seperti sudah dijabarkan di atas maka dipilihlah sensor *accelerometer*, *gyroscope* dan *magnetometer*.

Metode untuk mendapatkan karakteristik gelombang laut menggunakan *zero crossing analysis,* yaitu metode yang paling umum untuk mengukur tinggi, frekuensi atau periode gelombang [15]. Data yang dijadikan nilai ambang bawah sehingga ketika salah satu nilai terpenuhi atau melibihi maka otomatis dikategorikan sebagai gelombang tsunami. Gelombang tsunami memriliki karakteristik gelombang laut berfrekuensi rendah, gelombang dengan periode sekitar 12 menit, merambat pada kecepatan 180 m/s dan memiliki panjang gelombang 130 km di perairan 3,6 km [4].

Penelitian ini menggunakan Arduino Mega 2560 sebagai mikrokontrolernya dan MPU 9250 sebagai sensor IMU (*Inertia Measurement Unit*). Berdasarkan beberapa penelitian di atas tentang pengukuran parameter gelombang laut dapat terlihat bahwa sensor yang digunakan bisa berbeda-beda begitu juga dengan metodenya dengan setiap sensor dan metode memiliki kelebihan dan kekurangan masing-masing. Penelitian ini menggunakan sensor *accelerometer*, *gyroscope* dan *magnetometer* untuk mengukur percepatan, kecepatan sudut dan medan magnet pada 3 sumbu x, y dan menggunakan metode madgwick filter dan quaternion.

1.2 Rumusan Masalah

Rumusan permasalahan dari penelitian ini berdasarkan latar belakang masalah yaitu kurangnya *wave buoy* untuk mengukur karakteristik gelombang tsunami di Indonesia.

1.3 Tujuan Penelitian

Tujuan yang ingin dicapai dari penelitian ini yaitu membuat *prototype* untuk *wave buoy* yang dapat mengukur parameter gelombang laut (tinggi dan periode gelombang) sebagai pendeteksi gelombang tsunami.

1.4 Manfaat Penelitian

Penelitian ini mendapatkan beberapa manfaat yaitu mengetahui karakteristik gelombang laut di daerah sensor ditempatkan sehingga bisa dimanfaatkan untuk meminimalisir dampak bencana tsunami, memanfaatkan 9DoF (*Degree of Freedom*) untuk mengetahui karakteristik gelombang laut. Untirta ikut menjadi bagian dalam penelitian mengenai *wave buoy*.

1.5 Batasan Penelitian

Batasan masalah penelitian ini adalah sebagai berikut:

1. Mikrokontroler yang digunakan adalah Arduino Mega 2560.
2. Karakteristik gelombang laut yang diukur hanya tinggi dan periode gelombang.
3. Tinggi dan perioda gelombang hanya dihitung oleh sensor *accelerometer, gyroscope* dan *magnetometer*.
4. Alat hanya mengenai metode merubah data sensor menjadi karakteristik gelombang laut tidak mengenai design *buoy* atau pelampunyanya.
5. Gelombang yang diukur bukanlah gelombang laut yang asli.
6. Pengukuran dilakukan hanya skala lab atau simulasi.
7. Gelombang tsunami dideteksi dengan menggunakan parameter ambang batas minimal tinggi dan panjang gelombang.
8. Mikrokontroler digunakan sebagai pengambil data dan proses dilakukan di matlab.

9. Program untuk sensor MPU 9250 diprogram menggunakan *software* arduino IDE, kemudian data *logger* dari sensor tersebut direkam menggunakan *CoolTerm* yang berfungsi sebagai aplikasi terminal serial *port* sederhana untuk merekam data.
10. Data *logger* dari MPU 9250 kemudian diolah dengan *filter* Madgwick menggunakan *software* Matlab.

1.6 Sistematika Penulisan

BAB I PENDAHULUAN

Bab ini menjelaskan tentang latar belakang, perumusan masalah, tujuan penelitian, manfaat penelitian, batasan masalah, serta sistematika penulisan.

BAB II TINJAUAN PUSTAKA

Bab ini menjelaskan tinjauan pustaka dan dasar teori, didalamnya memuat kajian-kajian dari hasil artikel publikasi terkait dengan topik penelitian, dan landasan-landasan teori yang menunjang penyelesaian penelitian.

BAB III METODOLOGI PENELITIAN

Berisi tentang metode yang dipergunakan dalam penelitian, perangkat penelitian yang digunakan baik berupa *software* atau data-data, perancangan penelitian, serta tempat dan waktu penelitian

BAB IV HASIL DAN PEMBAHASAN

Dalam bab ini dijelaskan mengenai hasil dari penelitian serta analisa terhadap hasil penelitian yang dikaitkan dengan tinjauan pustaka dan metodologi penelitian.

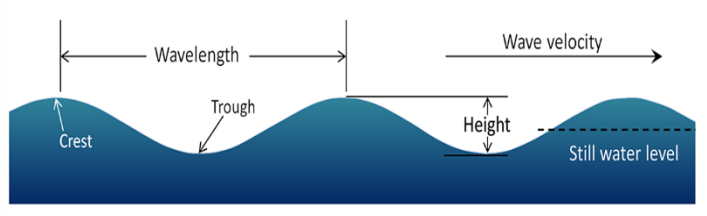
BAB V PENUTUP

Bab ini berisi kesimpulan dari penelitian dan saran untuk melengkapi kekurangan yang ada dalam penelitian ini.

BAB II
TINJAUAN PUSTAKA

2.1 Gelombang Laut

Gelombang laut secara umum terbentuk dari perpindahan energi dari gerakan angin ke permukaan laut yang melepaskan energi ke garis pantai [4]. Penyebab gelombang laut terdiri dari berbagai faktor, diantaranya yaitu oleh gaya tarik gravitasi, gempa bumi atau letusan gunung berapi [16]. Gelombang laut juga memiliki pergerakan yang acak dan kompleks, sehingga sulit diukur dan dirumuskan secara akurat [11]. Bentuk gelombang bisa dilihat pada (Gambar 2.1).



Gambar 2. 1 Karakteristik Gelombang [4]

Gelombang laut memiliki beberapa komponen (Gambar 2.1), diantaranya panjang gelombang (*wavelength*) yaitu jarak antara 2 titik yang sama, puncak gelombang (*crest*) yaitu titik tertinggi, lembah (*Trough*) yaitu titik terendah, tinggi gelombang (*Wave height*) merupakan jarak antara lembah dan puncak, dan kecepatan gelombang (*Wave velocity*) yaitu waktu yang dibutuhkan satu gelombang untuk melintasi titik yang sama. Jenis gelombang laut dapat dilihat pada (Tabel 2.1).

Tabel 2. 1 Jenis Gelombang Laut [17]

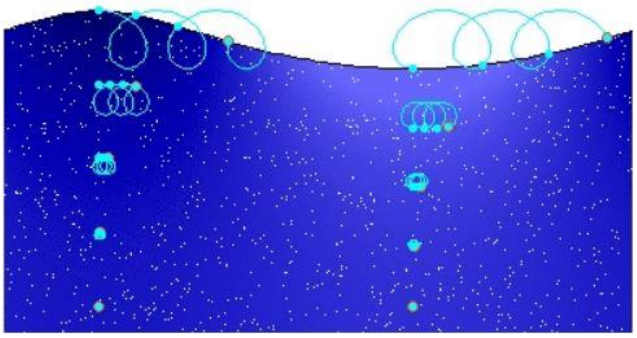
Gelombang	Kedalaman
Gelombang air dalam	≥ ½ panjang gelombang
Gelombang air sedang	1/20 – ½ panjang gelombang
Gelombang air rendah	≤ 1/20 panjang gelombang

Jenis gelombang laut (Tabel 2.1) dibedakan berdasarkan kedalaman air laut dimana gelombang itu terbentuk. Kelas pada gelombang laut dibedakan berdasakan karakteristik, penyebab terbentuknya dan jenis gelombang berdasarkan kedalaman lautnya. Kelas gelombang laut paling sering digunakan untuk membedakan gelombang laut. Kelas gelombang laut lihat (Tabel 2.2).

Tabel 2. 2 Kelas Gelombang Laut [17]

Gelombang	Perioda	Panjang	Sebab	Jenis
Capillary	< 0.1 detik	< 2 cm	Angin lokal	Dalam ke rendah
Chop	1-10 detik	1-10 m	Angin lokal	Dalam ke rendah
Swell	10-30 detik	Ratusan m	Badai	Dalam ke rendah
Seiche	10 m-10 Jam	Ratusan km	Angin, tsunami, tidal	Rendah ke sedang
Tsunami	10-60 m	Ratusan km	Gempa bumi, erupsi	Rendah ke sedang
Tide	12,4-24,8 jam	Ribuan km	Gravitasi	rendah

Gelombang laut memiliki karakteristik, yaitu perioda adalah waktu yang dibutuhkan 2 puncak gelombang melewati titik yang sama. Frekuensi, yaitu jumlah gelombang laut melewati titik yang sama dalam waktu tertentu, umumnya setiap satu detik. Kecepatan, yaitu waktu yang dibutuhkan untuk menempuh jarak tertentu. Gelombang laut memiliki sifat, apabila kedalaman laut minimal setengah dari panjang gelombangnya maka air laut hanya berpindah sedikit secara horizontal, melainkan energi dari gelombang laut yang berpindah, partikel air laut sendiri bergerak dalam orbit melingkar, dengan ukuran orbitnya sama dengan tinggi gelombang [4][16], lihat (Gambar 2.2).

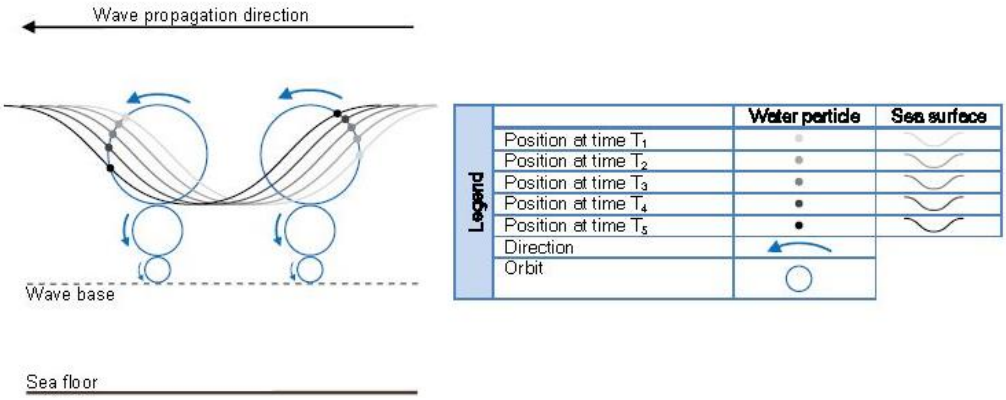


Gambar 2. 2 Pergerakan Orbital Gelombang Laut [4]

Pergerakan orbital ini berkurang seiring berkurangnya kedalaman laut yang dilintasi gelombang, lihat (Gambar 2.2). Sehingga pada kedalaman laut tertentu gelombang laut tidak membentuk gerakan melingkar. Gelombang laut rata-rata memliki panjang gelombang kurang dari ratusan meter, sehingga permukaan bawah laut dalam tidak terpengaruh oleh gelombang pada permukaan laut.

2.2 Pergerakan Buoy

Wave buoy mengukur karakteristik gelombang dengan mengikuti pergerakan orbital dari gelombang laut. *Wave buoy* diletakan di laut dengan dua cara, yaitu pertama dengan mengikat *wave buoy* pada pemberat yang diletakan dibawah laut sehingga *wave buoy* diam, cara kedua dengan tidak mengikatnya, sehingga *wave buoy* bisa bergerak terbawa oleh arus permukaan laut untuk mengukur gelombang laut dari titik yang berbeda. Pergerakan *buoy* dapat dilihat pada (Gambar 2.3).



Gambar 2. 3 Pergerakan Orbital *Wave Buoy* [17]

Jumlah titik tertentu pada orbit *wave buoy* yang dilewati sama dengan jumlah gelombang yang melewati *buoy*. Berdasarkan (Gambar 2.3) terlihat bahwa untuk mengihitung tinggi gelmbong, diperlukan jarak vertikal dari gerakan satu orbit penuh. Jarak vertikal rata-rata yang ditempuh per orbit dapat diketahui dengan menghitung total jarak yang ditempuh selama periode waktu tertentu dan membaginya dengan jumlah gelombang pada periode waktu tersebut.

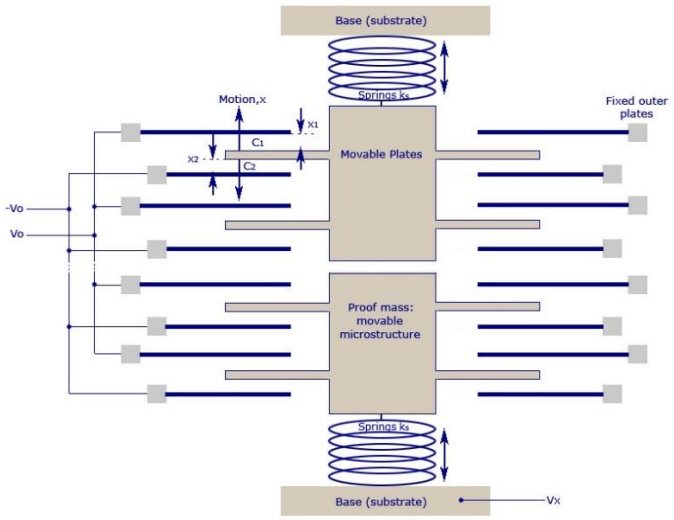
2.3 Sensor IMU

Inertial measurement unit (IMU) adalah modul yang terdiri dari accelerometer dan gyroscope atau IMU 6DoF yang digunakan untuk mendapatkan nilai orientasi sensor terhadap orientasi bumi [18][19]. Jenis lain terdapat tambahan sensor magnetometer, yang disebut *Magnetic Angular Rate and Gravity* (MARG) atau IMU 9 DoF. IMU 9 DOF mengukur orientasi dengan acuan gravitasi dan kutub utara medan magnet bumi. Kelebihan dari IMU 9 DoF dibandingkan 6 DoF memiliki keluaran yang lebih stabil jika dioperasikan dengan *filter madgwick*. Penggunaan sensor IMU 9DoF bertujuan untuk mengatasi kekurangan dari

giroskop saat menghitung orientasi sensor (*roll, pitch* dan *yaw*), yaitu *drift* dari giroskop dikarenakan dalam pengukurannya giroskop tidak mempunyai titik acuan, disini akselerometer dan magnetometer mengukur gravitasi dan medan magnet bumi sebagai acuan untuk mengatasi *drift* [18].

2.3.1 *Accelerometer*

Accelerometer adalah sensor yang digunakan untuk mengukur percepatan. *Micro Electro mechanical system* (MEMS) adalah teknologi yang memiliki bentuk fisik yang sangat kecil. MEMS *accelerometer* yang digunakan adalah jenis *capacitive accelerometer*. sensor mengkonversi perpindahan massa dari pergerakan sensor menjadi perubahan nilai kapasitansi yang kemudian dikonversi menjadi sinyal tegangan [20]. Sensor *accelerometer* memiliki dua elemen utama yang terdiri dari pegas dan massa yang berukuran mikro, lihat (Gambar 2.4).



Gambar 2. 4 Mekanikal MEMS *Accelerometer* [21]

Prinsip dasar *accelerometer* berdasarkan parameter pada (Gambar 2.4), yaitu ketika sensor mendapatkan percepatan dari luar, maka *proof mass* berpindah mendorong pegas yang membuat *proof mass* bergerak secara osilasi, pergerakan *proof mass* ini menyebabkan perubahan kapasitansi *antara proof* mass dan pelat elektroda yang dideteksi dengan perubahan tegangan keluaran dari sirkuit pada sensor [22]. Kapasitansi antara *movable plate* dengan dua pelat elektroda adalah C_1 dan C_2, pada saat tidak ada percepatan, maka nilai C_1 dan C_2 adalah sama, karena

perpindahan *movable plat* $x_1 = x_2$, Tetapi saat sensor diberikan gerakan, maka *proof mass* bergerak atau berpindah (x), menyebabkan terjadinya perbedaan nilai kapasitansi yang dapat dihitung menggunakan Persamaan (2.1).

$$C_2 - C_1 = 2\Delta C = 2\epsilon_A \frac{x}{d^2 - x^2} \tag{2.1}$$

Parameter $\epsilon_A = \epsilon_0 \epsilon A$ dimana A adalah luas permukaan elektroda, d adalah jarak antara elektroda dan ϵ adalah nilai permitivitas dari material pemisah elektroda. Jarak perubahan posisi x dicari menggunakan Persamaan (2.2).

$$\Delta C x^2 + \epsilon_A x - \Delta C d^2 = 0 \tag{2.2}$$

Persamaan (2.2) disederhanakan untuk nilai perubahan posisi yang kecil, parameter $\Delta C x^2$ bisa diabaikan, sehingga dengan menghilangkan $\Delta C x^2$ didapatkan Persamaan (2.3).

$$x \approx \frac{d^2}{\epsilon_A} \Delta C = d \frac{\Delta C}{C_0} \tag{2.3}$$

Persamaan (2.3) digunakan untuk semua kapasitor. Hal ini dikarenakan untuk semua kapasitor bagian atas dihubungkan secara pararel C_1 begitu juga dengan kapasitor bagian bawah C_2. Berdasarkan (Gambar 2.6), terdapat parameter V_0 dan V_x. Nilai V_0 adalah nilai keluaran dari pelat elektroda dan V_x adalah nilai keluaran dari *proof mass*, kedua parameter ini bekerja dengan Persamaan (2.4).

$$(V_x + V_0) C_1 + (V_x - V_0) C_2 = 0 \tag{2.4}$$

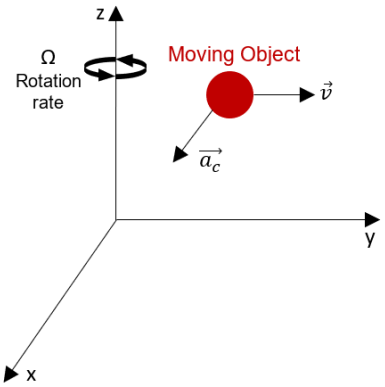
Nilai keluaran tegangan V_x berdasarkan Persamaan (2.3) bisa ditulis dengan Persamaan (2.5).

$$V_x = V_0 \frac{C_2 - C_1}{C_2 + C_1} = \frac{x}{d} V_0 \tag{2.5}$$

Nilai keluaran dari V_x inilah yang digunakan sensor untuk menghitung nilai percepatan yang dihitung setelah diolah oleh rangkaian *filter* dan penguat.

2.3.2 *Gyroscope*

Gyroscope digunakan untuk mengukur kecepatan sudut kerangka acuan inersia. Prinsip kerja sederhana dari *MEMS gyroscope* dengan menggunakan mekanikal getar (*vibrating mechanical*) untuk mendeteksi perubahan sudut sensor [23]. MEMS *gyroscope* dengan mekanikal getar memanfaatkan prinsip *coriolis acceleration* untuk mengukur kecepatan sudut, prinsip *coriolis acceleration* bisa dilihat pada (Gambar 2.5).

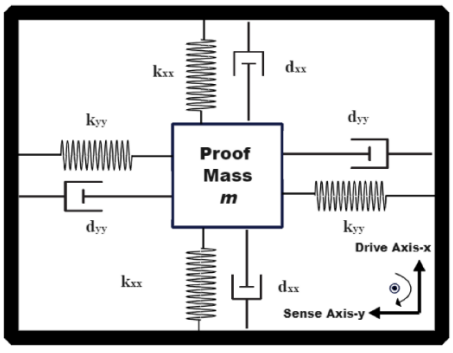


Gambar 2. 5 Percepatan *Coriolis Effect* [24]

Penjelasan lebih rinci mengenai *coriolis effect*, pada (Gambar 2.5) objek merah (*moving object*) pada bidang X, Y, Z yang bergerak lurus dengan kecepatan v ke arah sumbu Y saat bidang atau *reference frame* diam, lalu apabila *reference frame* diputar terhadap sumbu Z, maka objek merah memiliki gaya ke arah sumbu yang tegak lurus terhadap bidang yang terbentuk oleh arah *moving object* itu dan sumbu rotasi dimana benda itu diputar, dalam (Gambar 2.5) gaya ini memiliki percepatan a_c dan bergerak ke arah sumbu-x, fenomena ini yang disebut *coriolis effect* [24]. Percepatan dari *coriolis effect* dituliskan dengan Persamaan (2.6).

$$a_c = 2\text{v x } \Omega \tag{2.6}$$

Percepatan *coriolis effect* pada Persamaan (2.6) adalah besaran percepatan *moving object* merah (a_c) pada (Gambar 2.5), percepatan ini juga menjadi prinsip dasar dari sistem mekanikal *vibrating mass MEMS gyroscope*. Prinsip kerja mekanikal *gyroscope* dengan memannfaatkan *coriolis effect* bisa dilihat pada (Gambar 2.6).



Gambar 2. 6 Mekanikal MEMS *Gyroscope* [25]

Komponen utama dari MEMS *gyroscope* pada (Gambar 2.6) adalah *sensing element* yang mengukur kecepatan sudut berdasarkan *coriolis effect* tersusun dari massa, pegas dan peredam. *Read-out circuit* terdiri dari *drive mode* dan *sense mode*. *Case* digunakan untuk melindungi komponen sensor dan untuk I/O *interface*. Saat sensor digerakan *proof mass* berisolasi pada *drive* dan sumbu yang ortogonal, menghasilkan *coriolis effect* yang membuat osilasi dihasilkan pada *sense axis* [23]. Osilasi pada sense axis ini yang digunakan untuk menghitung kecepatan sudut. *Coriolis effect* dituliskan dengan Persamaan (2.7).

$$F_c = -2m\vec{\Omega}x\vec{v} \tag{2.7}$$

$\vec{\Omega}$: laju sudut aksial

$\vec{v}$: vektor osilasi

Model matematis gabungan dari *vibrating gyroscope* dituliskan pada Persamaan (2.8) dan (2.9) dengan asumsi bahwa $\Omega_x^2 \approx \Omega_y^2 \approx \Omega_z^2 \cong 0$ dan $\dot{\Omega}_z \equiv \dot{\Omega} \approx 0$, dimana nilai Ω yaitu kecepatan sudut tidak diketahui.

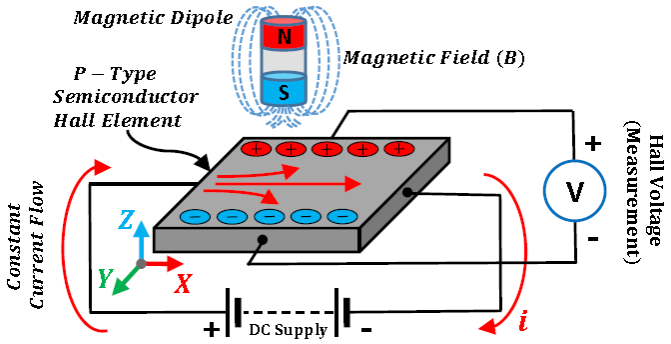
$$m\ddot{x} + d_{xx}\dot{x} + K_{xx}x + d_{xy}\dot{y} + K_{xy}y = u_x + 2m\Omega\dot{y} \tag{2.8}$$

$$m\ddot{y} + d_{yy}\dot{y} + K_{yy}y + d_{xy}\dot{x} + K_{xy}x = u_y - 2m\Omega\dot{x} \tag{2.9}$$

Persamaan (2.8) dan (2.9) memiliki variabel m yaitu massa dari proof mass, x dan y adalah koordinat dari proof mass lalu K_{xx} dan K_{yy} merupakan nilai koefisien dari pegas, parameter d_{xx} dan d_{xx} merupakan nilai peredam, selanjutnya u_x dan u_y merupakan nilai kontrol, terakhir $2m\Omega\dot{y}$ dan $2m\Omega\dot{x}$ keduanya adalah gaya kopling hasil dari *coriolis effect* serta d_{xx} dan d_{yy} adalah galat dari pegas [25].

2.3.3 *Magnetometer*

Magnetometer pada sensor IMU dapat mengukur besar medan magnet bumi disekitar sensor. Teknologi magnetometer dalam mendeteksi medan magnet bermacam-macam, *hall effect* sensor adalah yang paling banyak beredar dipasaran, ada juga *Anisotropic Magnetoresistors* (AMR), *Tunnel Magnetoresistors* (TMR) dan *Giant Magnetoresistors* (GMR) [26]. MPU9250 menggunakan AK8963 sebagai sensor magnetometer, AK8963 memanfaatkan fenomena *hall effect* [27]. Prinsip kerja magnetometer berdasarkan fenomena *hall effect* bisa dilihat pada (Gambar 2.7).



Gambar 2. 7 *Hall-effect Magnetic Sensor* [28]

Fenomena *hall effect* (efek *Hall*) sendiri bisa divisualisasi seperti pada (Gambar 2.7), ketika benda semikonduktor berbentuk pipih yang dialiri arus listrik secara *continue* didekatkan dengan medan magnet maka membuat muatan listrik (elektron dan *holes*) terpisah dikedua sisi pelat semikonduktor [23][28]. Hasil dari pemisahan elektron dan *holes* ke dua sisi yang berbeda menghasilkan perbedaan tegangan listrik, perbedaan tegangan listrik ini yang dikonversi menjadi besaran medan magnet pada sensor magnetometer. Konduktor yang didekatkan dengan medan magnet menghasilkan gaya dengan Persamaan (2.10).

$$\vec{F}_L = q(\vec{E} + (\vec{v}x\vec{B})) \tag{2.10}$$

Persamaan (2.10) memiliki variabel q untuk muatan elektron, $\vec{E}$ yaitu medan listrik, variabel $\vec{B}$ adalah besar medan magnet dan $\vec{F}_L$ adalah gaya lorentz. Gaya Lorentz yang dihasilkan berdasarkan Persamaan (2.10) merubah persebaran muatan pada konduktor yang menyebabkan perubahan tegangan listrik pada konduktor tersebut, perbedaan tegangan ini memiliki Persamaan (2.11) [29].

$$V_H = \frac{I\|\vec{B}\|}{qNd} \tag{2.11}$$

Tegangan Hall (V_H) memiliki variabel I untuk nilai arus litrik, N untuk nilai kerapatan elektron pembawa (*carrier density*) dan variabel d adalah besar dari ketebalan konduktor dalam meter (m). Besar tegangan sebanding dengan magnitude dari medan magnet $\vec{B}$ dan dapat ditentukan, dengan asumsi bahwa nilai I, N dan d diketahui dan konstan [29].

2.4 Orientasi

Penelitian ini bertujuan mendapatkan perpindahan alat secara vertikal dalam satuan meter, perpindahan dihitung dari data percepatan sumbu z terhadap bumi. Namun, saat pengujian alat tidak berada dalam posisi tegak lurus dengan bumi yang mengakibatkan sumbu z pada MPU 9250 berubah-ubah, sehingga orientasi sensor harus diubah berdasarakan referensi bumi. Data percepatan alat berdasarkan referensi bumi ini yang akan dikonversi menjadi data perpindahan vertikal.

2.4.1 *Reference Frame*

Kerangka acuan inersia adalah kerangka acuan yang asalnya berada di pusat bumi. Kerangka ini tidak berputar, tidak berakselerasi dan posisinya tetap terhadap benda-benda langit di alam semesta, orientasi kerangka seperti sumbu-Z sejajar dengan Sumbu bumi yang berputar, sumbu-X memanjang ke arah mean *vernal equinox*, dan sumbu Y terhadap kerangka ortogonal kanan [30]. Kerangka bumi adalah kerangka acuan yang asalnya tetap pada pusat bumi dan sumbunya sejajar dengan meridian rata-rata Greenwich dan sumbu Y melengkapi kerangka ortogonal tangan kanan. Kerangka bumi juga disebut sebagai *Earth Centered Earth Fixed* (bingkai ECEF) [30].

Kerangka acuan navigasi adalah kerangka acuan lokal yang titik asalnya bertepatan dengan kerangka acuan sensor, sumbu Z mengarah ke atas normal *ellipsoid*, sumbu X mengarah ke utara geodetik dan Sumbu Y mengarah ke timur geodetik. Kerangka ini disebut sebagai Kerangka ENU (*East North Up*) [30]. *Body frame* (b-frame) adalah kerangka acuan lokal yang asalnya bertepatan dengan pusat *body* (dalam penelitian adalah alat), sumbu X mengarah ke sisi depan alat dan sumbu Y mengarah ke sisi kanan alat dan Z- sumbu untuk melengkapi sisi kanan bingkai *orthogonal* [30].

2.4.2 Quaternion

Quaternion didefinisikan sebagai vektor tiga dimensi yang melacak perputarannya, menjadikannya objek 4 parameter. Parameter ini adalah w,x,y,z, di mana w adalah rotasi di sekitar sumbunya sendiri terhadap kerangka acuan dalam radian dan x,y,z adalah koordinat umum untuk menunjukkan vektor dalam tiga

dimensi [16]. Penggunaan quaternion ditujukan untuk menghindari masalah singularitas yang diberikan dalam sudut *euler* yaitu *gimbal lock* [31]. Penulisan vektor quaternion dilihat pada Persamaan (2.12).

$$q = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} = q_0 + q_1 i + q_2 j + q_3 k \tag{2.12}$$

Nilai q_0 adalah bilangan real sedangkan q_1, q_2, q_3 adalah imajiner. Variabel i, j, dan k yang menyertai setiap elemen vektor adalah unit vektor pada sumbu x, y, z ruang tiga dimensi. Dibandingkan dengan sudut Euler, quaternion lebih sederhana untuk menyusun dan menghindari masalah *gimbal lock* [32]. Penulisan lain dari quaternion bisa dilihat pada Persamaan (2.13)

$$q = cos\frac{0}{2} + e\ sin\frac{0}{2} = cos\frac{0}{2} + e_1 i + e_2 j\theta + e_3 k\ sin\frac{0}{2} = cos\frac{0}{2} + \begin{matrix} e_1 \\ e_2 \\ e_3 \end{matrix} sin\frac{0}{2} \tag{2.13}$$

Notasi e_n merupakan untit vektor baru yang nilainya 0 atau 1 tergantung pada sudut rotasi θ diputar dengan sumbu quaternion apa. Sebagai gambaran untuk perputaran quaternion dengan sumbu Z maka $e_1 = 0, e_2 = 0, e_3 = 1$, perputaran quaternion dengan sumbu X maka $e_1 = 1, e_2 = 0, e_3 = 0$ dan perputaran dengan sumbu Y $e_1 = 0, e_2 = 1, e_3 = 0$ [33]. Dari paparan tersebut maka quaternion hanya berlaku untuk satu perputaran sumbu saja.

2.5 Madgwick *Filter*

Madgwick *Filter* merupakan *filter* yang dikembangkan dari Mahony *filter*. Tingkat akurasi dari madgwick *filter* lebih tinggi dari Mahony *filter*, hal ini dikarenakan Madgwick menambahkan sensor *magnetometer* untuk mengurangi bias dari *gyroscope* [19]. Algoritma Madgwick *filter* menggunakan representasi quaternion untuk membuat estimasi baru dari orientasi sensor MARG dengan menerapkan algoritma penurunan gradien. Perhitungan arah giroskop dan kesalahan pengukuran dari turunan quaternion, algoritma madgwick menggunakan data dari *accelerometer* dan *magnetometer* dalam algoritma penurunan gradien yang diturunkan dan dioptimalkan [34]. Dasar dari algoritma *derivative* pada Madgwick *filter* adalah orientasi dari kecepatan sudut, proses algoritma *fusion*, kompensasi dari distorsi magnetik, dan parameter yang dapat disesuaikan [35].

Sensor *gyroscope* mengukur kecepatan sudut pada sumbu x, y dan z terhadap kerangka sensor, masing-masing ditulis ω_x, ω_y dan ω_z. Apabila parameter kecepatan sudut disusun dalam vektor ${}^{S}\omega$ ditulis dalam Persamaan (2.14), maka turunan quaternion untuk laju perubahan pada kerangka bumi relatif terhadap kerangka sensor ${}^{S}_{E}\dot{q}$ dapat dihitung dengan Persamaan (2.15). Dimana, simbol operasi $\otimes$ menunjukan produk dari quaternion dan aksen $\wedge$ menunjukan vektor normalisasi untuk satuan panjang.

$${}^{S}\omega = [0 \quad \omega_x \quad \omega_y \quad \omega_z] \tag{2.14}$$

$${}^{S}_{E}\dot{q} = \frac{1}{2}\,{}^{S}_{E}\hat{q} \otimes {}^{S}\omega \tag{2.15}$$

Orientasi kerangka bumi relatif terhadap kerangka sensor pada saat t, yaitu ${}^{S}_{E}q_{\omega,t}$, dihitung dengan mengitergral kan quaternion ${}^{S}_{E}\dot{q}_{\omega,t}$ seperti yang bisa dilihat pada Persamaan (2.16) dan (2.17), dengan keadaan kondisi awal diketahui. Pada Persamaan (2.16) dan (2.17), nilai ${}^{S}\omega_t$ adalah kecepatan sudut pada saat t, Δt adalah perioda sampling dan ${}^{S}_{E}\hat{q}_{est,t-1}$ merupakan estimasi orientasi sebelumnya. Subskrip ω menunjukan nilai quaternion merupakan hasil dari kecepatan sudut.

$${}^{S}_{E}\dot{q}_{\omega,t} = \frac{1}{2}\,{}^{S}_{E}\hat{q}_{est,t-1} \otimes {}^{S}\omega_t \tag{2.16}$$

$${}^{S}_{E}q_{\omega,t} = {}^{S}_{E}\hat{q}_{est,t-1} + {}^{S}_{E}\dot{q}_{\omega,t}\Delta t \tag{2.17}$$

Orientasi pada kerangka sensor yang relatif terhadap kerangka bumi dapat dihitung menggunakan data pengukuran arah medan magnet dalam kerangka sensor apabila arah medan magnet bumi dalam kerangka bumi diketahui. Rumusan masalah optimasi dimana orientasi sensor, ${}^{S}_{E}\hat{q}$, dapat dihitung ketika perkiraan sebelumnya selaras dengan arah referensi yang telah ditentukan sebelumnya dari bidang dalam kerangka bumi, ${}^{E}\hat{d}$ dengan bidang terukur dalam kerangka sensor ${}^{S}\hat{s}$. Sehingga solusinya adalah Persamaan (2.18), dimana apabila dijabarkan fungsi objektif untuk menghitung orientasi dalam quaternion menggunakan Persamaan (2.19).

$$\min_{{}^{S}_{E}\dot{q}\in\Re^4} f({}^{S}_{E}\hat{q},\ {}^{E}\hat{d},\ {}^{S}\hat{s}) \tag{2.18}$$

$$f\left({}^{S}_{E}\hat{q},\ {}^{E}\hat{d},\ {}^{S}\hat{s}\right) = {}^{S}_{E}\hat{q}^{*} \otimes {}^{E}\hat{d} \otimes {}^{S}_{E}\hat{q} - {}^{S}\hat{s} \tag{2.19}$$

Madgwick filter menggunakan algoritma *gradient descent* untuk menyelesaikan masalah optimalisasi ke-n iterasi yang menghasilkan estimasi

orientasi ${}^{S}_{E}\hat{q}_{n+1}$ berdasarkan orientasi perkiraan/tebakan awal ${}^{S}_{E}\hat{q}_{0}$ dan variabel *step* μ dengan Persamaan (2.20). Kesalahan pada permukaan solusi yang ditentukan oleh fungsi objektif, ***f*** dan jacobian, ***J*** menggunakan Persamaan (2.21).

$${}^{S}_{E}q_{k+1} = {}^{S}_{E}\hat{q}_{k} - \mu\frac{\nabla f({}^{S}_{E}\hat{q}_{k}, {}^{E}\hat{d}, {}^{S}\hat{s})}{\|\nabla f({}^{S}_{E}\hat{q}_{k}, {}^{E}\hat{d}, {}^{S}\hat{s})\|}, k = 0,1,2, \dots n \quad (2.20)$$

$$\nabla f\left({}^{S}_{E}\hat{q}_{k}, {}^{E}\hat{d}, {}^{S}\hat{s}\right) = J^{T}({}^{S}_{E}\hat{q}_{k}, {}^{E}\hat{d}) f({}^{S}_{E}\hat{q}_{k}, {}^{E}\hat{d}, {}^{S}\hat{s}) \quad (2.21)$$

Persamaan (2.20) dan (2.21) adalah bentuk umum dari algoritma *gradient descent* untuk menentukan orientasi. Komponen rotasi matrix dari f dan J berbeda untuk *accelerometer* dan *magnetometer*. Perhitungan menggunakan Persamaan (2.22)

$$\mu(t) = a\|{}^{S}_{E}\dot{q}_{ES}(w,t)\|\Delta t, a > 1 \quad (2.22)$$

Dimana pada ${}^{S}_{E}\dot{q}_{ES}(w,t)$, nilai a adalah perubahan dari orientasi yang diukur oleh *gyroscope* dan augmentasi dari μ untuk menghitung *noise* pada bidang homogen (*accelerometer* dan *magnetometer*). Madgwick *filter* juga memiliki variable β, dinyatakan dalam $\tilde{\omega}_{maxX}$, $\tilde{\omega}_{maxY}$ dan $\tilde{\omega}_{maxZ}$ (kemungkinan maksimal kesalahan pengukuran dari *gyroscope* sumbu x, y dan z pada kerangka sensor) ditulis dengan {ersamaan (2.23).

$$\beta = \left\|\frac{1}{2}\hat{q} \otimes [0 \quad \tilde{\omega}_{maxX} \quad \tilde{\omega}_{maxY} \quad \tilde{\omega}_{maxZ}]\right\| = \frac{3}{4}\tilde{\omega}_{max} \quad (2.23)$$

Parameter β adalah laju divergensi dari ${}^{S}_{E}\dot{q}_{ES}(w,t)$, yang dinyatakan sebagai besaran turunan quaternion dan parameter yang dapat diubah. Variabel ini ditentukan oleh besaran sudut $\tilde{\omega}_{max}$, yang merupakan kesalahan maksimal sensor *gyroscope* dari setiap sumbunya. Parameter β pada Persamaan (2.23) dijelaskan bahwa $\hat{q}$ adalah setiap unit quaternion.

2.6 Integral

Data perpindahan dapat dihitung dengan nilai percepatan dan nilai orientasi yang didapatkan dari filter madgwick. Percepatan dari MPU9250 dikonversi menjadi perpindahan dalam satuan cm atau m dengan cara integral. Apabila diketahui posisi terhadap waktu dari sebuah objek, maka kecepatan dapat dihitung dengan Persamaan (2.24).

$$v(t) = \frac{dx}{dt} \quad (2.24)$$

Mengacu pada Persamaan (2.24), dimana Posisi adalah x(t) dan kecepatan adalah v(t). Percepatan a(t) dapat dicari dengan menggunakan turunan ke dua dari posisi atau turunan pertama dari kecepatan. Dapat dilihat pada Persamaan (2.25).

$$a(t) = \frac{d^2x}{dt^2} = \frac{dv}{dt} \tag{2.25}$$

Berdasarkan Persamaan (2.25), untuk mencari posisi dari percepatan dengan membalik prosesnya. Yaitu, dengan cara melakukan integral dua kali terhadap data percepatan. Pada prinsipnya, menggunakan *double integration* pada percepatan untuk mendapatkan posisi, inisialisasi posisi dan inisialisasi kecepatan harus diketahui [36]. Setelah integral pertama dilakukan, inisialisasi kecepatan harus ditambahkan pada hasil integral, ditulis sebagai Persamaan (2.26).

$$v(t) = v(t_0) + \int_{t_0}^{t} a(t)dt \tag{2.26}$$

Variabel pada Persamaan (2.26) yaitu t_0 adalah waktu inisialisasi dan $v(t_0)$ adalah inisialisasi dari kecepatan, untuk mendapatkan posisi dari kecepatan yang diintegralkan kembali menggunakan prinsip yang sama, dilihat pada Persamaan (2.27).

$$x(t) = x(t_0) + \int_{t_0}^{t} v(t)dt \tag{2.27}$$

Nilai inisialisasi dari kecepatan dan posisi harus diketahui untuk menghidari kesalahan, tetapi niliai insialisasi didapatkan dengan langsung menghitung secara langsung yang membuat ini sangat tidak praktis [37]. Perhitungan posisi menggunakan metode integral tanpa perlu menggunakan nilai inisialisasi. Metode yang dipilih adalah *trapezoidal rule*, metode ini memiliki hasil yang lebih akurat dibandingkan dengan dengan metode *digital integration* seperti *rectangular method* [36]. Metode ini memiliki Persamaan (2.28) berikut.

$$y(n) = y(n-1) + \frac{1}{2f_s}[x(n-1) + x(n)], n > 0 \tag{2.28}$$

Trapezoidal integration memiliki kelebihan lain yaitu bisa digunakan secara *real time* [16]. Sehingga apabila penelitian ini dilanjut dengan membuat perhitungan secara *real time* maka metode integral ini bisa tetap digunakan.

2.7 Kajian Pustaka

Skripsi ini mengenai pengukuran gelombang menggunakan sensor MPU9250 dan Arduino Mega 2560 dengan *filter* madgwick dan integral trapeziodal. Beberapa

peneletian lainnya mengenai *wave buoy* menggunakan sensor *accelerometer* dan mengirimkan datanya menggunakan modul *GPRS* mendapatkan akurasi 94,95% [10]. Penelitian lainnya untuk mengitung tinggi gelombang laut menggunakan sensor *accelerometer* menggunakan metode *fourier transform* menghasilkan rata-rata perbedaan sebesar 0,17 m dibandingkan dengan data prediksi dari Badan Informasi Geospasial atau nilai akurasinya mencapai 85% [11].

Pengukuran karakteristik gelombang laut dengan memfilter data 9dof dengan kalman *filter* menghasilkan akurasi yang sangat baik. Akurasi pengukura tinggi gelombang lebih dari 3,75% x nilai pengukuran, untuk *error* arah gelombang ±1° dan akurasi perioda gelombang ± 0,1 detik, namun *filter* ini memiliki kekurangan yaitu memerlukan pemrosesan yang berat sehingga harus menggunakan mikrokontroler yang mahal [38].

Penelitian untuk mengukur tinggi dan perioda gelombang menggunakan sensor IMU. Data *acceleration* difilter menggunakan metode *anisotropic diffusion* kemudian dikonversi menjadi tinggi gelombang menggunakan *Fast Fourier Transform*. Alat diuji menggunakan simulator rotari sinusoidal sebanyak dua kali dengan nilai tinggi gelombang 3m. Pengujian dengan perioda 10s menghasilkan akurasi tinggi gelomang 97,07% dan perioda 97,6%. Pengujian dengan perioda 14,2s menghasilkan akurasi tinggi gelombang 97,88% dan perioda 99,01% [39].

Penelitian *wave buoy* untuk mengukur tinggi gelombang menggunakan arduino dan sensor IMU dengan modul BN005. Pengujian dilakukan secara simulasi pada wadah air untuk menggantikan gelombang laut. Hasil pengujian dari penelitian ini untuk tinggi gelombang sebanyak 20 kali dengan parameter tinggi yang berbeda menghasilkan akurasi sebesar 73,06% dan kesalahan rata-rata sebesar 26,94% [40].

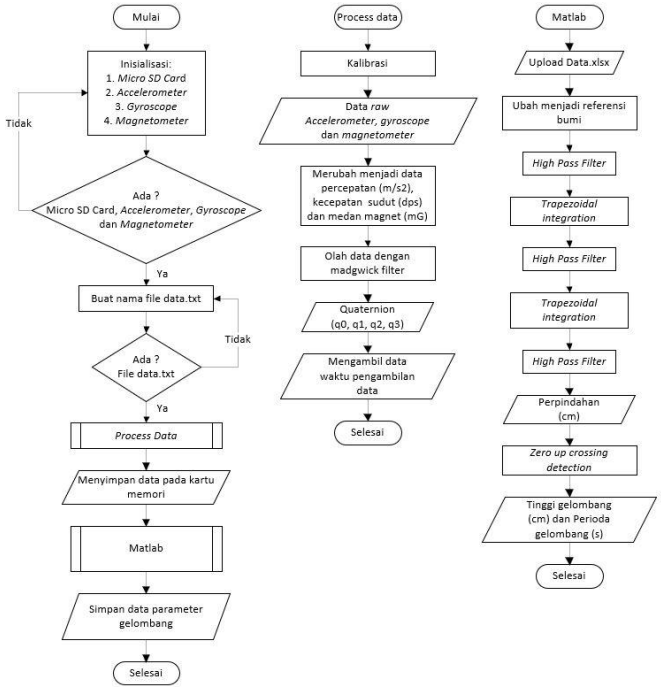
BAB III
METODE PENELITIAN

3.1 Metode Penelitian

Penelitian dalam skripsi ini menggunakan serangkaian kegiatan yang dilakukan untuk memperoleh tujuan yang ingin dicapai. Kegiatan-kegiatan tersebut harus tersusun dalam sebuah metodologi. Metodologi penelitian ini adalah sebagai berikut:

1. Studi literatur, mencari jurnal, artikel penelitian dan buku-buku yang berhubungan dengan permasalahan yang dibahas pada penelitian yang dapat membantu dalam pelaksanaan penelitian.
2. Merangcang *Hardware*, merancang sistem elektrikal MPU9250 dan *micro sd card module* untuk terintegrasi dengan Arduino.
3. Merangcang *software*, perancangan *software* Arduino IDE untuk membaca sensor MPU9250, menyimpan data dengan perantara *micro sd card module* juga perancangan *software* matlab untuk analisis data menjadi gelombang.
4. Menguji Sistem, dilakukan untuk mengetahui kinerja sistem yang telah dibuat. Pengujian sistem dilakukan secara simulasi. Pengujian antara lain simulasi statis dan simulasi dinamis, pada simulasi statis dilakukan pengujian sensor MPU9250 dalam kondisi diam sedangkan pegujian simulasi dinamis dilakukan pada saat kondisi bergerak
5. Menganalisis penelitian dan pengujian, berisi analisis hasil dan pembahasan dari sistem yang telah dibuat dan membuat kesimpulan dari pengujian tersebut.

Sensor *accelerometer*, *gyroscope,* dan *magnetometer* yang terdapat pada modul MPU9250 terintegrasi dengan Arduino mega 2560. Data dari ketiga sensor diproses oleh Atmega 2560 sehingga menghasilkan data percepatan, kecepatan sudut dan besar medan magnet. Data disimpan pada kartu memori melalui modul *SD card reader* yang datanya diproses pada laptop menggunakan *software* matlab. Laptop berfungsi sebagai pengolah data dan juga sebagai media penampil informasi karakteristik gelombang laut, kartu memori berfungsi sebagai penyimpan data. Flowchart penelitian bisa dilihat pada (Gambar 3.1).



Gambar 3. 1 Diagram Alir Sistem

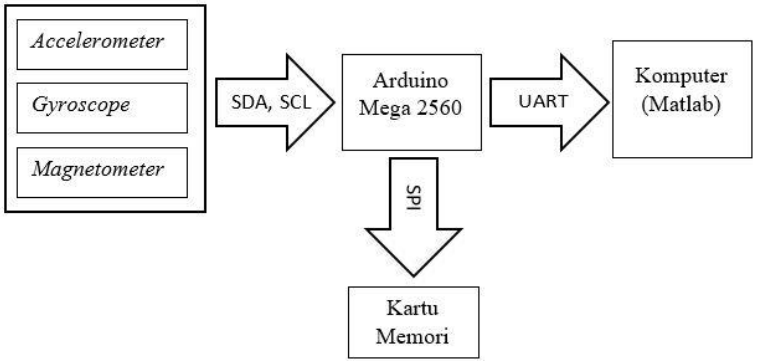
Alur program secara keseluruhan dari sistem pada penelitian ini digambarkan dengan *flowchart* pada (Gambar 3.1). Dimulai dari inisialisasi sensor dan modul. Pengambilan data beserta kalibrasi sensor. Menyimpan data pada kartu memori. Proses filter dan sekaligus menghitung nilai quaternion menggunakan madgwick filter. Hingga mengubah data dari percepatan menjadi perpindahan secara vertikal.

3.2 Instrumentasi Penelitian

Instrument penelitian yang digunakan pada penelitian ini meliputi perangkat keras (*hardware)* yaitu Arduino Mega 2560, MPU9250, *micro sd reader,* kartu memori 8Gb, dan laptop. Perangkat lunak (*software*) yang digunakan adalah Arduino IDE dan matlab.

3.2.1 Perangkat Keras

Perancangan perangkat keras bertujuan untuk akusisi data dari sensor IMU dan mengolah data menjadi data karakteristik gelombang laut. Data yang menjadi perhatian pada penelitian ini adalah data perpindahan alat secara vertikal.. Diagram blok perangkat keras AHRS disajikan pada (Gambar 3.2).



Gambar 3. 2 Diagram Blok Sistem Perangkat Keras

Diagram blok dari perangkat keras pada (Gambar 3.2). Perangkat keras terdiri dari tiga sensor inersia (*accelerometer*, *gyroscope* dan *magnetometer*) yang tergabung dalam sensor MPU9250 sebagai masukan dari sistem yang terhubung dengan mikrokontroler menggunakan komunikasi I^2C, mikrokontroler arduino mega 2560 sebagai pengolah data, kartu memori sebagai penyimpan data yang terhubung dengan mikrokontroler menggunakan komunikasi SPI dengan bantuan modul *card reader*, dan laptop sebagai pengolah dan penampil data.

1. Arduino Mega 2560

Penelitian ini menggunakan mikrokontroler Arduino mega 2560. Arduino digunakan untuk mengambil data dari sensor MPU9250, mengolah data hingga mendapatkan quaternion. Arduino Mega 2560 dipilih karena memiliki kristal sebesar 16MHz, sehingga mikrokontroler dapat berkomunikasi menggunakan UART (*Universal Asynchronous Receiver- Transmitter)* ke komputer dengan *baud rate* sebesar 115200bps. Konfigurasi pin dari Arduino Mega lihat (Tabel 3.1).

Tabel 3. 1 Konfigurasi pin Arduino Mega 2560

Pin	Konfigurasi	Keterangan
Vcc	5v	Hubungkan dengan *pin* Vcc MU9250
Vcc	3.3v	Hubungkan dengan *pin* Vcc *micro sd card reader*
GND	Ground	Hubungkan dengan *pin* GND
D20	SDA	Hubungkan dengan *pin* SDA sensor IMU 9250
D21	SCL	Hubungkan dengan *pin* SCL sensor IMU 9250
D50	MISO	Hubungkan dengan *pin* MISO *micro sd card reader*
D51	MOSI	Hubungkan dengan *pin* MOSI *micro sd card reader*
D52	Clk	Hubungkan dengan *pin* Clk *micro sd card reader*
D53	SS	Hubungkan dengan *pin* SS *micro sd card reader*

Spesifikasi pada (Tabel 3.1) merupakan daftar pin Arduino Mega 2560 yang digunakan. Pin 5v dan 3.3v merupakan pin yang menghasilkan sumber tegangan listrik yang digunakan sebagai sumber tegangan untuk sensor MPU9250 dan modul *micro sd card reader*. Pin SCL dan SDA adalah pin komunikasi I^2C dengan MPU9250, pin MISO, MOSI, Clk dan SS adalah pin komunikasi SPI dengan *micro sd card reader*. Sumber daya untuk Arduino sendiri berasal dari *battery* 9v.

2. MPU9250

MPU9250 pada penelitian ini berfungsi untuk mengambil data percepatan, medan magnet dan kecepatan sudut. Sensor ini memiliki fitur 16-bit ADCs *(analog to digital converters)* dan memiliki faktor skala yang dapat diprogram. Pada *accelerometer* faktor skala antara ±2g, ±4g, ±8g dan ±16g, pada *gyroscope* faktor skala antara ±250deg/s, ±500deg/s, ±1000deg/s dan ±2000deg/s dan *magnetometer* memiliki faktor skala ±4800µT. MPU9250 memiliki dua pilihan komunikasi yaitu I^2C pada 400kHz atau SPI pada frekuensi 20MHZ [27].

Tegangan operasi untuk MPU9250 antara 2.5 – 3.6V, sehinga pin vcc terhubung dengan sumber 3.3V dan pin gnd terhubung dengan pin GND. Komunikasi data dengan mikrokontroler menggunakan I^2C, sehingga pin SDA terhubung dengan pin D20 dan pin SCL terhubung dengan pin D21 pada Arduino, SDA (*Serial Data*) berfungsi untuk mengirim data antara *master* (mikrokontroler) dan *slave* (MPU9250) sedangkan SCL (*Serial Clock*) berfungsi untuk mengirim sinyal *clock* [16][27]. Faktor skala *accelerometer* yang pilih adalah ±2g, *gyroscope* adalah ±250deg/s dan untuk *magnetometer* ±4800µT.

3. *Micro Sd card* 8GB

Sd card digunakan untuk menyimpan data dari sensor yang telah diproses oleh mikrokontroler menjadi informasi percepatan ($^{m}/_{s^2}$), kecepatan sudut ($^{rad}/_{s}$), medan magnet (mG) serta waktu pencatatan untuk pengambilan setiap data. Informasi yang disimpan pada memori ini dimasukan pada *software* matlab untuk diproses menghasilkan keluaran yang diinginkan.

4. Modul *Micro SD Card*

Modul *micro SD card* diperlukan sebagai media penghubung antara kartu memori dengan mikrokontroler. Komunikasi dengan kartu memori menggunakan komunikasi SPI melalui pin MOSI (*Master Output Slave Input*) untuk mengirim

data dari *master* (mikrokontroler) ke *slave* (Modul *Micro SD Card*), MISO (*Master Input Slave Output*) fungsinya untuk mengirim data dari *slave* ke *master*, SCLK (SPI *Clock*) adalah pin untuk sinyal *clock* dan pin SCK (*Slave/Chip Select*) digunakan master untuk memilih *slave* yang dikirim data. Beberapa spesifikasi tambahan dari modul *micro SD card* (Tabel 3.2).

Tabel 3. 2 Modul *Micro SD Card*

Spesifikasi	**Nilai**
Support Card	*Micro SD*
Tegangan operasi	4.4 – 5.5 V
Arus operasi	80 – 200 mA

Spesifikasi pada (Tabel 3.2) merupakan informasi mengenai spisifikasi elektrikal modul *micro SD card* yang digunakan pada penielitian ini. Sumber tegangan dan arus berasal dari mikrokontroler arduino mega 2560, yaitu pin 5v. Tegangan ini diturunakan oleh regulator yang ada pada modul untuk diberikan sebagai masukan Vcc pada *micro sd card*, karena micro sd card sendiri hanya membutuhkan tegangan operasi sebesar 3.3v.

3.2.2 Perangkat Lunak

Perangkat lunak yang digunakan untuk memprogram Arduino Mega adalah Arduino IDE. Penggunaan Arduino IDE dengan menulis algoritma dan perhitungan menggunakan bahasa pemrograman kusus, men-*debug* serta memprogram. Ketika digunakan dengan perangkat keras yang terkait, kombinasi perangkat keras-perangkat lunak ini memungkinkan untuk menguji perangkat keras saat melihat dan melakukan *debug* pada perangkat lunak.

1. Arduino IDE

Arduino IDE merupakan perangkat lunak gratis yang resmi dikeluarkan oleh arduino. Fungsi dari program ini adalah melakukan *compile* program, pengisian kode program, dan serial monitor. Arduino IDE digunakan karena semua sensor dan modul yang digunakan terintegrasi dengan mikrokontroler arduino. Keunggulan dari Arduino IDE adalah dapat melakukan pengisian program dan serial monitor yang terpadu dalam satu perangkat lunak ini tidak membutuhkan perangkat lunak lain untuk membaca data *serial*.

2. MATLAB

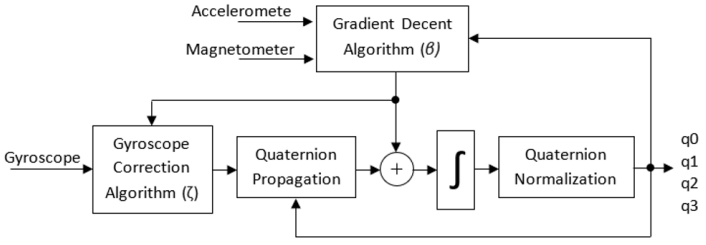
MATLAB merupakan kepanjangan dari *Matrix Laboratory*, perangkat lunak ini memiliki keunggulan untuk menyelesaikan komputasi dalam bidang matematika khususnya untuk melakukan perhitungan yang melibatkan matriks [41]. Penelitian ini menggunakan perangkat lunak MATLAB untuk menjalankan Algoritma *zero crossing* dengan memasukkan data perpindahan vertikal dari arduino, membuat grafik hasil.

3.3 Akuisisi Data Sensor

Akusisi data sensor MPU 9250 dilakukan dengan menghubungkan antara sensor tersebut ke Arduino Mega 2560. Data yang dikirimkan dari sensor merupakan *raw* data yang terdiri dari 3 sumbu *accelerometer*, 3 sumbu *gyroscope*, dan 3 sumbu *magnetometer*. Terdapat 9 *raw* data yang diambil melalui Arduino Mega. Pengambilan data pertama-tama dilakukan dengan menghubungkan sensor MPU 9250 dengan arduino Mega 2560 yang terkoneksi ke komputer dengan menggunakan *software* IDE untuk menampilkan hasil data pengukuran awal dari *accelerometer, gyroscope*, *magnetometer*.

3.4 Perancangan Madgwick Filter

Algoritma filter ini dimulai dengan menormalisasi data dari *accelerometer* dan *magnetometer*, menghitung medan magnet dalam referensi bumi dan efek dari pengukuran kemiringan yang salah terhadap arah medan magnet[41]. Algoritma ini menggabungkan kompensasi distorsi magnetik dan menggunakan representasi quaternion yang memungkinkan data *accelerometer* dan *magnetometer* digunakan dalam algoritma *gradien descent* dan dioptimalkan untuk menghitung kesalahan pengukuran *gyroscope* hasil dari turunan Quaternion (Gambar 3.3).

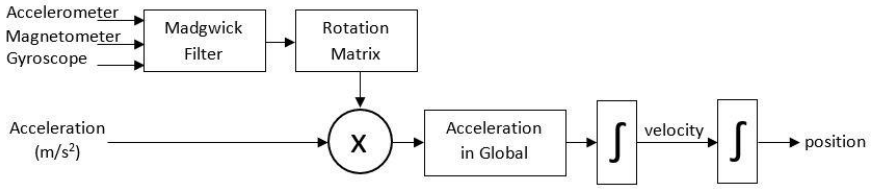


Gambar 3. 3 Diagram Blok Sistem

Algoritma madgwick berdasarkan diagram blok pada (Gambar 3.3) terbagi menjadi dua proses. Pertama, pengukuran *gyroscope* diuraikan dengan algoritma koreksi, yang tergantung pada parfameter ζ, untuk meminimalkan efek karena bias dan kesalahan *drift,* dan digunakan untuk menghitung orientasi *body* atau *object frame* (OF) dengan perambatan Quaternion mulai dari orientasi yang diperkirakan pada langkah sebelumnya. Pengukuran *accelerometer* dan *magnetometer* digabungkan dengan parameter konstanta pembobotan *β* melalui algoritma *gradient descent,* yang formulasinya dilaporkan dalam keluaran dari algoritma *gradient descent* kemudian digunakan untuk memperbaiki orientasi yang diperkirakan dengan hanya mempertimbangkan pengukuran *gyroscope*. Alur kerja dari metode Madgwick pada (Gambar 3.3), yaitu untuk mengetahui representasi pergeseran *frame* berdasarkan sumbu garis normal gravitasi bumi menggunakan Persamaan *gradient descents.* Apabila di *cross product* ketiga Quaternion tersebut menghasilkan matrik yang baru.

3.5 Perhitungan Posisi dan Orientasi

Posisi atau perpindahan diketahui berdasarkan data percepatan, data quaternion dan matrix rotasi yang didapatkan dari filter madgwick. Data percepatan (m/s^2) dari sensor *accelerometer* pada MPU 9250 yang diintegralkan sebanyak dua kali sehingga menghasilkan data perpindahan (m). Diagram blok untuk perhitungan posisi dapat dilihat pada (Gambar 3.4).



Gambar 3. 4 Diagram Blok Perhitungan Posisi

Berdasarkan (Gambar 3.4) madgwick filter berfungsi untuk mendapatkan nilai quaternion berdasarkan data percepatan, kecepatan sudut dan medan magnet dari MPU 9250. Nilai quaternion yang didapatkan kemudian diubah menjadi matrix rotasi untuk kemudian dikali dengan nilai percepatan dari sensor *accelerometer* yang sudah diproses sehingga memiliki satuan m/s^2.

3.5.1 Konversi Percepatan Menjadi Referensi Bumi

Sensor IMU pada buoy mendapatkan data percepatan, kecepatan sudut dan medan magnet pada referensi sensor itu sendiri (*body frame*). Sehingga untuk mengukur perpindahan perlu untuk merubah referensinya menjadi referensi bumi (*earth frame*). Konversi ini dilakukan dengan cara mengalikan data dengan matrix rotasi, data yang dikalikan adalah data percepatan (m/s^2) karena data percepatan ini yang digunakan secara langsung diintegralkan untuk mendapatkan nilai perpindahan [42]. Perhitungan konversi dilakukan berdasarkan Persamaan (3.1).

$$a_{(gs)}\begin{bmatrix}x\\y\\z\end{bmatrix} = a_{(bs)}\begin{bmatrix}x\\y\\z\end{bmatrix}.R_{(q)}\begin{bmatrix}q_0\\q_1\\q_2\\q_3\end{bmatrix} \tag{3.1}$$

Dimana $a_{(gs)}$ adalah nilai akselerasi pada referensi bumi, $a_{(bs)}$ adalah nilai percepatan pada referensi sensor itu sendiri dan $R_{(q)}$ adalah rotasi matrix dalam quaternion dengan q0 sebagai bilangan *real* dan q1, q2, q3 sebagai bilangan kompleks [29][42][43]. Nilai akselerasi baik dalam referensi bumi dan referensi sensor dalam bentuk matrix 3x1 sedangkan matrix rotasi dalam bentuk matrix 3x3. Matrix rotasi dapat dilihat pada Persamaan (3.2) berikut.

$$R_{(q)} = \begin{bmatrix} q_0^2+q_1^2-q_2^2-q_3^2 & 2(q_1q_2-q_0q_3) & 2(q_0q_2+q_1q_3) \\ 2(q_1q_2+q_0q_3) & q_0^2-q_1^2+q_2^2-q_3^2 & 2(q_2q_3-q_0q_1) \\ 2(q_1q_3-q_0q_2) & 2(q_0q_1+q_2q_3) & q_0^2-q_1^2-q_2^2+q_3^2 \end{bmatrix} \tag{3.2}$$

Matrix rotasi pada Persamaan (3.2) yang digunakan pada skripsi ini untuk merubah referensi alat menjadi referensi Bumi, perhitunggannya dilakukan pada aplikasi matlab dengan masukan data yang sudah di ambil sebelumnya.

3.5.2 Konversi Percepatan Menjadi Perpindahan

Data akselerasi 3 sumbu (x, y, z) dalam bentuk matrix 3x1 yang sudah diubah menjadi referensi bumi harus diintegral kan untuk mendapatkan data perpindahan. Integral pertama menghasilkan data kecepatan (m/s) dari sebelumnya (m/s^2). Data kecepatan harus diintegralkan kembali untuk menghasilkan data perpindahan (m). Dasar proses ini bisa dilakukan berdasarkan Persamaan (3.3) dan (3.4).

$$\triangle v = dT.a \tag{3.3}$$

$$\triangle s = dT.\triangle v \tag{3.4}$$

Lambang v adalah kecepatan dalam vektor, a adalah percepatan dalam bentuk vektor, s adalah perpindahan dalam bentuk vektor dan dT adalah *interval* waktu. Persamaan tidak bisa dilakukan secara *real time*, sedangkan penelitian ini membutuhkan data perpindahan secara *real time*, sehingga dipilihlah metode *trapezoidal integration*. Integral metode trapezoidal menggunakan data saat ini dan data pengukuran sebelumnya dalam perhitungannya [16]. Persamaan integral metode trapezoidal bisa dilihat pada Persamaan.

$$y(n) = y(n-1) + \frac{1}{2}.dT.[x(n-1) + x(n)], n = 1,2,3, \ldots \quad (3.5)$$

Pada Persamaan (3.5) y(n) adalah hasil dari integral, y(n-1) adalah hasil sebelumnya, dT adalah selisih waktu pengukuran dari sensor, x(n-1) adalah masukan sebelumnya dan x(n) adalah masukan baru atau masukan saat ini. Metode ini lebih akurat dari metode integral biasa pada Persamaan dan hanya perlu menyimpan satu data sebelumnya.

3.6 Perhitungan Gelombang Laut

Gelombang adalah gangguan dari media fluida melalui mana energi dipindahkan. Gelombang laut bergerak pada antarmuka antara lautan dan atmosfer. Gelombang disebabkan oleh gesekan antara angin dan permukaan air, gaya tarik gravitasi, gempa bumi atau letusan gunung berapi [4][17].

3.6.1 Tinggi Gelombang

Tinggi gelombang adalah perbedaan antara ketinggian permukaan maksimum dan minimum air dalam gelombang yang relevan [19]. Tinggi gelombang (H) dihitung menggunakan algoritma ZCD dengan data masukan adalah data perpindahan vertikan dari alat hasil integrasi ganda dari percepatan vertikal *earth frame*. Tinggi gelombang dihitung dari jarak tegak lurus puncak dan lembah gelombang.

3.6.2 Periode Gelombang

Periode gelombang adalah jarak antar dua puncak atau lembah gelombang. Periode gelombang (Tp) diperoleh dengan mengukur waktu antara puncak berurutan [4][17]. Perhitungan perioda gelombang dilakukan menggunakan algoritma ZCD.

BAB IV
HASIL DAN PEMBAHASAN

Bab ini berisi hasil pengujian dari dua jenis pengujian, pada saat kondisi statis dan kondisi dinamis. Kondisi statis merupakan kondisi ketika sensor tidak diberikan pergerakan dan kondisi dinamis ketika sensor diberi gerakan. Pengujian juga dilakukan dengan membandingkan pengukuran statis sebelum dan sesudah sensor dikalibrasi. Pelaksanaan pengujian yaitu dengan melakukan simulasi gerakan gelombang. Data dari sensor kemudian dimasukan kedalam matlab untuk menghitung nilai jarak vertikal dan horizontal. Contoh data lihat (Gambar 4.1)

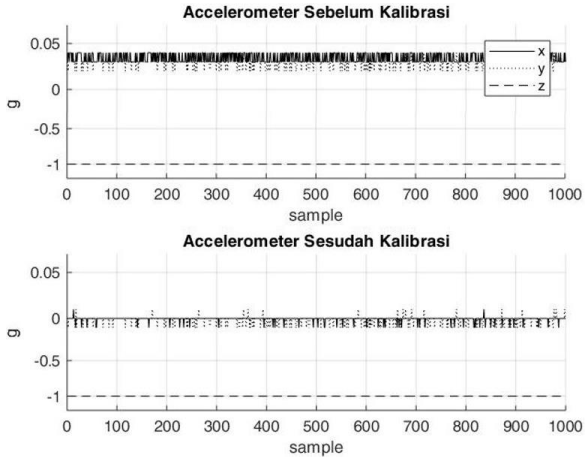
	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	t(s)	Ax	Ay	Az	Gx	Gy	Gz	Mx	My	Mz	q0	q1	q2	q3
2	0.000	0.02	-0.03	1	0	0	0.01	0.79	-0.39	-0.48	0.52	-0.05	0.01	-0.85
3	0.031	0.02	-0.03	1	0	0	0.01	0.81	-0.4	-0.42	0.52	-0.06	0	-0.85
4	0.063	0.02	-0.03	1	0	0	0.01	0.77	-0.39	-0.5	0.51	-0.05	0.01	-0.86
5	0.094	0.02	-0.03	1	0	0	0.01	0.79	-0.41	-0.46	0.51	-0.05	0	-0.86
6	0.125	0.02	-0.03	1	0	0	0	0.79	-0.39	-0.46	0.52	-0.05	0.01	-0.86
7	0.156	0.02	-0.03	1	0	0	0	0.8	-0.38	-0.47	0.52	-0.05	0.01	-0.85
8	0.188	0.02	-0.03	1	0	0	0.01	0.77	-0.43	-0.48	0.5	-0.05	0	-0.86
9	0.219	0.02	-0.03	1	0	0	0.01	0.8	-0.39	-0.46	0.52	-0.05	0	-0.85
10	0.250	0.02	-0.03	1	0.01	0	0	0.8	-0.37	-0.47	0.52	-0.05	0	-0.85
11	0.281	0.02	-0.03	1	0	0	0.01	0.79	-0.4	-0.47	0.52	-0.05	0	-0.86
12	0.313	0.02	-0.03	1	0	0	0.01	0.8	-0.4	-0.45	0.52	-0.05	0.01	-0.85
13	0.344	0.02	-0.03	1	0	0	0.01	0.77	-0.42	-0.48	0.51	-0.05	0.01	-0.86
14	0.375	0.02	-0.03	1	0	0	0	0.78	-0.41	-0.47	0.51	-0.05	0.01	-0.86
15	0.406	0.02	-0.03	1	0	0	0.01	0.78	-0.41	-0.48	0.51	-0.06	0	-0.86
16	0.438	0.02	-0.03	1	0	0	0.01	0.78	-0.41	-0.47	0.51	-0.05	0.01	-0.86
17	0.469	0.02	-0.03	1	0	0	0	0.79	-0.42	-0.45	0.51	-0.06	0	-0.86
18	0.500	0.02	-0.03	1	0	0	0.01	0.78	-0.41	-0.47	0.51	-0.05	0.01	-0.86
19	0.531	0.02	-0.03	1	0	0	0	0.78	-0.41	-0.48	0.51	-0.05	0	-0.86
20	0.563	0.02	-0.03	1	0	0	0.01	0.8	-0.37	-0.47	0.52	-0.05	0	-0.85
21	0.594	0.02	-0.03	1	0	0	0.01	0.77	-0.41	-0.49	0.51	-0.05	0	-0.86
22	0.625	0.02	-0.03	1	0	0	0.01	0.79	-0.41	-0.46	0.51	-0.05	0	-0.86
23	0.656	0.02	-0.03	1	0	0	0.01	0.82	-0.37	-0.44	0.53	-0.05	0	-0.85
24	0.688	0.02	-0.03	1	0	0	0	0.76	-0.42	-0.49	0.5	-0.05	0	-0.86
25	0.719	0.02	-0.03	1	0	0	0.01	0.8	-0.37	-0.47	0.52	-0.05	0.01	-0.85
26	0.750	0.02	-0.03	1	0	0	0.01	0.77	-0.43	-0.48	0.5	-0.05	0.01	-0.86
27	0.781	0.02	-0.03	1	0	0	0.01	0.78	-0.4	-0.48	0.51	-0.05	0.01	-0.86
28	0.813	0.02	-0.03	1	0	0	0	0.8	-0.36	-0.48	0.53	-0.05	0	-0.85

Gambar 4. 1 Contoh Data Keluaran Arduino

Data yang kirimkan oleh ardino ke laptop sebanyak 14 jenis data, bisa dilihat pada (Gambar 4.1). MPU9250 membaca dan mengirimkan 9 data mentah dari tiga sensor dengan masing-masing sensor mengirimkan tiga data. Data-data itu lalu diproses oleh Arduino Mega 2560 berdasarkan *datasheet* dari MPU9250 sehingga mengasilkan data percepatan dalam G (*gravity*), data kecepatan sudut dalam dps (*degree per second*) dan data medan magnet dalam mG (*milli Gauss*). Data yang lain didapatkan dari hasil pengolahan data dari MPU 9250 dan juga dari arduino itu sendiri. Semua data 9 dof yang diperoleh dimasukan sebagai masukan dari madgwick filter, hasil dari filter ini berupa 4 data quaternion, yaitu q0, q1, q2 dan q3. Data terakhir adalah data selisih watu dalam detik pengiriman antar kesulurah data, data *delay* ini berasal dari Arduino itu sendiri melalui fungsi *millis()* yang sudah tersedia.

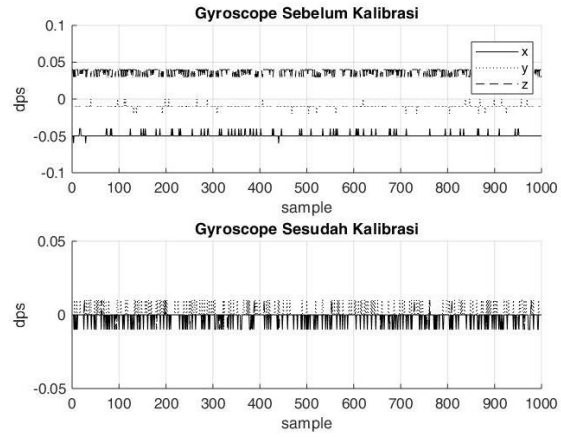
4.1 Hasil Kalibrasi

Pengambilan *sample* pada percobaan ini dilakukan dalam dua keadaan, yaitu tanpa dan dengan mengurangi nilai *offset* dari sensor. Data untuk *accelerometer* dan *gyroscope* diambil saat sensor dalam keadaan diam, sedangkan sensor *magnetometer* mengambil data dengan menggerakan sensor secara tiga dimensi. Tujuan dari pengujian ini untuk melihat perbedaan data sebelum dan sesudah dikalibrasi. Hasil pengujian *accelerometer* dapat dilihat pada (Gambar 4.2).



Gambar 4. 2 Pengujian Kalibrasi Accelerometer

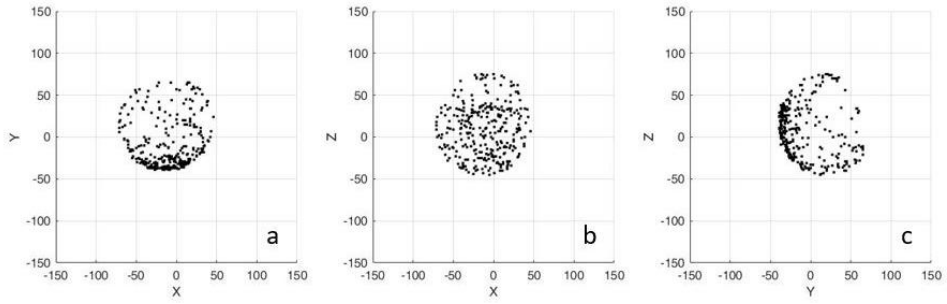
Perbandingan data *accelerometer* pada (Gambar 4.2) menunjukan bahwa saat sensor belum dikurang dengan *offset*, data untuk sumbu X dan Y tidak berada pada titik nol, setelah dikurangi dengan *offset* data yang didapatkan terpusat pada nilai nol. Sumbu Z baik sebelum dan setelah dikurangi dengan *offset* selalu mengukur percepatan satu. Nilai sumbu Z juga sebenarnya merupakan *offset* karena itu adalah nilai percepatan dari gravitasi. Pengujian status *gyroscope* lihat (Gambar 4.3).



Gambar 4. 3 Pengujian Kalibrasi *Gyroscope*

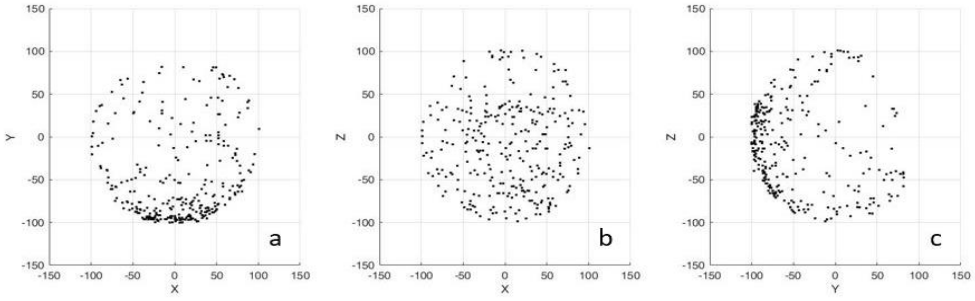
Sensor *gyroscope* juga memiliki data yang tidak sama dengan nol sebelum dikurangi dengan nilai *offset*. Setelah kalibrasi data yang didapat untuk ketiga sumbunya terpusat pada nilai nol. Nilai kesalahan dari kedua sensor ini adalah *drift* yang disebabkan oleh *DC bias* pada sinyal *acceleration*. *Drift* pada data percepatan menyebabkan nilai kesalahan yang besar saat data diintegralkan. Kalibrasi ini dilakukan untuk mengurangi nilai kesalahan pada sensor *accelerometer* dan *gyroscope*. Sehingga, saat integrasi dilakukan, nilai kesalahan yang dihasilkan menjadi kecil. Grafik yang dihasilkan pada (Gambar 4.2) dan (Gambar 4.3) masih memiliki riak setelah kalibrasi, ini adalah kesalahan yang gagal dihilangkan.

Data dari *magnetometer* sebenarnya susah untuk direpresentasikan, hal ini karena kalibrasi dari sensor ini berbeda dari *accelerometer* dan *gyroscope* yang nilainya harus terpusat pada nol terhadap waktu saat sensor diam. *Magnetometer* dikalibrasi sehingga menghasilkan data yang membentuk bola pada 2 sumbu sensor. Kalibrasi juga dilakukan dengan menggerakan sensor secara 3D. Data sebelum kalibrasi bisa dilihat pada (Gambar 4.4).



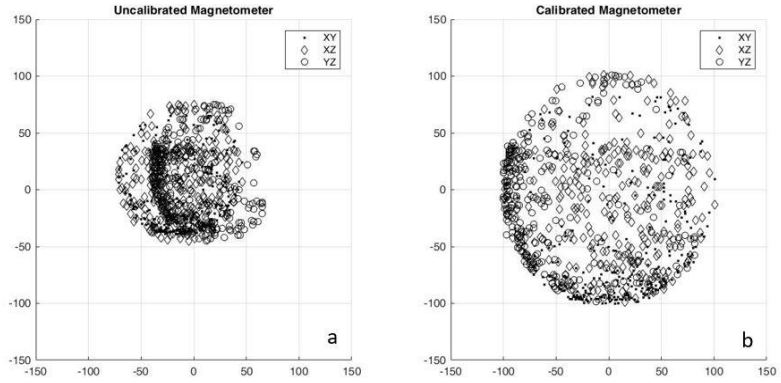
Gambar 4. 4 Magnetometer Sebelum Kalibrasi (a)XY, (b)XZ dan (c)YZ

Data mentah dari *magnetometer* ketika sensor digerakan secara 3D terpresentasikan dengan grafik 2D untuk setiap 2 kemungkinan sumbu. Grafik pada (Gambar 4.4) (a) untuk sumbu X dan Y, Grafik pada (Gambar 4.4) (b) untuk sumbu X dan Z, terakhir Grafik pada (Gambar 4.4) (c) untuk sumbu Y dan Z. Terlihat bahwa data medan megnet yang sudah dikonversi menjadi berbentuk lingkaran tidak terpusat ditengah bidang 2D untuk semua kemungkinan sumbu, ini disebabkan oleh *hard iron losses*. Lingkaran data juga tidak memiliki bentuk lingkaran yang sempurna melainkan sedikit berbentuk *oval*, untuk masalah ini penyebabnya adalah *soft iron losses*. Hasil kalibrasi magngetometer bisa dilihat pada (Gambar 4.5).



Gambar 4. 5 Magnetometer Sesudah Kalibrasi (a)XY, (b)XZ dan (c)YZ

Setelah melakukan kalibrasi dengan metode *Li's ellipsoid specific fitting algorithm,* data mentah dari *magnetometer* pada semua kemungkinan sumbu seperti yang terlihat pada (Gambar 4.5) (a) untuk sumbu X dan Y, (Gambar 4.5) (b) untuk sumbu X dan Z, (Gambar 4.5) (c) untuk sumbu Y dan Z hampir terpusat pada titik nol dan memiliki bentuk lingkaran yang lebih proporsional dibandingkan dengan grafik sebelum kalibrasi, lihat (Gambar 4.4), namun data yang sudah dikoreksi tetap memiliki bentuk lingkaran yang belum sempurna dan belum sepenuhnya terpusat pada titik nol, hal ini bisa lebih terlihat lebih jelas apabila visualisasi data digabungkan untuk semua kemungkinan sumbu, bisa dilihat pada (Gambar 4.6).

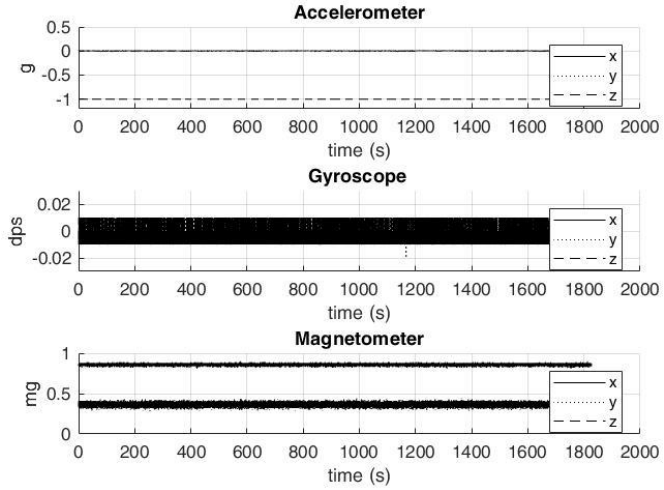


Gambar 4. 6 Kalibrasi Magnetometer (a)*Uncalibrated* (b)*Calibrated*

Gabungan nilai mentah *magnetometer* sebelum (Gambar 4.6) (a) dan sesudah kalibrasi lihat (Gambar 4.6) (b), nilai setiap sumbu digabung untuk melihat detail yang lebih jelas. Data mentah hasil kalibrasi masih memiliki nilai *offset,* ini terlihat dari bentuk lingkaran yang tidak sempurna, sehingga proses kalibrasi untuk sensor *magnetometer* tidak memenuhi kebutuhan yang diinginkan, karena *offset* tidak sepenuhnya hilang.

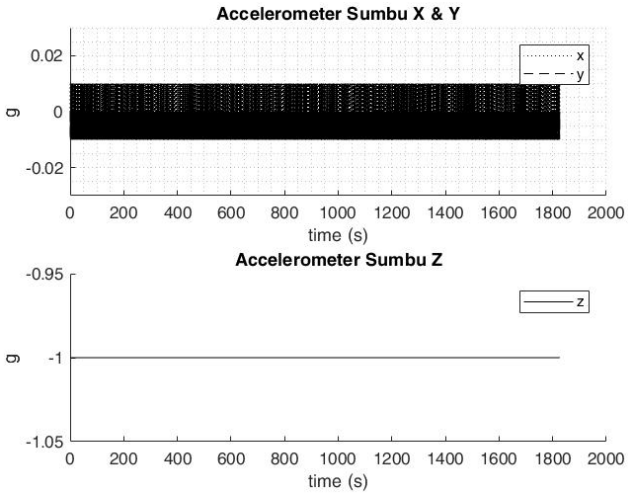
4.2 Pengukuran Statis

Pengambilan *sample* data pada percobaan ini dilakukan pada saat sensor dalam kondisi diam dengan frekuensi 30 Hz, pengambilan *sample* berlangsung selama 30 menit. Pengujian ini dilakukan setelah dilakukan kalibrasi pada semua sensor. Pengujian ini bertujuan untuk melihat apakan data dari sensor stabil atau tidak jika diambil dalam waktu yang lama, tujuan lainnya yaitu untuk melihat apakah *drift* dari pengukuran konstan atau tidak. Berikut adalah data mentah hasil pengukuran saat kondisi statis/diam (Gambar 4.7).



Gambar 4. 7 Data Statis

Sensor *gyroscope* berdasarkan (Gambar 4.7) menghasilkan nilai yang terpusat pada titik nol untuk semua sumbunya. Hal ini dikarenakan sensor *gyroscope* tidak mengukur gaya dari luar sensor itu sendiri seperti gravitasi. Sehingga apabila sensor tidak mendapatkan gerakan maka data yang didapat harus bernilai nol, jika tidak maka nilai itu adalah kesalahan atau *offset* dari sensor tersebut. Data yang diukur oleh sensor *gyroscope* masih memiliki *drift* yang belum dihilangkan saat kalibrasi. Data dari *magnetometer* adalah besar medan magnet yang berasal dari bumi bukan dari sensor itu sendiri, sehingga penjelasan untuk data *magnetometer* mengacu pada sub bab 4.1 yaitu pada (Gambar 4.5) dan (Gambar 4.6). Data percepatan pada (Gambar 4.7) terlihat bahwa sumbu X dan Y selama 30 menit berada dalam titik nol, sedangkan sumbu Z bernilai negatif satu, namun grafik sebenarnya memiliki riak terutama untuk sumbu X dan Y, untuk lebih jelasnya lihat (Gambar 4.8).



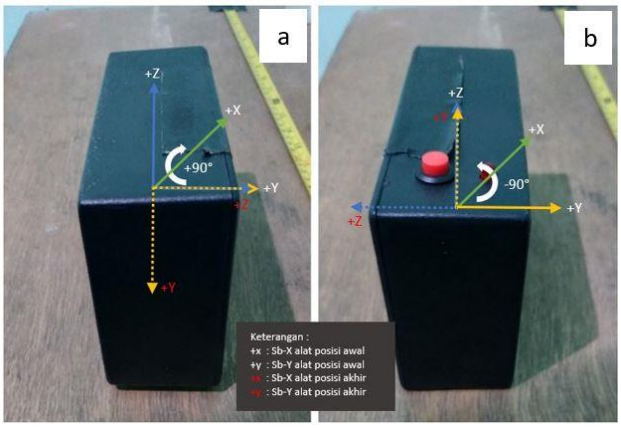
Gambar 4. 8 Data Statis *Accelerometer*

Grafik pada (Gambar 4.8) adalah data percepatan dari sensor *accelerometer* untuk semua sumbu. Data dari *accelerometer* untuk sumbu X dan sumbu Y memiliki nilai nol yang tidak stabil, sedangkan sumbu Z memiliki nilai -1 yang stabil, hal ini dikarenakan sumbu Z pada sensor *accelerometer* mengukur nilai gravitasi bumi, dimana gravitasi sendiri bernilai 1 G atau 9.81 m/s^2, nilai gravitasi yang terukur menjadi negatif karena pada program nilai sumbu Z dikalikan -1.

Dapat dilihat pada (Gambar 4.8), data grafik untuk sumbu X dan Y pada sensor *accelerometer* masih memiliki riak, dimana ini berarti nilai pengukuran pada kondisi diam tidak sepenuhnya benilai 0, Hal ini disebabkan kalibrasi sensor yang tidak sempurna sehingga masih menyisakan *drift*, dimana riak yang disebabkan oleh *drift* ini berasal dari mekanikal sensor itu sendiri.

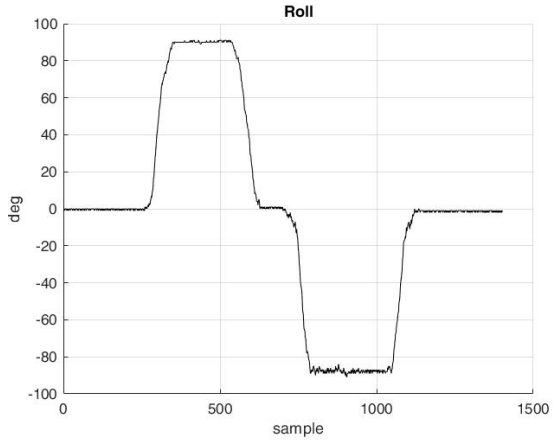
4.3 Pengujian Orientasi

Pengujian kali ini dilakukan untuk mengukur akurasi orientasi sensor dalam tiga dimensi. Pengujian dengan menggerakan sensor 90^0 mengikuti orientasi gerak *roll*, *pitch* dan *yaw* secara manual menggunakan tangan. Data diambil dengan frekuensi 30 Hz. Pengujian ini dilakukan setelah data mentah dari semua sensor diproses melalui madgwick filter untuk menghasilkan nilai quaternion yang kemudian dari nilai quaternion itu dikonversi menjadi nilai sudut euler. Pengujian orientasi *roll* lihat (Gambar 4.9).



Gambar 4. 9 Orientasi Orientasi *Roll* (a)90^0 dan (b)-90^0

Orientasi *roll* dilakukan dengan memutar sensor 90^0 terhadap sumbu X, dimana sumbu X tetap diam dan yang berputar adalah sumbu Y dan Z. Sensor diputar 90^0 ke kanan (Gambar 4.8) (a), diam sejenak dan kemudian kembali ke posisi awal lalu diputar -90^0 ke kiri (Gambar 4.9) (b), dan terakhir sensor dikembalikan ke posisi semula. Data yang diperoleh sensor untuk orientasi sudut *roll* bisa dilihat pada (Gambar 4.10). Data untuk sudut *roll* diproses langsung oleh arduino, data lalu disimpan di *micro sd card* yang kemudian dimasukan kedalam aplikasi matlab untuk ditampilkan dalam bentuk grafik.

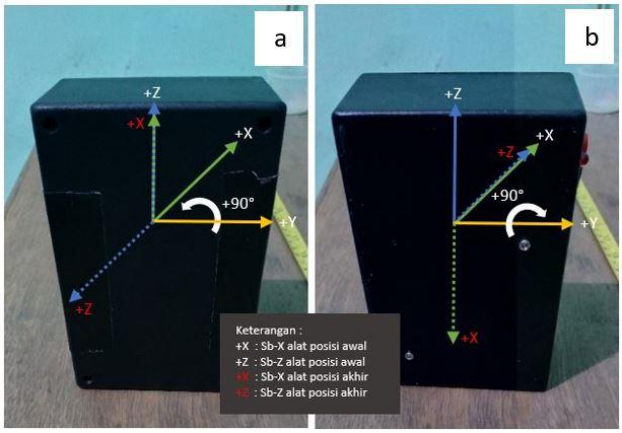


Gambar 4. 10 Data Pengujian Orientasi *Roll*

Grafik pada (Gambar 4.10) adalah data hasil dari pengujian sudut roll. Berdasarkan (Gambar 4.10) terlihat bahwa data untuk gerakan *roll* mengikuti dengan baik pergerakan sensor. Nilai yang dibaca oleh sensor juga kurang lebih 90° untuk sudut positif dan negatif. Data keluaran ketika membaca gerakan euler pada

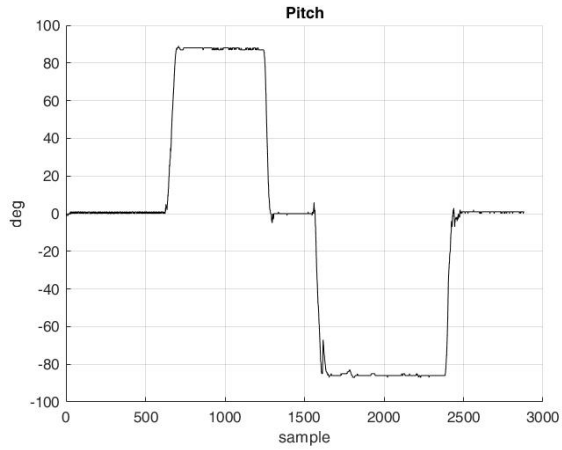
hal ini sudut *roll* masih memiliki nilai kesalahan, terlihat pada (Gambar 4.10) nilai yang dihasilkan masih memiliki riak untuk semua data yang diambil baik saat sensor sedang diam juga saat sedang bergerak, terutama ketika sensor dalam keadaan 90°, dimana riak pada grafik terlihat sangat jelas. Hal ini seharusnya tidak terjadi, data sudut seharusnya stabil mengeluarkan nilai 90°, karena saat dalam sudut 90° sensor dalam keadaan diam.

Kesalahan ini dihasilkan oleh gabungan *drift* dari *accelerometer*, *gyroscope* dan *magnetometer*, *drift* yang terbaca sebenarnya sudah difilter dengan algoritma madgwick dan juga saat melakukan kalibrasi sensor, namun kedua perlakuan itu tetap tidak bisa sepenuhnya menghilangkan *drift* dari sensor. Nilai dari *drift* ini terakumulasi saat nilai quaternion hasil dari filter madgwick dirubah menjadi nilai sudut *roll*. Nilai *drift* inilah yang membuat nilai pengukuran menjadi tidak stabil meski tidak diberikan gerakan pada alat. Data untuk sudut *roll* terutama nilai kesalahan yang terhitung terlihat lebih jelas pada pengujian selanjutnya, yaitu pengujian untuk sudut *pitch*, lihat (Gambar 4.11).



Gambar 4. 11 Pengujian Orientasi *Pitch* (a)90^0 dan (b)-90^0

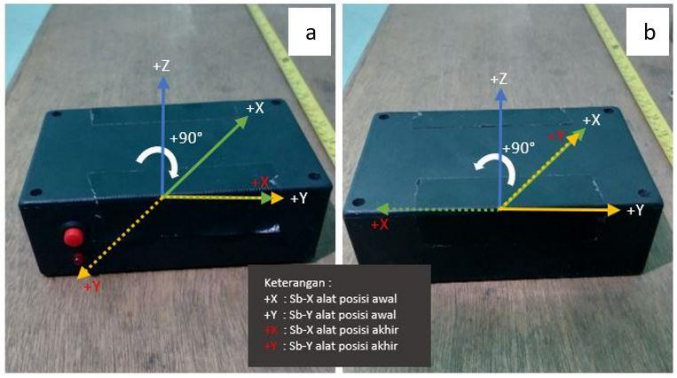
Pengujian gerakan *pitch* dilakukan dengan cara memutar sensor 90^0 terhadap sumbu Y, yaitu sumbu Y diam sedangkan sumbu X dan Z bergerak dari posisi awal. Pada (Gambar 4.11) (a) alat diputar 90^0 ke atas, lalu diam sesaat dan dikembalikan ke posisi semula, kemudian diputar kembali -90^0 ke bawah (Gambar 4.11) (b), terakhir sensor dikembalikan pada kedudukan semula. Pengujian ini dilakukan secara manual menggunakan tangan dan bantuan alat ukur sudut penggaris bujur. Hasil dalam bentuk grafik lihat (Gambar 4.12).



Gambar 4. 12 Data Pengujian Orientasi *Pitch*

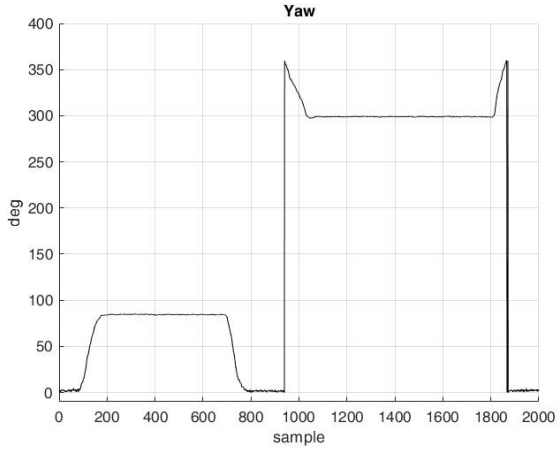
Berdasarkan (Gambar 4.12) terlihat bahwa data untuk gerakan *pitch* mengikuti dengan baik sesuai pergerakan sensor. Sensor mengambil data sudut kurang lebih 90° untuk setiap gerakan yang dilakukan oleh alat. Baik sudut positif dan juga sudut negatif berhasil diperoleh oleh sensor. Terlihat juga dari (Gambar 4.12) grafik *pitch* memiliki riak.

Sensor *fusion* dilakukan menggunakan algoritma filter madgwick untuk bisa menghasilkan sudut euler. Pada saat data dimasukan ke dalam algoritma madgwick, data di-*filter* untuk mengurangi *drift* yang masih belum bisa dihilangkan saat melakukan kalibrasi, lihat (Gambar 4.6). Hasil data dari madgwick tetap memiliki *drift*, ini terlihat dari riak pada data (Gambar 4.12) dimana riak menandakan bahwa data keluaran untuk sudut *pitch* tidak stabil. Pada saat sensor diam seharusnya data konstan atau stabil kecuali saat sensor digerakan. Selanjutnya dilakukan pengujian orientasi *yaw*, lihat (Gambar 4.13).



Gambar 4. 13 Pengujian Orientasi *Yaw* (a)90^0 dan (b)-90^0

Pengujian sudut *yaw* dilakukan dengan memutar alat terhadap sumbu Z. Alat digerakan 90^0 ke arah positif (Gambar 4.13) (a) dan 90° negatif (Gambar 4.13) (b). Sama seperti gerakan pada sudut *roll* dan *pitch*, pengujian kali ini juga dilakukan menggunakan tangan. Hasil pengujian gerakan euler yaw bisa dilihat pada (Gambar 4.14) di bawah ini.



Gambar 4. 14 Data Pengujian Sudut *Yaw*

Hasil pengungujian sudut *yaw* berdasarkan (Gambar 4.14), terlihat jelas perbedaan jika dibandingkan dengan data dari *roll* pada (Gambar 4.10) dan data *pitch* pada (Gambar 4.12). Sudut *yaw* tidak memiliki nilai negatif, hal ini dikarenakan sudut *yaw* difungsikan sama seperti kompas sehingga hanya memiliki besaran nilai sudut 0° sampai 360°. Pengukuran sudut *yaw* ini juga mendapatkan peran yang besar dari sensor *magnetometer* yang mengukur medan magnet bumi, sehingga nilai 0° dari *yaw* didapatkan bukan ketika sensor sedang diam, melainkan saat sumbu +X dari sensor *magnetometer* menghadap ke arah utara bumi. Saat sensor digerakan 90° derajat kekanan (X menghadap arah timur) data pada grafik terlihat berada kurang lebih pada nilai 90°, sedangkan saat diputar 90° ke kiri (X menghadap arah barat) data yang didapatkan adalah 300°. Hal ini tentu saja tidak benar karena data yang dihasilkan seharusnya adalah 280°.

Merujuk pada data kalibrasi *magnetometer* pada (Gambar 4.4), (Gambar 4.5) dan (Gambar 4.6) diketahui bahwa sensor *magnetometer* belum terkalibrasi dengan baik, sehingga nilai kesalahan yang dihasilkan *magnetometer* terlihat jelas pada pengujian sudut *yaw* ini. Sedangkan saat pengujian *roll* dan *pitch*, peran dari sensor *magnetometer* tidak terlihat pada data grafik pada (Gambar 4.12) dan (Gambar

4.10), hal ini karena data grafik hanya memperlihatkan data perubahan sudut yang dihasilkan sedangkan orientasi sensor terhadap bumi tidak terlihat. Pada paragraf sebelumnya dijelaskan bahwa *yaw* berfungsi sebagai kompas, maksudnya adalah nilai dari *yaw* yang mempengaruhi kedudukan alat terhadap bumi.

Pengujian orientasi sudut *roll* dan *pitch* dilakukan lebih lanjut, dimana sudut ini diuji dengan nilai -20°, -40°, -60°, 20°, 40° dan 60°. Hal ini dilakukan karena pengukuran gelombang terpengaruh oleh kedua sudut orientasi ini, sedangkan sudut *yaw* tidak diuji lebih lanjut karena tidak berpengaruh besar terhadap nilai posisi yang dihitung, yaw mempengaruhi jika menghitung arah gerakan sensor atau arah gelombang. Namun, parameter arah gelombang tidak menjadi parameter gelombang yang di uji pada penilitian ini. Hasil pengujian kemiringan sensor masing-masing untuk *roll* dan *pitch* bisa dilihat pada (Tabel 4.1).

Tabel 4. 1 Data Pengujian Sudut *Roll dan Pitch* (°)

No	**Pengujian Sudut *Roll* (°)**						**Kesalahan Total**
	-20	-40	-60	20	40	60	
1	-20,80	-40,65	-61,70	18,93	39,27	60,38	
2	-21,17	-40,85	-60,47	19,01	39,12	60,50	
3	-20,86	-41,54	-59,91	20,14	40,03	60,74	
4	-20,56	-40,12	-60,30	20,04	40,05	61,45	
5	-21,23	-39,89	-60,33	20,63	39,38	61,34	
Kesalahan Rata-rata	**0,92**	**0,61**	**0,54**	**0,25**	**0,43**	**0,88**	**3,64**
No	**Pengujian Sudut *Pitch* (°)**						**Kesalahan Total**
	-20	-40	-60	20	40	60	
1	-21,75	-41,36	-62,19	20,41	41,04	61,43	
2	-21,26	-40,52	-61,81	20,73	41,89	62,25	
3	-19,41	-39,74	-60,41	19,96	40,35	60,76	
4	-20,42	-40,16	-61,64	20,68	40,87	61,14	
5	-20,95	-41,16	-63,47	20,63	40,92	61,17	
Kesalahan Rata-rata	**0,76**	**0,59**	**1,90**	**0,48**	**1,01**	**1,35**	**6,10**

Hasil pengujian kemiringan sensor lihat (Tabel 4.1), pengujian dilakukan sebanyak lima kali untuk setiap masing-masing besar sudut yang diuji. Kesalahan paling rendah dari pengujian ini adalah 0.25°. Kesalahan tertinggi pada pengujian ini adalah 0.92°. Total kesalahan sudut *roll* untuk semua pengujian adalah 3.64°. Data pengujian kemiringan sensor pada sudut *pitch*, lihat (Tabel 4.1). Kesalahan paling rendah dari pengujian ini adalah 0.48°. Nilai kesalahan tertingi pada pengujian ini adalah 1.9°. Total kesalahan sudut *pitch* adalah 6.10°.

Adanya selisih pengukuran ini menandakan bahwa *drift* ketiga sensor yang digunakan tidak sepenuhnya bisa dihilangkan. Nilai Kesalahan ini sangat mempengaruhi saat pengujian perpindahan dilakukan, karena nilai kesalahan ini teraukumulasi saat dilakukan integral sebanyak dua kali, dan terus bertambah dengan *drift* dari *accelerometer*.

4.4 Pengujian Perpindahan

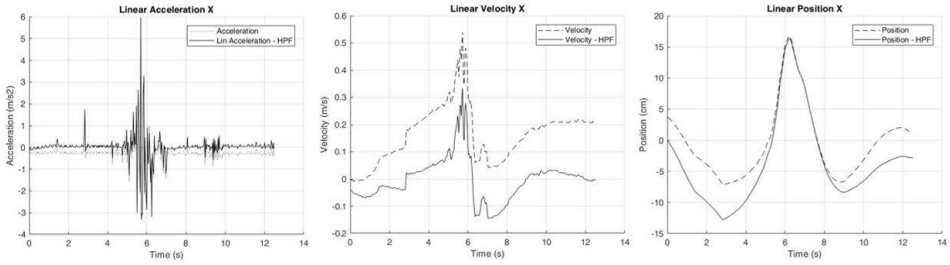
Pengujian ini dilakukan dengan menggerakan alat ke satu arah sumbu saja, setelah itu dihitung nilai perpindahannya. Alat digerakan sepanjang 30 cm untuk masing-masing arah sumbu dengan menggunakan *roll meter* sebagai perbandingan. Semua pengujian untuk setiap sensor dilakukan sebanyak 10 kali. Alat digerakan menggunakan tangan pada bidang datar, waktu yang ditempuh alat untuk jarak 30 cm tidak ditentukan dan tidak dihitung. Data keluaran dengan frekuensi 30 Hz.

Tabel 4. 2 Pengujian Perpindahan Sb-X

No	**Sensor (cm)**	***Roll Meter* (cm)**	**Kesalahan (cm)**
1	16,74	30	13,26
2	23,36	30	6,64
3	16,23	30	13,77
4	19,13	30	10,87
5	19,42	30	10,58
6	21,45	30	8,5
7	14,83	30	15,17
8	17,49	30	12,51
9	18,34	30	11,66
10	18,56	30	11,44
RMSE			11.68%

Berdasarkan (Tabel 4.3) dapat terlihat bahwa hasil pengujian alat memiliki nilai selisih yang sangat besar dibandingkan dengan nilai dari *roll meter*. Dimana dari 10 pengujian nilai selisih paling kecil adalah 8.55 cm dan nilai selisih terbesar mencapai 15.17 cm. Nilai kesalahan yang besar pada sumbu X ini dipengaruhi oleh *drift* gabungan dari *accelerometer*, *gyroscope* dan juga *magnetometer*. Nilai *Drift* pada sumbu x juga berasal dari mekanikal sensor *accelerometer* itu sendiri, setelah itu dipengaruhi juga oleh sensor yang tidak bergerak lurus sempurna, melainkan sumbu X pada sensor *accelerometer* miring dengan nilai kemiringan yang tidak

diketahui. Menggerakan sensor menggunakan tangan juga tentu memberikan nilai *drift* dari ketidakstabilan gerakan, kesalahan bisa lebih jelas terlihat dengan melihat grafik pada (Gambar 4.15).



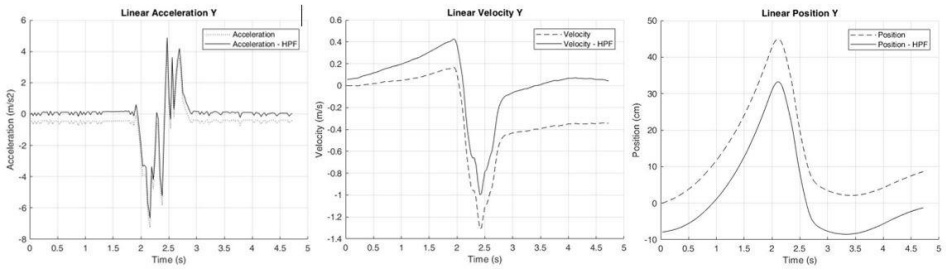
Gambar 4. 15 Grafik Pengujian Sumbu X (a)Percepatan, (b)Kecepatan dan (c) Posisi

Grafik pada (Gambar 4.15) terlihat perubahan data untuk setiap hasil integral, juga terlihat perbedaan data sebelum dan sesudah diberikan filter HPF. Data percepatan pada (Gambar 4.15) (a) sebelum diberikan HPF memiliki data awal dan akhir tidak sama dengan nol, data saat alat diam yaitu pada detik 0-4 dan 8-12 nilai yang terbaca oleh alat kurang dari nol, dimana seharusnya nilai adalah nol. Setelah diberikan HPF maka data menjadi terpusat pada nol. Grafik pada (Gambar 4.15) (b) adalah data kecepatan hasil integral dari percepatan, data kecepatan memiliki nilai awal dan akhir tidak sama dengan nol, sedangkan data percepatan memiliki nilai awal dan akhir berada pada titik nol, setelah diberikan HPF barulah data ditarik ke titik nol. Data posisi pada (Gambar 4.15) (c) juga memiliki hasil yang sama dengan percepatan dan kecepatan saat diberikan HPF. Pengujian perpindahan pada sumbu Y bisa dilihat pada (Tabel 4.4).

Tabel 4. 3 Pengujian Perpindahan Sumbu Y

No	**Sensor (cm)**	***Roll Meter* (cm)**	**Kesalahan (cm)**
1	26,85	30	3,15
2	28,65	30	1,35
3	31,84	30	1,84
4	34,44	30	4,44
5	28,80	30	1,2
6	33,38	30	3,38
7	27,37	30	2,63
8	35,85	30	5,85
9	26,52	30	3,48
10	28,94	30	1,06
RMSE			3,19%

Hasil pengujian pada (Tabel 4.4) dapat terlihat bahwa hasil yang diukur alat memiliki nilai selisih yang cukup besar dengan nilai dari *roll meter*, namun jika dibandingkan dengan pengukuran pada sumbu X maka selisih nilai dari sumbu Y relatif kecil. Dimana dari 10 pengujian nilai selisih paling kecil adalah 1.06 cm dan nilai selisih terbesar mencapai 5.85 cm. Data grafik pengujian sumbu Y dapat dilihat pada (Gambar 4.16)



Gambar 4. 16 Grafik Pengujian Sumbu Y (a)Percepatan, (b)Kecepatan dan (c) Posisi

Grafik pada (Gambar 4.16) (a) adalah data percepatan, terlihat bahwa data awalnya tidak terpusat dititik nol hingga diberikan HPF barulah data terpusat dititik nol. Grafik pada (Gambar 4.16) (b) adalah data kecepatan, HPF membuat data akhir kembali ketitik nol dari sebelumnya bernilai minus. Grafik pada (Gambar 4.16) (c) adalah data posisi, pada data ini HPF dapat menarik data akhir menjadi nol, tetapi data awal gagal untuk dijadikan nol.

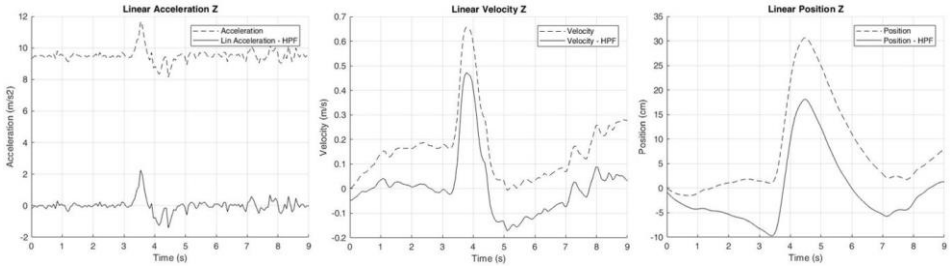
Nilai kesalahan yang pada sumbu Y ini dipengaruhi oleh *drift* dari *accelerometer, gyroscope* dan juga *magnetometer*. *Drift* pada sumbu Y berasal dari mekanikal sensor *accelerometer* itu sendiri, setelah itu dipengaruhi juga oleh sensor yang tidak bergarak lurus, dikarenakan sumbu Y pada *accelerometer* miring dengan nilai kemiringan yang tidak diketahui. Menggerakan sensor menggunakan tangan juga menambah *drift* dari ketidakstabilan gerakan. Berdasarkan perbandingan (Tabel 4.3) dengan (Tabel 4.4) maka akurasi sumbu Y lebih baik dari sumbu X. Hasil pengujian sumbu Z lihat (Tabel 4.4).

Tabel 4. 4 Pengujian Perpindahan Sumbu Z

No	**Sensor (cm)**	***Roll Meter* (cm)**	**Kesalahan (cm)**
1	10,85	30	19,15
2	8,92	30	21,08
3	12,87	30	17,13

4	13,56	30	16,44
5	13,9	30	16,1
6	13,62	30	16,38
7	18,10	30	11,9
8	13,48	30	16,52
9	12,34	30	17,66
10	15,9	30	14,1
RMSE			16,82%

Berdasarkan (Tabel 4.5) dapat terlihat bahwa hasil pengujian alat memiliki nilai selisih yang sangat besar dengan nilai dari *roll meter*, bahkan ssangat besar dibandingkan dengan sumbu Y pada (Tabel 4.4). Dimana dari 10 pengujian nilai selisih paling kecil adalah 11.9 cm dan nilai selisih terbesar mencapai 21.08 cm. Nilai kesalahan yang besar pada sumbu Z ini dipengaruhi oleh *drift* dari *accelerometer*, *gyroscope* dan juga *magnetometer*. Grafik pengujian ini bisa dilihat pada (Gambar 4.17).



Gambar 4. 17 Grafik Pengujian Sumbu Z (a)Percepatan, (b)Kecepatan dan (c) Posisi

Data percepatan pada (Gambar 4.17) (a) sebelum diberikan HPF hampir 10 m/s^2, ini adalah nilai gravitasi yang besarnya adalah 9.81 $m/s^{2,}$ setelah diberikan HPF dan dikurangi nilai gravitasi data menjadi terpusat pada nilai nol. Grafik pada (Gambar 4.17) (b) adalah data kecepatan hasil integral dari percepatan, HPF berhasil menarik data kecepatan awal dan akhir ke titik nol. Data posisi pada (Gambar 4.17) (c) juga diberikan HPF dan HPF berhasil menarik data awal dan akhir menjadi nol. Nilai kesalahan yang pada sumbu Z ini disebabkan oleh hal yang sama pada sumbu X dan Y, yaitu *drift*, kesalahan akibat pengujian dengan tangan dan penempatan sensor pada wadah yang tidak tegak lurus.

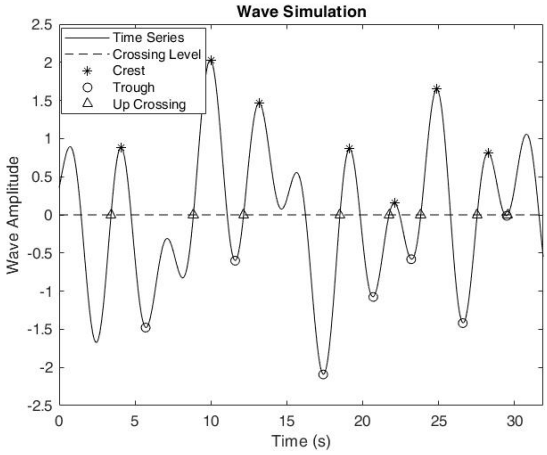
Adanya *Drift* pada sumbu Z berasal dari mekanikal sensor *accelerometer* itu sendiri, setelah itu dipengaruhi juga oleh sensor yang tidak bergarak lurus, melainkan sumbu X pada sensor *accelerometer* miring dengan nilai kemiringan

yang tidak diketahui. Menggerakan sensor menggunakan tangan juga memberikan nilai *drift* dari ketidakstabilan gerakan. *Drift* terbesar dari sumbu Z ini adalah gravitasi, dimana nilai gravitasi yang terbaca oleh sensor mempengaruhi pembacaan nilai pergerakan sensor itu sendiri. Nilai gravitasi bumi adalah 9.81 m/s^2 yang mana itu sangatlah besar dibandingkan nilai yang terbaca oleh sensor, sehingga nilai gravitasi bisa menutupi nilai pembacaan yang kecil.

Nilai *drift* dari tiga sensor yang tidak sepenuhnya hilang saat kalibrasi terus bertambah ketika sensor bergerak dan menjadi berkali lipat saat di integralkan. *High pass filter* yang diberikan terhadap nilai percepatan sebelum dilakukan integral juga tidak sepenuhnya bisa menghilangkan *drift*. Data percepatan yang masih memiliki *drift* juga dikalikan dengan matrix rotasi yang didapatkan dari quaternion hasil dari filter madgwick. Jadi *drift* dari ketiga sensor yang terbaca terus terakumulasi oleh pembacaan sensor saat pengujian, saat melakukan filter madgwick dan juga tentu saat dilakukan integral sebanyak dua kali.

4.5 Pengujian Deteksi Parameter Gelombang

Parameter gelombang yang dukur, yaitu tinggi gelombang dan periode gelombang didapatkan dengan menggunakan algoritma *zero crossing detection*. Algoritma ini mendeteksi nilai 0 dari gelombang yang diuji. Pengujian metode ini dilakukan pertama kali dengan mengunakan sinyal *sinusoidal* untuk memastikan bahwa algoritma bisa bekerja dengan baik. Setelah memastikan algoritma bisa mendeteksi parameter gelombang dengan baik barulah digunakan masukan sinyal dari alat yang dibuat. Hasil pengujian algoritma lihat (Gambar 4.18).



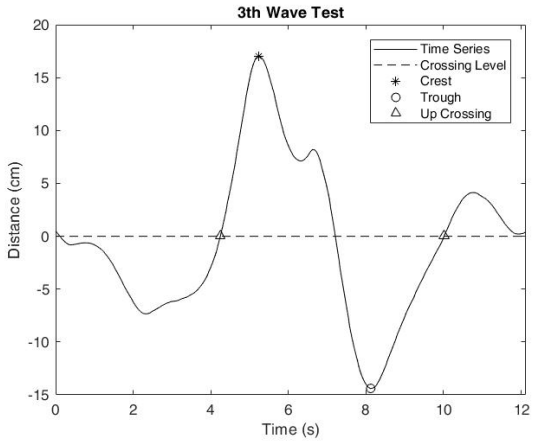
Gambar 4. 18 Pengujian Algoritma ZCD

Sinyal uji adalah sinyal sinusoidal *random* dengan parameter yang ditentukan hanya frekuensi sampling, yaitu 10Hz dan durasi yaitu 32 detik. Berdasarkan (Gambar 4.18) dapat diketahui bahwa algoritma ZCD ini bekerja dengan baik, algoritma dapat mendeteksi setiap nilai nol dalam keadaan sinyal naik, berhasil mendeteksi titik puncak dan lembah untuk setiap satu gelombang. Parameter tinggi dan perioda gelombang juga dapat dihitung, Hasilnya bisa dilihat pada (Tabel 4.5).

Tabel 4. 5 Parameter Gelombang Uji

Gelombang ke-	**Tinggi Gelombang (H)**	**Perioda Gelombang (T)**
1	2,36	5,42
2	2,63	3,31
3	3,56	6,35
4	1,94	3,24
5	0,74	2,09
6	3,07	3,72
7	0,83	2,01

Data pada (Tabel 4.5) merupakan data parameter gelombang dari sinyal masukan acak yang dihitung oleh algoritma ZCD. Algoritma ZCD berhasil mendeteksi parameter gelombang sesuai dengan tujuan peneletian, yaitu tinggi dan perioda gelombang, selanjutnya melakukan pengujian menggunakan data yang didapatkan dari sensor. Pengujian ini dilakukan dengan menggerakan sensor dengan gerakan seakan-akan terbawa gelombang laut, alat digerakan keatas dan kebawah mengikuti gerakan gelombang menggunakan tangan. Salah satu hasil pengujian bisa dilihat pada (Gambar 4.19).



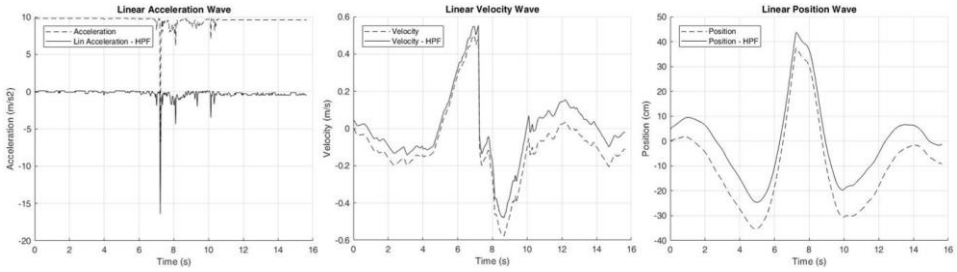
Gambar 4. 19 Hasil Pengujian ZCD

Algoritma ZCD untuk data pengujian dari alat bisa dilihat pada (Gambar 4.19). berdasarkan (Gambar 4.19), parameter tinggi gelombang dan perioda gelombang dapat dideteksi oleh algoritma ZCD. Titik nol yang dideteksi juga berhasil hanya mendeteksi titik nol dengan data yang naik, sehingga satu gelombang bisa didapatkan dengan baik. Tinggi gelombang dihitung dari lembah sampai puncak gelombang, nilai perioda dihitung berdasarkan waktu yang diperlukan oleh satu gelombang. Berikut adalah data lengkap hasil pengujian ini, lihat (Tabel 4.6).

Tabel 4. 6 Hasil Pengujian Tinggi Gelombang

No	**Tinggi Gelombang (cm)**		**Kesalahan (cm)**
	Uji	Aktual	
1	26,89	60	33,1
2	21,84	60	38,16
3	30,76	60	29,24
4	63,33	60	3,33
5	89,67	60	29,67
RMSE			29,32%

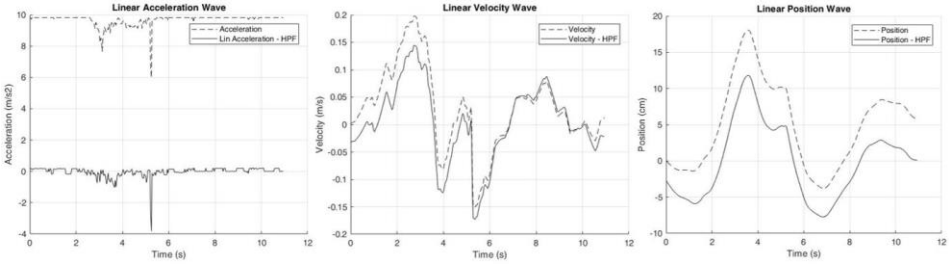
Tinggi gelombang didapatkan dari sumbu Z sensor *accelerometer*. Hasil pengujian berdasarkan data pada (Tabel 4.6) memiliki selisih yang sangat besar dan beragam untuk setiap pengambilan data. Nilai RMSE bahkan mencapai 29.32% dari 5 kali percobaan. Kesalahan yang terjadi disebabkan oleh banyak faktor, nilai *drift* yang belum sepenuhnya hilang terakumulasi ketika sensor bergerak. Data hasil pengujian ke-4, yaitu data dengan kesalahan paling kecil dalam bentuk grafik bisa dilihat pada (Gambar 4.20).



Gambar 4. 20 Grafik Pengujian Ke-4 (a)Percepatan, (b)Kecepatan dan (c) Posisi

Data percepatan yang didapatkan alat untuk pengujian ke-4 memiliki nilai percepatan tertinggi sebesar 15 m/s^2, lihat (Gambar 4.20) (a). Nilai 15 m/s^2 didapat setelah dikurangi nilai gravitasi dan juga setelah diberikan HPF. Data percepatan

diintegralkan menjadi kecepatan (Gambar 4.20) (b), data sebelum dan sesudah diberikan HPF memiliki perbedaan, yaitu filter mengembalikan data akhir menjadi ketitik nol. Data Posisi (Gambar 4.20) (c) yang diberikan HPF memiliki kekurangan yaitu data awal yang sudah berada pada titik nol berubah menjadi lebih tinggi. Sebagai perbandingan dengan pengujian yang memiliki nilai kesalahan tinggi, lihat (Gambar 4.21).



Gambar 4. 21 Grafik Pengujian Ke-2 (a)Percepatan, (b)Kecepatan dan (c) Posisi

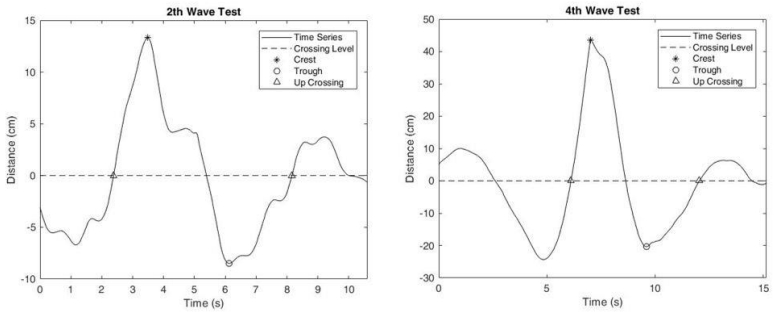
Pengujian data ke-2 berdasarkan (Tabel 4.6) yang memilki nilai kesalahan tertinggi dimana hasil posisi terukur adalah 21,84 cm yang seharusnya adalah 60 cm, grafiknya dilihat pada (Gambar 4.21). Nilai percepatan pada (Gambar 4.21) (a) yang terukur oleh alat kurang dari ±2 m/s^2, dimana apabila dibandingkan dengan data percepatan yang memiliki nilai kesalahan terkecil, lihat (Gambar 4.20) (a), data percepatan tertinggi yang terukur sebesar ±15 m/s^2, selisih antara kedua data sebesar ± 13 m/s^2. Data kecepatan pada (Gambar 4.21) (b) saat diberikan HPF memiliki perubahan yaitu data awal dan akhir menjadi kurang dari nol. Data Posisi (Gambar 4.21) (c) yang diberikan HPF memiliki perubahan, dimana data kesuluruhan diturunkan. Berdasarkan perbandingan ini terlihat bahwa nilai kesalahan yang besar terjadi karena sensor gagal mengukur data percepatan saat sensor bergerak.

Penyebab lainnya yang membuat RMSE sebesat 29,32% berdasarkan (Tabel 4.6) pada pengujian perpindahan secara vertikal atau pada sumbu Z memiliki nilai RMSE yang paling besar, lalu saat nilai di intergral kan sebanyak dua kali maka nilai dari *drift* ikut terintegral yang mengakibatkan kesalahan menjadi semakin besar. Upaya pencegahan sudah dilakukan dengan mengimplementasika *high pass filter* sebelum data diintegralkan dan juga menggunakan madgwick filter dan quaternion. Hasil perioda yang berhasil dihitung dari pengujian bisa dilihat pada (Tabel 4.7).

Tabel 4. 7 Hasil Pengujian Perioda Gelombang

No	1	2	3	4	5
Period (s)	7,66	5.96	5,63	5,88	4,91

Berdasarkan data pada (Tabel 4.7) perioda yang dihitung adalah selang waktu (dalam detik) yang ditempuh antara satu titik *wave up zero* dengan titik *wave up zero* selanjutnya, atau dihitung dalam satu gelombang. Nilai perioda tidak ada perbandingan dikarenakan tidak ada alat yang bisa menghitung gerakan alat menggunakan tangan untuk satu gelombang penuh, dimana ujicoba dilakukan setiap satu gelombang sekali. Hasil algiritma ZCD untuk sinyal uji dari alat lihat (Gambar 4.22).



Gambar 4. 22 Grafik Perbandingan ZCD (a)Data Ke-2 dan (b)Data Ke-4

Grafik pada (Gambar 4.22) (a) adalah garfik hasil dari algoritma ZCD pada data pengujian ke-2 dan Grafik pada (Gambar 4.22) (b) adalah data uji ke-4. Berdasarkan (Gambar 4.22) maka dapat diketahui bahwa algoritma berhasil mendeteksi dan menghitung nilai dari karakteristik gelombang yang diukur pada penilitian ini berdasarkan tujuan penelitian, dimana karakteristik yang harus dideteksi adalah tinggi gelombang dan perioda gelombang.

BAB V
PENUTUP

5.1 Kesimpulan

Serangakaian hasil pengujian pada Bab 4 dapat disimpulkan bahwa Alat yang dibuat berhasil memenuhi tujuan dari skripsi, yaitu mendeteksi parameter gelombang laut berdasarkan data dari pengujian, parameter yang harus dideteksi adalah tinggi dan periode gelombang, namun alat yang dibuat memiliki nilai kesalahan. Pengujian untuk menentukan Tinggi gelombang dengan nilai aktual 60 cm memiliki selisih 33,1 cm, 38,16 cm, 29,24 cm, 3,33 cm dan 29,67 cm dengan nilai RMSE sebesar 29,32 %. Pengujian untuk perioda gelombang memiliki nilai yang terukur adalah 7,66 detik, 5,96 detik, 5,63 detik, 5,88 detik dan 4,91 detik.

Alat yang digunakan tidak sesuai dengan judul dari skripsi ini, yaitu mikrokontroler PSoC diganti dengan Arduino Mega 2560. Pergantian mikrokontroler dikarenakan kurangnya komunitas PSoC sehingga mnyebabkan kurangnya juga literatur yang tesedia yang membuat pengerjaan skripsi tidak terfokus pada tujuan dari skripsi ini sendiri. Arduino Mega 2560 dipilih karena memiliki komunitas yang besar dan aktif sehingga memudahkan untuk melanjutkan penelitian ini.

5.2 Saran

1. Mengganti sensor MPU9250 dengan icm-20948 serta perlu penelitian mengenai kalibrasi sensor untuk sensor *accelerometer*, *gyroscope* dan *magnetometer* pada IMU dengan metode yang lebih rinci dan juga dengan menggunakan bantuan alat untuk menggantikan tangan saat menggerakan sensor.
2. Menggunakan *microcontroller* yang memiliki spesifikasi yang jauh lebiih tinggi dari Arduino mega 2560, sehingga semua proses bisa dilakukan pada mikrokontroler tanpa harus memindahkan data ke matlab, juga untuk membuat frekuensi lebih tinggi dari 30Hz yang membuat data menjadi lebih detail setiap detiknya.

3. Perlu dibuatkan pelampung atau *buoy* untuk tempat dari sensornya sehingga alat bisa dilakukan uji coba di kolam, *buoy* yang dibuat juga perlu memiliki bentuk yang membuat sensor tidak terlalu miring saat terkena gelombang air.
4. Perlu ditambahkan modul telemetry untuk membuat alat bisa mengirimkan data langsung tanpa harus melepas kartu memori atau kabel, sehingga pengujian bisa dilakukan secara *real time*.

DAFTAR PUSTAKA

[1] Solihuddin T., H. L. Salim, S. Husrin, A. Daulat, and D. Purbani, "Dampak Tsunami Selat Sunda di Provinsi Banten dan Upaya Mitigasinya," *Jurnal Segara*, vol. 16, no. 1, pp. 15–28, Apr. 2020.

[2] Widiatmoko F. R., A. Zamroni, M. A. Siamashari, A. N. Maulina, and Budiarto, "Rekaman stasiun GPS sebagai pendeteksi pergerakan tektonik, studi kasus: bencana Tsunami Aceh 26 Desember 2004," *e-Jurnal ITATS (Institut Teknologi Adhi Tama Surabaya*, vol. 1, no. 1, pp. 236–240, Aug. 2019.

[3] Intergovernmental Oceanographic Commission, *Tsunamy Glossary*, 4th ed. Paris: UNESCO, 2019.

[4] Webb P., *introduction to oceanography*. Rebus Community, 2019.

[5] Kurniawan R., N. M. Habibie, and Suratno, "Variasi Bulanan Gelombang Laut Di Indonesia," *Jurnal Meteorologi dan Geofisika*, vol. 12, no. 3, pp. 221–232, Des. 2011.

[6] Azzahra A. N., A. Nugraha, S. Kuncorojati, H. N. Rahmadini, and B. Gauttama, *Buletin Meteorologi Edisi Mei 2022*. Sintang: Badan Meteorologi Klimatologi dan Geofisika, 2022.

[7] Dwipanegara A. D., "Evaluasi Kinerja Sistem Peringatan Dini Tsunami di Indonesia," *INDEPT*, vol. 4, no. 1, Feb. 2014.

[8] Sutrisno E., *Buoy, Pendeteksi Tsunami Super Cepat Buatan Indonesia*, 2021, Tersedia dari: https://indonesia.go.id/ [URL dikunjungi pada 15 Oktober 2022].

[9] Maulana F., *22 Buoy Tsunami Indonesia Tak Aktif Sejak 2012*, 2018, Tersedia dari: https://news.detik.com/ [URL Dikunjungi pada 15 Oktober 2022].

[10] Noptian S. R., A. Suhendi, and R. A. Salam, "Sistem Monitoring Ketinggian Permukaan Air Laut Menggunakan Accelerometer Berbasis IoT," *eProceedings of Engeneering*, vol. 7, no. 2, pp. 4517–4522, Aug. 2020.

[11] Munanda E., I. Jaya, and A. S. Atmadipoera, "Rancang Bangun dan Uji Kinerja Wave Buoy Sebagai Alat Pengukur Tinggi Gelombang Pesisir,"

Jurnal Ilmu dan Teknologi Kelautan Tropis, vol. 10, no. 1, pp. 1–14, Apr. 2018.

[12] Yurovsky Y. Y. and V. A. Dulov, "Compact Low-Cost Arduino-Based Buoy for Sea Surface Wave Measurements," in *2017 Progress in Electromagnetics Research Symposium- Fall (PIERS - FALL)*, Singapore, 2017, pp. 2315–2322.

[13] Liu Y., X. Wang, J. You, and C. Chen, "Ocean Wave Measurement Using GPS Buoys," *Journal of Geodetic Science*, vol. 3, no. 3, pp. 163–172, 2013.

[14] Patra S. K., and B. K. Jena, "Inter-comparison of wave measurement by accelerometer and GPS wave buoy in shallow water off Cuddalore, east coast of India," *Indian Journal of Geo-Marine Science*, vol. 43, no. 1, pp. 45–49, Jan. 2014.

[15] Wall R. W., "Simple Methods for Detecting Zero Crossing," in *Proceedings of The 29th Annual Conference of the IEEE Industrial Electronics Society*, Roanoke, VA, 2003, pp. 2477-2481.

[16] Treffers C. and V. Wietmarschen. L, "Position and orientation determination of a probe with use of the IMU MPU9250 and a ATmega328 microcontroller," Bachelor Thesis, Dept. Microelectronics, Technische Universiteit Delft, Delft, 2016..

[17] Kuperus J., "Wave Monitoring using Wireless Sensor Nodes," Master Thesis, Dept. Embedded Systems, University Of Melbourne, Melbourne, 2009.

[18] Pramana Y. A., "Implementasi Sensor Accelerometer, Gyroscope Dan Magnetometer Berbasis Mikrokontroler Untuk Menampilkan Posisi Benda Menggunakan Inertial Navigation System," Bachelor Thesis, Dept. Electrical Eng., Universitas Komputer Indonesia, Bandung, 2013.

[19] Munarto R., R. Wiryadinata, and D. P. Utama, "Implementation of AHRS (Attitude Heading and Reference Systems) With Madgwick Filter as Hexapods Robot Navigation," in *2022 International Conference on Informatics Electrical and Electronics (ICIEE)*, Yogyakarta, 2021, pp. 1-9.

[20] Mohammed Z., I. M. Elfadel, and M. Rasras, "Monolithic Multi Degree of Freedom (MDoF) Capacitive MEMS Accelerometers," *Micromachines*

(Basel), vol. 9, no. 11, 2018.

[21] John., "MEMS Accelerometer," 2017, Tersedia dari: http://www.instrumentationtoday.com/ [URL Dikunjungi pada 01 November 2022].

[22] Sinha S., S. Shakya, R. Mukhiya, R. Gopal, and B. Pant, "Design and Simulation of MEMS Differential Capacitive Accelerometer," in *ISSS International Conference on Smart Materials, Structures and Systems*, Bangalore, India, 2014.

[23] Northrop R. B., "*Introduction to Instrumentation and Measurements*, 3rd ed, Boca Raton: CRC Press, 2014.

[24] Passaro V. M. N., A. Cuccovillo, L. Vaiani, M. De Carlo, and C. E. Campanella, "Gyroscope Technology and Applications: A Review in the Industrial Perspective," *Sensors (Basel)*, vol. 17, no. 10, Oct. 2017.

[25] Öztürk D. Ü. and A. M. Erkmen, "Coriolis Vibratory MEMS Gyro Drive Axis Control with Proxy-Based Sliding Mode Controller," *Micromachines*, vol. 13, no. 3, 2022.

[26] Chiva J. M. S., J. Valle, D. Fernández, and J. Madrenas, "A Mixed-Signal Control System for Lorentz-Force Resonant MEMS Magnetometers," *IEEE Sensors Journal*, vol. 19, no. 17, pp. 7479–7488, 2019.

[27] Anonymous, *MPU-9250 Product Specification*, San Jose: Inve Sense, 2016.

[28] Karimi M., E. Babaians, M. Oelsch, T. Aykut, and E. Steinbach, "Skewed-redundant Hall-effect Magnetic Sensor Fusion for Perturbation-free Indoor Heading Estimation," in *Proceedings 2020 Fourth IEEE International Conference on Robotic Computing (IRC)*, Taichung, Taiwan2020, pp. 367–374.

[29] Leporcq J., "Position Estimation Using IMU Without Tracking System," Master's Programme in Computer, Communication and Information Sciences (TS2013), Aalto University, Espoo, 2018.

[30] Yang Z., H. Jing, B. Kuang, S. Zhong, J. Qin, and Y. Tong, "GPS and Lidar Fusion Positioning Based on Unscented Kalman Filtering and Long Short-Term Memory Cosidering GPS Failure," in *Proceedings 2021 IEEE 4th International Conference on Automation, Electronics and Electrical*

Engineering (AUTEEE), Shenyang, China, 2021, pp. 18–23.

[31] Ludwig S. A., "Optimization of Control Parameter for Filter Algorithms for Attitude and Heading Reference Systems," in *Proceedings 2018 IEEE Congress on Evolutionary Computation (CEC)*, Rio de Janeiro, Brazil, 2018, pp. 1–8.

[32] Al-Fahoum A. S. and M. S. Abadir, "Design of a modified Madgwick filter for quaternion-based orientation estimation using AHRS," *International Journal of Computer Electrical Engineering*, vol. 10, no. 3, pp. 174–186, 2018.

[33] Kasau M. I. and A. D. Ghani, "ANALISIS KINEMATIK ROBOT KOORDINAT BOLA, ENAM DERAJAT KEBEBASAN DENGAN METODE QUATERNION & ROTATION VECTOR," in *proceeding SISITI: Seminar Ilmiah Sistem Informasi dan Teknologi Informasi*, vol. 7, no. 2, 2018.

[34] Hasan M. A. and M. H. Rahman, "Smart Phone Based Sensor Fusion by Using Madgwick Filter for 3D Indoor Navigation," *Wireless Personal Communinications*, vol. 113, no. 4, pp. 2499–2517, 2020.

[35] Madgwick S. O. H., A. J. L. Harrison, and R. Vaidyanathan, "*Estimation of IMU and MARG orientation using a gradient descent algorithm*", in *Proceedings 2011 IEEE International Conference on Rehabilitation Robotics*, Zurich, Switzerland, 2011, pp. 1-7.

[36] Slifka D. L, *An Accelerometer Based Approach to Measuring Displacement*, Master Thesis, Dept. Elect. and Comp. Eng., University of Michigan, 2004.

[37] Kamal A. M, S. H. Hemel, and M. U. Ahmad, "Comparison of Linear Displacement Measurements Between A Mems Accelerometer and Hc-Sr04 Low-Cost Ultrasonic Sensor," in *Proceedings 2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT)*, Dhaka, Bangladesh, 2019, pp. 1-6.

[38] Liu Y., X. Wang, J. You, and C. Chen, "Ocean Wave Buoy Based on Parallel Six-Dimensional Accelerometer," *IEEE Access*, vol. 8, pp. 29627–29638, 2020.

[39] Zhang Y., L. Qi, D. Junyung, Q. Wen, and M. Lv, "Data Processing Based

on Low-Precision IMU Equipment to Predict Wave Height and Wave Period," in *Proceedings 2019 2nd International Conference on Data Intelligence and Security*, South Padre Island, USA, 2019, pp. 103–107.

[40] Kurnia M. H., R. E. Saputra, and C. Setianingsih, "High-Low Detection of Sea Water Waves With Multi-Sensor System Based on IOT," *e-Proceeding of Engineering*, vol. 8, no. 5, p. 6175, Okt. 2021.

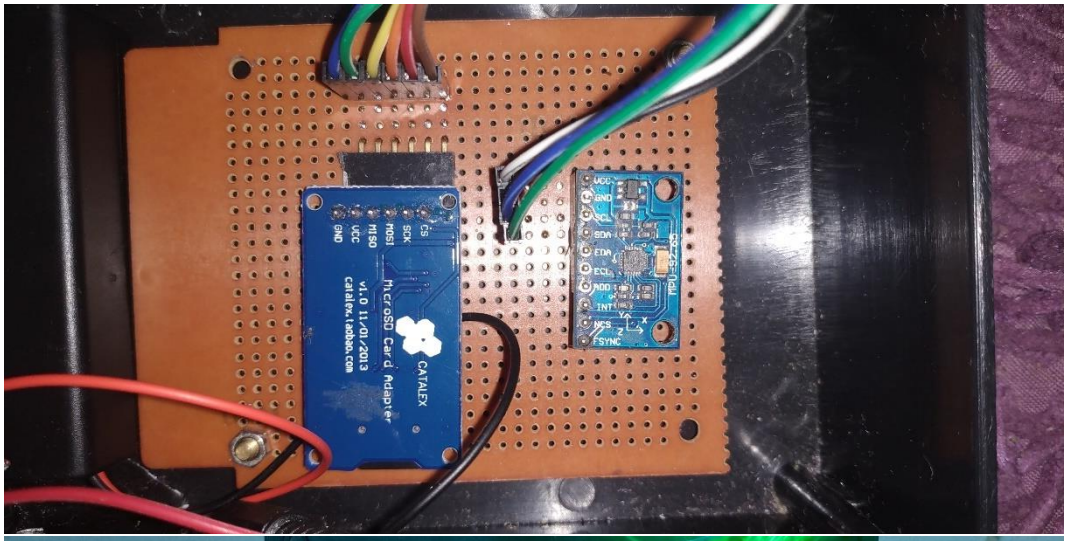
[41] D. Segarra, J. Caballeros, W. G. Aguilar, A. Samà, and D. Rodríguez-Martín, "Orientation Estimation Using Filter-Based Inertial Data Fusion for Posture Recognition," in *Algorithms for Sensor Systems*, Helsinki, Finland, 2019, pp. 220–233.

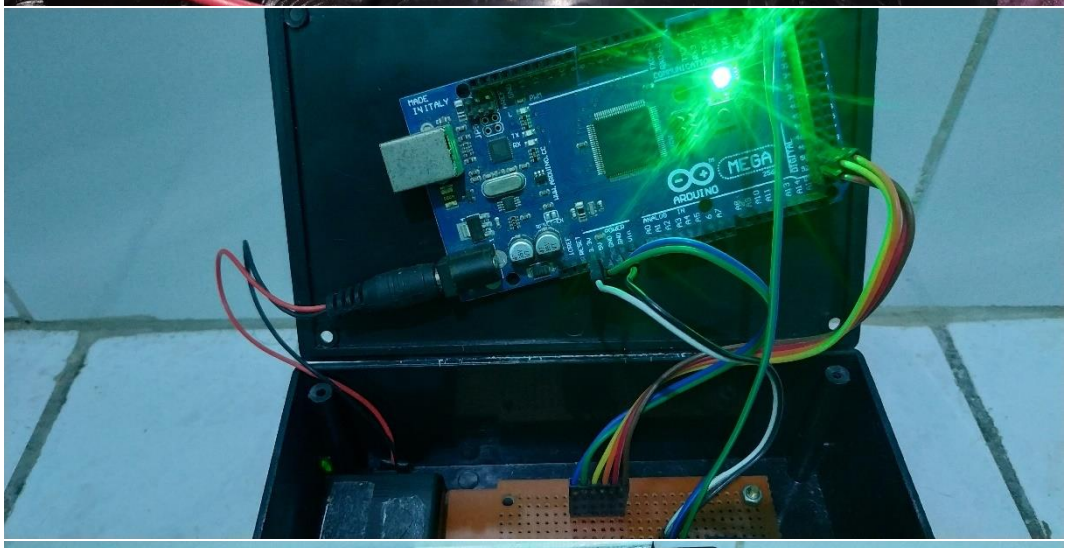
[42] Yurovsky Y. Y. and V. A. Dulov, "MEMS-based wave buoy: Towards short wind-wave sensing," *Ocean Engineering*, vol. 217, p. 108043, 2020.

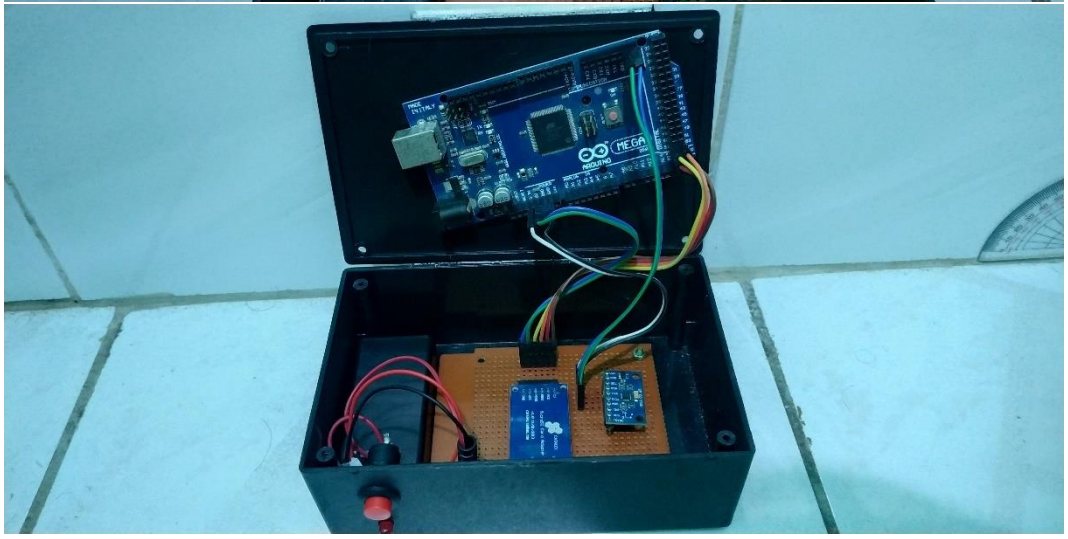
[43] Esser P., H. Dawes, J. Collett, and K. Howells, "IMU: Inertial sensing of vertical CoM movement," *Journal of Biomechanics*, vol. 42, no. 10, pp. 1578–1581, 2009.

LAMPIRAN

LAMPIRAN A Foto Alat







LAMPIRAN B Perbandingan PSoC dan Arduino

	Arduino Mega 2560	**PSoC CY8CKIT-044**
Microcontroller	Atmega2560	Arm Cortex M0
Operating Voltage	5V	3,3V & 5V
Digital I/O Pins	54	55
Analog Input Pins	16	6
I2C	✓	✓
SPI	✓	✓
UART	✓	✓
Flash Memory	256 KB (8 KB for bootloader)	128 KB
SRAM	8 Kb	16 KB
FRAM	-	1 Mb
EEPROM	4 Kb	-
ROM	-	8 KB
Clock Speed	16 MHz	32 KHz

LAMPIRAN C *Listing Program* Arduino

Sumber listing program arduino :

[1] Remington, S.J. *Jremington/MPU-9250-AHRS*. 2020. Tersedia dari: https://github.com/jremington/MPU-9250-AHRS. [URL dikunjungi pada 13 Januari 2022]

[2] Winer, K. *kriswiner/MPU9250*. 2017 Tersedia dari: https://github.com/kriswiner/MPU9250. [URL dikunjungi pada 04 Oktober 2021]

[3] Tai, H. *hideaiktai/MPU9250*. 2020. Tersedia dari: https://github.com/hideakitai/MPU9250. [URL dikunjungi pada 17 Desember 2022]

```
#include "Wire.h"
#include "/libs/I2Cdev.cpp"
#include " libs/MPU9250.cpp"
#include "SD.h"
#include "SPI.h"

MPU9250 mpu;
I2Cdev   I2C_M;
File sdCard;

int CS_pin = 53;

// offsets and correction matrix for accel and mag
float A_B[3]
{ 1529.76,  467.31,  -25.25};

float A_Ainv[3][3]
{ {  0.59100, -0.04303,  0.02990},
  { -0.04303,  0.63281, -0.00879},
  {  0.02990, -0.00879,  0.62263}
};

// mag offsets and correction matrix
float M_B[3]
{   -14.41,   21.43,   13.82};

float M_Ainv[3][3]
{ {  1.84445, -0.01730,  0.00046},
  { -0.01730,  1.90095, -0.01225},
  {  0.00046, -0.01225,  1.78335}
};

float G_off[3] = { -335.7, -86.1, 275.8}; //raw offsets, determined for
gyro at rest

float pi = 3.14159f;
float GyroMeasError = pi * (40.0f / 180.0f);   // gyroscope measurement
error in rads/s (start at 40 deg/s)
```

```
float GyroMeasDrift = pi * (0.0f  / 180.0f);   // gyroscope measurement
drift in rad/s/s (start at 0.0 deg/s/s)
float beta = sqrt(3.0f / 4.0f) * GyroMeasError;   // compute beta
float zeta = sqrt(3.0f / 4.0f) * GyroMeasDrift;   // compute zeta, the
other free parameter in the Madgwick scheme usually set to a small or
zero value

//raw data and scaled as vector
int16_t ax, ay, az;
int16_t gx, gy, gz;
int16_t mx, my, mz;
float Axyz[3];
float Gxyz[3];
float Mxyz[3];

float Araw[3];
float Graw[3];
float Mraw[3];

float Acal[3];
float Gcal[3];
float Mcal[3];
#define gscale (250./32768.0)*(PI/180.0)

// Vector to hold quaternion
static float q[4] = {1.0, 0.0, 0.0, 0.0};
float yaw, pitch, roll; //Euler angle output
float lin_acc[6] {0.f, 0.f, 0.f, 0.f, 0.f, 0.f};  // linear acceleration
(acceleration with gravity component subtracted)

unsigned long now = 0, last = 0; //micros() timers
float deltat = 0;  //loop time in seconds
unsigned long now_ms, last_ms = 0; //millis() timers
unsigned long print_ms = 20; //print every "print_ms" milliseconds

void setup() {
  // put your setup code here, to run once:
  Wire.begin();
  Serial.begin(115200);
  while (!Serial); //wait for connection

  // initialize device
  mpu.initialize();
  // verify connection
  Serial.println(mpu.testConnection() ? "MPU9250 OK" : "MPU9250 ??");
  last = micros();

  //---SD Card SetUp---//
/*
  Serial.print("Initializing SD card...");
  if(!SD.begin(CS_pin)) {
    Serial.println("initialization failed!");
    return;
  }
  Serial.println("initialization done.");
  sdCard = SD.open("DATA.txt", FILE_WRITE);
  if (sdCard) {
```

```
    Serial.println("File opened ok");
    // print the headings for our data
    sdCard.println("Ax, Ay, Az, Gx, Gy, Gz, Mx, My, Mz, q0, q1, q2,
q3"); //Ax, Ay, Az, Gx, Gy, Gz, Mx, My, Mz, LinAx, LinAy, LinAz, q0, q1,
q2, q3
  }
  sdCard.close();*/
}

void loop() {
  // put your main code here, to run repeatedly:
  get_MPU_scaled();
  //get_MPU_cal();
  now = micros();
  deltat = (now - last) * 1.0e-6; //seconds since last update
  last = now;

  /* Sensors x (y)-axis of the accelerometer/gyro is aligned with the y
(x)-axis of the magnetometer;
   *  the magnetometer z-axis (+ down) is misaligned with z-axis (+ up)
of accelerometer and gyro!
   * We have to make some allowance for this orientation mismatch in
feeding the output to the quaternion filter.
   * For the MPU9250+MS5637 Mini breakout the +x accel/gyro is North,
then -y accel/gyro is East. So if we want te quaternions properly
aligned
   * we need to feed into the Madgwick function Ax, -Ay, -Az, Gx, -Gy, -
Gz, My, -Mx, and Mz. But because gravity is by convention
   * positive down, we need to invert the accel data, so we pass -Ax,
Ay, Az, Gx, -Gy, -Gz, My, -Mx, and Mz into the Madgwick
   * function to get North along the accel +x-axis, East along the accel
-y-axis, and Down along the accel -z-axis.
   * This orientation choice can be modified to allow any convenient
(non-NED) orientation convention.
  */
  //filter.update(gyr.y(),gyr.x(),-gyr.z(),-acc.y(),-acc.x(),acc.z(),-
hag.y(),-hag.x(),hag.z());
  //filter.update(Gxyz[1], Gxyz[0], -Gxyz[2], -Axyz[1], -Axyz[0],
Axyz[2], -Mxyz[1], -Mxyz[0], Mxyz[2]);
  //MadgwickQuaternionUpdate(-ax, ay, az, gx * pi / 180.0f, -gy * pi /
180.0f, -gz * pi / 180.0f,  my,  -mx, mz);
  //MadgwickQuaternionUpdate(Axyz[0], Axyz[1], Axyz[2], Gxyz[0],
Gxyz[1], Gxyz[2], Mxyz[1], Mxyz[0], -Mxyz[2]);
  MadgwickQuaternionUpdate(-Axyz[0], Axyz[1], Axyz[2], Gxyz[0], -
Gxyz[1], -Gxyz[2], Mxyz[1], -Mxyz[0], Mxyz[2]);
  //MadgwickQuaternionUpdate(Gxyz[1], Gxyz[0], -Gxyz[2], -Axyz[1], -
Axyz[0], Axyz[2], -Mxyz[1], -Mxyz[0], Mxyz[2]);

  //linear_acceleration(q[0], q[1], q[2], q[3]);
  /*roll = atan2f(q[0]*q[1] + q[2]*q[3], 0.5f - q[1]*q[1] - q[2]*q[2]);
  pitch = asinf(-2.0f * (q[1]*q[3] - q[0]*q[2]));
  yaw = atan2f(q[1]*q[2] + q[0]*q[3], 0.5f - q[2]*q[2] - q[3]*q[3]);

  // to degrees
  yaw   *= 57.29578f;
  pitch *= 57.29578f;
  roll *= 57.29578f;
```

```
  // corrected for local magnetic declination
  yaw = yaw - 0.29;
  if(yaw <= 0) {
    yaw = 360 + yaw;
  }*/
  /*if(yaw >= 180) {
    yaw = 360 - yaw;
  }
  if(yaw <= -180.0) {
    yaw = 360.0 + yaw;
  }*/

  /*roll  = atan2((q[0] * q[1] + q[2] * q[3]), 0.5 - (q[1] * q[1] + q[2]
* q[2]));
  pitch = asin(2.0 * (q[0] * q[2] - q[1] * q[3]));
  yaw   = atan2((q[1] * q[2] + q[0] * q[3]), 0.5 - ( q[2] * q[2] + q[3]
* q[3]));

  // to degrees
  yaw   *= 180.0 / PI;
  pitch *= 180.0 / PI;
  roll *= 180.0 / PI;

  // http://www.ngdc.noaa.gov/geomag-web/#declination
  //conventional nav, yaw increases CW from North, corrected for local
magnetic declination

  yaw = -yaw + 0.29;
  if (yaw < 0) yaw += 360.0;
  if (yaw >= 360.0) yaw -= 360.0;*/

  now_ms = millis(); //time to print?
  if (now_ms - last_ms >= print_ms) {
    last_ms = now_ms;
    // print angles for serial plotter...
    Serial.print(now_ms*0.001); Serial.print(", ");

    //print_accRawCal(); Serial.println("");
    //print_gyrRawCal(); //Serial.println("");

    //Serial.print(Axyz[2], 2); Serial.println("");
    print_acc(); Serial.print(", ");
    print_gyro(); Serial.print(", ");
    print_mag(); Serial.print(", ");

    print_q(); Serial.println("");
    //Logging();

    //print_linAcc(); Serial.println("");
    //print_linAcc2(); Serial.println("");

    /*Serial.print(yaw, 2);
    Serial.print(", ");
    Serial.print(pitch, 2);
    Serial.print(", ");
    Serial.println(roll, 2);
```

```
  }
}

void get_MPU_scaled(void) {
  float temp[3];
  int i;
  mpu.getMotion9(&ax, &ay, &az, &gx, &gy, &gz, &mx, &my, &mz);

  Gxyz[0] = ((float) gx - G_off[0]) * gscale; //250 LSB(d/s) default to
radians/s
  Gxyz[1] = ((float) gy - G_off[1]) * gscale;
  Gxyz[2] = ((float) gz - G_off[2]) * gscale;

  Axyz[0] = (float) ax;
  Axyz[1] = (float) ay;
  Axyz[2] = (float) az;
  //apply offsets (bias) and scale factors from Magneto
  for (i = 0; i < 3; i++) temp[i] = (Axyz[i] - A_B[i]);
  Axyz[0] = A_Ainv[0][0] * temp[0] + A_Ainv[0][1] * temp[1] +
A_Ainv[0][2] * temp[2];
  Axyz[1] = A_Ainv[1][0] * temp[0] + A_Ainv[1][1] * temp[1] +
A_Ainv[1][2] * temp[2];
  Axyz[2] = A_Ainv[2][0] * temp[0] + A_Ainv[2][1] * temp[1] +
A_Ainv[2][2] * temp[2];
  vector_normalize(Axyz);

  Mxyz[0] = (float) mx;
  Mxyz[1] = (float) my;
  Mxyz[2] = (float) mz;
  //apply offsets and scale factors from Magneto
  for (i = 0; i < 3; i++) temp[i] = (Mxyz[i] - M_B[i]);
  Mxyz[0] = M_Ainv[0][0] * temp[0] + M_Ainv[0][1] * temp[1] +
M_Ainv[0][2] * temp[2];
  Mxyz[1] = M_Ainv[1][0] * temp[0] + M_Ainv[1][1] * temp[1] +
M_Ainv[1][2] * temp[2];
  Mxyz[2] = M_Ainv[2][0] * temp[0] + M_Ainv[2][1] * temp[1] +
M_Ainv[2][2] * temp[2];
  vector_normalize(Mxyz);
}

void get_MPU_cal(void) {
  float temp[3];
  int i;
  mpu.getMotion9(&ax, &ay, &az, &gx, &gy, &gz, &mx, &my, &mz);

  Graw[0] = (float) gx * gscale;
  Graw[1] = (float) gy * gscale;
  Graw[2] = (float) gz * gscale;

  Araw[0] = (float) ax;
  Araw[1] = (float) ay;
  Araw[2] = (float) az;
  //vector_normalize(Araw);
}

float vector_dot(float a[3], float b[3])
{
```

```
  return a[0] * b[0] + a[1] * b[1] + a[2] * b[2];
}

void vector_normalize(float a[3])
{
  float mag = sqrt(vector_dot(a, a));
  a[0] /= mag;
  a[1] /= mag;
  a[2] /= mag;
}

/*void AccelEarthFrame() { //(float ax, float ay, float az) {
  //Acceleration earth reference
  //Rotation matrix
  q0 = q[0];
  q1 = q[1];
  q2 = q[2];
  q3 = q[3];

  a11 = (q0 * q0) + (q1 * q1) - (q2 * q2) - (q3 * q3); //a11 = -2.0f *
(q2 * q2 + q3 * q3) + 1.0f;
  a21 = 2.0f * (q1 * q2 + q0 * q3);
  a31 = 2.0f * (q1 * q3 - q0 * q2);

  a12 = 2.0f * (q1 * q2 - q0 * q3);
  a22 = (q0 * q0) - (q1 * q1) + (q2 * q2) - (q3 * q3); //a22 = -2.0f *
(q1 * q1 + q3 * q3) + 1.0f;
  a32 = 2.0f * (q0 * q1 + q2 * q3);

  a13 = 2.0f * (q0 * q2 + q1 * q3);
  a23 = 2.0f * (q2 * q3 - q0 * q1);
  a33 = (q0 * q0) - (q1 * q1) - (q2 * q2) + (q3 * q3); //a11 = -2.0f *
(q1 * q1 + q2 * q2) + 1.0f;

  //Matrix multiplication aR = Rxa
  aRx = (a11 * Axyz[0]) + (a12 * Axyz[1]) + (a13 * Axyz[2]);
  aRy = (a21 * Axyz[0]) + (a22 * Axyz[1]) + (a23 * Axyz[2]);
  aRz = (a31 * Axyz[0]) + (a32 * Axyz[1]) + (a33 * Axyz[2]);
  //aRz = (aRz*0.98)-0.98;

  //Serial.print("lin aRx, lin aRy, lin aRz: ");
  Serial.print(aRx, 2); Serial.print(","); Serial.print(aRy, 2);
Serial.print(","); Serial.println(aRz, 2);
}*/

void linear_acceleration(float qw, float qx, float qy, float qz)
{
  lin_acc[0] = Axyz[0] + (2.0f * (qw * qx + qy * qz));
  lin_acc[1] = Axyz[1] + (2.0f * (qx * qz - qw * qy));
  lin_acc[2] = Axyz[2] - (qw * qw - qx * qx - qy * qy + qz * qz);
  lin_acc[3] = Axyz[0] + (qw * qx + qy * qz);
  lin_acc[4] = Axyz[1] + (qw * qy - qx * qz);
  lin_acc[5] = Axyz[2] - (qx * qx + qy * qy);
}

void print_gyrRawCal() {
```

```
  Serial.print(Graw[0]); Serial.print(", "); Serial.print(Graw[1]);
Serial.print(", "); Serial.print(Graw[2]); //Serial.print(", ");
  //Serial.print(Gcal[0]); Serial.print(", "); Serial.print(Gcal[1]);
Serial.print(", "); Serial.print(Gcal[2]);
}

void print_accRawCal() {
  Serial.print(Araw[0]); Serial.print(", "); Serial.print(Araw[1]);
Serial.print(", "); Serial.print(Araw[2]); //Serial.print(", ");
  //Serial.print(Acal[0]); Serial.print(", "); Serial.print(Acal[1]);
Serial.print(", "); Serial.print(Acal[2]);
}

void print_acc() {
  Serial.print(Axyz[0], 2); Serial.print(", "); Serial.print(Axyz[1],
2); Serial.print(", "); Serial.print(Axyz[2], 2);
}

void print_gyro() {
  Serial.print(Gxyz[0], 2); Serial.print(", "); Serial.print(Gxyz[1],
2); Serial.print(", "); Serial.print(Gxyz[2], 2);
}

void print_mag() {
  Serial.print(Mxyz[0], 2); Serial.print(", "); Serial.print(Mxyz[1],
2); Serial.print(", "); Serial.print(Mxyz[2], 2);
}

void print_q() {
  Serial.print(q[0]); Serial.print(", "); Serial.print(q[1]);
Serial.print(", "); Serial.print(q[2]); Serial.print(", ");
Serial.print(q[3]);
  //Serial.print(q0); Serial.print(", "); Serial.print(q1);
Serial.print(", "); Serial.print(q2); Serial.print(", ");
Serial.print(q3);
}

void print_linAcc() {
  Serial.print(lin_acc[0], 2); Serial.print(", ");
Serial.print(lin_acc[1], 2); Serial.print(", ");
Serial.print(lin_acc[2], 2);
}

void print_linAcc2() {
  Serial.print(lin_acc[3], 2); Serial.print(", ");
Serial.print(lin_acc[4], 2); Serial.print(", ");
Serial.print(lin_acc[5], 2);
}

void Logging() {
  sdCard = SD.open("DATA.txt", FILE_WRITE);
  if(sdCard) {
    sdCard.print(now_ms*0.001); sdCard.print(", ");
    sdCard.print(Axyz[0], 2); sdCard.print(", "); sdCard.print(Axyz[1],
2); sdCard.print(", "); sdCard.print(Axyz[2], 2); sdCard.print(", ");
    sdCard.print(Gxyz[0], 2); sdCard.print(", "); sdCard.print(Gxyz[1],
2); sdCard.print(", "); sdCard.print(Gxyz[2], 2); sdCard.print(", ");
```

```
    sdCard.print(Mxyz[0], 2); sdCard.print(", "); sdCard.print(Mxyz[1],
2); sdCard.print(", "); sdCard.print(Mxyz[2], 2); sdCard.print(", ");
    sdCard.print(q[0]); sdCard.print(", "); sdCard.print(q[1]);
sdCard.print(", "); sdCard.print(q[2]); sdCard.print(", ");
sdCard.println(q[3]);
  }
  else {
    Serial.println("can't open sd card");
  }
  sdCard.close();
}
```

LAMPIRAN D *Listing Program* Matlab

Sumber listing program matlab :

[1] Madgwick, S. *Oscillatory-Motion-Tracking-With-x-IMU*. 2017. Tersedia dari: https://github.com/xioTechnologies/Oscillatory-Motion-Tracking-With-x-IMU. [URL dikunjungi pada 17 November 2022]

[2] Karimpour, A. *Wave Upward Zero Crossing Fungction.* 2020 Tersedia dari: https://www.mathworks.com/matlabcentral/fileexchange/59342-wave-upward-zero-crossing-function. [URL dikunjungi pada 26 Januari 2022]


```
%addpath('D:\Uyung\TA\Tsunami\Program\testing\ximu_matlab_library');
% include x-IMU MATLAB library
addpath('D:\Uyung\TA\Tsunami\Program\program_Matlab_Madgwick_FIX\quatern
ion_library');  % include quatenrion library
addpath('D:\Uyung\TA\Tsunami\Program\program_Matlab_Madgwick_FIX\Gyrosco
peIntegration');    % include Gyroscope integration library
addpath('D:\Uyung\TA\Tsunami\Program\program_Matlab_Madgwick_FIX\Acceler
ometerMagnetometer');    % include Accelerometer and Magnetometer
Integration library

close all;                      % close all figures
clear;                          % clear all variables
clc;                            % clear the command terminal

%% Import data
[~, ~, raw] = xlsread('D:\DataGelombang2.xlsx','sheet1','A3:N1278');
%Data 1
%[~, ~, raw] = xlsread('D:\DataGelombang2.xlsx','sheet2','A2:N831');
%Data 2
%[~, ~, raw] = xlsread('D:\DataGelombang2.xlsx','sheet3','A3:N831');
%Data 3
%[~, ~, raw] = xlsread('D:\DataGelombang2.xlsx','sheet5','A2:N1023');
%Data 4
%[~, ~, raw] = xlsread('D:\DataGelombang2.xlsx','sheet4','A2:N831');
%Data 5

% Allocate imported array to column variable names
% Create output variable
data = reshape([raw{:}],size(raw));
tim = data(:,1);
acce = data(:,2:4);
gyr = data(:,5:7);
magn = data(:,8:10);
q = data(:,11:14);

% convert degree to radian
% gyr = deg2rad(gyro);

samplePeriod = 1/33;

% Plot
figure('NumberTitle', 'off', 'Name', 'Accelerometer');
```

```
hold on;
plot(acce(:,1), 'r');
plot(acce(:,2), 'g');
plot(acce(:,3), 'b');
xlabel('sample');
ylabel('g');
title('Accelerometer');
legend('X', 'Y', 'Z');
 %
%% Cuplik;
m1 = 500; n1 = 1000; t1 = n1-m1+1;
% m2 = 350; n2 = 700; t2 = n2-m2+1;
% m3 = 200; n3 = 600; t3 = n3-m3+1; %m4 = 780; n4 = 880; t4 = n4-m4+1;
%570:800 #333333333
% m4 = 300; n4 = 800; t4 = n4-m4+1; %150:400, 500:750, 800:1050-800:840
% m5 = 300; n5 = 600; t5 = n5-m5+1;

% Data 1
accX_ = acce(m1:n1,:); %
gyr_ = gyr(m1:n1,:);
mag_ = magn(m1:n1,:);
q_ = q(m1:n1,:);
time_ = tim(1:t1);

% % Data 2
% accX_ = acce(m2:n2,:); %
% gyr_ = gyr(m2:n2,:);
% mag_ = magn(m2:n2,:);
% q_ = q(m2:n2,:);
% time_ = tim(1:t2);

% % Data 3
% accX_ = acce(m3:n3,:); %
% gyr_ = gyr(m3:n3,:);
% mag_ = magn(m3:n3,:);
% q_ = q(m3:n3,:);
% time_ = tim(1:t3);

% % Data 4
% accX_ = acce(m4:n4,:); %
% gyr_ = gyr(m4:n4,:);
% mag_ = magn(m4:n4,:);
% q_ = q(m4:n4,:);
% time_ = tim(1:t4);

% % Data 4
% accX_ = acce(m5:n5,:); %
% gyr_ = gyr(m5:n5,:);
% mag_ = magn(m5:n5,:);
% q_ = q(m5:n5,:);
% time_ = tim(1:t5);


% time = t;
% quat = q;
% accX = acc;
```

```
time = time_; %5,6
quat = q_;
acc = accX_;
gyro = gyr_;
mag = mag_;

% % Plot
% figure('NumberTitle', 'off', 'Name', 'Acceleration Cuplik');
% hold on;
% plot(accX(:,2), 'g');
% xlabel('sample');
% ylabel('m/s2');
% title('acceleration Cuplik');

%% Process data through AHRS algorithm (calcualte orientation)
% See: http://www.x-io.co.uk/open-source-imu-and-ahrs-algorithms/

R1 = zeros(3,3,length(gyro));     % rotation matrix describing sensor
relative to Earth

AHRS = MadgwickAHRS('SamplePeriod', samplePeriod, 'Beta', 0.1);

quaternion = zeros(length(time),4);
for t = 1:length(time)
    AHRS.Update(gyro(t,:), acc(t,:), mag(t,:)); % gyroscope units must
be radians
    quaternion(t, :) = AHRS.Quaternion;
    R1(:,:,t) = quatern2rotMat(AHRS.Quaternion)';
end

% Quaternion to rotation matrix
R2 = quatern2rotMat(quat);

R = R2;

%% Calculate 'tilt-compensated' accelerometer

tcAcc = zeros(size(acc));  % accelerometer in Earth frame

for i = 1:length(acc)
    tcAcc(i,:) = R(:,:,i) * acc(i,:)';
end

%% Calculate linear acceleration in Earth frame (subtracting gravity)

linAcc = tcAcc - [zeros(length(tcAcc), 1), zeros(length(tcAcc), 1),
ones(length(tcAcc), 1)];
linAcc = linAcc * 9.81;     % convert from 'g' to m/s/s

% Plot
figure('NumberTitle', 'off', 'Name', 'Linear Acceleration');
hold on;
plot(linAcc(:,1), 'r');
plot(linAcc(:,2), 'g');
plot(linAcc(:,3), 'b');
xlabel('sample');
ylabel('g');
```

```
title('Linear acceleration');
legend('X', 'Y', 'Z');

%% High-pass filter linear Acceleration to remove drift
order = 4;
filtCutOff = 0.1;
[b, a] = butter(order, (2*filtCutOff)/(1/samplePeriod), 'high');
linAccHP = filtfilt(b, a, linAcc);

%% Calculate linear velocity (integrate acceleartion)
linVel = zeros(size(linAccHP));

% for i = 2:length(linAccHP)
%     linVel(i,:) = linVel(i-1,:) + linAccHP(i,:) * samplePeriod;
% end
linVel = cumtrapz(time,linAccHP);

% Plot
figure('NumberTitle', 'off', 'Name', 'Linear Velocity');
hold on;
plot(linVel(:,1), 'r');
plot(linVel(:,2), 'g');
plot(linVel(:,3), 'b');
xlabel('sample');
ylabel('g');
title('Linear velocity');
legend('X', 'Y', 'Z');

%% High-pass filter linear velocity to remove drift
[b, a] = butter(order, (2*filtCutOff)/(1/samplePeriod), 'high');
linVelHP = filtfilt(b, a, linVel);

% Plot
figure('NumberTitle', 'off', 'Name', 'High-pass filtered Linear
Velocity');
hold on;
plot(linVelHP(:,1), 'r');
plot(linVelHP(:,2), 'g');
plot(linVelHP(:,3), 'b');
xlabel('sample');
ylabel('g');
title('High-pass filtered linear velocity');
legend('X', 'Y', 'Z');

%% Calculate linear position (integrate velocity)

linPos = zeros(size(linVelHP));

% for i = 2:length(linVelHP)
%     linPos(i,:) = linPos(i-1,:) + linVelHP(i,:) * samplePeriod;
% end
linPos = cumtrapz(time,linVelHP);
linPoscm = linPos*100;

% Plot
figure('NumberTitle', 'off', 'Name', 'Linear Position');
hold on;
```

```
plot(linPoscm(:,1), 'r');
plot(linPoscm(:,2), 'g');
plot(linPoscm(:,3), 'b');
xlabel('sample');
ylabel('g');
title('Linear position');
legend('X', 'Y', 'Z');

%% High-pass filter linear position to remove drift
[b, a] = butter(order, (2*filtCutOff)/(1/samplePeriod), 'high');
linPosHP = filtfilt(b, a, linPos);
linPosHP = linPosHP*100;

% Plot
figure('NumberTitle', 'off', 'Name', 'High-pass filtered Linear
Position');
hold on;
plot(linPosHP(:,1), 'r');
plot(linPosHP(:,2), 'g');
plot(linPosHP(:,3), 'b');
xlabel('sample');
ylabel('g');
title('High-pass filtered linear position');
legend('X', 'Y', 'Z');

%% Play animation

% SamplePlotFreq = 8;
%
% SixDOFanimation(linPosHP, R, ...
%                 'SamplePlotFreq', SamplePlotFreq, 'Trail', 'Off', ...
%                 'Position', [9 39 1280 720], ...
%                 'AxisLength', 0.1, 'ShowArrowHead', false, ...
%                 'Xlabel', 'X (cm)', 'Ylabel', 'Y (cm)', 'Zlabel', 'Z
(cm)', 'ShowLegend', false, 'Title', 'Unfiltered',...
%                 'CreateAVI', false, 'AVIfileNameEnum', false,
'AVIfps', ((1/samplePeriod) / SamplePlotFreq));
%
%% End of script
waveZ2 = linPosHP(:,3);
waveZ = linPoscm(:,3);
fs = 33;
t = time;

% close all;
%
[H,T,Time,UpCrossIndex,UpCrossTime,UpCrossValue,TroughTime,TroughValue,C
restTime,CrestValue]=WaveUpZeroCrossing(waveZ,'zero',fs,'yes');
```