TUGAS AKHIR - TE145561

PERANCANGAN *INERTIAL NAVIGATION SYSTEM* MENGGUNAKAN *INERTIAL MEASUREMENT UNIT* 10 DOF PADA *DIVER PROPULSION VEHICLE*

Alief Ardiansyah
NRP. 10311500000070

Pembimbing
Imam Arifin, S.T., M.T.
Mahardhika P., S.T., M.Sc., Ph.D.
Agung Imam R., S.T.

DEPARTEMEN TEKNIK ELEKTRO OTOMASI
Fakultas Vokasi
Institut Teknologi Sepuluh Nopember
Surabaya 2018

FINAL PROJECT – TE145561

INERTIAL NAVIGATION SYSTEM DESIGN USING 10 DOF INERTIAL MEASUREMENT UNIT ON DIVER PROPULSION VEHICLE

Alief Ardiansyah
NRP. 10311500000070

Supervisors
Imam Arifin, S.T., M.T.
Mahardhika P., S.T., M.Sc., Ph.D.
Agung Imam R., S.T.

DEPARTMENT OF ELECTRICAL AUTOMATION ENGINEERING
Faculty of Vocations
Institut Teknologi Sepuluh Nopember
Surabaya 2018

PERNYATAAN KEASLIAN TUGAS AKHIR

Dengan ini saya menyatakan bahwa isi sebagian maupun keseluruhan dari Tugas Akhir saya dengan judul:

"PERANCANGAN *INERTIAL NAVIGATION SYSTEM* MENGGUNAKAN *INERTIAL MEASUREMENT UNIT* 10 DOF PADA *DIVER PROPULSION VEHICLE*"

adalah benar-benar hasil karya intelektual mandiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diizinkan dan bukan merupakan karya pihak lain yang saya akui sebagai karya sendiri.

Semua referensi yang dikutip maupun dirujuk telah ditulis secara lengkap di dalam daftar pustaka. Apabila ternyata pernyataan ini tidak benar, saya bersedia menerima sanksi sesuai dengan peraturan yang berlaku.

Surabaya, 25 Juni 2018

Alief Ardiansyah
NRP. 10311500000070

PERANCANGAN *INERTIAL NAVIGATION SYSTEM* MENGGUNAKAN *INERTIAL MEASUREMENT UNIT* 10 DOF PADA *DIVER PROPULSION VEHICLE*

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Memperoleh Gelar Ahli Madya Teknik
Pada
Departemen Teknik Elektro Otomasi
Fakultas Vokasi
Institut Teknologi Sepuluh Nopember

Menyetujui

Pembimbing I,

Imam Arifin, S.T., M.T.
NIP. 197302222002121001

Pembimbing II,

Mahardhika P., S.T., M.Sc., Ph.D.

SURABAYA
JULI, 2018

Halaman ini sengaja dikosongkan

PERANCANGAN *INERTIAL NAVIGATION SYSTEM* MENGGUNAKAN *INERTIAL MEASUREMENT UNIT* 10 DOF PADA *DIVER PROPULSION VEHICLE* DI PT BHIMASENA RESEARCH AND DEVELOPMENT

TUGAS AKHIR

Disusun oleh:

Alief Ardiansyah NRP. 10311500000070

Menyetujui,

Kepala *Human Resources Department*,

Fadli Tirmissi
NIK. 020492016

Pembimbing Perusahaan,

Agung Imam R., S.T.
NIK. 020412016

Chief Executive Officer,

BHIMASENA Research & Development

Dipl. -Ing. Aris Budiyarto

Halaman ini sengaja dikosongkan

PERANCANGAN *INERTIAL NAVIGATION SYSTEM* MENGGUNAKAN *INERTIAL MEASUREMENT UNIT* 10 DOF PADA *DIVER PROPULSION VEHICLE*

Alief Ardiansyah
10311500000070

Pembimbing I : Imam Arifin, S.T., M.T.
Pembimbing II : Mahardhika P., S.T., M.Sc., Ph.D.
Pembimbing III : Agung Imam R., S.T.

ABSTRAK

Sistem navigasi pada *Diver Propulsion Vehicle* (DPV) menggunakan kompas analog. Alat tersebut digunakan untuk menampilkan arah dari perpindahan sudut kendaraan yang ditunjukkan melalui derajat sudut dan arah mata angin. Namun, alat tersebut tidak bisa menampilkan data lain, seperti sudut orientasi dan kedalaman penyelaman, yang dibutuhkan untuk mengoptimalkan kinerja dari DPV.

Inertial Navigation System (INS) digunakan sebagai sistem navigasi pada kendaraan, dimana di dalam sistem tersebut terdapat *Inertial Measurement Unit* (IMU) dan perangkat komputer. INS bekerja dengan mengukur percepat an linier, laju rotasi, dan medan magnet menggunakan akselerometer, giroskop, dan magnetometer yang terdapat pada IMU. Hasil pembacaan tersebut diolah oleh perangkat komputer menggunakan perhitungan *digital low-pass filter*, *quaternion*, dan *euler angles* untuk mendapatkan data sudut orientasi berupa sudut *roll*, *pitch*, dan *yaw* dari kendaraan. IMU pada INS dikonfigurasikan dengan sebuah sensor tekanan yang bekerja dengan mengukur tekanan hidrostatis dan mengkonversikannya menjadi data kedalaman penyelaman.

Dari pengujian yang telah dilakukan, didapatkan hasil pengukuran sudut orientasi yang memiliki nilai kesalahan terjauh sebesar 3.00^o, dengan nilai kesalahan rata-rata terjauh sebesar 0.96^o, dan nilai derau terjauh dalam kondisi tunak sebesar 1.00^o. Sedangkan untuk hasil pengukuran kedalaman memiliki nilai kesalahan terjauh sebesar 2.00 cm, dengan nilai kesalahan rata-rata terjauh sebesar 1.30 cm, dan nilai derau terjauh dalam kondisi tunak sebesar 0 cm.

Kata kunci: Navigasi, DPV, INS, IMU, orientasi, kedalaman

Halaman ini sengaja dikosongkan

INERTIAL NAVIGATION SYSTEM DESIGN USING 10 DOF INERTIAL MEASUREMENT UNIT ON DIVER PROPULSION VEHICLE

Alief Ardiansyah
10311500000070

Supervisor I : Imam Arifin, S.T., M.T.
Supervisor II : Mahardhika P., S.T., M.Sc., Ph.D.
Supervisor III: Agung Imam R., S.T.

ABSTRACT

Navigation system on Diver Propulsion Vehicle (DPV) uses an analog compass. It is used to display the direction of the angular displacement of the vehicle through the degree of angle and the cardinal direction. However, it can not display other variables, such as the orientation angle and the dive depth, which are needed to optimize the performance of DPV.

Inertial Navigation System (INS) is used as a navigation system on the vehicle, where within the system there are Inertial Measurement Unit (IMU) and computer device. INS works by measuring linear acceleration, rotation rate, and magnetic field using accelerometers, gyroscopes, and magnetometers, that contained in the IMU. The reading results are processed by a computer device using digital calculations of low-pass filters, quaternion, and euler angles to obtain angle orientation data in the form of roll, pitch, and yaw angles from vehicle. The IMU on INS is configured with a pressure sensor that works by measuring the hydrostatic pressure and converting it into depth data.

From the experiments that has been done, obtained results of angle orientation measurements that has the furthest error value of 3.00°, with the furthest average error value of 0.96°, and the furthest noise value at steady state of 1.00°. While for the measurement of the depth has the furthest error value of 2.00 cm, with the furthest average error value of 1.30 cm, and the furthest noise value at steady state of 0 cm.

Keywords**:** *Navigation,* DPV, INS, IMU, *orientation, depth*

Halaman ini sengaja dikosongkan

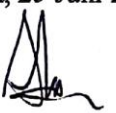
KATA PENGANTAR

Puji syukur kami panjatkan kepada Allah SWT, karena atas rahmat dan karunia-Nya, penulis dapat menyelesaikan laporan Tugas Akhir dengan judul "**Perancangan *Inertial Navigation System* Menggunakan *Inertial Measurement Unit* 10 DoF pada *Diver Propulsion Vehicle***" untuk memenuhi syarat kelulusan pada Program Studi Komputer Kontrol, Departemen Teknik Elektro Otomasi, Fakultas Vokasi, Institut Teknologi Sepuluh Nopember (ITS) Surabaya.

Laporan ini dapat diselesaikan oleh penulis berkat bantuan dari berbagai pihak. Oleh karena itu, penulis mengucapkan terimakasih kepada keluarga, khususnya orang tua dan adik tercinta, yang selalu memberi dukungan dalam berbagai bentuk untuk keberhasilan penulis. Terimakasih kepada Bapak Imam Arifin, S.T., M.T., selaku kepala Laboratorium Sistem Komputer dan Otomasi, dosen wali, dan pembimbing I dari ITS, serta Bapak Mahardhika P., S.T., M.Sc., Ph.D., selaku pembimbing II dari Nanyang Technological University, atas bantuan dan bimbingannya, sehingga penulis dapat menyelesaikan Tugas Akhir ini dengan baik. Terimakasih kepada Bapak Dipl. -Ing. Aris Budiyarto, selaku *Chief Executive Officer* Bhimasena Research & Development dan Bapak Hujjatul Anam, S.T., selaku pemimpin proyek *Diver Propulsion Vehicle*, yang telah memberikan kesempatan kepada penulis untuk bergabung di dalam proyek DPV Bhimasena, sehingga penulis dapat menimba ilmu dan mengerjakan Tugas Akhir di tempat tersebut. Terimakasih kepada Bapak Agung Imam R., S.T., selaku pembimbing III dari perusahaan, atas bantuan dan bimbingan selama di perusahaan, sehingga penulis dapat menimba ilmu dan mengerjakan Tugas Akhir dengan baik. Serta seluruh pihak-pihak yang tidak bisa disebutkan satu persatu, atas bantuan dan dukungan, sehingga penulis diberi kelancaran dalam pengerjaan Tugas Akhir ini.

Penulis mengharapkan kritik dan saran yang membangun dari pembaca. Semoga laporan ini dapat bermanfaat bagi kita semua, khusunya bagi pembaca dan penulis.

Surabaya, 25 Juni 2018



Alief Ardiansyah
NRP. 10311500000070

Halaman ini sengaja dikosongkan

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

Halaman ini sengaja dikosongkan

BAB I
PENDAHULUAN

1.1. Latar Belakang

Bhimasena Research and Development merupakan perusahaan yang melakukan penelitian dan pengembangan teknologi di bidang militer. Perusahaan tersebut bertujuan untuk memenuhi kebutuhan industri militer dalam skala nasional maupun internasional yang terus mengalami perkembangan teknologi dari masa ke masa. Penelitian dan pengembangan yang dilakukan dibagi menjadi beberapa bidang atau divisi yang terdiri dari *Unmanned Aerial Vehicle* (UAV), *Vertical Take-Off and Landing* (VTOL), *Ground Vehicle*, dan *Underwater Vehicle*.

Divisi *Underwater Vehicle* atau kendaraan bawah air merupakan divisi yang melakukan penelitian dan pengembangan mengenai teknologi kendaraan dalam air. Hal tersebut bertujuan untuk memenuhi kebutuhan militer dalam melakukan operasi di dalam air. Di dalam divisi tersebut, terdapat dua proyek penelitian dan pengembangan, yaitu *Diver Propulsion Vehicle* (DPV) dan Kapal Tempur Bawah Air (KTBA).

DPV adalah kendaraan yang dapat melakukan penyelaman bawah air dengan dorongan propeler. Kendaraan ini merupakan perlengkapan militer yang berfungsi untuk membantu pergerakan personil ketika melakukan tugas di dalam air. Tugas yang dapat dilakukan oleh kendaraan ini diantaranya adalah pengintaian, penyergapan, dan patroli bawah air. Selain di bidang militer, kendaraan tersebut juga dapat digunakan dalam bidang ilmu pengetahuan, yaitu untuk melakukan penelitian bawah air [1].

Untuk memenuhi fungsi-fungsi kerja dari DPV, diperlukan elemen-elemen pendukung agar kendaraan tersebut dapat bekerja sesuai dengan fungsinya dengan baik. Salah satu elemen tersebut adalah sistem navigasi yang merupakan penunjuk posisi dan arah tujuan kendaraan. Dengan menggunakan sistem navigasi, pengemudi dapat mengetahui posisi dan arah yang dituju oleh kendaraan [2, 3].

Pada DPV seri sebelumnya menggunakan sebuah kompas analog sebagai alat navigasi dengan menampilkan derajat sudut dan arah mata angin. Namun, perangkat tersebut tidak bisa menampilkan data lain, seperti sudut orientasi dan kedalaman penyelaman, yang dibutuhkan untuk mengoptimalkan kinerja dari kendaraan. Oleh karena itu, *Inertial Navigation System* (INS) digunakan sebagai sistem navigasi untuk

menggantikan kerja dari kompas analog pada penelitian dan pengembangan DPV seri terbaru.

INS adalah sistem navigasi yang melakukan pengukuran data untuk mencari posisi dan orientasi suatu benda berdasarkan titik awal yang diketahui. Sistem tersebut bekerja dengan mengukur data-data yang ada disekitar kendaraan menggunakan *Inertial Measurement Unit* (IMU). Data yang diukur berupa akselerasi linier, kecepatan rotasi, serta medan magnet bumi masing-masing pada 3-sumbu yang dikonfigurasikan dengan tekanan dalam air, sehingga membentuk pengukuran 10 DoF (*Degree of Freedom*). INS dapat menghasilkan pengukuran sudut orientasi dan kedalaman penyelaman kendaraan yang memiliki nilai kesalahan pengukuran dan derau pada kondisi tunak yang kecil.

Melalui penelitian ini, dapat diperoleh indikator tambahan pada sistem navigasi berupa pengukuran sudut orientasi yang terdiri dari sudut *roll*, *pitch*, dan *yaw*, untuk menunjukkan arah yang dituju oleh kendaraan. Selain itu, juga diperoleh pengukuran kedalaman penyelaman yang menunjukkan posisi kedalaman kendaraam berada. Hasil tersebut bertujuan untuk mengoptimalkan kinerja dan tingkat keamanan penggunaan dari DPV.

1.2. Rumusan Masalah

Kompas analog yang digunakan sebagai sistem navigasi pada DPV sebelumnya memiliki beberapa keterbatasan. Perangkat tersebut hanya bisa menampilkan arah yang dituju oleh kendaraan melalui derajat sudut dan arah mata angin, serta tidak bisa menampilkan data lain, seperti sudut orientasi dan kedalaman penyelaman, untuk mengoptimalkan kinerja dari DPV.

1.3. Batasan Masalah

Pada penelitian ini, terdapat sensor MPU9250 digunakan untuk menampilkan sudut orientasi berupa *roll*, *pitch*, dan *yaw*, serta MS5837 untuk menampilkan pengukuran kedalaman penyelaman. Sensor-sensor tersebut diprogram menggunakan perangkat lunak Arduino IDE untuk dapat melakukan pengukuran dimana data hasil pengukuran ditampilkan melalui *Serial Monitor*. Pengujian data sudut orientasi dan kedalaman dilakukan untuk mengetahui akurasi serta derau dari hasil pengukuran.

1.4. Tujuan

Penelitian ini bertujuan agar INS dapat menghasilkan pengukuran data berupa sudut orientasi dan kedalaman penyelaman dari DPV yang memiliki kesalahan dan derau di bawah batas dengan menggunakan akselerometer, giroskop, magnetometer masing-masing pada 3-sumbu, dan sensor tekanan yang terdapat di dalam IMU.

1.5. Metodologi

Pada penelitian ini, metode-metode yang dilaksanakan diawali dengan studi pustaka dan survei literatur. Metode ini dilakukan dengan melakukan pencarian materi yang berkaitan dengan topik penelitian dari berbagai sumber, dimana sumber tersebut dapat berupa buku pustaka, *manual book*, atau *reference book* dari suatu perangkat. Materi yang telah didapat, kemudian dipelajari dan dijadikan sebagai dasar dalam diskusi, perancangan perangkat lunak, perancangan perangkat keras, pengujian, dan analisis.

Setelah dasar teori didapatkan, Sistem yang terdiri dari perangkat keras dan lunak dari INS dirancang di dalam area ruang kerja perusahaan dengan bimbingan pembimbing dari perusahaan. Perancangan tersebut dilaksanakan dengan berdasarkan pada studi pustaka dan literatur yang telah dipelajari.

Untuk mengetahui hasil kinerja dari sistem yang telah dirancang, dilakukan pengujian dan analisis data. Pengujian dilakukan berdasarkan literatur dan hasil diskusi bersama pembimbing. Hasil pengujian yang meliputi meliputi sudut *roll*, *pitch*, *yaw*, dan kedalaman penyelaman, dimana data-data tersebut kemudian di analisis dengan membandingkan data yang didapatkan dengan perangkat pembanding untuk mengetahui tingkat kesalahan dan derau hasil pengukuran sensor.

Penelitian yang telah dilaksanakan kemudian disusun menjadi sebuah laporan dalam bentuk *soft file* dan *hard file* sebagai bukti dan dokumentasi dari penelitian. Laporan tersebut berguna untuk digunakan sebagai referensi dalam penelitian selanjutnya

1.6. Sistematika Penulisan

Penelitian yang dilakukan disusun di dalam laporan dengan pembagian sebanyak lima bab, dengan susunan sebagai berikut:

BAB I PENDAHULUAN

Membahas tentang latar belakang, rumusan masalah, batasan masalah, tujuan, metodologi, dan sistematika laporan dari penelitian yang dilakukan.

BAB II *INERTIAL NAVIGATION SYSTEM* PADA *DIVER PROPULSION VEHICLE*

Menjelaskan mengenai teori-teori penunjang yang dijadikan sebagai landasan dan pendukung dalam penelitian.

BAB III PERANCANGAN *INERTIAL NAVIGATION SYSTEM*

Menjelaskan langkah-langkah yang dilakukan dalam perancangan perangkat keras dan lunak pada INS.

BAB IV PENGUJIAN DAN ANALISIS

Membahas proses, dan data hasil pengujian yang dilakukan untuk menganalisis kinerja dari INS.

BAB V PENUTUP

Berisi kesimpulan yang didapatkan dari hasil peneletian yang telah dilakukan dan saran untuk penelitian selanjutnya.

BAB II
INERTIAL NAVIGATION SYSTEM PADA *DIVER PROPULSION VEHICLE*

2.1. *Diver Propulsion Vehicle*

Diver Propulsion Vehicle (DPV) merupakan kendaraan yang dapat melakukan penyelaman bawah air dengan dorongan baling-baling. Kendaraan tersebut digunakan oleh penyelam bawah air untuk meningkatkan jangkauan jelajah penyelaman di dalam air. Pada Gambar 2.1, menampilkan bentuk dari DPV yang terdiri dari bodi tahan air dan tekanan dimana didalamnya terdapat motor listrik bertenaga baterai yang menggerakkan baling-baling. Kendaraan ini didesain dengan memastikan bahwa baling-baling tidak membahayakan penyelam, peralatan selam, dan kehidupan laut disekitar [1].

Gambar 2.1 *Diver Propulsion Vehicle*

Di dalam DPV, terdapat sistem informasi yang digunakan untuk memudahkan pengoperasian kendaraan dimana sistem tersebut juga berfungsi untuk menjaga keamanan penyelam serta kendaraan itu sendiri. Informasi yang ditampilkan berupa indikator lampu, status sistem (Nyala/mati), daya baterai, waktu jelajah, navigasi, dan kecepatan jelajah kendaraan dimana ditampilkan pada sebuah layar LCD (*Liquid Crystal Display*) yang terdapat pada bagian atas kendaraan [1].

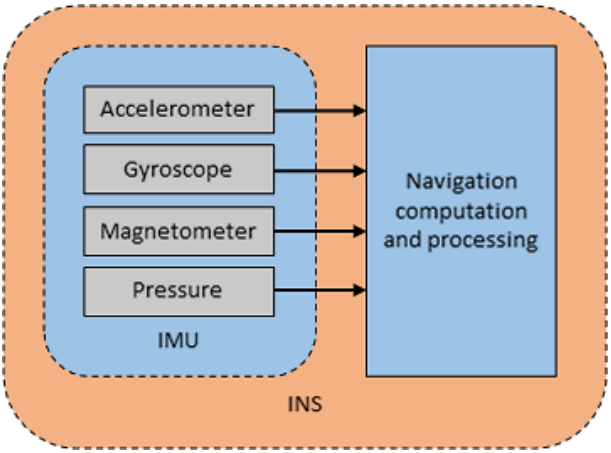
DPV dapat dikendarai oleh satu orang pengendara untuk mengendalikan arah, kecepatan dan kedalaman penyelaman kendaraan. Seperti yang ditunjukkan pada Gambar 2.2, pengendara berada pada posisi di atas kendaraan, dengan kedua tangan berpegang pada setang

besi yang berada di sisi kanan dan kiri. Pada setang tersebut tedapat tombol-tombol yang berfungsi untuk menaikkan dan menurunkan kecepatan jelajah kendaraan.

Gambar 2.2 Posisi Pengendara DPV

2.2. *Inertial Navigation System*

Inertial Navigation System (INS) merupakan sistem navigasi mandiri yang melakukan pengukuran untuk mencari posisi dan orientasi dari sebuah objek yang relatif berdasarkan pada orientasi, kecepatan, dan titik awal yang diketahui [4]. Sistem tersebut terdiri dari *Inertial Measurement Unit* (IMU) dan perangkat komputer.



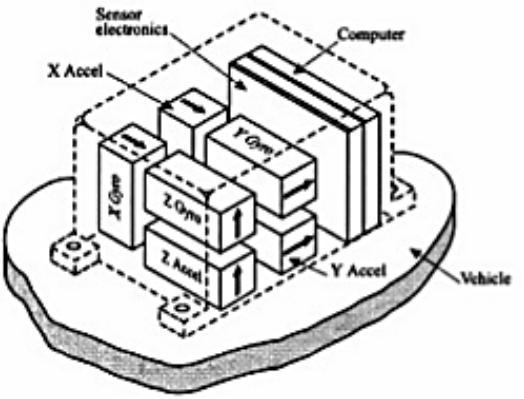
Gambar 2.3 Konfigurasi Umum INS

IMU merupakan perangkat yang melakukan pengukuran data pada INS yang pada umunya terdiri dari tiga-sumbu ortogonal akselerometer, tiga-sumbu ortogonal giroskop, dan tiga-sumbu ortogonal magnetometer. Data yang telah terukur diolah menggunakan perangkat komputer untuk mendapatkan posisi dan orientasi kendaraan [5]. Dengan memproses sinyal dari sensor-sensor tersebut, dapat dimungkinkan untuk mencari posisi dan orientasi dari suatu benda [4, 6].

INS digunakan secara luas termasuk untuk navigasi pesawat, rudal, pesawat luar angkasa, kapal selam, dan kapal laut. Perkembangan terbaru dari konstruksi perangkat MEMS (*Micro Electro-Mechanical System*) membuat perancangan sistem navigasi inersia yang kecil dan ringan menjadi memungkinkan [4].

Pada Gambar 2.3, menampilkan diagram blok INS yang terdiri dari akselerometer, giroskop, magnetometer, dan sensor tekanan pada IMU yang berfungsi untuk melakukan pengukuran data, dimana hasil pengukuran tersebut dikirim dan diolah oleh perangkat komputer.

IMU dibagi menjadi dua kategori yang dikenal dengan *Stable Platform System* dan *Strapdown System* [4]. Sistem yang digunakan pada penelitian ini adalah *Strapdown System.*



Gambar 2.4 *Strapdown Platform*

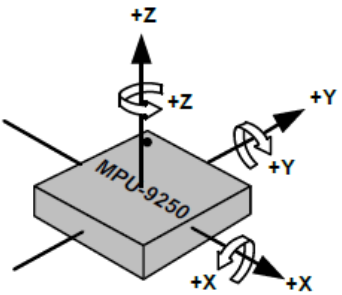
Pada *Strapdown System*, sensor inersia disematkan secara kaku didalam perangkat, yang menyebabkan jumlah keluaran diukur di dalam

body frame, bukan di dalam *global frame*. Untuk melakukan pengukuran orientasi, sinyal dari giroskop mengalamai integrasi dengan sinyal dari sensor lain (terutama akselerometer maupun magnetometer).

Untuk melakukan pengukuran posisi, sinyal dari ketiga sumbu akselerometer dipecahkan menjadi koordinat global menggunakan orientasi yang diketahui, yang didapatkan dari integrasi dengan sinyal giroskop [4]. Ilustrasi sistem dapat dilihat pada Gambar 2.4.

2.3. MPU9250

MPU9250 adalah sebuah MCM (*Multi-Chip Module*) yang terdiri dari dua cip yang saling terintegrasi satu sama lain. Cip pertama terdiri dari akselerometer 3-sumbu dan giroskop 3-sumbu. Sedangkan pada cip yang lain terdiri dari magnetometer AK8963 3-sumbu. Oleh karena itu, sensor tersebut merupakan sebuah modul sensor 9-sumbu pembaca gerakan yang mengkombinasikan akselerometer, giroskop, dan magnetometer masing-masing pada 3-sumbu, serta sebuah *Digital Motion Processor* (DMP) [7]. Pada Gambar 2.5 menampilkan sumbu *x*, *y*, dan *z* yang dijadikan sebagai sumbu pengukuran akselerometer, giroskop, dan magnetometer.



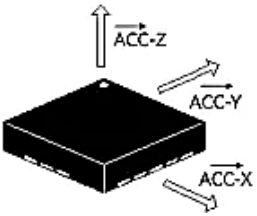
Gambar 2.5 Sumbu Pengukuran pada MPU9250

Di dalam MPU9250 terdapat ADC (*Ananlog-to-Digital Converter*) 16-bit untuk mengonversi data analog menjadi data digital. Sensor tersebut dapat menggunakan komunikasi I^2C 400 kHz dan SPI (*Serial Peripheral Interface*) 1 Mhz - 20 Mhz sebagai *slave* maupun *master*. Bentuk dari sensor MPU9250 dapat dilihat pada Gambar 2.6.

Gambar 2.6 MPU9250

2.3.1. Akselerometer

Akselerometer merupakan sensor elektro-mekanikal yang sensitif terhadap gaya statis maupun dinamis pada akselerasi. Gaya statis meliputi gravitasi, sedangkan gaya dinamis meliputi getaran dan gerakan. Sensor tersebut dapat mengukur akselerasi pada satu, dua, atau tiga sumbu [8], seperti yang ditampilkan pada Gambar 2.7. Sensor tersebut bekerja dengan menginduksi perpindahan dari *proof mass* pada sumbu x, y, z, dan pengukuran perpindahan diferensial sensor kapasitif [7].



Gambar 2.7 Akselerometer

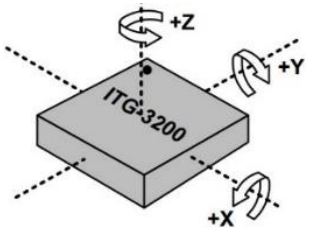
Sensor tersebut dapat digunakan untuk mengukur kemiringan yang diakibatkan oleh percepatan yang ditimbulkan oleh gaya gravitasi [5, 9], serta mengukur akselerasi yang merupakan besar perubahan dari kecepatan suatu benda. Sensor ini melakukan pengukuran dalam meter per sekon kuadrat (m/s^2) atau dalam *gforce* (g), dimana 1 g setara dengan 9.8 m/s^2 [8].

2.3.2. Giroskop

Giroskop adalah sensor yang dapat melakukan pengukuran kecepatan sudut yang berfungsi untuk untuk mengukur atau

mempertahankan orientasi, yang berdasarkan pada prinsip-prinsip momentum sudut [5, 4, 10]. Sensor tersebut dapat mengukur kecepatan sudut dalam *degrees per second* (dps). Kecepatan sudut pada dasarnya adalah pengukuran kecepatan rotasi [10]. Sensor tersebut mengukur rotasi disekitar sumbu x, y, dan z. Efek Coriolis menyebabkan getaran yang dideteksi oleh *capacitive pickoff* menghasilkan sinyal. Sinyal yang dihasilkan kemudian dikuatkan, dimodulasi, dan difilter untuk menghasilkan gelombang yang proporsional dengan kecapatan sudut [7].

Benda yang berotasi pada suatu sumbu mempunyai kecepatan sudut. Jika sensor berada tegak lurus dengan sumbu *z*, maka sensor dapat mengukur kecepatan sudut pada sumbu *z*. Sedangkan kedua sumbu yang lain tidak melakukan pengukuran rotasi apapun. Seperti yang ditampilkan pada Gambar 2.8, sensor tersebut dapat mengukur kecepatan rotasi yang terjadi di sekitar sumbu *x*, *y*, dan *z* [10, 11].

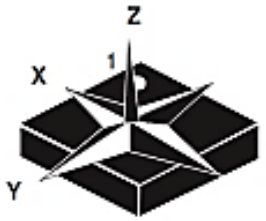


Gambar 2.8 Giroskop

2.3.3. Magnetometer

Magnetometer adalah sensor yang dapat melakukan pengukuran kekuatan dan arah medan magnet [5, 11, 12]. Sensor ini juga digunakan untuk melakukan kalibrasi elektromagnet dan magnet permanen serta untuk mengetahui magnetisasi pada sebuah benda [12]. Sensor tersebut melakukan pengukuran pada sumbu *x*, *y*, dan *z* seperti yang ditunjukkan pada Gambar 2.9.

Teknologi sensor Hall dengan sensitivitas tinggi digunakan pada magnetometer. Bagian sensor dari *intergrated circuit* (IC) menggabungkan sensor magnetik untuk mendeteksi magnet terestrial pada sumbu x-y-z, rangkaian pengendali sensor, rantai penguat sinyal, dan rangkaian aritmatika untuk memproses sinyal dari tiap sensor [7].



Gambar 2.9 Magnetometer

Berdasarkan metode kalibrasi, terdapat dua jenis magnetometer yang digunakan untuk mengukur medan magnet bumi, yaitu *absolute* dan *relative*. Magnetometer *absolute* dikalibrasi dengan mengacu pada konstanta internal sensor yang telah diketahui. Sedangkan magnetometer *relative* harus dikalibrasi dengan mengacu pada medan magnet yang diukur dengan akurat [12].

2.4. MS5837

MS5837 adalah sebuah modul sensor tekanan beresolusi tinggi dengan komunikasi I^2C yang berfungsi sebagai sistem pengukur kedalaman air dengan resolusi sebesar 2 mm. Sensor ini juga dapat mengukur temperatur dengan resolusi tinggi yang dapat diimplementasikan dalam sistem pengukuran kedalaman [13].

Gambar 2.10 MS5837

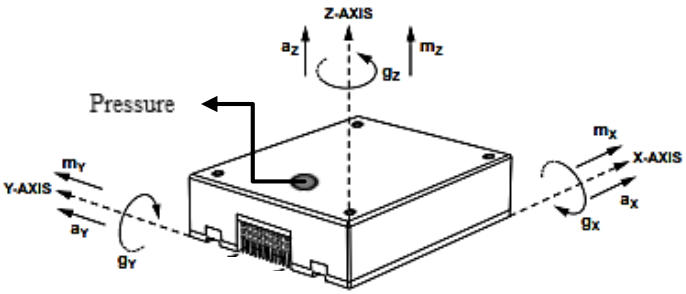
Sensor tersebut bekerja dengan mengukur tekanan dan suhu sekitar yang bekerja pada sensor. Hasil pengukuran suhu tersebut kemudian diolah untuk menghasilkan nilai tekanan akhir yang dijadikan data dalam pengukuran tekanan air dengan menggunakan persamaan tekanan hidrostatis. Bentuk dari sensor dapat dilihat pada Gambar 2.10.

2.5. *Degree of Freedom*

Degree of Freedom atau DoF adalah jumlah sumbu dan sensor yang dikombinasikan pada sistem navigasi untuk menyeimbangkan sebuah pesawat, helikopter, atau robot [14]. Jumlah sumbu yang umum digunakan pada sistem navigasi meliputi:

a. 3 DoF, terdiri dari masing-masing 3-sumbu akselerometer atau giroskop
b. 6 DoF, terdiri dari masing-masing 3-sumbu akselerometer dan giroskop
c. 9 DoF, terdiri dari masing-masing 3-sumbu akselerometer, giroskop, dan magnetometer
d. 10 DoF, terdiri dari masing-masing 3-sumbu akselerometer, giroskop, magnetometer dan 1-sumbu sensor tekanan
e. 11 DoF, terdiri dari masing-masing 3-sumbu akselerometer, giroskop, dan magnetometer dikombinasikan dengan 1-sumbu sensor tekanan DoF dan modul GPS

Pada Gambar 2.11 menampilkan sensor dengan 10 DoF yang terdiri dari akselerometer, giroskop, dan magnetometer yang masing-masing mengukur akselerasi linier, kecepatan sudut, dan medan magnet pada tiga sumbu, yaitu sumbu *x*-*y*-*z*, yang dikombinasikan dengan sensor tekanan yang mengukur tekanan hidrostatis.



Gambar 2.11 9 *Degrees of Freedom*

2.6. *Quaternion Filter*

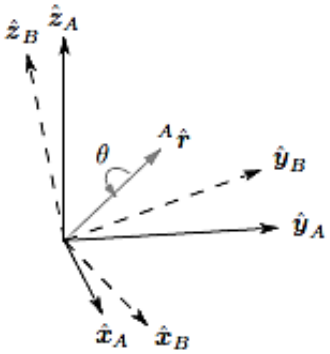
Quaternion adalah bilangan kompleks empat-dimensi yang bisa digunakan untuk menampilkan orientasi dari benda tegar atau kerangka koordinat suatu benda dalam ruang tiga-dimensi. Filter ini menggunakan

represenrasi *quaternion* yang memungkinkan data akselerometer dan magnetometer untuk digunakan dalam algoritima turunan gradien yang diturunkan dan dioptimalkan untuk menghitung arah kesalahan pengukuran giroskop sebagai turunan *quaternion* [5].

Bilangan tersebut dapat ditulis sebagai kombinasi linier dari elemen basis berupa $al + bi + cj + dk$, dimana a, b, c, dan d adalah bilangan riil. Sedangkan i, j, k adalah unit imajiner yang mengikuti hukum Hamilton. Tetapi ketika aljabar vektor lebih ditekankan daripada aljabar kuaternion, maka i, j, k menjadi unit vektor Cartesian [15].

Operasi yang dapat dilakukan pada perhitungan ini ada tiga, yaitu penjumlahan, perkalian skalar, dan perkalian *quaternion*. Perkalian tersebut tidak bersifat komutatif, hal ini disebabkan oleh hukum Hamilton [15].

Pada Gambar 2.12, menampilkan sebuah rotasi dari kerangka *B* terhadap kerangka *A* yang dapat diperoleh melalui rotasi sudut θ disekitar sumbu ${}^{A}\hat{r}$ yang didefinisikan didalam kerangka *A*.



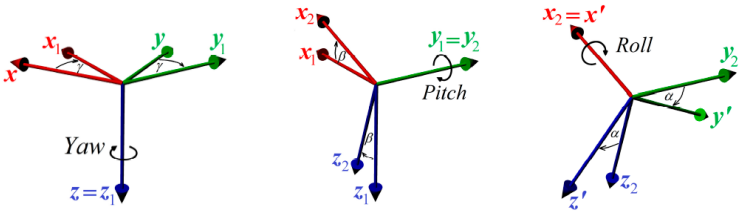
Gambar 2.12 Orientasi Kerangka B terhadap Kerangka A

Vektor saling ortogonal $\hat{x}_A, \hat{y}_A, \hat{z}_A$, dan $\hat{x}_B, \hat{y}_B, \hat{z}_B$ mendefinisikan sumbu dasar dari masing-masing koordinat kerangka *A* dan *B*.

2.7. *Euler Angles*

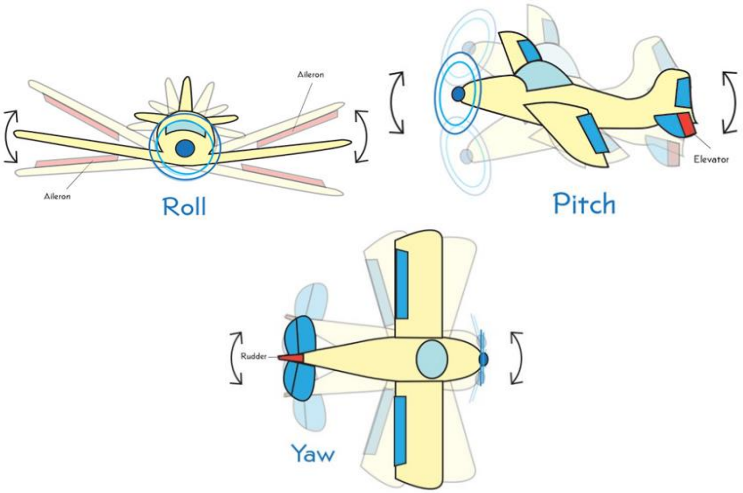
Euler angles adalah tiga sudut yang mendeskripsikan orientasi dari benda tegar yang berhubungan dengan sistem koordinat tetap. Sudut tersebut juga dapat menampilkan rotasi dari suatu objek sebagai tiga

rotasi sekuensial disekitar sumbu koordinat lokal objek. Metode tersebut merupakan cara yang paling interpretatif dan memiliki redudansi data nol karena hanya tiga bilangan real yang diperlukan [16].



Gambar 2.13 *Euler angles* dalam Konvensi *Yaw-Pitch-Roll*

Urutan rotasi sumbu yang berbeda menghasilkan rotasi resultan yang berbeda. Oleh karena itu, *Euler angles* didefinisikan sesuai dengan urutan yang dipilih (aturan). Dalam navigasi, aturan yang paling banyak digunakan adalah konvensi *z-y-x* (disebut juga dengan *Yaw-Pitch-Roll* atau *3-2-1*) [16].



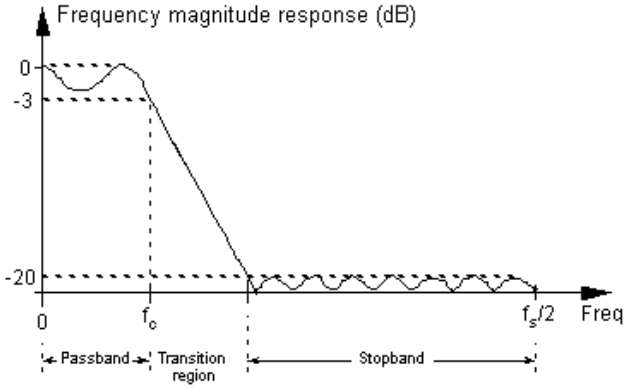
Gambar 2.14 Orientasi pada Pesawat Terbang

Pada Gambar 2.13, menampilkan rotasi pada sumbu *x*, *y*, dan *z*. Rotasi suatu objek disekitar sumbu *z* disebut dengan sudut *Yaw* (ψ). Rotasi suatu objek disekitar sumbu *y* disebut dengan sudut *Pitch* (θ). Dan Rotasi suatu objek disekitar sumbu *x* disebut dengan sudut *Roll* (ϕ) [16].

Aturan sudut *roll*, *pitch*, *yaw* pada umumnya digunakan untuk menampilkan orientasi dari suatu kendaraan. Orientasi tersebut ditampilkan melalui kerangka acuan yang sudah diatur di dalam ISO 1151 tahun 1985. Kerangka tersebut terdiri dari sudut *roll* yang dihasilkan sudut miring kendaraan, sudut *pitch* yang dihasilkan dari sudut elevasi kendaraan, dan sudut yaw yang dihasilkan dari sudut arah kendaraan. Pada Gambar 2.14, menampilkan sudut orientasi dari pesawat terbang.

2.8. *Digital Low-Pass Filter*

Low-pass filter merupakan filter yang melakukan peredaman terhadap sinyal berfrekuensi tinggi, dan melewatkan sinyal berfrekuensi rendah, seperti yang ditampilkan pada Gambar 2.15. Filter tersebut digunakan untuk mengurangi *noise* pada suatu frekuensi atau sinyal yang disebabkan oleh sinyal berfrekuensi tinggi [17].



Gambar 2.15 Visualisasi Hasil *Digital Low-Pass Filter*

Untuk melakukan proses filterisasi pada suatu sinyal, dapat digunakan perhitungan yang ditampilkan pada Persamaan 2.1.

$$S_t = \alpha \times Y_t + (1 - \alpha) \times S_{t-1} \quad (2.1)$$

Dimana S_t adalah hasil filter ketika waktu *t*. Y_t adalah data bernilai awal. α adalah koefisien yang menentukan jumlah sampel yang dihitung dengan rentang nilai 0-1. Koefisien dengan nilai kecil mengakibatkan perhitungan yang lambat karena jumlah sampel yang banyak, namun hasil filter menjadi lebih baik, dan sebaliknya [17].

2.9. Tekanan dan Kedalaman Zat Cair

Tekanan adalah ukuran dari gaya yang bekerja pada suatu benda. Tekanan pada zat cair, yang sering disebut juga dengan tekanan hidrostatis, dipengaruhi oleh tingkat kedalaman, gaya gravitasi, dan massa jenis zat cair [18].

Rumus untuk menghitung tekanan zat cair memiliki persamaan yang ditampilkan pada Persamaan 2.2.

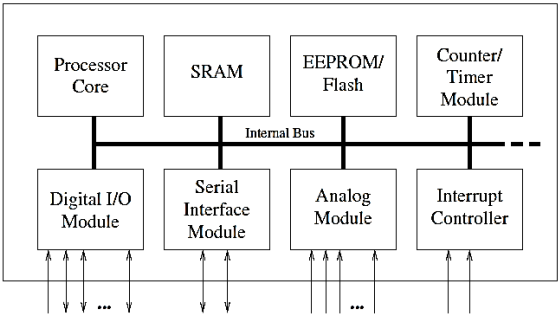
$$P_h = \rho \times g \times h \quad (2.2)$$

Dimana P_h adalah tekanan hidrostatis, dengan satuan Paskal (Pa). ρ adalah massa jenis dari zat cair, dengan satuan kg/m^3. g adalah gaya gravitasi bumi, dengan satuan m/s^2. *h* adalah kedalaman zat cair, dengan satuan meter (m). Semakin dalam benda berada dalam zat cair, semakin besar tekanan yang bekerja. Semakin besar massa jenis dari zat cair, semakin besar pula tekanan yang bekerja pada benda [18].

2.10. Mikrokontroler

Mikrokontroler adalah sebuah prosesor yang dilengapi dengan memori, pengatur waktu, pin I/O paralel, dan periferal lainnya [19]. Perangkat tersebut digunakan untuk mengolah suatu data, mengontrol rangkaian elektronik, dan menyimpan suatu program di dalam memori.

Pada Gambar 2.16, menampilkan blok diagram dari konfigurasi umum mikrokonroler. Semua komponen terkoneksi pada *internal bus* dan terintegrasi pada satu cip [19].

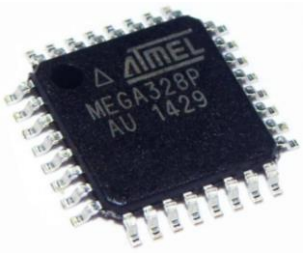


Gambar 2.16 Konfigurasi Dasar Mikrokontroler

2.10.1. ATmega328P

ATmega328P merupakan sebuah mikrokontroler CMOS (*Complementary Metal–Oxide–Semiconductor*) 8-bit berdaya rendah yang berbasis pada arsitektur RISC (*Reduced Instruction Set Computer*) pada AVR (*Alf and Vegard's Risc Processor*) [20].

Inti Atmel AVR menggabungkan set instruksi yang terdiri dari 32 register kerja yang terkoneksi secara langsung pada *Arithmetic Logic Unit* (ALU), memungkinkan dua register independed untuk bisa diakses dalam sebuah instruksi tunggal yang dieksekusi dalam satu siklus *clock* [20]. Bentuk dari ATmega328P dapat dilihat pada Gambar 2.17 [20].



Gambar 2.17 ATmega328P

2.10.2. Arduino IDE

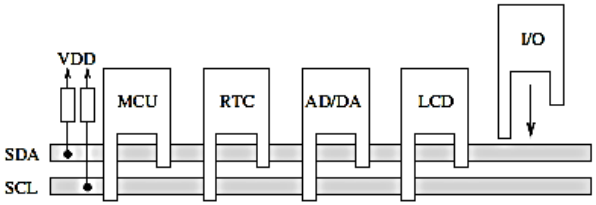
Arduino IDE (*Integrated Development Environment*) merupakan sebuah perangkat lunak pengembangan yang terintegrasi dengan

mikrokontroler Arduino dan Genuino. Perangkat lunak tersebut berisi sebuah editor teks untuk menulis kode, area pesan, konsol teks, *toolbar* dengan tombol untuk fungsi umum, dan serangkaian menu. Perangkat lunak ini terhubung dengan perangkat keras Arduino dan Genuino untuk mengunggah program dan melakukan komunikasi [21].

Pada Arduino IDE terdapat *serial monitor* yang berfungsi untuk menampilkan pengiriman serial dari mikrokontroler melalui USB (*Universal Serial Bus*) atau konektor serial [21].

2.10.3. I^2C

I^2C (*Inter-Integrated Circuit*) merupakan protokol komunikasi yang dilakukan antar mikrokontroler, mikrokontroler ke perangkat lain, atau antar perangkat, secara serial tersinkronisasi yang menggunakan menggunakan dua jalur untuk melakukan komunikasi, yaitu SCL (*Serial Clock Line*) dan SDA (*Serial Data Line*), dimana SCL merupakan jalur untuk pengiriman *clock* dan SDA merupakan jalur untuk pengiriman data. Protokol tersebut merupakan jenis komunikasi *serial synchronous half-duplex bidirectional*, dimana data ditransmisikan dan diterima hanya melalui satu jalur SDA (serial), komunikasi menggunakan *clock* yang bersal dari perangkat master melalui jalur SCL (*synchronous*), penggunaan jalur data secara bergantian antar perangkat (*half-duplex*), dan data dapat ditransmisikan dari dan ke sebuah perangkat (*bidirectional*) [22].

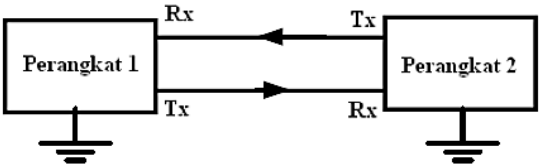


Gambar 2.18 Konfigurasi I^2C

Salah satu keunggulan dari konfigurasi komunikasi I^2C adalah mudiah untuk diperpanjang. Perangkat baru dapat ditambahkan ke dalam konfigurasi dengan cara mengkoneksikannya, seperti yang ditampilkan pada Gambar 2.18.

2.10.4. Komuniaksi Serial

Komunikasi serial adalah pengiriman data secara sekuensial atau pengiriman data secara satu persatu secara berurutan melalui sebuah kanal informasi. *Port* serial pada komputer memungkinkan untuk melakukan komunikasi dua arah atau *full-duplex* yaitu dapat mengirim dan menerima data secara bersamaan atau lebih dikenal dengan istilah komunikasi serial *asynchronous* [9].



Gambar 2.19 Komunikasi serial

Komunikasi serial digunakan untuk melakukan komunikasi antara mikrokontoler dengan komputer, mikrokontroler dengan perangkat lain, dan komunikasi antar mikrokontroler. Komunikasi tersebut mengirim data melalui pin Tx (*Transmitter*) dan menerima data melalui pin Rx (*Receiver*), seperti yang ditampilkan pada Gambar 2.19. Jumlah data yang dikirim atau diterima dalam satu detik pada komunikasi serial disebut dengan *baud rate* yang memiliki satuan bps (*bit per second*) [9].

Halaman ini sengaja dikosongkan

BAB III
PERANCANGAN *INERTIAL NAVIGATION SYSTEM*

3.1. Persyaratan Perancangan Sistem

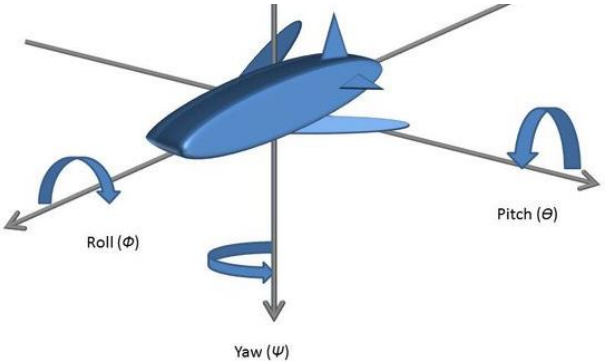
Sistem navigasi pada DPV seri terbaru memerlukan peningkatan teknologi untuk mengoptimalkan kinerja kendaraan. Oleh karena itu, INS yang didalamnya terdapat IMU dan komputer dipilih sebagai solusi untuk meningkatkan sistem navigasi pada kendaraan. IMU yang digunakan terdiri dari MPU9250 dan MS5837 yang didalamnya terdapat akselerometer yang melakukan pengukuran akselerasi linier pada 3-sumbu, giroskop yang melakukan pengukuran kecepatan sudut pada 3-sumbu, magnetometer yang melakukan pengukuran medan magnet pada 3-sumbu, dan sensor tekanan yang mengukur tekanan hidrostatis, dimana jumlah sensor-sensor yang dikombinasikan tersebut membentuk 10 DoF dalam INS. Hasil pengukuran IMU diolah oleh *board* INS yang berbasis pada mikrokontroler ATmega328P untuk menghasilkan data berupa sudut orientasi dan kendalaman air.

INS yang dirancang sebagai sistem navigasi pada DPV memiliki beberapa persyaratan sistem yang harus dipenuhi dimana bertujuan agar sistem yang dibuat memenuhi kebutuhan kendaraan dan menjadi acuan perancangan sistem, dimana syarat-syarat tersebut meliputi:

- Menggunakan tegangan suplai sebesar 5 VDC
- Mendukung komunikasi serial UART
- Kedap terhadap air
- Menghasilkan pengukuran sudut orientasi *roll*, *pitch*, *yaw* dan kedalaman penyelaman kendaraan
- Memiliki nilai kesalahan dan derau maksimal sebesar 3^{o} pada hasil pengukuran sudut orientasi kendaraan.
- Memiliki nilai kesalahan dan derau maksimal sebesar 5 cm pada hasil pengukuran kedalaman penyelaman kendaraan.

Syarat-syarat tersebut merupakan tolok ukur berhasil atau tidaknya sistem yang telah dibuat.

Pengukuran sudut orientasi kendaraan yang terdapat pada INS berdasarkan pada aturan sumbu kendaraan air yang telah ada, seperti yang ditampilkan pada Gambar 3.1. Dimana sudut *roll* menunjukkan besar sudut kemiringan, sudut *pitch* menunjukkan besar sudut elevasi, dan sudut *yaw* menunjukkan arah dari kendaraan.



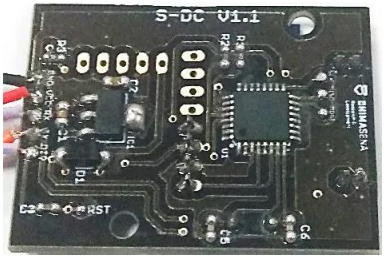
Gambar 3.1 Sumbu orientasi pada DPV

3.2. Perancangan Perangkat Keras

Perancangan perangkat keras pada INS berupa perancangan *board* INS, konfigurasi sensor MPU9250 dan MS5837 dengan board INS, serta pengedapan *board* INS.

3.2.1. Perancangan *Board* INS

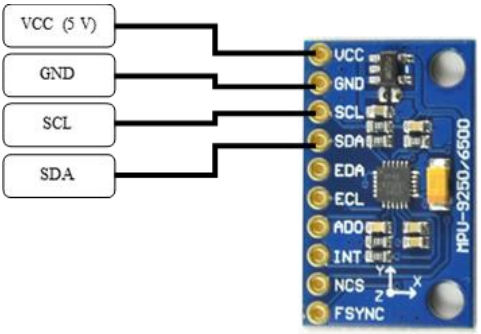
Board INS merupakan sebuah mikrokontroler yang berbasis pada ATmega328P, *crystal oscillator* (XTAL) 16 MHz, *voltage regulator* AMS1117 3.3 V, serta komponen elektronika pendukung seperti kapasitor dan resistor. *Board* tersebut berfungsi sebagai tempat sensor MPU9250 dan MS5837 disematkan, serta sebagai pengolah data pengukuran 9 DoF untuk menjadi data orientasi dan kedalaman. Bentuk dari board tersebut dapat dilihat pada Gambar 3.2.



Gambar 3.2 *Board* INS

3.2.2. Konfigurasi MPU9250

MPU9250 disematkan pada *board* INS dengan menggunakan tegangan *input* sebesar 5 V dan menggunakan protokol I^2C untuk melakukan komunikasi dengan mikrokontroler.



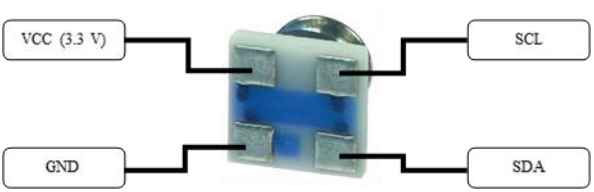
Gambar 3.3 Konfigurasi MPU9250

Seperti yang ditampilkan pada Gambar 3.3, tegangan mengalir dari pin VCC mikrokontroler menuju pin VCC pada sensor. Sambungan *ground* terhubung antara pin GND pada mikrokontroler dengan pin GND pada sensor. Pin SCL pada mikrokontroler terhubung dengan pin SCL pada sensor untuk pengiriman *clock*. Data pengukuran yang berupa akselerasi, kecepatan sudut, serta medan magnet pada sumbu *x*, *y*, dan *z* dikirim melewati pin SDA pada sensor dan diterima oleh pin SDA pada mikrokontroler.

3.2.3. Konfigurasi MS5837

MS5837 disematkan pada *board* INS dengan menggunakan tegangan *input* sebesar 3.3 V dan menggunakan protokol I^2C untuk melakukan komunikasi dengan mikrokontroler.

Seperti yang ditampilkan pada Gambar 3.4, tegangan mengalir dari pin VCC *voltage regulator* 3.3 V menuju pin VCC pada sensor. Sambungan *ground* terhubung antara pin GND pada mikrokontroler dengan pin GND pada sensor. Pin SCL pada mikrokontroler terhubung dengan pin SCL pada sensor untuk pengiriman *clock*. Data pengukuran yang tekanan dan temperatur dikirim melewati pin SDA pada sensor dan diterima oleh pin SDA pada mikrokontroler.



Gambar 3.4 Konfigurasi MS5837

3.2.4. Pengedapan *Board* INS

Salah satu fungsi dari INS pada penelitian ini adalah untuk mengukur kedalaman penyelaman dari DPV. Untuk mendapatkan data tersebut, sensor harus melakukan kontak langsung dengan air untuk mengukur tekanan hidrostatis.

Agar *board* INS tidak mengalami kerusakan ketika melakukan pengukuran tersebut, maka *board* dilapisi dengan sebuah lapisan padat untuk mencegah air masuk dan merusak sistem INS. Pelapisan tersebut dilakukan dengan teknik pengecoran menggunakan *sealant* pada seluruh bagian INS, kecuali pada ujung sensor MS5837 untuk melakukan kontak dengan air. Hasil dari pengedapan board INS dapat dilihat pada Gambar 3.5.

Gambar 3.5 Board MPU9250 yang Telah Dikedapkan

3.3. Perancangan Perangkat Lunak

Perancangan perangkat lunak pada INS berupa pemrograman sensor MPU9250, MS5837, dan integrasi INS. Program dibuat menggunakan perangkat lunak Arduino IDE.

3.3.1. Pemrograman Sensor MPU9250

Pemrograman MPU9250 dimulai dengan inisialiasi *library* <Wire.h> untuk komunikasi I^2C dengan mikrokontroler. Sensor MPU9250 bekerja berdasarkan register yang dibaca. Register diinialisasi pada program yang nantinya akan digunakan untuk mengaktifkan fungsi-fungsi pada sensor. Variabel-variabel diinisialisasi pada program untuk menyimpan nilai atau data.

Program dilanjutkan dengan inisialisasi Wire.begin() untuk memulai komunikasi I^2C antara sensor dengan mikrokontroler dan Serial.begin(38400) untuk memulai komunikasi serial antara mikrokontroler dengan komputer dengan *baudrate* sebesar 38400 bps. Komunikasi antara mikrokontroler dengan sensor dimulai dengan mengirimkan register kepada sensor. Register WHO_AM_I digunakan untuk mengetahui apakah komunikasi berjalan dengan sukses atau tidak.

Untuk mengetahui sensor dalam keadaan layak untuk digunakan, dilakukan proses *self-test*. Proses tersebut memungkinkan pengujian bagian mekanis dan elektronik dari sensor untuk masing-masing sumbu pengukuran dengan menggunakan register pada giroskop dan akselerometer.

Self-test (ST) menghasilkan sebuah respon (STR) yang digunakan untuk menentukan apakah sensor telah lulus atau gagal. Batas ideal nilai perbedaan dari FT pada STR untuk giroskop sebesar 3.6% dan sebesar 2.9% untuk akselerometer.

Untuk pengukuran data, diawali dengan pengiriman register dan pemilihan rentang skala penuh pada sensor. Pengukuran awal menghasilkan data akselerasi, kecepatan sudut, dan medan magnet masing-masing pada sumbu *x*, *y*, dan *z*. Data-data tersebut kemudian diolah menggunakan *digital low-pass filter* pada Persamaan 3.1 untuk mengurangi *noise* pada data.

$$Filter_t = 0.98 \times Data + 0.02 \times Filter_{t-1} \tag{3.1}$$

Data yang telah difilter kemudian diolah kembali menggunakan perhitungan kuaternion dengan metode *Madgwick Quaternion Update*. Giroskop mengukur kecepatan sudut disekitar sumbu *x*, *y*, dan *z*, masing-masing disebut dengan ω_x, ω_y, dan ω_z yang diolah menjadi vektor ${}^S\omega$, menggunakan Persamaan 3.2.

$${}^S\omega = \begin{bmatrix} 0 & \omega_x & \omega_y & \omega_z \end{bmatrix} \tag{3.2}$$

Orientasi pada kerangka bumi terhadap kerangka sensor pada waktu *t*, ${}^{S}_{E}q_{\omega,t}$, bisa dihitung menggunakan turunan kuaternion ${}^{S}_{E}\dot{q}_{\omega,t}$ pada Persamaan 3.3 dengan kondisi awal orientasi yang diketahui.

$$ {}^{S}_{E}\dot{q}_{\omega,t} = \frac{1}{2} {}^{S}_{E}\hat{q}_{est,t-1} \otimes {}^{S}\omega_t \tag{3.3} $$

Dimana ${}^{S}\omega_t$ adalah kecepatan sudut pada waktu *t* dan ${}^{S}_{E}\hat{q}_{est,t-1}$ adalah perkiraan orientasi sebelumnya.

Akselerometer 3-sumbu mengukur besar dan arah gaya gravitasi di dalam kerangka sensor yang terhubung dengan akselerasi linier. Magnetometer mengukur besar dan arah medan magnet bumi yang terhubung dengan fluks magnet lokal dan distorsi. Orientasi sensor, ${}^{S}_{E}\hat{q}$, menyejajarkan referensi arah medan magnet yang telah ditentukan dalam kerangka bumi, ${}^{E}\hat{d}$, dengan arah medan magnet terukur di dalam kerangka sensor, ${}^{S}\hat{s}$.

Algoritma penurunan gradient digunakan untuk mengoptimalkan perhitungan. Algoritma tersebut menghasilkan perkiraan orientasi, ${}^{S}_{E}\hat{q}_{n+1}$, untuk n-iterasi berdasarkan pada tebakan awal orientasi, ${}^{S}_{E}\hat{q}_0$, dan ukuran langkah μ. Perhit ungan $\nabla f\left({}^{S}_{E}\hat{q}_k, {}^{E}\hat{d}, {}^{S}\hat{s}\right)$ menghitung gradien dari permukaan solusi yang didefinisikan oleh fungsi objektif dan Jacobian.

Jika arah medan gravitasi diasumsikan memiliki komponen dalam 1 atau 2 sumbu utama dalam kerangka koordinat global, arah gravitasi mendefinisikan sumbu vertical sumbu z sebagai ${}^{E}\hat{g}$ pada Persamaan 3.4. Mensubtitusi ${}^{E}\hat{g}$ dan menormalkan pengukuran akselerometer pada Persamaan 3.5, ${}^{S}\hat{a}$, untuk masing-masing ${}^{E}\hat{d}$ dan ${}^{S}\hat{s}$, dalam persamaan fungsi objektif dan Jacobian menghasilkan persamaan $f_g\left({}^{S}_{E}\hat{q}, {}^{S}\hat{a}\right)$ dan $J_g\left({}^{S}_{E}\hat{q}\right)$ yang ditunjukkan pada Persamaan 3.6 dan Persamaan 3.7.

Medan magnet bumi dapat memiliki bagian dalam sumbu horisontal dan vertikal, ${}^{E}\hat{b}$, seperti yang ditunjukkan pada Persamaan 3.8. Mensubstitusi ${}^{E}\hat{b}$ dan menormalkan pengukuran magnetometer, ${}^{S}\hat{m}$, yang terdapat pada Persamaan 3.9 masing-masing untuk ${}^{E}\hat{d}$ dan ${}^{S}\hat{s}$, dalam persamaan fungsi objektif dan Jacobian menghasilkan persamaan $f_b\left({}^{S}_{E}\hat{q}, {}^{E}\hat{b}, {}^{S}\hat{m}\right)$ dan $J_b\left({}^{S}_{E}\hat{q}, {}^{E}\hat{b}\right)$ yang terdapat pada Persamaan 3.11.

$$^{E}\hat{g} = [0 \quad 0 \quad 0 \quad 1] \tag{3.4}$$

$$^{S}\hat{a} = \begin{bmatrix} 0 & a_x & a_y & a_z \end{bmatrix} \tag{3.5}$$

$$f_g\left(^{S}_{E}\hat{q},\ ^{S}\hat{a}\right) = \begin{bmatrix} 2(q_2q_4 - q_1q_3) - a_x \\ 2(q_1q_2 + q_3q_4) - a_y \\ 2\left(\frac{1}{2} - q_2^2 - q_3^2\right) - a_z \end{bmatrix} \tag{3.6}$$

$$J_g\left(^{S}_{E}\hat{q}\right) = \begin{bmatrix} -2q_3 & 2q_4 & -2q_1 & 2q_2 \\ 2q_2 & 2q_1 & 2q_4 & 2q_3 \\ 0 & -4q_2 & -4q_3 & 0 \end{bmatrix} \tag{3.7}$$

$$^{E}\hat{b} = [0 \quad b_x \quad 0 \quad b_z] \tag{3.8}$$

$$^{S}\hat{m} = \begin{bmatrix} 0 & m_x & m_y & m_z \end{bmatrix} \tag{3.9}$$

$$\begin{aligned} &f_b\left(^{S}_{E}\hat{q},\ ^{E}\hat{b},\ ^{S}\hat{m}\right) \\ &= \begin{bmatrix} 2b_x\left(\frac{1}{2} - q_3^2 - q_4^2\right) + 2b_z(q_2q_4 - q_1q_3) - m_x \\ 2b_x(q_2q_3 - q_1q_4) + 2b_z(q_1q_2 + q_3q_4) - m_y \\ 2b_x(q_1q_3 + q_2q_4) + 2b_z\left(\frac{1}{2} - q_2^2 - q_3^2\right) - m_z \end{bmatrix} \end{aligned} \tag{3.10}$$

$$J_b\left(^{S}_{E}\hat{q},\ ^{E}\hat{b}\right) = \begin{bmatrix} 2b_zq_3 & 2b_zq_4 & -4b_xq_3\text{-}2b_zq \\ -2b_xq_4\text{+}2b_zq_2 & 2b_xq_3\text{+}2b_zq_1 & 2b_xq_2\text{+}2b_zq \\ 2b_xq_3 & 2b_xq_4\text{-}4b_zq_2 & 2b_xq_1\text{-}4b_zq \\ -4b_xq_4\text{-}2b_zq_2 & & \\ -2b_xq_1\text{+}2b_zq_3 & & \\ 2b_xq_2 & & \end{bmatrix} \tag{3.11}$$

$$f_{g,b}\left(^{S}_{E}\hat{q},\ ^{S}\hat{a},\ ^{E}\hat{b},\ ^{S}\hat{m}\right) = \begin{bmatrix} f_g\left(^{S}_{E}\hat{q},\ ^{S}\hat{a}\right) \\ f_b\left(^{S}_{E}\hat{q},\ ^{E}\hat{b},\ ^{S}\hat{m}\right) \end{bmatrix} \tag{3.12}$$

$$J_{g,b}\left(^{S}_{E}\hat{q},\ ^{E}\hat{b}\right) = \begin{bmatrix} J_g^T\left(^{S}_{E}\hat{q}\right) \\ J_b^T\left(^{S}_{E}\hat{q},\ ^{E}\hat{b}\right) \end{bmatrix} \tag{3.13}$$

Pengukuran gravitasi atau medan magnet sendiri tidak bisa menghasilkan pengukkuran orientasi. Untuk bisa menghasilkan data tersebut, pengukuran dan arah referensi dari kedua medan dapat dikombinasikan menjadi $f_{g,b}\left({}^{S}_{E}\hat{q},\ {}^{S}\hat{a},\ {}^{E}\hat{b},\ {}^{S}\hat{m}\right)$ dan $J_{g,b}\left({}^{S}_{E}\hat{q},\ {}^{E}\hat{b}\right)$ seperti yang terdapat pada Persamaan 3.12 dan Persamaan 3.13.

Optimalisasi menggunakan iterasi per waktu sampel diambil yang dinotasikan dengan μ menggunakan persamaan ${}^{S}_{E}q_{\nabla,t}$ yang ditampilkan pada Persamaan 3.14. Persamaan tersebut menghitung perkiraan orientasi pada waktu t berdasarkan pada perkiraan orientasi sebelumnya, ${}^{S}_{E}\hat{q}_{est,t-1}$, dan fungsi objektif gradien ∇f didefinisikan oleh pengukuran sensor ${}^{S}\hat{a}_t$ dan ${}^{S}\hat{m}_t$ pada waktu t, seperti yang ditampilkan pada Persamaan 3.15.

$$ {}^{S}_{E}q_{\nabla,t} = {}^{S}_{E}\hat{q}_{est,t-1} - \mu_t \frac{\nabla f}{\|\nabla f\|} \tag{3.14} $$

$$ \nabla f = J^{T}_{g,b}\left({}^{S}_{E}\hat{q}_{est,t-1},\ {}^{E}\hat{b}_t\right) f_{g,b}\left({}^{S}_{E}\hat{q}_{est,t-1},\ {}^{S}\hat{a},\ {}^{E}\hat{b}_t,\ {}^{S}\hat{m}\right) \tag{3.15} $$

Sudut Euler kemudian mengolah bilangan kuaternion yang telah didapatkan untuk menjadi data orientasi yang meliputi sudut *roll*, *pitch*, dan *yaw* dimana ditunjukkan secara berturut-turut pada Persamaan 3.16, Persamaan 3.17, dan Persamaan 3.18.

$$ \phi = \text{Atan2}(2q_3q_4 - 2q_1q_2, 2q_1^2 + 2q_4^2 - 1) \tag{3.16} $$

$$ \theta = -\sin^{-1}(2q_2q_4 - 2q_1q_3) \tag{3.17} $$

$$ \psi = \text{Atan2}(2q_2q_3 - 2q_1q_4, 2q_1^2 + 2q_2^2 - 1) \tag{3.18} $$

Dimana data *yaw* (ψ) digunakan sebagai data pada kompas.

Data orientasi kemudian dikirim menuju komputer menggunakan komunikasi serial dan ditampilkan didalam *serial monitor* pada Arduino IDE.

3.3.2. Pemrograman Sensor MS5837

Pemrograman MS5837 dimulai dengan inisialiasi *library* <Wire.h> untuk komunikasi I^2C dengan mikrokontroler. Sensor MS5837 bekerja berdasarkan register yang dibaca. Register diinialisasi

pada program yang nantinya akan digunakan untuk mengaktifkan fungsi-fungsi pada sensor. Variabel-variabel diinisialisasi pada program untuk menyimpan nilai atau data.

Program dilanjutkan dengan inisialisasi Wire.begin() untuk memulai komunikasi I^2C antara sensor dengan mikrokontroler dan Serial.begin(38400) untuk memulai komunikasi serial antara mikrokontroler dengan komputer dengan *baudrate* sebesar 38400 bps. Komunikasi antara mikrokontroler dengan sensor dimulai dengan mengirimkan register kepada sensor.

Pengukuran data diawali dengan pengiriman register kepada sensor untuk mengukur tekanan awal dan temperatur. Data tersebut dikirim dari sensor menuju mikrokontroler dan diolah untuk menjadi data pengukuran tekanan akhir , dimana nilai temperatur yang terukur mempengaruhi nilai tekanan.

Data pengukuran tekanan akhir digunakan untuk menghitung data kedalaman air. Perhitungan tersebut menggunakan persamaan tekanan hidrostatis yang ditampilkan pada Persamaan 3.20.

$$Depth = \frac{P}{\rho \times g} \tag{3.20}$$

Dimana *Depth* merupakan kedalaman air dalam meter (m). *P* merupakan tekanan hidrostatis dalam bar. ρ merupakan massa jenis zat cair sebesar 997.77 kg/m^3, sesuai dengan massa jenis air pada suhu ruangan (25^o C). *g* merupakan gaya gravitasi bumi sebesesar 9.807 m/s^2.

Data kedalaman yang telah terukur kemudian dikirim menuju komputer menggunakan komunikasi serial dan ditampilkan pada *serial monitor* dalam Arduino IDE.

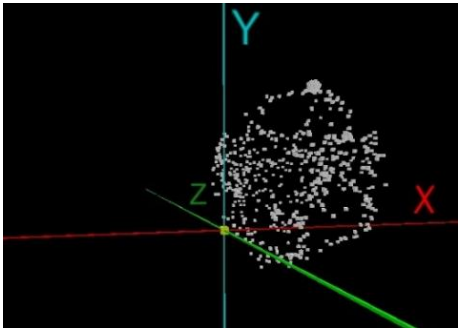
3.3.3. Program Integrasi IMU

Program dari sensor MPU9250 dan MS5837 diintegrasikan agar dapat bekerja secara bersama-sama dalam satu *sketch* program Arduino. Dalam program integrasi, terdapat *selector*, yang terdiri dari mode "*indirect"* dan mode "*direct"*, dimana befungsi sebagai pengatur program dengan mengerjakan mode yang dipilih.

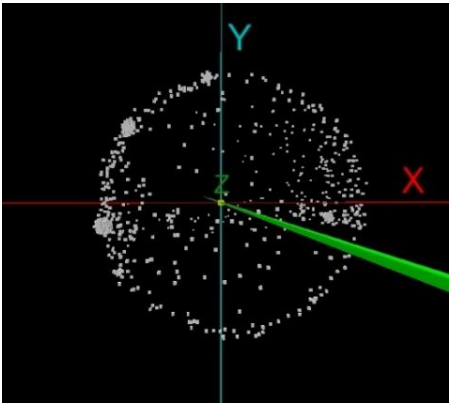
Pada mode *indirect*, program *self-test* akselerometer dan giroskop dijalankan. Kemudian dilanjutkan dengan proses pengukuran dan penampilan data. Pada mode *direct*, program pengukuran dan penampilan data dijalankan tanpa menjalankan *self-test* terlebih dahulu.

3.4. Kalibrasi Magnetometer

Kalibrasi magnetometer bertujuan untuk memperbaiki hasil pengukuran medan magnet dari MPU9250. Kalibrasi ini dilakukan secara terpisah dari kalibrasi akselerometer dan giroskop, karena pada MPU9250 tidak ada register yang berfungsi untuk melakukan kalibrasi pada magnetometer.



Gambar 3.6 Bias Belum Sesuai



Gambar 3.7 Bias Sudah Sesuai

Proses kalibrasi magnetometer dilakukan dengan menjalankan program kalibrasi magnetometer pada *sketch* yang terpisah dari program IMU. Sensor kemudian diputar dengan membentuk gerakan menyerupai

angka 8. Setelah kalibrasi selesai, pencarian nilai bias lingkungan (*environmental bias*) dilakukan menggunakan program MPU9250 dengan mengeluarkan data pengukuran medan magnet oleh magnetometer pada tiga sumbu. Data-data tersebut kemudian diolah menggunakan sebuah perangkat lunak bernama MagView. Perangkat lunak tersebut menampilkan titik-titik disekitar sumbu *x*, *y*, dan *z*. Jika titik-titik tersebut berbentuk tidak beraturan dan tidak berada di pusat pertemuan tiga sumbu seperti yang ditunjukkan pada Gambar 3.6, maka bias pada pengukuran magnetometer masih belum sesuai. Dan jika titik-titik telah membentuk lingkaran dan berada di pusan pertemuan tiga sumbu seperti yang ditunjukkan pada Gambar 3.7, maka nilai bias sudah sesuai.

Nilai bias hasil dari kalibrasi dimasukkan kedalam variabel pada program. Variabel tersebut kemudian dimasukkan dalam perhitungan medan magnet pada MPU9250.

Halaman ini sengaja dikosongkan

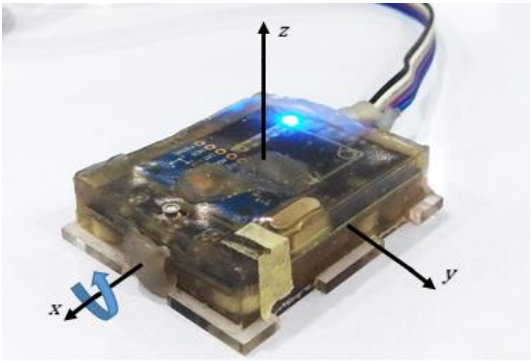
BAB IV
PENGUJIAN DAN ANALISIS

4.1. Pengujian MPU9250

Pengujian sensor MPU9250 yang digunakan sebagai IMU bertujuan untuk mengetahui kinerja dari perangkat tersebut. Pengujian diawali dengan melakukan pengukuran akselerasi linier, kecepatan sudut, dan medan magnet masing-masing pada sumbu *x*, *y*, dan *z*. Hasil pengukuran tersebut digunakan untuk mendapatkan data berupa sudut *roll*, *pitch*, dan *yaw* yang merepresentasikan orientasi dari sensor. Data orientasi tersebut ditampilkan dalam *serial monitor* pada Arduino IDE dan dianalisis untuk mengetahui nilai kesalahan, derau dalam kondisi tunak, dan faktor-faktor yang mempengaruhi kinerja sensor.

4.1.1. Pengujian Sudut *Roll*

Sudut *roll* didapatkan melalui rotasi yang dilakukan oleh sumbu *y* di sekitar sumbu *x*, seperti yang ditampilkan pada Gambar 4.1. *Roll* bernilai positif jika sumbu *y* positif menujuh ke bawah, dan bernilai negatif ketika menuju ke atas, dengan sudut awal 0^o sampai dengan 90^o.



Gambar 4.1 Rotasi *Roll* pada Sensor

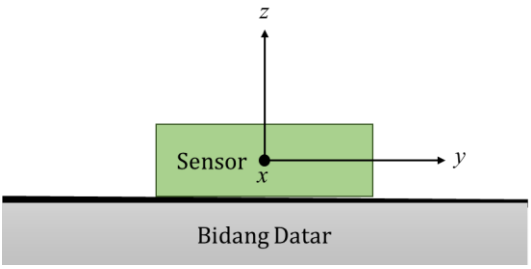
Proses pengujian dimulai dengan melakukan rotasi sumbu *y* sensor terhadap sumbu *x* dengan selisih pengambilan data setiap 10 ^{o} dan rentang derajat mulai dari 0^o sampai dengan 90^o. Data yang diperoleh

kemudian ditampilkan pada *serial monitor* dan dibandingkan dengan nilai yang ada pada busur derajat untuk mengetahui nilai kesalahan dari hasil pengukuran sensor.

Tabel 4.1 Data Pengujian Akurasi Sudut *Roll*

Busur Derajat (°)	*Roll* IMU (°)	Selisih (°)
0	0.6	0.6
10	9.5	0.5
20	19.6	0.4
30	29.7	0.3
40	40.0	0.0
50	49.1	0.9
60	59.7	0.3
70	70.4	0.4
80	79.9	0.1
90	89.2	0.8

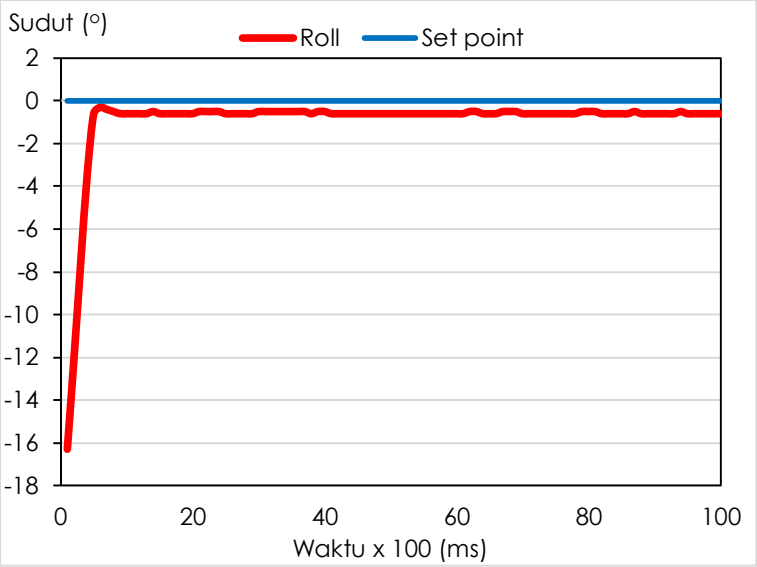
Data hasil pengujian dapat dilihat pada Tabel 4.1. Sudut *roll* yang terukur memiliki selisih terjauh sebesar 0.9° dengan selisih rata-rata sebesar 0.43°. Selisih yang didapatkan dipengaruhi oleh beberapa faktor, diantaranya adalah keterbatasan kinerja sensor dan tingkat akurasi perbandingan antara data sensor dengan perangkat pembanding.



Gambar 4.2 Pengujian Nilai Derau Sudut *Roll*

Pengujian dilanjutkan dengan menguji nilai derau sudut *roll* sensor dalam kondisi tunak dengan mengukur sudut *roll* dalam keadaan diam di atas bidang horisontal, seperti yang ditampilkan pada Gambar

4.2. Data hasil pengukuran selama rentang waktu t = 0 ms sampai dengan t = 10000 ms dikonversikan menjadi sebuah grafik yang ditampilkan pada Gambar 4.3. Berdasarkan grafik yang ditampilkan, diketahui bahwa waktu untuk sensor mencapai kondisi tunak sebesar 800 ms dengan nilai derau dalam kondisi tunak sebesar 0.1°.

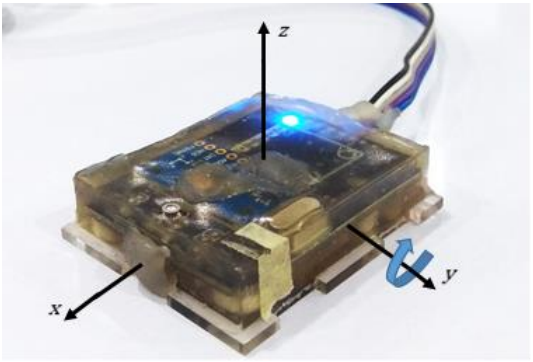


Gambar 4.3 Grafik Sudut *Roll* dalam Kondisi Tunak

4.1.2. Pengujian Sudut *Pitch*

Sudut *pitch* didapatkan melalui rotasi yang dilakukan oleh sumbu *x* di sekitar sumbu *y*, seperti yang ditampilkan pada Gambar 4.4. *Pitch* bernilai positif jika sumbu *x* positif menujuh ke atas, dan bernilai negatif ketika menuju ke bawah, dengan sudut awal 0° sampai dengan 90°.

Proses pengujian dimulai dengan melakukan rotasi sumbu *x* sensor terhadap sumbu *y* dengan rentang derajat mulai dari 0° sampai dengan 90°. Data yang diperoleh kemudian ditampilkan pada *serial monitor* dan dibandingkan dengan nilai yang ada pada busur derajat untuk mengetahui nilai kesalahan pengukuran sensor.

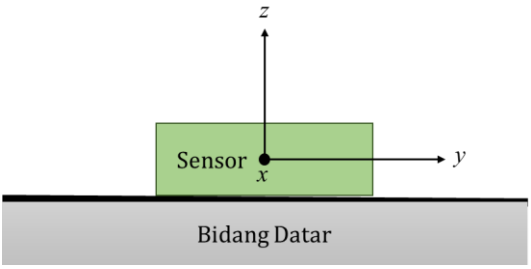


Gambar 4.4 Rotasi *Pitch* pada Sensor

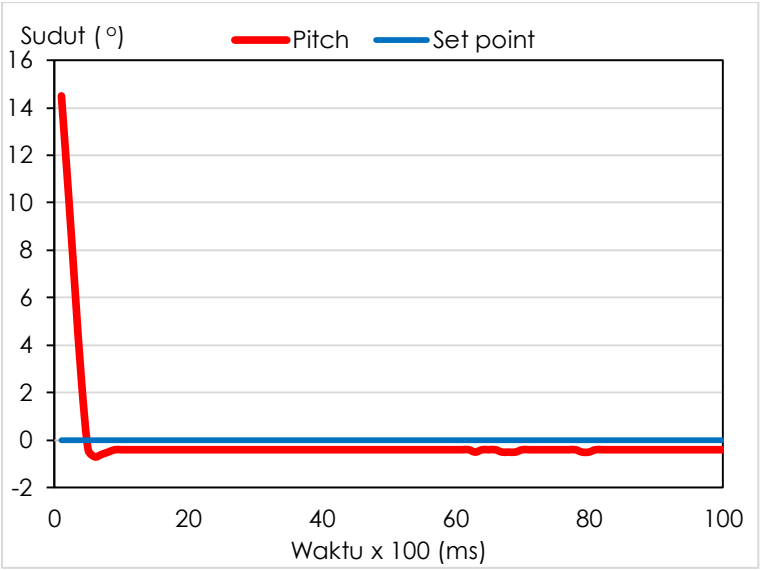
Tabel 4.2 Data Pengujian Akurasi Sudut *Pitch*

Busur Derajat (°)	*Pitch* IMU (°)	Selisih (°)
0	0.4	0.4
10	10.4	0.4
20	20.3	0.3
30	30.4	0.4
40	40.4	0.4
50	50.5	0.5
60	60.3	0.3
70	70.0	0.0
80	80.1	0.1
90	88.0	2.0

Data hasil pengujian dapat dilihat pada Tabel 4.2. Sudut *pitch* yang terukur memiliki selisih terjauh sebesar 2° dengan selisih rata-rata sebesar 0.48°. Selisih yang didapatkan dipengaruhi oleh beberapa faktor, diantaranya adalah keterbatasan kinerja sensor dan tingkat akurasi perbandingan antara data sensor dengan perangkat pembanding.



Gambar 4.5 Pengujian Nilai Derau Sudut *Pitch*



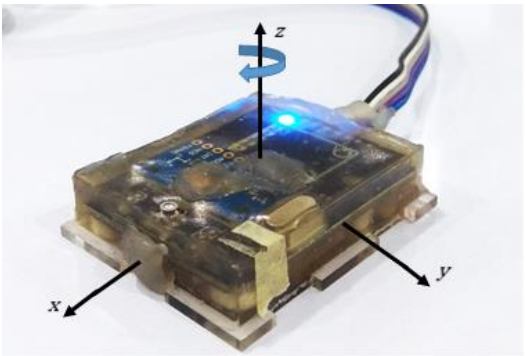
Gambar 4.6 Grafik Sudut *Pitch* dalam Kondisi Tunak

Pengujian dilanjutkan dengan menguji nilai derau pengukuran sudut *pitch* sensor dalam kondisi tunak dengan mengukur sudut *pitch* dalam keadaan diam di atas bidang horisontal, seperti yang ditampilkan pada Gambar 4.5. Data hasil pengukuran selama rentang waktu t = 0 ms

sampai dengan t = 10000 ms dikonversikan menjadi sebuah grafik yang ditampilkan pada Gambar 4.6. Berdasarkan grafik yang ditampilkan, diketahui bahwa waktu untuk sensor mencapai kondisi tunak sebesar 800 ms dengan nilai derau dalam kondisi tunak sebesar 0.1 °.

4.1.3. Pengujian Sudut *Yaw*

Sudut *yaw* didapatkan melalui rotasi yang dilakukan oleh sumbu *x* dan *y* di sekitar sumbu *z*, seperti yang ditampilkan pada Gambar 4.7. *Yaw* bernilai positif jika sumbu *z* positif berotasi searah jarum jam, dan bernilai negatif jika berorientasi berlawan arah jarum jam, dengan sudut awal 0 ° sampai dengan 180 °. Data *yaw* digunakan sebagai kompas setelah mengalami konversi, sehingga rentang sudut menjadi 0 ° sampai dengan 360 °.



Gambar 4.7 Rotasi *Yaw* pada Sensor

Proses pengujian dimulai dengan melakukan rotasi sumbu *x* dan *y* sensor terhadap sumbu *z* dengan rentang derajat mulai dari 0 ° sampai dengan 360 °. Data yang diperoleh kemudian ditampilkan pada *serial monitor* dan dibandingkan dengan nilai yang ada pada kompas analog untuk mengetahui nilai kesalahan pengukuran sensor.

Data hasil pengujian dapat dilihat pada Tabel 4.3. Sudut *yaw* pada kompas yang terukur memiliki selisih terjauh sebesar 3 ° dengan selisih rata-rata sebesar 0.96 °. Selisih yang didapatkan dipengaruhi oleh beberapa faktor, diantaranya adalah kalibrasi magnetometer yang belum optimal, radiasi medan magnet di sekitar sensor, keterbatasan kinerja

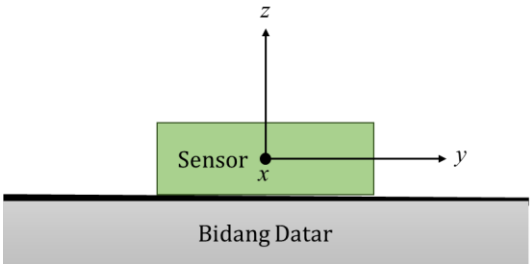
sensor, tingkat akurasi perbandingan antara data sensor dengan perangkat pembanding.

Tabel 4.3 Data Pengujian Akurasi Kompas

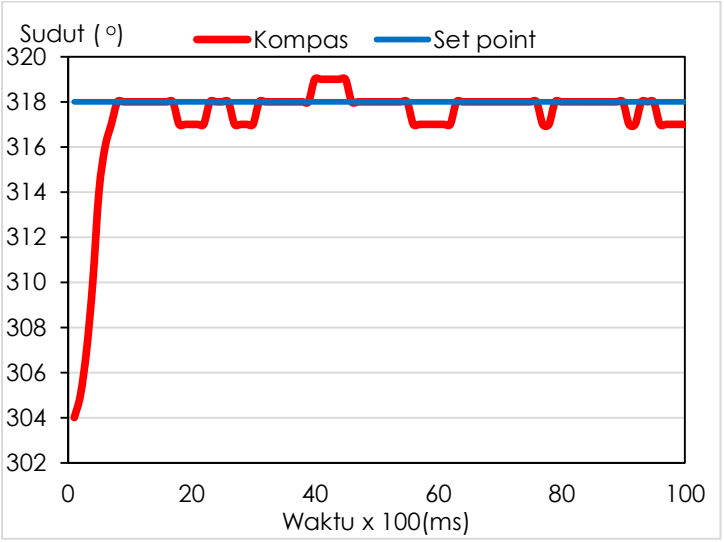
Kompas Analog (º)	*Yaw* Kompas IMU (º)	Selisih (º)
0	0	0
15	16	0
30	30	0
45	45	0
60	60	0
75	75	0
90	89	1
105	104	1
120	118	2
135	134	1
150	149	1
165	165	0
180	180	0
195	195	0
210	211	1
225	225	0
240	242	2
255	257	2
270	273	3
285	288	3
300	303	3
315	317	2
330	331	1
345	345	0

Pengujian dilanjutkan dengan menguji kestabilan pengukuran sudut *yaw* pada kompas dalam kondisi tunak dengan melakukan pengukuran dalam keadaan diam di atas bidang horisontal, seperti yang ditunjukkan pada Gambar 4.8. Data hasil pengukuran selama rentang waktu t = 0 ms sampai dengan t = 10000 ms dikonversikan menjadi sebuah grafik yang ditampilkan pada Gambar 4.9. Berdasarkan grafik yang ditampilkan, diketahui bahwa waktu untuk sensor mencapai

kondisi tunak sebesar 800 ms dengan nilai derau dalam kondisi tunak sebesar 1 °.



Gambar 4.8 Pengujian Nilai Derau Kompas



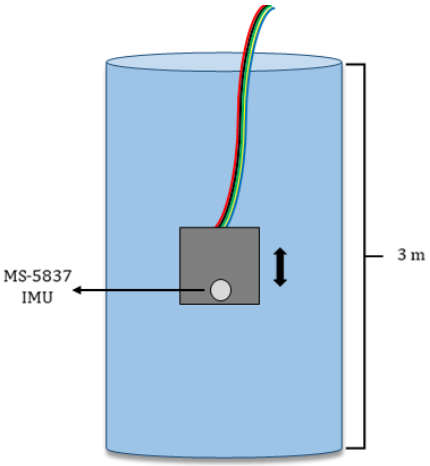
Gambar 4.9 Grafik Kompas dalam Kondisi Tunak

4.2. Pengujian MS5837

Pengujian sensor MS5837 yang digunakan sebagai IMU bertujuan untuk mengetahui kinerja dari perangkat tersebut. Pengujian

diawali dengan melakukan pengukuran tekanan hidrostatis yang bekerja terhadap sensor. Hasil pengukuran tersebut digunakan untuk mendapatkan data kedalaman penyalaman yang dijangkau oleh sensor. Data kedalaman tersebut ditampilkan dalam *serial monitor* pada Arduino IDE dan dianalisis untuk mengetahui nilai kesalahan, nilai derau dalam kondisi tunak, dan faktor-faktor yang mempengaruhi kinerja sensor.

Seperti yang ditunjukkan pada Gambar 4.10, proses pengujian dimulai dengan memasukkan sensor ke dalam kolam air dengan kedalaman lebih dari 300 cm dengan selisih pengambilan data setiap 500 cm dengan rentang dimulai dari kedalaman 0 cm sampai dengan 300 cm. Data yang diperoleh kemudian ditampilkan pada *serial monitor* dan dibandingkan dengan kedalaman kolam sesungguhnya untuk mengetahui nilai kesalahan pengukuran sensor.



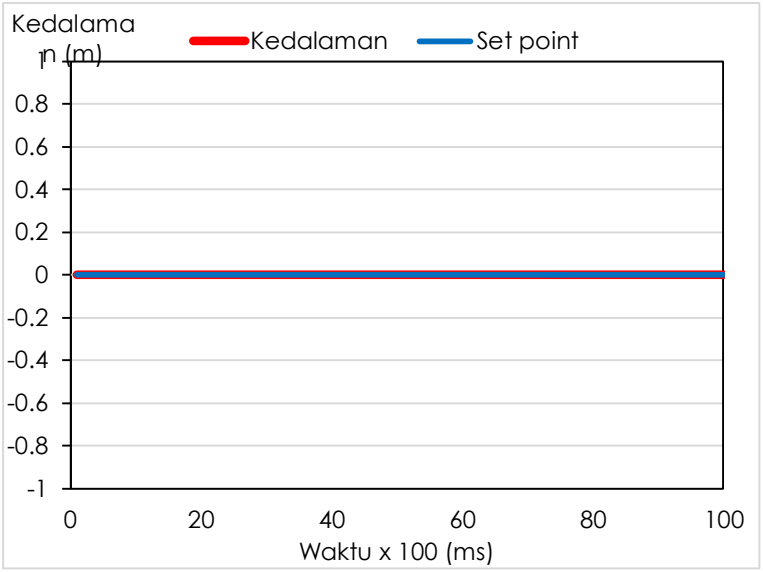
Gambar 4.10 Uji Coba Pengukuran Kedalaman

Data hasil pengujian dapat dilihat pada Tabel 4.1. Data kedalaman yang terukur memiliki selisih terjauh sebesar 2 cm dengan selisih rata-rata sebesar 1.3 cm. Selisih yang didapatkan dipengaruhi oleh beberapa faktor, diantaranya adalah keterbatasan kinerja sensor dan tingkat akurasi perbandingan antara data sensor dengan kedalaman kolam.

Tabel 4.4 Data Pengukuran Kedalaman

Kedalaman Air (cm)	Kedalaman IMU (cm)	Selisih (cm)
0	0	0
50	49	1
100	98	2
150	148	2
200	199	1
250	252	2
300	299	1

Gambar 4.11 Pengujian Nilai Derau Kedalaman



Gambar 4.12 Grafik Kedalaman dalam Kondisi Tunak

Pengujian dilanjutkan dengan menguji nilai derau data kedalaman sensor dalam kondisi tunak dengan melakukan pengukuran dalam keadaan diam di atas bidang horisontal, seperti yang ditunjukkan pada Gambar 4.11. Data hasil pengukuran selama rentang waktu t = 0 ms sampai dengan t = 10000 ms dikonversikan menjadi sebuah grafik yang ditampilkan pada Gambar 4.12. Berdasarkan grafik yang ditampilkan, diketahui bahwa waktu untuk sensor mencapai kondisi tunak sebesar 100 ms dengan nilai derau dalam kondisi tunak sebesar 0 cm.

Halaman ini sengaja dikonsongkan

BAB V
PENUTUP

Berdasarkan pengujian dan analisis yang telah dilakukan terhadap INS pada DPV, dengan menggunakan *digital low-pass filter*, serta perhitungan *Madgwick quaternion* dan *Euler angles*, didapatkan pengukuran orientasi sensor dengan tingkat kesalahan maksimal sebesar 3.00^{o} dengan nilai derau dalam kondisi tunak sebesar 0.96^{o}. Selain itu, dengan menggunakan perhitungan tekanan dan kedalaman hidrostatis, didapatkan hasil pengukuran kedalaman penyelaman dengan tingkat kesalahan maksimal sebesar 2.00 cm dengan nilai derau dalam kondisi tunak sebesar 0 cm. Dengan hasil tersebut, INS yang telah dibuat telah berhasil memenuhi semua persyaratan perancangan sistem yang wajib dipenuhi.

Untuk pengembangan selanjutnya, disarankan untuk melakukan kalibrasi magnetometer di tempat yang jauh dari perangkat yang menimbulkan radiasi elektromagnetik untuk mendapatkan hasil yang lebih baik. Selain itu, untuk pengujian nilai derau pada hasil pengukuran kedalaman penyelaman kendaraan, disarankan untuk dilakukan di dalam air agar mendapatkan data derau yang lebih valid.

Halaman ini sengaja dikosongkan

DAFTAR PUSTAKA

[1] H. Anam, L. Haris, I. A. S, P. A. R., S. G. W., A. Hidayat, A. I. R., G. Choirun, W. Nugroho, C. D. S., A. Budiyarto dan A. Budiyono, "Recent Advances in Diver Propulsion Vehicle (DPV) Ganendra RI-1: Design, Analysis, Manufacture and Performance," Bhimasena Research and Development, Sumedang, 2016.

[2] Y. Rahmi, "Navigasi," Sekolah Tinggi Teknologi Telematika Telkom, Purwokerto, 2015.

[3] H. P. Prasetyo, A. S. A. dan F. A. Iskandarianto, "Perancangan Sistem Navigasi pada Kapal (MCST-1 Ship Autopilot) untuk Mendukung Sistem Autopilot," Institut Teknologi Sepuluh Nopember, Surabaya, 2012.

[4] O. J. Woodman, "An Introduction to Inertial Navigation," University of Cambridge, Cambridge, 2007.

[5] S. O. H. Madgwick, "An Efficient Orientation Filter for Inertial and Magnetic Sensor Arrays," University of Bristol, Bristol, 2010.

[6] R. Nurfansyah, Wahyudi dan B. Setiyono, "Estimasi Sudut Orientasi Benda Menggunakan Sensor 6 DoF IMU dan Sensor Magnetometer 3 Aksis," Universitas Diponegoro, Semarang, 2013.

[7] InvenSense Inc., "MPU-9250 Product Spesification," InvenSense Inc., San Jose, 2016.

[8] T. Corinne, "Accelerometer Basics," SparkFun Electronics, 28 Maret 2013. [Online]. Available: https://www.sparkfun.com/users/194976?_ga=2.240552963.1400991660.1522659430-574162345.1519800178. [Diakses 2 April 2018].

[9] A. I. Rahmanto, "Perancangan Stabilisasi Sudut Orientasi Pitch pada Remotely Operated Vehicle (ROV) dengan Metode Kontrol Proporsional Integral Derivatif," Universitas Diponegoro, Semarang, 2015.

[10] A. Weiss, "Gyroscope," SparkFun Electronics, 1 Februari 2013. [Online]. Available:

https://learn.sparkfun.com/tutorials/gyroscope/all. [Diakses 2 April 2018].

[11] L. Tran, "Data Fusion with 9 Degrees of Freedom Inertial Measurement Unit to Determine Object's Orientation," California Polytechnic State University, San Luis Obispo, 2017.

[12] The Editors of Encyclopaedia Britannica, "Magnetometer," Encyclopaedia Britannica, 21 Maret 2016. [Online]. Available: https://www.britannica.com/technology/magnetometer. [Diakses 2 April 2018].

[13] TE Connectivity, "MS5837-30BA," TE Connectivity, Fremont, 2015.

[14] Arduino, "Arduino Playground," Arduino, 1 January 2015. [Online]. Available: https://playground.arduino.cc/Main/WhatIsDegreesOfFreedom6DOF9DOF10DOF11DOF. [Diakses 20 November 2017].

[15] J. J. Lasiman, "Penggunaan Quaternion dan Matriks pada Perputaran Spasial," Institut Teknologi Bandung, Bandung, 2016.

[16] A. Janota, V. Šimák, D. Nemec dan J. Hrbček, "Improving the Precision and Speed of Euler Angles Computation from Low-Cost Rotation Sensor Data," University of Žilina, Žilina, 2015.

[17] M. Aasvik, "Tutorial: Potentiometers with Arduino and Filtering," Norwegian Creations, 28 Oktober 2015. [Online]. Available: https://www.norwegiancreations.com/2015/10/tutorial-potentiometers-with-arduino-and-filtering/. [Diakses 3 April 2018].

[18] C. Hodanbosi dan J. G. Fairman, "Fluids Pressure and Depth," NASA, 1 Agustus 1996. [Online]. Available: https://www.grc.nasa.gov/www/k-12/WindTunnel/Activities/fluid_pressure.html. [Diakses 2 Maret 2018].

[19] G. Gridling dan B. Weiss, "Introduction to Microcontrollers," Vienna University of Technology, Wien, 2007.

[20] Atmel Corporation, "ATmega328/P," Atmel Corporation, San Jose, 2016.

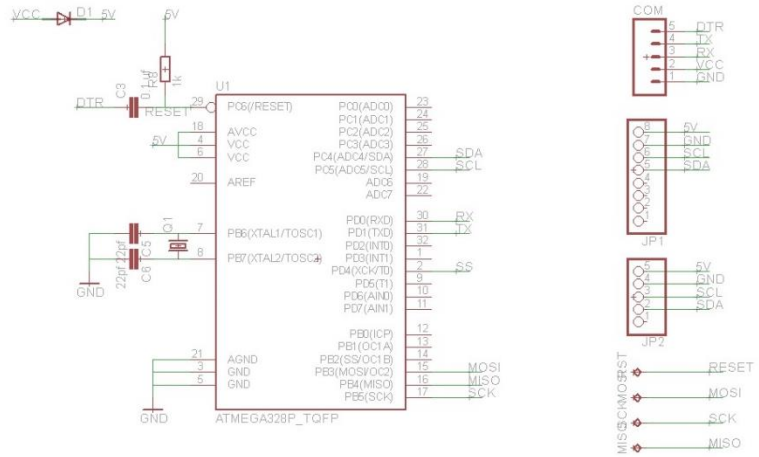
[21] Arduino, “Arduino Software (IDE),” Arduino, 7 September 2015. [Online]. Available: https://www.arduino.cc/en/Guide/Environment. [Diakses 3 April 2018].

[22] F. Surya, “I2C Protokol,” Binus University, Jakarta, 2007.

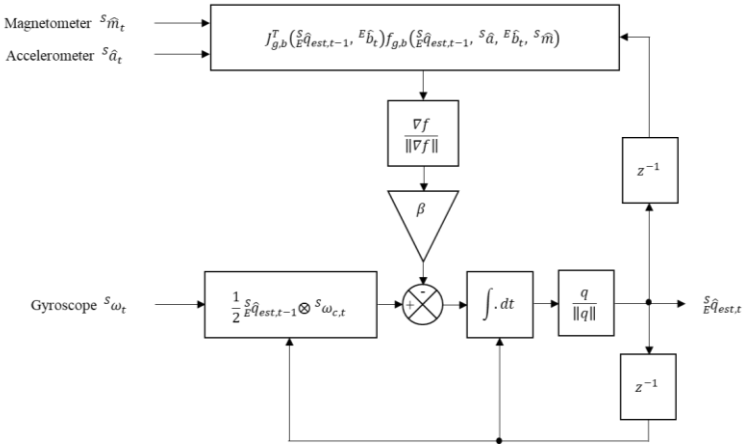
Halaman ini sengaja dikosongkan

LAMPIRAN

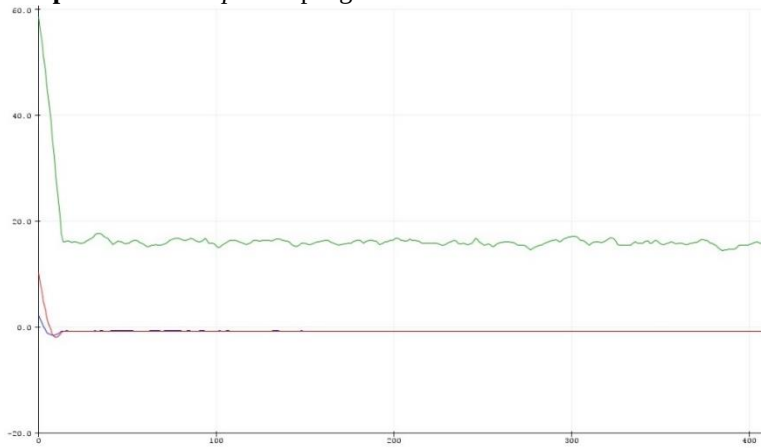
Lampiran 1: *Schematic board* IMU



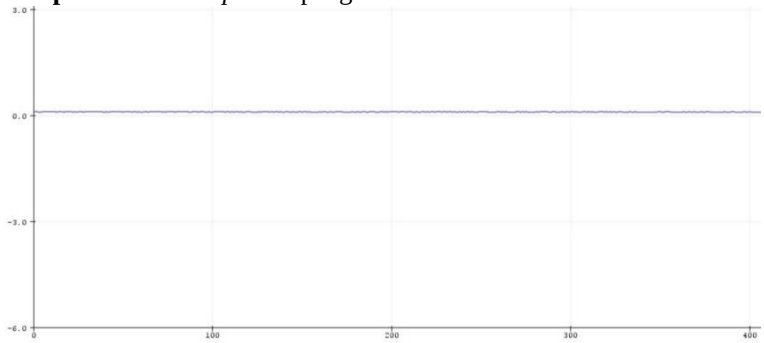
Lampiran 2: Diagram blok kuaternion



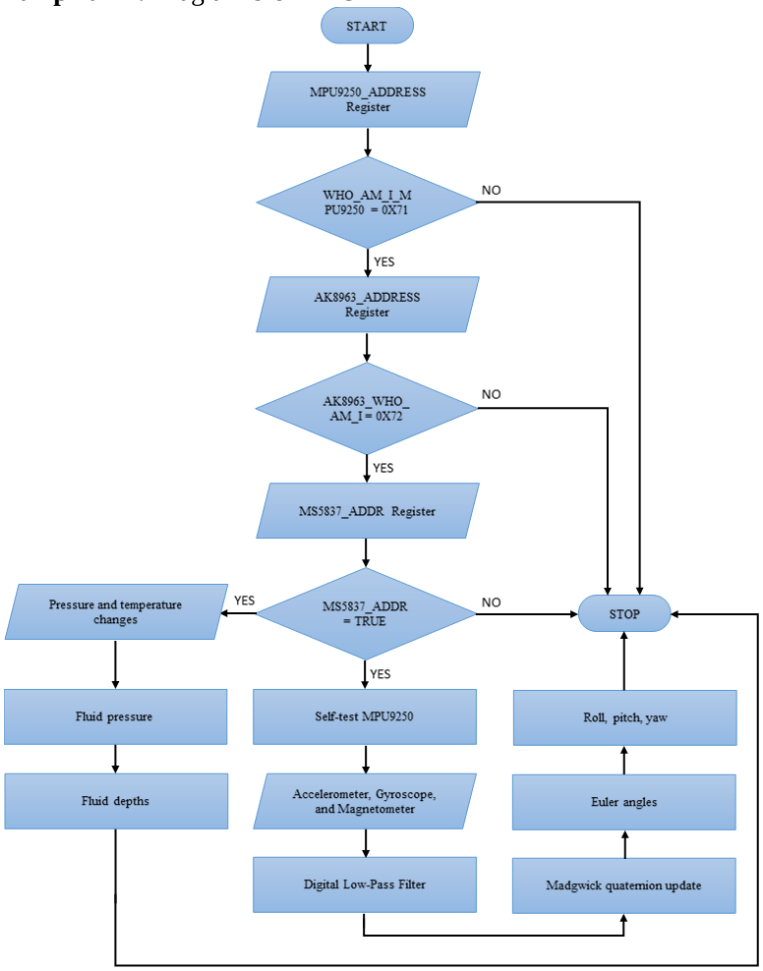
Lampiran 3: *Serial plotter* pengukuran orientasi MPU9250



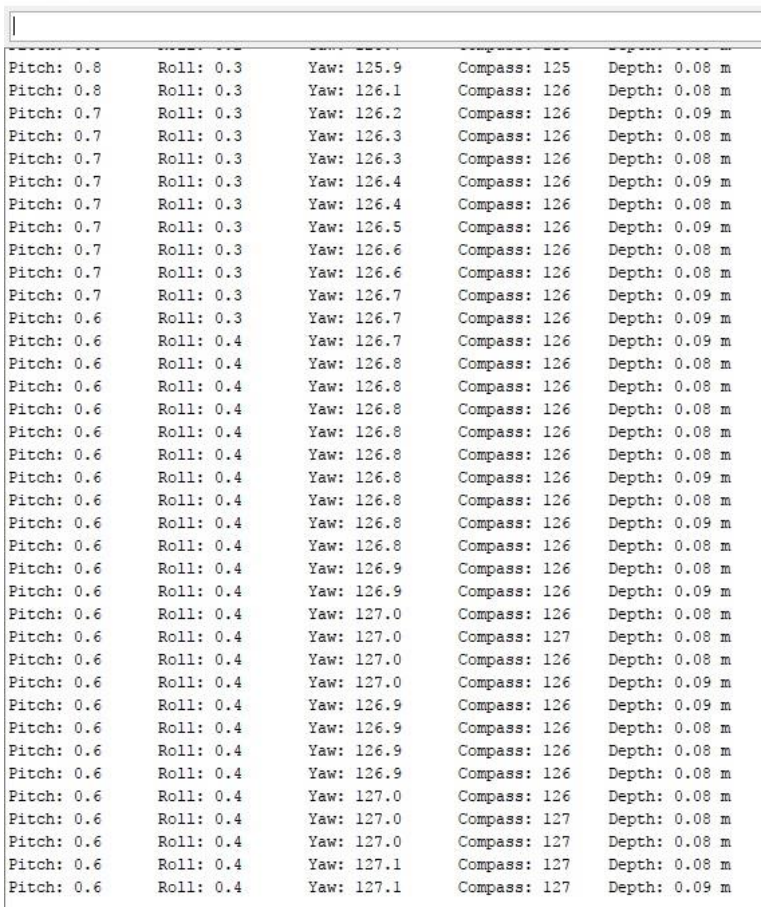
Lampiran 4: *Serial plotter* pengukuran kedalaman MS5837



Lampiran 4: Diagram blok IMU



Lampiran 6: Tampilan Data pada *Serial Monitor* Arduino IDE



Lampiran 7: *Sketch* program IMU

```
/*  MPU9250 + MS5837
 *
 *  Proyek Akhir
 *  Inertial Navigation System Menggunakan
Inertial Measurement Unit 10 DoF pada Diver
Propulsion Vehicle
```

```
 *
 *  Alief Ardiansyah
 *
 *  Bhimasena Research and Development
 *  Underwater Division - Diver Propulsion Vehicle
(DPV)
 *  Jatinangor, 2018
 *
 *  MPU9250 - Arduino Uno     MS5837 - Arduino Uno
 *  VCC ----- 5 V             VCC ---- 3.3 V
 *  SDA ----- SDA             SDA ---- SDA
 *  SCL ----- SCL             SCL ---- SCL
 *  GND ----- GND             GND ---- GND
 */

// Library untuk melakukan komunikasi secara I2C
#include <Wire.h>

// MPU9250 Register
// Register untuk Aksel dan Giro
#define MPU9250_ADDRESS   0x68
#define SELF_TEST_X_GYRO  0x00
#define SELF_TEST_Y_GYRO  0x01
#define SELF_TEST_Z_GYRO  0x02
#define SELF_TEST_X_ACCEL 0x0D
#define SELF_TEST_Y_ACCEL 0x0E
#define SELF_TEST_Z_ACCEL 0x0F
#define SMPLRT_DIV        0x19
#define CONFIG            0x1A
#define GYRO_CONFIG       0x1B
#define ACCEL_CONFIG      0x1C
#define ACCEL_CONFIG2     0x1D
#define INT_PIN_CFG       0x37
#define INT_ENABLE        0x38
#define INT_STATUS        0x3A
#define ACCEL_XOUT_H      0x3B
#define ACCEL_XOUT_L      0x3C
#define ACCEL_YOUT_H      0x3D
#define ACCEL_YOUT_L      0x3E
#define ACCEL_ZOUT_H      0x3F
#define ACCEL_ZOUT_L      0x40
#define GYRO_XOUT_H       0x43
#define GYRO_XOUT_L       0x44
#define GYRO_YOUT_H       0x45
```

```
#define GYRO_YOUT_L        0x46
#define GYRO_ZOUT_H        0x47
#define GYRO_ZOUT_L        0x48
#define PWR_MGMT_1         0x6B
#define WHO_AM_I_MPU9250   0x75

// Register untuk Magnetometer
#define AK8963_ADDRESS   0x0C
#define AK8963_WHO_AM_I  0x00
#define AK8963_ST1       0x02
#define AK8963_XOUT_L    0x03
#define AK8963_XOUT_H    0x04
#define AK8963_YOUT_L    0x05
#define AK8963_YOUT_H    0x06
#define AK8963_ZOUT_L    0x07
#define AK8963_ZOUT_H    0x08
#define AK8963_CNTL      0x0A
#define AK8963_ASAX      0x10
#define AK8963_ASAY      0x11
#define AK8963_ASAZ      0x12

// Register untuk MS5837
#define MS5837_ADDR               0x76
#define MS5837_RESET              0x1E
#define MS5837_ADC_READ           0x00
#define MS5837_PROM_READ          0xA0
#define MS5837_CONVERT_D1_8192    0x4A
#define MS5837_CONVERT_D2_8192    0x5A

#define AHRS true      // Set "true" untuk membaca
data orientasi, "false" untuk membaca data dasar
#define senMode false  // Set "true" untuk dual
mode (kalibrasi dan pengukuran), "false" untuk
single mode (pengukuran)

// Memasukkan parameter input
// Tabel parameter Aksel
enum Ascale {AFS_2G, AFS_4G, AFS_8G, AFS_16G};

// Tabel parameter Giro
enum Gscale {GFS_250DPS, GFS_500DPS, GFS_1000DPS,
GFS_2000DPS};

// Tabel parameter Magnetometer
```

```
enum Mscale {MFS_14BITS, MFS_16BITS};

// Pemilihan skala parameter input
byte Gscale = GFS_250DPS;
byte Ascale = AFS_2G;
byte Mscale = MFS_16BITS;

// Pemilihan frekuensi Magnetometer, 0x02 untuk 8
Hz, 0x06 untuk 100 Hz
byte Mmode = 0x02;

// Resolusi skala per LSB (Least Significant Bit)
untuk sensor
float aRes, gRes, mRes;

int accelCount[3], gyroCount[3], magCount[3];   //
Menyimpan 16-bit output sensor
float magCalibration[3] = {0, 0, 0};            //
Kalibrasi pabrik Magneto
float magBias[3] = {0, 0, 0};                   //
Bias pabrik Magneto
float SelfTest[6];                              //
Hasil self test Aksel dan Giro

unsigned int delt_t = 0;                    //
Mengontrol kecepatan keluaran pada display
unsigned int count = 0;
unsigned int sumCount = 0;
unsigned int compass;
float pitch, roll, yaw;                     //
Hasil keluaran pitch, roll, yaw
float deltat = 0.0f;                        //
Integrasi interval waktu pada filter
float sum = 0.0f;
unsigned long lastUpdate = 0;               //
Menghitung integrasi interval
unsigned long firstUpdate = 0;
unsigned long Now = 0;

// Konstanta AHRS ( Attitude and Heading Reference
System
// Perhitungan nilai kesalahan Giro dalam rad/s
(>> 40 dps)
float GyroMeasError = PI*(40.0f / 180.0f);
```

```
float beta = sqrt(3.0f / 4.0f) * GyroMeasError;

float ax, ay, az, gx, gy, gz, mx, my, mz;    //
Variabel output sensor
float q[4] = {1.0f, 0.0f, 0.0f, 0.0f};       //
Vektor quaternion

// Variabel low-pass filter
float axFil, ayFil, azFil;
float gxFil, gyFil, gzFil;
float mxFil, myFil, mzFil;

// Variabel massa jenis dan kedalaman air
float fluidDensity;
float waterDepth;

unsigned int C[8];
unsigned long D1, D2;
long P;

void setup()
{
  Wire.begin();          // Memulai komunikasi I2C
  Serial.begin(38400);  // Baud rate 38400 bps

  Serial.println("Inertial Measurement Unit");

  // Membaca WHO_AM_I register untuk tes
komunikasi
  byte MPU_1 = readByte(MPU9250_ADDRESS,
WHO_AM_I_MPU9250);
  if(MPU_1 == 0x71)
  {
    Serial.println("MPU9250 is online....");
  }
  else
  {
    Serial.print("Could not connect to
MPU9250....");
    while(1);
  }

  // Inisialisasi MPU9250
  initMPU9250();
```

```
  Serial.println("MPU9250 initialized....");

  // Membaca WHO_AM_I register untuk tes
komunikasi Magneto
  byte MPU_2 = readByte(AK8963_ADDRESS,
AK8963_WHO_AM_I);

  // Inisialisasi Magnetometer (AK8963)
  initAK8963(magCalibration);
  Serial.println("AK8963 initialized....");

  // Inisialisasi sensor tekanan
  // Return true jika inisialisasi berhasil
  while(!initMS5837())
  {
    Serial.println("Init failed!");
  }

  Serial.println("MS5837 is online....");

  // Set nilai massa jenis air
  // Massa jenis air tawar = 997.77, air laut =
1025 (kg/m^3)
  fluidDensity = 997.77f;
}

void loop()
{
  Selector();
}

// ================================================
// === Fungsi pendukung program sensor MPU9250 ===
// ================================================

// Mengirim register kepada sensor MPU9250
void writeByte(byte regAddress, byte sensorReg,
byte data)
{
  Wire.beginTransmission(regAddress);   //
Inisialisasi Tx
  Wire.write(sensorReg);                //
Menempatkan alamat register slave ke Tx
```

```
  Wire.write(data);                       //
Menempatkan data ke Tx
  Wire.endTransmission();                 //
Mengirim Tx
}

// Membaca register dari sensor MPU9250
byte readByte(byte regAddress, byte sensorReg)
{
  byte data;
  Wire.beginTransmission(regAddress);       //
Inisialisasi Tx
  Wire.write(sensorReg);                    //
Menempatkan alamat register slave ke Tx
  Wire.endTransmission(false);              //
Mengirim Tx, komunikasi tetap terhubung
  Wire.requestFrom(regAddress, (byte) 1);   //
Membaca 1 byte dari alamat register slave
  data = Wire.read();                       //
Mengisi Rx dengan hasil
  return data;                              //
Return data dari register slave
}

// Membaca data sensor MPU9250
void readBytes(byte regAddress, byte sensorReg,
byte count, byte * dest)
{
  Wire.beginTransmission(regAddress);           //
Inisialisasi Tx
  Wire.write(sensorReg);                        //
Menempatkan alamat register slave ke Tx
  Wire.endTransmission(false);                  //
Mengirim Tx, komunikasi tetap terhubung
  Wire.requestFrom(regAddress,(byte) count);    //
Membaca 1 byte dari alamat register slave
  byte i = 0;
  while (Wire.available())
  {
    dest[i++] = Wire.read();                    //
Menaruh hasil pembacaan ke dalam Rx
  }
}
```

```
void getAres()
{
  switch(Ascale)
  {
    // Skala Aksel
    // 2 g (00), 4 g (01), 8 g (10), 16 g (11)
    case AFS_2G: aRes = 2.0/32768.0;    // 32768.0
= konversi 16 bit dalam desimal
    break;
    case AFS_4G: aRes = 4.0/32768.0;
    break;
    case AFS_8G: aRes = 8.0/32768.0;
    break;
    case AFS_16G: aRes = 16.0/32768.0;
    break;
  }
}

void getGres()
{
  switch(Gscale)
  {
    // Skala Giro
    // 250 dps (00), 500 dps (01), 1000 dps (10),
2000 dps (11)
    case GFS_250DPS: gRes = 250.0/32768.0;    //
32768.0 = konversi 16 bit dalam desimal
    break;
    case GFS_500DPS: gRes = 500.0/32768.0;
    break;
    case GFS_1000DPS: gRes = 1000.0/32768.0;
    break;
    case GFS_2000DPS: gRes = 2000.0/32768.0;
    break;
  }
}

void getMres()
{
  switch(Mscale)
  {
    // Skala Magnetometer
    // 14 bit (0) and 16 bit (1)
```

```
    case MFS_14BITS: mRes = 10.*4912.0/8192.0;
// 4912 = kerapatan fluks magnetik
    break;
// 8192 = konversi 14 bit dalam desimal
    case MFS_16BITS: mRes = 10.*4912.0/32768.0;
// 32768 = konversi 16 bit dalam desimal,
    break;
// 10 = untuk konversi dari uT menjadi mG
  }
}

// Inisialisasi MPU9250
void initMPU9250()
{
  // Menyalakan sensor
  writeByte(MPU9250_ADDRESS, PWR_MGMT_1, 0x00);
// Menghapus bit mode tidur, mengaktifkan semua
sensor
  delay(100);

  // Mendapatkan sumber waktu yang stabil
  writeByte(MPU9250_ADDRESS, PWR_MGMT_1, 0x01);
// Auto select sumber clock sebagai referesni PLL
(Phase Locked Loop) Giro
  delay(200);

  // Set bandwidth Giro menjadi 41 Hz
  writeByte(MPU9250_ADDRESS, CONFIG, 0x03);

  // Set frekuensi keluaran Giroskop menjadi 200
Hz
  writeByte(MPU9250_ADDRESS, SMPLRT_DIV, 0x04);

  // Mengatur skala penuh Giro
  byte IMU_1 = readByte(MPU9250_ADDRESS,
GYRO_CONFIG);  // Mendapatkan nilai register
GYRO_CONFIG sekarang
  IMU_1 = IMU_1 & ~0x03;        // Menghapus
Fchoice bit
  IMU_1 = IMU_1 & ~0x18;        // Menghapus GFS
bits
  IMU_1 = IMU_1 | Gscale << 3;  // Mengatur skala
penuh Giro
```

```
  writeByte(MPU9250_ADDRESS, GYRO_CONFIG, IMU_1 );
// Menulis nilai GYRO_CONFIG register baru

  // Mengatur skala penuh Aksel
  IMU_1 = readByte(MPU9250_ADDRESS, ACCEL_CONFIG);
// Mendapatkan nilai register ACCEL_CONFIG
sekarang
  IMU_1 = IMU_1 & ~0x18;          // Menghapus AFS
bit
  IMU_1 = IMU_1 | Ascale << 3;    // Mengatur
skala penuh Aksel
  writeByte(MPU9250_ADDRESS, ACCEL_CONFIG, IMU_1);
// Menulis nilai AKSEL_CONFIG register baru

  // Mengatur konfigurasi frekuensi data Aksel
  IMU_1 = readByte(MPU9250_ADDRESS,
ACCEL_CONFIG2);     // Mendapatkan nilai register
ACCEL_CONFIG2
  IMU_1 = IMU_1 & ~0x0F;    // Menghapus
accel_fchoice_b dan A_DLPFG
  IMU_1 = IMU_1 | 0x03;     // Mengatur frekusensi
Aksel menjadi 1 kHz dan Bandwidth menjadi 41 kHz
  writeByte(MPU9250_ADDRESS, ACCEL_CONFIG2,
IMU_1);     // Menulis nilai AKSEL_CONFIG2
register baru
  // Frekuensi dari Aksel dan Giro sebesar 1 kHz,
tapi dibagi 5 menjadi 200 Hz karena fungsi
SMPLRT_DIV

  // Konfigurasi interrupt dan bypass aktif
  writeByte(MPU9250_ADDRESS, INT_PIN_CFG, 0x22);
  writeByte(MPU9250_ADDRESS, INT_ENABLE, 0x01);
  delay(100);
}

// Inisialisasi AK8963
void initAK8963(float * dest)
{
  byte rawData[3];
// Menyimpan data kalibrasi Giro
  writeByte(AK8963_ADDRESS, AK8963_CNTL, 0x00);
// Power down Magneto
  delay(10);
```

```
  writeByte(AK8963_ADDRESS, AK8963_CNTL, 0x0F);
// Masuk mode akses ROM gabungan
  delay(10);
  readBytes(AK8963_ADDRESS, AK8963_ASAX, 3,
&rawData[0]);         // Membaca nilai kalibrasi
sumbu xyz
  dest[0] =  (float)(rawData[0] - 128)/256. + 1.;
// Mengembalikan nilai pengaturan sensitivitas
sumbu x
  dest[1] =  (float)(rawData[1] - 128)/256. + 1.;
// 128 = 2 byte data
  dest[2] =  (float)(rawData[2] - 128)/256. + 1.;
// 256 = 4 byte data
  writeByte(AK8963_ADDRESS, AK8963_CNTL, 0x00);
// Power down Magneto
  delay(10);

  // Konfigurasi magnetometer untuk pembacaan
terus menerus dengan resolusi tinggi
  writeByte(AK8963_ADDRESS, AK8963_CNTL, Mscale <<
4 | Mmode);    // Mengatur resolusi data Magneto
  delay(10);
}

// Membaca register data Aksel sensor
void readAccelData(int * dest)
{
  byte rawData[6];
// Menyimpan register data xyz Aksel
  readBytes(MPU9250_ADDRESS, ACCEL_XOUT_H, 6,
&rawData[0]);   // Membaca enam register data
mentah menjadi array
  dest[0] = ((int)rawData[0] << 8) | rawData[1] ;
// Mengubah MSB dan LSB menjadi 16-bit
  dest[1] = ((int)rawData[2] << 8) | rawData[3] ;
  dest[2] = ((int)rawData[4] << 8) | rawData[5] ;
}

// Membaca register data Giro sensor
void readGyroData(int * dest)
{
  byte rawData[6];
// Menyimpan register data xyz Giro
```

```
  readBytes(MPU9250_ADDRESS, GYRO_XOUT_H, 6,
&rawData[0]);    // Membaca enam register data
mentah menjadi array
  dest[0] = ((int)rawData[0] << 8) | rawData[1] ;
// Mengubah MSB dan LSB menjadi 16-bit
  dest[1] = ((int)rawData[2] << 8) | rawData[3] ;
  dest[2] = ((int)rawData[4] << 8) | rawData[5] ;
}

// Membaca register data Magneto sensor
void readMagData(int * dest)
{
  byte rawData[7];
// Menyimpan register data xyz dan ST2, ST2 harus
dibaca untuk mendapatkan data
  if(readByte(AK8963_ADDRESS, AK8963_ST1) & 0x01)
// Menunggu data Magneto siap
  {
    readBytes(AK8963_ADDRESS, AK8963_XOUT_L, 7,
&rawData[0]);   // Membaca enam register data
mentah dan ST2 menjadi array
    byte IMU_1 = rawData[6];
// MEngakhiri pembacaan data dengan membaca
register ST2
    if(!(IMU_1 & 0x08))
    {
      dest[0] = ((int)rawData[1] << 8) |
rawData[0] ;           // Mengubah MSB dan LSB
menjadi 16-bit
      dest[1] = ((int)rawData[3] << 8) |
rawData[2] ;           // Data disimpan dalam
bentuk Little-Endian
      dest[2] = ((int)rawData[5] << 8) |
rawData[4] ;
    }
  }
}

// Self test Aksel dan Giro. Nilai deviasi harus
dibawah 14 %
void MPU9250SelfTest(float * dest)
{
  byte rawData[6] = {0, 0, 0, 0, 0, 0};
  byte selfTest[6];
```

```
  long gAvg[3] = {0}, aAvg[3] = {0}, aSTAvg[3] =
{0}, gSTAvg[3] = {0};
  float factoryTrim[6];
  byte FS = 0;

  writeByte(MPU9250_ADDRESS, SMPLRT_DIV, 0x00);
// Set frekuensi Giro menjadi 1 kHz
  writeByte(MPU9250_ADDRESS, CONFIG, 0x02);
// Set frekuensi Giro menjadi 1 kHz dan DLPF
(Digital Low Pass Filter) menjadi 92 Hz
  writeByte(MPU9250_ADDRESS, GYRO_CONFIG, FS<<3);
// Set skala penuh Giro menjadi 250 dps
  writeByte(MPU9250_ADDRESS, ACCEL_CONFIG2, 0x02);
// Set frekuensi Aksel menjadi 1 kHz dan Bandwidth
menjadi 92 Hz
  writeByte(MPU9250_ADDRESS, ACCEL_CONFIG, FS<<3);
// Set skala penuh Aksel menjadi 2 g

  for( int ii = 0; ii < 200; ii++)    //
Mendapatkan rata-rata nilai aktual Aksel dan Giro
  {
    readBytes(MPU9250_ADDRESS, ACCEL_XOUT_H, 6,
&rawData[0]);   // Membaca 6 register data mentah
menjadi data aray
    aAvg[0] += (int)(((int)rawData[0] << 8) |
rawData[1]) ;     // Mengubah MSB dan LSB menjadi
nilai 16-bit
    aAvg[1] += (int)(((int)rawData[2] << 8) |
rawData[3]) ;
    aAvg[2] += (int)(((int)rawData[4] << 8) |
rawData[5]) ;

    readBytes(MPU9250_ADDRESS, GYRO_XOUT_H, 6,
&rawData[0]);    // Membaca 6 register data mentah
secara sekuensial menjadi data aray
    gAvg[0] += (int)(((int)rawData[0] << 8) |
rawData[1]) ;     // Mengubah MSB dan LSB menjadi
nilai 16-bit
    gAvg[1] += (int)(((int)rawData[2] << 8) |
rawData[3]) ;
    gAvg[2] += (int)(((int)rawData[4] << 8) |
rawData[5]) ;
  }
```

```
  for (int ii =0; ii < 3; ii++)        //
Mendapatkan rata rata dari 200 nilai dan disimpan
sebagai rata rata pembacaan selftest
  {
    aAvg[ii] /= 200;
    gAvg[ii] /= 200;
  }
  // Konfigurasi Aksel untuk selftest
  writeByte(MPU9250_ADDRESS, ACCEL_CONFIG, 0xE0);
// Mengaktifkan selftest di semua sumbu Aksel
dengan skala 2 g
  writeByte(MPU9250_ADDRESS, GYRO_CONFIG,  0xE0);
// Mengaktifkan selftest di semua sumbu Giro
dengan skala 250 dps
  delay(25);  // Untuk menstabilkan sensor

  for( int ii = 0; ii < 200; ii++)    //
Mendapatkan rata rata nilai selftest Aksel dan
Giro
  {
    readBytes(MPU9250_ADDRESS, ACCEL_XOUT_H, 6,
&rawData[0]);      // Membaca 6 register data
mentah menjadi data aray
    aSTAvg[0] += (int)(((int)rawData[0] << 8) |
rawData[1]) ;      // Mengubah MSB dan LSB menjadi
nilai 16-bit
    aSTAvg[1] += (int)(((int)rawData[2] << 8) |
rawData[3]) ;
    aSTAvg[2] += (int)(((int)rawData[4] << 8) |
rawData[5]) ;

    readBytes(MPU9250_ADDRESS, GYRO_XOUT_H, 6,
&rawData[0]);       // Membaca 6 register data
mentah secara sekuensial menjadi data aray
    gSTAvg[0] += (int)(((int)rawData[0] << 8) |
rawData[1]) ;      // Mengubah MSB dan LSB menjadi
nilai 16-bit
    gSTAvg[1] += (int)(((int)rawData[2] << 8) |
rawData[3]) ;
    gSTAvg[2] += (int)(((int)rawData[4] << 8) |
rawData[5]) ;
  }
```

```
  for (int ii =0; ii < 3; ii++)        //
Mendapatkan rata rata dari 200 nilai dan disimpan
sebagai rata rata pembacaan selftest
  {
    aSTAvg[ii] /= 200;
    gSTAvg[ii] /= 200;
  }

  // Konfigurasi untuk penggunaan normal Aksel dan
Giro
  writeByte(MPU9250_ADDRESS, ACCEL_CONFIG, 0x00);
  writeByte(MPU9250_ADDRESS, GYRO_CONFIG,  0x00);
  delay(25);     // Untuk menstabilkan sensor

  // Mendapatkan kode selftest pabrik Aksel dan
Giro dari USR_Reg
  selfTest[0] = readByte(MPU9250_ADDRESS,
SELF_TEST_X_ACCEL); // Hasil selftest sumbu X
Aksel
  selfTest[1] = readByte(MPU9250_ADDRESS,
SELF_TEST_Y_ACCEL); // Hasil selftest sumbu Y
Aksel
  selfTest[2] = readByte(MPU9250_ADDRESS,
SELF_TEST_Z_ACCEL); // Hasil selftest sumbu Z
Aksel
  selfTest[3] = readByte(MPU9250_ADDRESS,
SELF_TEST_X_GYRO);  // Hasil selftest sumbu X Giro
  selfTest[4] = readByte(MPU9250_ADDRESS,
SELF_TEST_Y_GYRO);  // Hasil selftest sumbu Y Giro
  selfTest[5] = readByte(MPU9250_ADDRESS,
SELF_TEST_Z_GYRO);  // Hasil selftest sumbu Z Giro

  // Mendapatkan nilai selftest pabrik dari
pembacaan kode selftest
  factoryTrim[0] = (float)(2620 / 1 << FS) *
(pow(1.01, ((float)selfTest[0] - 1.0))); //
Perhitungan FT[Xa]
  factoryTrim[1] = (float)(2620 / 1 << FS) *
(pow(1.01, ((float)selfTest[1] - 1.0))); //
Perhitungan FT[Ya]
  factoryTrim[2] = (float)(2620 / 1 << FS) *
(pow(1.01, ((float)selfTest[2] - 1.0))); //
Perhitungan FT[Za]
```

```
  factoryTrim[3] = (float)(2620 / 1 << FS) *
(pow(1.01, ((float)selfTest[3] - 1.0))); //
Perhitungan FT[Xg]
  factoryTrim[4] = (float)(2620 / 1 << FS) *
(pow(1.01, ((float)selfTest[4] - 1.0))); //
Perhitungan FT[Yg]
  factoryTrim[5] = (float)(2620 / 1 << FS) *
(pow(1.01, ((float)selfTest[5] - 1.0))); //
Perhitungan FT[Zg]

  // Hasil berupa rasio dari (STR - FT) / FT. STR
= Selftest response, FT = Factory trim.
  // Untuk mendapatkan nilai persentase, hasil
tersebut dikali dengan 100.
  for (int i = 0; i < 3; i++)
  {
    dest[i]   = 100.0 * ((float)(aSTAvg[i] -
aAvg[i])) / factoryTrim[i] - 100.;
    dest[i+3] = 100.0 * ((float)(gSTAvg[i] -
gAvg[i])) / factoryTrim[i+3] - 100.;
  }
}

void SelfTestResult()
{
  // Memulai self testing
  MPU9250SelfTest(SelfTest);
  // Menampilkan hasil self testing
  Serial.print("Self testing Aksel sumbu X: ");
Serial.print(SelfTest[0],1); Serial.println(" %");
  Serial.print("Self testing Aksel sumbu Y: ");
Serial.print(SelfTest[1],1); Serial.println(" %");
  Serial.print("Self testing Aksel sumbu Z: ");
Serial.print(SelfTest[2],1); Serial.println(" %");
  Serial.print("Self testing Giro sumbu X: ");
Serial.print(SelfTest[3],1); Serial.println(" %");
  Serial.print("Self testing Giro sumbu Y: ");
Serial.print(SelfTest[4],1); Serial.println(" %");
  Serial.print("Self testing Giro sumbu Z: ");
Serial.print(SelfTest[5],1); Serial.println(" %");
  delay(1000);
}

void LowpassFilter()
```

```
{
  axFil = (0.02f * ax) + (0.98f * axFil); ayFil =
(0.02f * ay) + (0.98f * ayFil); azFil = (0.02f *
az) + (0.98f * azFil);
  gxFil = (0.02f * gx) + (0.98f * gxFil); gyFil =
(0.02f * gy) + (0.98f * gyFil); gzFil = (0.02f *
gz) + (0.98f * gzFil);
  mxFil = (0.02f * mx) + (0.98f * mxFil); myFil =
(0.02f * my) + (0.98f * myFil); mzFil = (0.02f *
mz) + (0.98f * mzFil);
}

void MadgwickQuaternionUpdate(float ax, float ay,
float az, float gx, float gy, float gz, float mx,
float my, float mz)
{
  float q1 = q[0], q2 = q[1], q3 = q[2], q4 =
q[3];   // short name local variable for
readability
  float norm;
  float hx, hy, _2bx, _2bz;
  float s1, s2, s3, s4;
  float qDot1, qDot2, qDot3, qDot4;

  // Auxiliary variables to avoid repeated
arithmetic
  float _2q1mx, _2q1my, _2q1mz, _2q2mx, _4bx,
_4bz;
  float _2q1 = 2 * q1;
  float _2q2 = 2 * q2;
  float _2q3 = 2 * q3;
  float _2q4 = 2 * q4;
  float _2q1q3 = 2.0f * q1 * q3;
  float _2q3q4 = 2.0f * q3 * q4;
  float q1q1 = q1 * q1;
  float q1q2 = q1 * q2;
  float q1q3 = q1 * q3;
  float q1q4 = q1 * q4;
  float q2q2 = q2 * q2;
  float q2q3 = q2 * q3;
  float q2q4 = q2 * q4;
  float q3q3 = q3 * q3;
  float q3q4 = q3 * q4;
  float q4q4 = q4 * q4;
```

```
  // Normalise accelerometer measurement
  norm = sqrtf(ax * ax + ay * ay + az * az);
  if (norm == 0.0f) return; // handle NaN
  norm = 1.0f/norm;
  ax *= norm; ay *= norm; az *= norm;

  // Normalise magnetometer measurement
  norm = sqrtf(mx * mx + my * my + mz * mz);
  if (norm == 0.0f) return; // handle NaN
  norm = 1.0f / norm;
  mx *= norm; my *= norm; mz *= norm;

  // Reference direction of Earth's magnetic field
  _2q1mx = 2.0f * q1 * mx;
  _2q1my = 2.0f * q1 * my;
  _2q1mz = 2.0f * q1 * mz;
  _2q2mx = 2.0f * q2 * mx;
  hx = mx * q1q1 - _2q1my * q4 + _2q1mz * q3 + mx
* q2q2 + _2q2 * my * q3 + _2q2 * mz * q4 - mx *
q3q3 - mx * q4q4;
  hy = _2q1mx * q4 + my * q1q1 - _2q1mz * q2 +
_2q2mx * q3 - my * q2q2 + my * q3q3 + _2q3 * mz *
q4 - my * q4q4;
  _2bx = sqrtf(hx * hx + hy * hy);
  _2bz = -_2q1mx * q3 + _2q1my * q2 + mz * q1q1 +
_2q2mx * q4 - mz * q2q2 + _2q3 * my * q4 - mz *
q3q3 + mz * q4q4;
  _4bx = 2.0f * _2bx;
  _4bz = 2.0f * _2bz;

  // Gradient decent algorithm corrective step
  s1 = -_2q3 * (2.0f * q2q4 - _2q1q3 - ax) + _2q2
* (2.0f * q1q2 + _2q3q4 - ay) - _2bz * q3 * (_2bx
* (0.5f - q3q3 - q4q4) + _2bz * (q2q4 - q1q3) -
mx) + (-_2bx * q4 + _2bz * q2) * (_2bx * (q2q3 -
q1q4) + _2bz * (q1q2 + q3q4) - my) + _2bx * q3 *
(_2bx * (q1q3 + q2q4) + _2bz * (0.5f - q2q2 -
q3q3) - mz);
  s2 = _2q4 * (2.0f * q2q4 - _2q1q3 - ax) + _2q1 *
(2.0f * q1q2 + _2q3q4 - ay) - 4.0f * q2 * (1.0f -
2.0f * q2q2 - 2.0f * q3q3 - az) + _2bz * q4 *
(_2bx * (0.5f - q3q3 - q4q4) + _2bz * (q2q4 -
q1q3) - mx) + (_2bx * q3 + _2bz * q1) * (_2bx *
```

```
(q2q3 - q1q4) + _2bz * (q1q2 + q3q4) - my) + (_2bx
* q4 - _4bz * q2) * (_2bx * (q1q3 + q2q4) + _2bz *
(0.5f - q2q2 - q3q3) - mz);
  s3 = -_2q1 * (2.0f * q2q4 - _2q1q3 - ax) + _2q4
* (2.0f * q1q2 + _2q3q4 - ay) - 4.0f * q3 * (1.0f
- 2.0f * q2q2 - 2.0f * q3q3 - az) + (-_4bx * q3 -
_2bz * q1) * (_2bx * (0.5f - q3q3 - q4q4) + _2bz *
(q2q4 - q1q3) - mx) + (_2bx * q2 + _2bz * q4) *
(_2bx * (q2q3 - q1q4) + _2bz * (q1q2 + q3q4) - my)
+ (_2bx * q1 - _4bz * q3) * (_2bx * (q1q3 + q2q4)
+ _2bz * (0.5f - q2q2 - q3q3) - mz);
  s4 = _2q2 * (2.0f * q2q4 - _2q1q3 - ax) + _2q3 *
(2.0f * q1q2 + _2q3q4 - ay) + (-_4bx * q4 + _2bz *
q2) * (_2bx * (0.5f - q3q3 - q4q4) + _2bz * (q2q4
- q1q3) - mx) + (-_2bx * q1 + _2bz * q3) * (_2bx *
(q2q3 - q1q4) + _2bz * (q1q2 + q3q4) - my) + _2bx
* q2 * (_2bx * (q1q3 + q2q4) + _2bz * (0.5f - q2q2
- q3q3) - mz);
  norm = sqrtf(s1 * s1 + s2 * s2 + s3 * s3 + s4 *
s4);    // normalise step magnitude
  norm = 1.0f / norm;
  s1 *= norm; s2 *= norm; s3 *= norm; s4 *= norm;

  // Compute rate of change of quaternion
  qDot1 = 0.5f * (-q2 * gx - q3 * gy - q4 * gz) -
beta * s1;
  qDot2 = 0.5f * (q1 * gx + q3 * gz - q4 * gy) -
beta * s2;
  qDot3 = 0.5f * (q1 * gy - q2 * gz + q4 * gx) -
beta * s3;
  qDot4 = 0.5f * (q1 * gz + q2 * gy - q3 * gx) -
beta * s4;

  // Integrate to yield quaternion
  q1 += qDot1 * deltat;
  q2 += qDot2 * deltat;
  q3 += qDot3 * deltat;
  q4 += qDot4 * deltat;
  norm = sqrtf(q1 * q1 + q2 * q2 + q3 * q3 + q4 *
q4);    // normalise quaternion
  norm = 1.0f / norm;
  q[0] = q1 * norm;
  q[1] = q2 * norm;
  q[2] = q3 * norm;
```

```
  q[3] = q4 * norm;
}

void IMUResult()
{
  if(readByte(MPU9250_ADDRESS, INT_STATUS) & 0x01)
  {
    readAccelData(accelCount);  // Membaca nilai
ADC xyz
    getAres();
    // Menghitung nilai akselerasi menjadi gforce
    ax = (float)accelCount[0] * aRes;
    ay = (float)accelCount[1] * aRes;
    az = (float)accelCount[2] * aRes;

    readGyroData(gyroCount);    // Membaca nilai
ADC xyz
    getGres();
    // Menghitung nilai giro menjadi dps
    gx = (float)gyroCount[0] * gRes;
    gy = (float)gyroCount[1] * gRes;
    gz = (float)gyroCount[2] * gRes;

    readMagData(magCount);      // Membaca nilai
ADC xyz
    getMres();
    magBias[0] = +300.0f;   // Koreksi bias
Magnetometer di lingkungan sekitar
    magBias[1] = +180.0f;
    magBias[2] = -175.0f;
    // Menghitung nilai medan magnet menjadi
miliGauss
    mx = (float)magCount[0] * mRes *
magCalibration[0] - magBias[0];
    my = (float)magCount[1] * mRes *
magCalibration[1] - magBias[1];
    mz = (float)magCount[2] * mRes *
magCalibration[2] - magBias[2];
  }

  // Mengatur waktu integrasi dengan filter
Quaternion
  Now = micros();
  deltat = ((Now - lastUpdate)/1000000.0f);
```

```
  lastUpdate = Now;

  sum += deltat;    // Jumlah dari frekuensi rata-
rata update filter
  sumCount++;

  LowpassFilter();

  // Memasukkan filter Quaternion
  MadgwickQuaternionUpdate(axFil, ayFil, azFil,
gxFil*PI/180.0f, gyFil*PI/180.0f, gzFil*PI/180.0f,
myFil, mxFil, mzFil);

  if(!AHRS)
  {
    delt_t = millis() - count;
    if(delt_t > 50)
    {
      // Menampilkan data Aksel dalam mg
      Serial.print("AX: "); Serial.print(1000*ax);
Serial.print(" mg");
      Serial.print("\tAY: ");
Serial.print(1000*ay); Serial.print(" mg");
      Serial.print("\tAX: ");
Serial.print(1000*ax); Serial.print(" mg");

      // Menampilkan data Giro dalam dps
      Serial.print("\tGX: "); Serial.print(gx,3);
Serial.print(" dps");
      Serial.print("\tGY: "); Serial.print(gy,3);
Serial.print(" dps");
      Serial.print("\tGZ: "); Serial.print(gz,3);
Serial.print(" dps");

      // Menampilkan data Magneto dalam mG
      Serial.print("\tMX: "); Serial.print(mx);
Serial.print(" mG");
      Serial.print("\tMY: "); Serial.print(my);
Serial.print(" mG");
      Serial.print("\tMZ: "); Serial.print(mz);
Serial.print(" mG");

      // MEnampilkan data kuaternion
      Serial.print("\tq0: "); Serial.print(q[0]);
```

```
      Serial.print("\tqx: "); Serial.print(q[1]);
      Serial.print("\tqy: "); Serial.print(q[2]);
      Serial.print("\tqz: ");
Serial.println(q[3]);

      count = millis();
    }
  }
  else
  {
    delt_t = millis() - count;
    if (delt_t > 50)
    {
      // Mengkonversikan data dalam Quarernion
menjadi Euler Angles
      pitch = asin(2.0f * (q[1] * q[3] - q[0] *
q[2]));
      roll  = -atan2(2.0f * (q[0] * q[1] + q[2] *
q[3]), q[0] * q[0] - q[1] * q[1] - q[2] * q[2] +
q[3] * q[3]);
      yaw   = atan2(2.0f * (q[1] * q[2] + q[0] *
q[3]), q[0] * q[0] + q[1] * q[1] - q[2] * q[2] -
q[3] * q[3]);
      pitch *= 180.0f / PI;
      roll  *= 180.0f / PI;
      yaw   *= 180.0f / PI;
      yaw   *= -1.0f;
      yaw   -= 1.0f;              // Deklanasi
(6°56'17.5"S 107°45'28.9"E) 0.72°E ± 0.28° (1°)

      // Pitch bernilai positif jika sumbu x
positif menujuh ke atas, dan bernilai negatif
ketika menuju ke bawah
      // Roll bernilai positif jika sumbu y
positif menujuh ke bawah, dan bernilai negatif
ketika menuju ke atas
      // Yaw bernilai positif jika berputar searah
jarum jam , dan bernilai negatif ketika berputar
berlawanan arah jarum jam
      // Menampilkan data akhir berupa posisi dan
orientasi
      Serial.print("Pitch: ");
Serial.print(pitch, 1);
```

```
      Serial.print("\tRoll: "); Serial.print(roll,
1);
      Serial.print("\tYaw: ");  Serial.print(yaw,
1);

      if(yaw < 0)
      {
        compass = yaw + 360;
      }
      else
      {
        compass = yaw;
      }

      Serial.print("\tCompass: ");
Serial.print(compass);

      MS5837Result();

      count = millis();
      sumCount = 0;
      sum = 0;
    }
  }
}

// ================================================
// === Fungsi pendukung program sensor MS5837 ====
// ================================================

// Inisialisasi sensor MS5837
bool initMS5837()
{
  Wire.beginTransmission(MS5837_ADDR);
  Wire.write(MS5837_RESET);
  Wire.endTransmission();
  delay(10);

  // Membaca nilai kalibrasi dan CRC
  for (byte i = 0; i < 7; i++)
  {
    Wire.beginTransmission(MS5837_ADDR);
    Wire.write(MS5837_PROM_READ+i*2);
    Wire.endTransmission();
```

```
    Wire.requestFrom(MS5837_ADDR,2);
    C[i] = (Wire.read() << 8) | Wire.read();
  }

  // Verifikasi data dengan CRC
  byte crcRead = C[0] >> 12;
  byte crcCalculated = crc4(C);

  if(crcCalculated == crcRead)
  {
    return true;     // Inisialisasi berhasil
  }
  return false;      // CRC gagal
}

// Pembacaan hasil sensor MS5837
void readMS5837()
{
  // Meminta konversi D1
  Wire.beginTransmission(MS5837_ADDR);
  Wire.write(MS5837_CONVERT_D1_8192);
  Wire.endTransmission();
  delay(20);

  Wire.beginTransmission(MS5837_ADDR);
  Wire.write(MS5837_ADC_READ);
  Wire.endTransmission();
  Wire.requestFrom(MS5837_ADDR,3);
  D1 = 0;
  D1 = Wire.read();
  D1 = (D1 << 8) | Wire.read();
  D1 = (D1 << 8) | Wire.read();

  // Meminta konversi D2
  Wire.beginTransmission(MS5837_ADDR);
  Wire.write(MS5837_CONVERT_D2_8192);
  Wire.endTransmission();
  delay(20);

  Wire.beginTransmission(MS5837_ADDR);
  Wire.write(MS5837_ADC_READ);
  Wire.endTransmission();
  Wire.requestFrom(MS5837_ADDR,3);
  D2 = 0;
```

```
  D2 = Wire.read();
  D2 = (D2 << 8) | Wire.read();
  D2 = (D2 << 8) | Wire.read();

  calculate();
}

void calculate()
{
  long TEMP;
  long dT = 0;
  double SENS = 0, SENS2 = 0;
  double OFF = 0, OFF2 = 0;
  long SENSi = 0;
  long OFFi = 0;
  long Ti = 0;

  dT = D2 - uint32_t(C[5]) * 256l;

  SENS = double(C[1]) * 32768l + (double(C[3]) *
dT) / 256l;
  OFF = double(C[2]) * 65536l + (double(C[4]) *
dT) / 128l;
  P = (D1 * SENS / (20971521) - OFF) / (8192l);
  TEMP = 2500l;

  Ti = 2 * (dT * dT) / (137438953472LL);
  OFFi = (1 * (TEMP - 2000) * (TEMP - 2000)) / 16;
  SENSi = 0;

  OFF2 = OFF-OFFi;
  SENS2 = SENS-SENSi;

  TEMP = (TEMP - Ti);
  P = (((D1 * SENS2) / 20971521 - OFF2) / 8192l) /
10;
}

float pressure(float conversion = 1.0f)
{
  return P * conversion;
}

float depth()
```

```
{
  float Pa = 100.0f;
  return(pressure(Pa) - 101300) / (fluidDensity *
9.807);
}

byte crc4(unsigned int n_prom[])
{
  unsigned int n_rem = 0;
  n_prom[0] = ((n_prom[0]) & 0x0FFF);
  n_prom[7] = 0;

  for(byte i = 0; i < 16; i++)
  {
    if(i%2 == 1)
    {
      n_rem ^= (unsigned int)((n_prom[i >> 1]) &
0x00FF);
    }
    else
    {
      n_rem ^= (unsigned int)(n_prom[i>>1] >> 8);
    }
    for(byte n_bit = 8; n_bit > 0; n_bit--)
    {
      if(n_rem & 0x8000)
      {
        n_rem = (n_rem << 1) ^ 0x3000;
      }
      else
      {
        n_rem = (n_rem << 1);
      }
    }
  }
  n_rem = ((n_rem >> 12) & 0x000F);
  return n_rem ^ 0x00;
}

void MS5837Result()
{
  readMS5837();

  waterDepth = depth() + 1.23f;
```

```
  if( waterDepth <= 0)
  {
    waterDepth = 0;
  }

  // Serial.print("\tPress: ");
Serial.print(pressure()); Serial.print(" mbar");
  // Serial.print("\tUnfixed Depth: ");
Serial.print(depth());  Serial.print(" m");
  Serial.print("\tDepth: ");
Serial.print(waterDepth);  Serial.println(" m");
}

// ================================================
// ========== Fungsi integrasi program ===========
// ================================================

void Selector()
{
  char select;       // Variabel untuk selector
  if(!senMode)
  {
    Serial.println("Reading EEPROM");
    delay(500);
    while(!senMode)
    {
      IMUResult();
    }
  }
  else
  while(Serial.available() > 0)
  {
    select = Serial.read();
    if(select == 's')
    {
      SelfTestResult();
      delay(500);
      Serial.println("Self-test done");
    }
    if(select == 'r')
    {
      Serial.println("Display data");
      delay(500);
      while(select == 'r')
```

```
        {
          IMUResult();
          while(Serial.available() > 0)
          select = Serial.read();
          {
            if(select == 's')
            {
              SelfTestResult();
              Serial.println("Self-test done");
            }
          }
        }
      }
    }
}
```

Halaman ini sengaja dikosongkan

RIWAYAT PENULIS

Alief Ardianyah, lahir di Kota Malang pada 1 Oktober 1997. Anak pertama dari dua bersaudara. Mempunyai kampung halaman di Losari, Singosari, Kabupaten Malang, Jawa Timur. Pernah menempuh pendidikan di TK Muslimat 03 Singosari, SDN Pagentan 5 Singosari, SMPN 3 Singosari, dan SMAN 1 Lawang. Pada saat ini sedang menempuh jenjang pendidikan mengenai studi komputer kontrol di Departemen Teknik Elektro Otomasi, Fakultas Vokasi, Institut Teknologi Sepuluh Nopember Surabaya. Mempunyai pikiran optimis, realistis, dan humoris. Menyukai kegiatan berenang dan jalan-jalan. Bercita-cita menjadi pribadi yang berguna bagi semua orang serta sukses di dunia dan di akhirat.

Email: alief.ardiansyah70@gmail.com