TUGAS AKHIR – TE 145561

AKUISISI DAN *LOGGING* DATA *INERTIAL NAVIGATION SYSTEM* PADA KAPAL TEMPUR BAWAH AIR MENGGUNAKAN ARDUINO-SD CARD

Varisa Rahmawati
NRP. 1031150000088

Pembimbing
Dr. Ir. Achmad Affandi, DEA
Imam Arifin, ST.,MT.
Muhammad Asrofi, ST., M.Eng

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

FINAL PROJECT – TE 145561

ACQUISITION AND DATA LOGGING INERTIAL NAVIGATION SYSTEM ON COMBAT SUBMARINE USING ARDUINO-SD CARD

VARISA RAHMAWATI
NRP. 1031150000088

Supervisor
Dr. Ir. Achmad Affandi, DEA
Imam Arifin, ST.,MT.
Muhammad Asrofi, ST., M.Eng

Department of Electrical Automation
Faculty of Vocational
Institut Teknologi Sepuluh Nopember
Surabaya 2018

PERNYATAAN KEASLIAN TUGAS AKHIR

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

"**Akuisisi dan *Logging* Data *Inertial Navigation System* (INS) pada Kapal Tempur Bawah Air Menggunakan Arduino-SD Card**"

adalah benar-benar hasil karya intelektual mandiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diijinkan dan bukan merupakan karya pihak lain.

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

Surabaya, 30 Juni 2018

Varisa Rahmawati
NRP.1031150000088

Halaman ini sengaja dikosongkan

AKUISISI DAN *LOGGING* DATA *INERTIAL NAVIGATION SYSTEM* (INS) PADA KAPAL TEMPUR BAWAH AIR MENGGUNAKAN ARDUINO-SD CARD

TUGAS AKHIR

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

Menyetujui,

Dosen Pembimbing I	Dosen Pembimbing II
Dr. Ir. Achmad Arifandi DEA	Imam Arifin, S.T., M.T.
NIP. 196510141990021001	NIP. 197302222002121001

SURABAYA
JUNI, 2018

Halaman ini sengaja dikosongkan

AKUISISI DAN *LOGGING DATA INERTIAL NAVIGATION SYSTEM* (INS) PADA KAPAL TEMPUR BAWAH AIR MENGGUNAKAN ARDUINO-SD CARD DI PT. BHIMASENA RESEARCH AND DEVELOPMENT

TUGAS AKHIR

Disusun oleh:

Varisa Rahmawati NRP. 10311500000088

Menyetujui,

Kepala *Human Resources Department*,	Pembimbing Perusahaan,
Fadli Tirmissi	Muhammad Asrofi, ST.,M.Eng
NIK. 020492016	NIK. 020792017

Chief Executive Officer,

BHIMASENA Research & Development

Dipl. -Ing. Aris Budiyarto

Halaman ini sengaja dikosongkan

AKUISISI DAN *LOGGING* DATA *INERTIAL NAVIGATION SYSTEM* (INS) PADA KAPAL TEMPUR BAWAH AIR MENGGUNAKAN ARDUINO-SD CARD

Varisa Rahmawati
10311500000088

Pembimbing I : Dr. Ir. Achmad Affandi, DEA
Pembimbing II : Imam Arifin, ST., MT.
Pembimbing III : Muhammad Asrofi, ST., M.Eng

ABSTRAK

Pada sistem pengoperasian kapal, alat navigasi berperan sebagai pemberi informasi lokasi di dalam atau permukaan laut. Informasi tersebut berupa data posisi dan orientasi dari kapal. Alat navigasi yang digunakan pada Kapal Tempur Bawah Air adalah INS bernama Sublocus. Sensor inersia ini melakukan interaksi menggunakan komunikasi data serial, maka dibutuhkan rancangan sistem akuisisi dan *logging* data untuk pengambilan, pengumpulan dan penyimpanan data yang masuk agar dapat diolah dengan mudah. Akuisisi diperlukan untuk mengolah protokol pada sensor dengan melakukan *parse* rangkaian *byte* ke dalam bentuk informasi. Sublocus menggunakan protokol ANPacket, dimana terdapat banyak rangkaian informasi yang harus dipisahkan sesuai dengan alamat dalam *datasheet*. Berikutnya dilakukan proses *logging* data sebagai penyimpanan data menggunakan SD Card. Perlunya perangkat tambahan, seperti SD Card karena pada umumnya sistem *embedded* dibekali dengan memori berkapasitas sedikit. Arduino merupakan mikrokontroler yang dapat membantu dalam proses akuisisi karena data-data yang diperoleh dari sensor diolah menggunakan jalur komunikasi serial (RX dan TX).

Hasil yang diperoleh dari rancangan sistem akuisisi dan *logging* data adalah *Roll*, *Pitch*, *Heading*, *Latitude*, *Longitude*, *Microsecond*s, *Gyroscope* pada sumbu X, Y, Z, *Accelerometer* pada sumbu X, Y, Z, *Magnetometer* pada sumbu X, Y, Z, *Velocity North*, *East*, *Down*. Hal ini membuktikan bahwa perancangan sistem akuisisi dan *logging* data menggunakan Arduino dan SD Card dapat membantu dalam perolehan informasi posisi dan orientasi KTBA.

Kata kunci : IMU (*Inertial Measurement Unit*), INS (*Inertial Navigation System*), *logging*

Halaman ini sengaja dikosongkan

ACQUISITION AND DATA LOGGING INERTIAL NAVIGATION SYSTEM (INS) ON COMBAT SUBMARINE USE ARDUINO-SD CARD

Varisa Rahmawati
10311500000088

Supervisor I : Dr. Ir. Achmad Affandi, DEA
Supervisor II : Imam Arifin, ST., MT.
Supervisor III : Muhammad Asrofi, ST., M.Eng

ABSTRACT

In a submarine operating system, navigation device acts as a location information provider on the inside or at sea level. Such information is position data and ship orientation data. Navigation device used on combat submarine is INS from Advanced Navigation named Sublocus. This inertial sensor interacts using serial data communication, it requires design of data acquisition and logging systems for collection, collection and storage of incoming data in order to be processed easily. Acquisition is required to process protocol exposed to sensor by parsing the byte series into information. Sublocus used ANPacket protocol, where there is a lot of information to be adjusted to address in datasheet. Next done, process of logging data as a data storage using SD Card. Needed for additional devices, such as SD Card because generally embedded systems equipped with a small capacity memory. Arduino is a microcontroller that can assist in process of data acquisition because data obtained from sensor is processed using serial communication lines (RX and TX).

The results obtained from design of data acquisition and logging systems are Roll, Pitch, Heading, Latitude, Longitude, Microseconds, Gyroscope on X axis, Y, Z, Accelerometer on X axis, Y, Z, Magnetometer on X, Y, Z, Velocity North, East, Down. This proves that design of data acquisition and logging system using Arduino and SD Card can assist in obtaining position information and KTBA orientation.

Keyword : *IMU (Inertial Measurement Unit), INS (Inertial Navigation System), logging*

Halaman ini sengaja dikosongkan

KATA PENGANTAR

Puji dan syukur penulis ucapkan atas kehadirat Tuhan Yang Maha Esa karena atas rahmat dan karuniaNya kami dapat menyelesaikan laporan Tugas Akhir ini dengan judul : **AKUISISI DAN *LOGGING* DATA *INERTIAL NAVIGATION SYSTEM* (INS) PADA KAPAL TEMPUR BAWAH AIR MENGGUNAKAN ARDUINO-SD CARD**. Tugas Akhir merupakan salah satu syarat kelulusan yang wajib ditempuh oleh setiap mahasiswa di Departemen Teknik Elektro Otomasi ITS. Kegiatan tersebut meliputi pembuatan rancangan, pengambilan data maupun studi literatur dibawah bimbingan *engineer*.

Pada kesempatan ini penulis mengucapkan terimakasih kepada Allah SWT karena telah memberikan kelancaran dan keselamatan selama kegiatan. Terima kasih untuk kedua orang tua sekaligus keluarga besar penulis yang tidak pernah putus dalam mendo'akan. Terima kasih untuk Direktur Teknologi yang memberi kesempatan penulis untuk belajar dan melaksanakan penelitian di PT.Bhimasena Research and Development. Berikutnya terima kasih kepada kepala proyek serta anggota KTBA yang telah membimbing penulis untuk menyelesaikan pengerjaan tugas akhir. Adapun ucapan terima kasih diberikan kepada kepala Departemen Teknik Elektro Otomasi dan dosen pembimbing penulis yang telah memberikan arahan dan bimbingan berupa materi kepada penulis. Ahmad Nur Riza, Alief Ardiansyah, Livian Tjandra, Tata Tanjung Tamara, dan Amirotul Khoiro rekan dalam proses magang yang selalu membantu dalam menyelesaikan tugas. Rekan-rekan Anggota Laboratorium Automation Control System telah memberikan dukungan kepada penulis dan pihak-pihak lain yang belum dapat disebutkan satu per satu. Terima kasih untuk semuanya telah memberikan yang terbaik untuk penulis, sehingga dapat menyelesaikan penelitian ini.

Penulis menyadari bahwa laporan penelitian ini belum sempurna, oleh karena itu, saran dan masukan sangat diharapkan untuk perbaikan di masa yang akan datang. Semoga laporan tugas akhir ini bermanfaat bagi pembaca dan masyarakat untuk pengetahuan atau sebagai referensi penelitian selanjutnya.

Surabaya, 30 Juni 2018

Varisa Rahmawati
10311500000088

Halaman ini sengaja dikosongkan

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

Halaman ini sengaja dikosongkan

BAB I
PENDAHULUAN

1.1. Latar Belakang

PT. Bhimasena Research and Development merupakan perusahaan penelitian dan pengembangan yang bergerak dibidang industri otomotif, *aerospace*, dan teknologi. Perusahaan ini bekerja sama dengan kemiliteran Republik Indonesia mengembangkan sebuah peralatan maupun perlengkapan para Tentara Republik Indonesia. Dalam perusahaan tersebut terdapat beberapa kelompok penelitian yaitu divisi *Unmmaned Arial Vihacle* (Pesawat tanpa awak), *Ground Vihacle* (Kendaraan darat), dan *Underwater Vehicle* (Kendaraan bawah air). Divisi *Underwater Vihacle* mengembangkan dua proyek yang dinamakan DPV (*Driver Propulsion Vihacle*) dan KTBA (Kapal Tempur Bawah Air). KTBA merupakan jenis kapal selam yang dimanfaatkan sebagai pertahanan kawasan perairan Indonesia. Kapal merupakan kendaraan pengangkut penumpang dan barang di laut, sungai, dan sebagainya[1]. Seiring berjalannya waktu, beberapa penelitian memunculkan kapal mesin yang berjalan mengapung di atas air dan menyelam di bawah lautan, yaitu kapal selam. Jenis kapal selam adalah kendaraan yang bergerak di bawah permukaan air, umumnya digunakan untuk kepentingan militer, seperti pada penelitian dan pengembangan. Gaya apung pada kapal ini dapat diatur, sehingga menyebabkan kendaraan ini tenggelam dan muncul ke permukaan sesuai kebutuhan. Namun, cahaya tidak dapat menembus lebih jauh ke dalam lautan, akibatnya kapal selam harus dikendalikan melewati air dengan pandangan buta[2].

Alat navigasi kapal merupakan suatu yang penting dalam menentukan lokasi sebuah titik kapal di permukaan bumi atau di laut. Sistem navigasi pada kapal diperlukan untuk memperoleh data posisi terkini dan juga data orientasi kapal[3]. Dengan demikian kapal selam dapat dikendalikan melewati laut yang gelap. Data posisi dapat dihasilkan dari kinerja GPS (*Global Positioning System*) dan data orientasi dapat dihasilkan oleh peran IMU (*Inertial Measurement Unit*). Perangkat IMU memiliki tiga sensor di dalamnya, yaitu *gyroscope*, *accelerometer*, dan *magnetometer*. Sensor *accelerometer* bekerja untuk memberikan data percepatan gerak, *gyroscope* memberikan data percepatan sudut yang terjadi, sedangkan *magnetometer* memberikan data medan magnet yang terdapat pada lokasi. Navigasi umumnya menggunakan INS (*Inertial*

Navigation System) dimana alat ini merupakan gabungan antara GPS dan IMU[4]. INS yang digunakan pada penelitian ini adalah produk dari Advanced Navigation bernama Sublocus. Perangkat ini memiliki data *output* berupa rangkaian *byte* dalam ANPacket Protokol, sehingga data tidak dapat diterjemahkan secara langsung[5], [6].

Untuk mendapatkan data dari sensor dibutuhkan rancangan sistem akuisisi untuk mengambil informasi. Salah satu perangkat yang dapat menunjang sistem ini, yaitu Arduino[7]. Perangkat ini merupakan sebuah mikrokontroler yang mampu melakukan komunikasi serial dengan sensor[8].

Arduino memiliki memori internal dengan kapasitas penyimpanan terbatas. Hal ini menghambat pengguna dalam melakukan analisis data dari sensor. Diperlukannya sebuah media *logging* yaitu data *logger*, sehingga dapat dilakukan penyimpanan data. Dengan demikian, data yang tersimpan dapat dianalisis dikemudian hari secara *offline*[9], [10].

1.2. Rumusan Masalah

Pengendara Kapal Tempur Bawah Air tidak dapat mengoperasikan atau menentukan haluan di dalam laut yang gelap sehingga, membutuhkan alat navigasi untuk menentukan posisi dan arah kapal di bawah laut. Alat navigasi yang digunakan adalah INS tipe Sublocus. Perangkat ini menggunakan protokol serial ANPacket berisi informasi yang terdiri dari rangkaian *byte*. Hal ini menyulitkan pegendara dalam menerjemahkan informasi kapal, maka data tersebut diolah dan dipisah sesuai kebutuhan. Dalam memisahkan masing-masing informasi pada rangkaian *byte* tersebut, diperlukan sebuah perangkat yang mampu mengolah protokol ANPacket sesuai dengan jalur data pada sensor yaitu jalur serial. Perangkat yang digunakan adalah sistem *embedded* yang terpasang secara portable. Memori sistem *embedded* yang disediakan memiliki kapasitas penyimpanan sedikit, dan menyebabkan data yang disimpan terbatas. Penyimpanan data diperlukan untuk melakukan analisis data dari sensor INS secara *offline*.

1.3. Tujuan

INS berperan sebagai penghitung sikap kendaraan di bawah air secara kontinu. Alat navigasi yang digunakan adalah INS bernama Sublocus dari Advanced Navigation yang mampu bekerja hingga kedalaman 3000 meter. Dengan demikian, dirancang sebuah sistem untuk

pengambilan, pengumpulan dan penyimpanan data posisi maupun kecepatan dari perangkat navigasi kapal menggunakan perancangan proses akuisisi dan *logging* data.

1.4. Batasan Masalah

Terdapat beberapa hal yang membatasi dari penelitian ini, yaitu :

1. Perangkat sensor yang digunakan adalah INS bernama Sublocus.
2. Standar komunikasi yang digunakan adalah RS 422.
3. Arduino digunakan sebagai mikrokontroler.
4. Data yang diambil dari sensor Sublocus hanya *Latitude*, *Longitude*, *Velocity*, *Roll*, *Pitch*, *Yaw*, *Acceleration*, *magnetometer*, *accelerometer*, *gyroscope*.
5. Penelitian difokuskan pada akuisisi dan penyimpanan data ke dalam modul penyimpanan menggunakan sistem *embedded*.

1.5. Metode Penelitian

Terdapat beberapa tahapan metodologi dalam penelitian ini, yaitu studi literatur, perancangan sistem, pengujian dan pengambilan data, dan yang terakhir adalah penyusunan laporan berupa buku Tugas Akhir. Selain itu dilakukan diskusi dan bimbingan bersama pembimbing lapangan dan dosen pembimbing. Hal ini membantu peneliti untuk merencanakan sebuah proyek melalui saran dan evaluasi dari pembimbing.

Pada tahap studi literatur dilakukan pencarian data, bahan, dan sumber atau pedoman yang diperoleh dari jurnal, *paper*, penelitian-penelitian sebelumnya, web, dan *Sublocus Referance Manual*. Dalam tahap perancangan sistem terdiri dari dua, yaitu perancangan perangkat keras dan perancangan perangkat lunak. Perancangan perangkat keras meliputi konfigurasi *input*/*output* yang terdiri dari Sublocus, RS 422 ke TTL, Arduino, modul SD Card dan komputer. Perangkat lunak yang dirancang menggunakan *flowchart* untuk bahasa pemrograman C pada Arduino IDE. Tahap berikutnya pengambilan data percobaan melalui sensor Sublocus yang diuji dengan menghubungkan ke Arduino menggunakan RS 422. Data yang telah diperoleh dianalisis menurut pergerakan data dari waktu ke waktu. Dari hasil data tersebut, dapat ditarik kesimpulan dari penelitian yang telah dilakukan. Hasil dari keseluruhan tugas akhir disusun dalam laporan penelitian.

1.6. Sistematika Penulisan

Pembahasan Tugas Akhir ini dibagi menjadi lima bab dengan sistematika sebagai berikut :

BAB I : PENDAHULUAN

Pada bab pertama berisi latar belakang masalah yang dibahas, rumusan masalah, tujuan, batasan masalah, metode penelitian, sistematika penulisan laporan.

BAB II : TINJAUAN PUSTAKA

Dalam pembahasan bab dua, diulas mengenai pengertian, cara kerja dari kapal selam, INS (Sublocus), Akuisisi, *Logging* Data, dan Sistem *Embedded* maupun komunikasi serial.

BAB III : PERANCANGAN SISTEM DAN REALISASI

Pembahasan bab tiga berisi desain dan perancangan perangkat keras yang meliputi *input*/*output* komponen. Perangkat lunak, yaitu pembuatan *flowchart* untuk pemrograman bahasa C pada Arduino.

BAB IV : PENGUJIAN DAN ANALISIS

Bab empat dilakukan realisasi perancangan dan hasil dari data yang telah diambil melalui pengujian perangkat INS dan juga hubungan dengan rangkaian-rangkaian terkait.

BAB V : PENUTUP

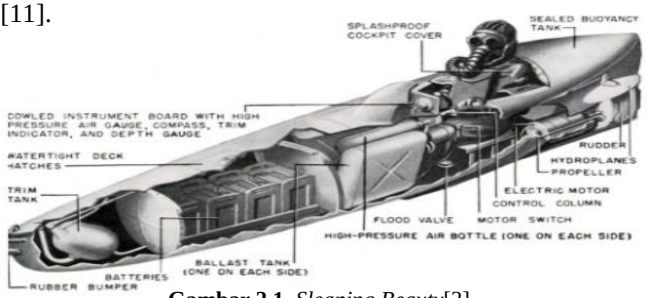
Kesimpulan dan saran dari hasil pembahasan yang telah diperoleh dibahas dalam bab kelima.

BAB II
TINJAUAN PUSTAKA

Dalam penelitian Akuisisi dan *Logging* Data *Inertial Navigation System* pada Kapal Tempur Bawah Air ini, terdapat beberapa studi literatur yang membantu dan merekomendasikan teknik-teknik pengolahan maupun pengetahuan dalam pengerjaan penelitian Tugas Akhir. Kapal Tempur Bawah Air merupakan kendaraan kapal selam yang sedang dikembangkan oleh PT. Bhimasena Research and Development. Kendaraan ini dibutuhkan oleh Komando Pasukan Katak (Kopaska) untuk misi infiltrasi maupun sabotase. Kegiatan tersebut dilakukan dengan menggunakan bahan peledak ke daerah lawan atau obyek vital musuh. Dengan adanya kapal tersebut, infiltrasi ke sasaran di permukaan akan lebih aman, karena tidak terdeteksi secara kasat mata oleh lawan. Penulis mengambil beberapa contoh penelitian yang telah dilakukan sebelumnya dan teori-teori terkait.

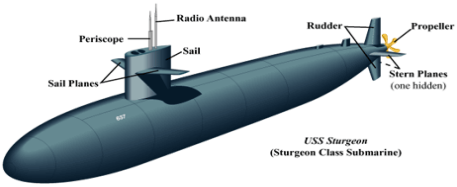
2.1. Kapal Selam

Pengenalan kapal selam pertama kali diketahui oleh William Bourne pada tahun 1580. Kapal selam adalah kapal yang bergerak di bawah permukaan air, umumnya digunakan untuk tujuan dan kepentingan militer. Dalam penelitiannya, William Bourne menulis tentang prinsip bahwa dengan memasukan dan mengeluarkan air secara bergantian, kapal tersebut mengubah ketinggiannya. Pada Perang Dunia kedua, kapal selam digunakan untuk memotong jalur pasokan, menghancurkan kapal musuh, atau menjelajahi wilayah musuh. Penggunaan kapal selam selama Perang Dunia II mengarah pada penemuan *Sleaping Beauty*, yaitu kapal selam berbentuk kapal bermotor, seperti yang diilustrasikan pada Gambar 2.1[3],[11].



Gambar 2.1. *Sleaping Beauty*[3]

Selama perang dunia kedua, penemuan ini diimplementasikan secara luas untuk mengeksplorasi wilayah musuh tanpa terdeteksi, tetapi dapat menempuh jarak yang lebih jauh[3]. Kapal selam menggunakan prinsip penerapan Hukum Archimedes yang tekait dengan terapung, melayang dan tenggelam. Hukum archimedes menyatakan: “Jika sebuah benda tercelup seluruh atau sebagian di dalam zat cair (fluida) mengalami gaya ke atas yang besarnya adalah sama dengan berat zat cair yang dipindahkan". Secara matematis, gaya archimedes dinyatakan dengan perkalian massa jenis dengan percepatan gravitasi dan volume benda yang tercelup. Kapal tersebut mampu mengapung dan menyelam ke dasar samudra dengan bantuan tangki pemberat antara dinding luar dan dinding dalam dapat diisi air laut sehingga meningkatkan bobot keseluruhan dan mengurangi kemampuan mengapungnya. Dengan dorongan baling-baling ke depan dan pengarahan bilah kemudi datar ke bawah, kapal dapat menyelam. Setelah berada di dalam air, kapal mempertahankan posisinya dengan bantuan tangki-tangki pemberat. Lain halnya ketika kapal naik ke permukaan atau dalam keadaan mengapung, kapal selam mengeluarkan air dari tangki pemberat[11]. Pada Gambar 2.2 telah ditinjukan ilustrasi kapal selam beserta komponen atau alat pendukungnya.



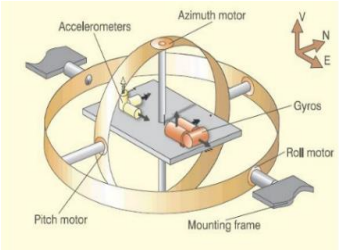
Gambar 2.2. Kapal Selam[2]

Terdapat pula perangkat-perangkat navigasi yang membatu dalam kinerja kapal selam, seperti periskop, radar, sonar dan jaringan satelit INS (*Inertial Navigation System*)[11], [12]. INS berperan sebagai penghitung sikap kendaraan di bawah air secara kontinu, perhitungan sikap diperbarui saat kendaraan bergerak dan koreksi gravitasi diperhitungkan dengan benar sehingga diperoleh perkiraan kecepatan dan posisi pada sistem navigasi lokal[12].

2.2. Sistem Navigasi Kapal

Navigasi atau pandu arah adalah penentuan kedudukan dan arah perjalanan di medan sebenarnya maupun di peta. Perangkat dari navigasi

bervariasi, contohnya seperti GPS (*Global Positioning System*), Loran-C, INS (*Inertial Navigation System*), Kompas dan lain sebagainya. Penggunaan perangkat tersebut bergantung pada kebutuhan letaknya, karena navigasi dapat dilakukan di darat, di laut, maupun di udara. INS adalah sistem navigasi inersia yang dapat memberikan posisi dan kecepatan pada kapal. Dengan memproses sinyal dari perangkat ini[13], navigasi inersia dapat mengandalkan pengetahuan posisi awal, kecepatan (*velocity*), dan sikap (*attitude*) untuk mengukur tingkat sikap dan akselerasi. Alat ini yang bersifat otonom dan tidak bergantung pada alat bantu atau kondisi visibilitas eksternal[4]. Sebelum kapal dan sensor ini menyelam, sistem didahului oleh penerima GPS (*Global Positioning System*) atau dengan kalibrasi. Dari hal tersebut, lokasi saat ini diperkirakan dengan menghitung lokasinya relatif terhadap lokasi yang diukur sebelumnya. Istilah ini dikenal dengan nama *dead-reckoning*[3]. Navigasi inersia digunakan dalam berbagai aplikasi, seperti navigasi pesawat terbang, pesawat ruang angkasa, kapal selam dan kapal laut[13]. Prinsip navigasi inersia adalah pengukuran percepatan atau akselerasi. Percepatan ini kemudian diintegrasikan ke dalam kecepatan dan diintegrasikan ke dalam posisi[14]–[16]. Dalam sistem INS, tingkat angular dan pengukuran kekuatan spesifik dari IMU (*Inertial Measurement Unit*) diproses untuk menghasilkan keadaan posisi, kecepatan dan sikap kapal selam[16]. Ilustrasi perangkat IMU dapat dilihat pada Gambar 2.3.



Gambar 2.3. Diagram Blok INS

IMU memiliki tiga sumbu *accelerometer*, yaitu satu arah utara-selatan, satu timur-barat, dan satu ke atas, semuanya terpasang pada *platform* yang stabil dan berorientasi ortogonal[4]. Selain itu juga terdapat tiga sumbu *gyroscope*, tiga sumbu *magnetometer* dan *reciver* GPS yang digunakan sebagai alat pemantau posisi dan sikap[3], [7], [16], [17].

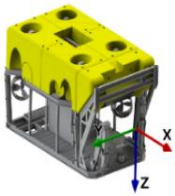
Berdasarkan literatur nomor tiga dirancang sistem navigasi yang dapat diintegrasikan ke dalam kapal selam Ortega. Kapal selam keluaran Ortega dapat melakukan perjalanan baik di atas permukaan dan di bawah air dengan kecepatan 25 km/jam, dan memiliki jangkauan 200 km. Ortega perusahaan yang telah meneliti tiga konsep kapal selam. Konsep pertama MK1A, kapal selam duduk tunggal yang dirancang khusus untuk survei dan industri lepas pantai. Kedua, yaitu MK1B, versi duduk ganda di mana penelitian bawah laut dan perlindungan pelabuhan. Ketiga, MK1C merupakan kapal selam berisi tiga orang yang dikembangkan terutama untuk tujuan pertahanan[3]. Pada kapal selam Ortega ini terdapat sedikit masalah dimana navigasi tidak dapat menembus air[2]. Hal ini dikarenakan sistem navigasi erat hubungannya dengan satelit yang menggunakan kinerja sinyal[13]. Hal pertama yang perlu dilakukan sistem navigasi yaitu meminta informasi tentang lokasi kapal selam, sedangkan sinyal tidak mendukung untuk proses di dalam laut. Kesalahan akumulatif dan keakuratan sistem mudah menurun seiring berjalannya waktu. Dalam penelitian de Kuil digunakan kombinasi dengan metode bantu lainnya, seperti memanfaatkan kinerja dari INS dan DVL (*Doppler Velocity Log*). Melalui gabungan sistem antara INS dan DVL, sensor mencapai akurasi sebesar 0,08% dari total jarak yang ditempuh dan lebih akurat dibanding sensor lainnya. Komunikasi data kombinasi dua sistem tersebut menggunakan RS 232 dan RS 422[3].

Dalam dunia industri terdapat tipe INS yang untuk membantu sistem navigasi kapal selam, seperti Sublocus. Tipe INS tersebut merupakan sistem navigasi inersia bawah air yang memberikan posisi, kecepatan dan orientasi akurat pada kedalaman hingga 3000 meter dan memiliki akurasi tinggi[14]. Bentuk dari sublocus dapat dilihat pada Gambar 2.4.

Gambar 2.4. Sublocus[6]

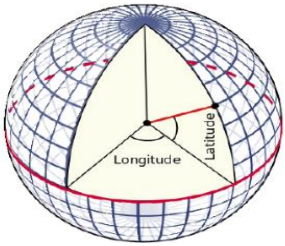
Sublocus memiliki sensor inersia kinerja tinggi yang dikombinasikan dengan algoritme fusi revolusioner Advanced Navigation. Hal ini dapat memberikan data posisi, kecepatan, *roll*, *pitch* dan *heading* kendaraan

yang sangat akurat. Spesifikasi dari Sublocus dapat dilihat pada lampiran A-3[6]. Selain itu, sublocus dapat memberikan akurasi posisi bawah laut sebesar 0,08% jarak tempuh. Perangkat lunak tersebut dilengkapi dengan sensor kedalaman tekanan dan penerima GPS dengan nilai kedalaman 6000 meter. Penerima GPS memberikan posisi awal sebelum kapal menyelam dengan akurat tanpa entri manual. Sensor inersia memiliki 3 sumbu, yaitu X, Y, dan Z sebagai penentu arah disekitar sudut dan akselerasi yang diukur. Sumbu ini ditunjukan pada Gambar 2.5 dengan sumbu X mengarah ke depan, sumbu Z mengarah ke bawah dan sumbu Y menunjuk papan kanan[6].



Gambar 2.5. Sumbu Sensor Inersia[6]

Orientasi pada Gambar 2.5 dapat diartikan dengan tiga sumbu *roll*, *pitch* dan *heading*, yang dikenal dengan sebutan *Euler*. *Roll* merupakan sudut di sekitar sumbu X, *Pitch* adalah sudut disekitar sumbu Y, sedangkan *Heading* sudut disekitar sumbu Z. Selain itu sublocus dapat memberikan sistem koordinat geodetik untuk menggambarkan posisi absolut di Bumi. sistem koordinat ini terdiri dari sudut lintang (*latitude*) dan bujur (*longitude*). *Latitude* sudut yang menentukan posisi utara ke selatan dari sebuah titik di permukaan bumi, sedangkan *longitude* merupakan sudut yang menentukan posisi timur ke barat dari sebuah titik di permukaan bumi. Pada Gambar 2.6 dapat dilihat ilustrasi dari *latitude* dan *longitude*[6], [14].



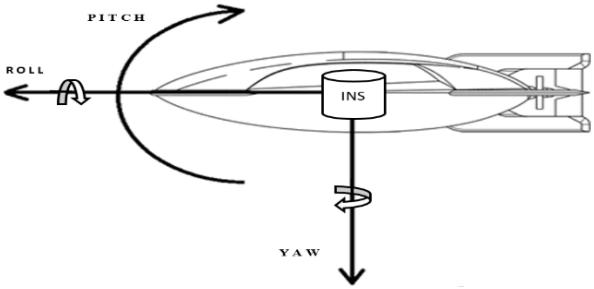
Gambar 2.6. Garis Lintang (*Latitude*) dan Garis Bujur (*Longitude*)[6]

Garis lintang nol adalah garis khatulistiwa dan garis nol bujur merupakan garis meridian utama[14]. Sublocus memiliki algoritme pencarian utara yang sangat cepat dan mampu memberikan arah akurat dalam waktu 10 detik setelah perangkat diaktifkan. Algoritme pencarian utara Sublocus berjalan secara terus menerus saat beroperasi dan tidak dipengaruhi oleh kecepatan atau gerakan sudut.

Perangkat navigasi yang digunakan dalam Kapal Tempur Bawah Air adalah Sonar (*Sound Navigation and Ranging*), INS dan Kamera seperti pada Gambar 2.7. Alat-alat tersebut dapat bekerja sesuai fungsinya di bawah air.

Gambar 2.7. Navigasi pada Kapal Tempur Bawah Air

Pada sistem navigasi Kapal Tempur Bawah Air, Kamera digunakan untuk mengambil citra lingkungan sekitar. Sonar berperan sebagai pendeteksi benda-benda sekitar, sedangkan INS untuk mengukur keadaan, posisi, dan kecepatan kapal. Letak INS pada Kapal Tempur Bawah Air dapat diilustrasikan pada Gambar 2.8.

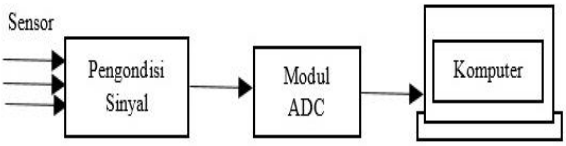


Gambar 2.8. INS pada Kapal Tempur Bawah Air

Pemasangan perangkat mempengaruhi keakuratan perhitungan, dimana idealnya sumbu Sublocus harus di sejajarkan dengan sumbu kendaraan. Dengan demikian, sumbu X menunjuk ke depan kendaraan dan sumbu Z menunjuk ke bawah. Letak perangkat INS berada di bawah pengemudi, dimana hal ini bertujuan untuk acuan perhitungan keadaan, posisi serta kecepatan ketika kapal dioperasikan.

2.3. Akuisisi dan Logging Data

Pengertian dari Akuisisi data adalah proses pengambilan, pengumpulan data, transformasi maupun pengukuran fenomena listrik atau fisik, seperti tegangan, arus, suhu, tekanan, atau suara dengan komputer[5], [9]. Sistem akuisisi terdiri dari sensor, perangkat keras pengukuran, dan komputer dengan perangkat lunak yang dapat diprogram[5]. Secara aktual, akuisisi berupa *interface* antara lingkungan analog dengan lingkungan digital. Lingkungan analog meliputi tranduser atau sensor dan pengkondisi sinyal dengan segala kelengkapannya, sedangkan lingkungan digital meliputi pengubah analog ke digital (ADC) dan pengolah digital dilakukan oleh mikroprosesor dengan bantuan program aplikasi[18]. Sistem akuisisi data dapat dilihat pada Gambar 2.9.



Gambar 2.9. Sistem Akuisisi Data Berbasis Komputer

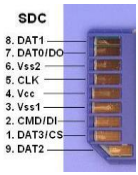
Sistem akuisisi data mengubah sinyal yang berasal dari sensor menjadi urutan nilai numerik digital yang dapat dianalisis menggunakan komputer[7], [19], [20]. Komponen sistem akuisisi data meliputi sensor untuk mengubah parameter fisik menjadi sinyal listrik, sirkuit pengkondisian sinyal untuk mengubah sinyal sensor menjadi bentuk yang bisa dikonversi menjadi nilai digital dan konverter ADC untuk mengubah sinyal sensor terkondensasi ke nilai digital. Terakhir, sebagai operasi perangkat lunak yang mendukung hasil yang diinginkan, yaitu sebuah komputer[21], [22]. Aplikasi akuisisi data biasanya dikendalikan oleh perangkat lunak yang dikembangkan dengan menggunakan berbagai

bahasa pemrograman tujuan umum, seperti Assembly, BASIC, C, C ++, C #, Fortran, Java, LabVIEW, Lisp, Pascal, dan sebagainya[22].

Lain halnya tentang *Logging* data yang memiliki pengertian, yaitu aplikasi pengukuran dan pencatatan data atau parameter fisik atau listrik selama periode waktu tertentu[9]. Dengan kata lain, proses pada komputer digunakan untuk mengumpulkan data melalui sensor dimana *output* adalah hasil data yang dikumpulkan, dianalisis dan disimpan. Data *logger* merupakan instrumen elektronik yang merekam pengukuran dari waktu ke waktu untuk sistem akuisisi data [23]. Di sisi lain data *logger* memiliki kemampuan untuk menerima lebih banyak saluran *input*, dengan resolusi dan akurasi yang lebih baik. Selain itu, data *logger* biasanya memiliki beberapa bentuk kecerdasan *on board* yang memberi pengguna kemampuan beragam. Sebagai contoh, data mentah dapat dianalisis untuk memberikan laju alir, sensor dan data yang diinterpretasikan lainnya jika tidak memerlukan analisis manual oleh operator[24]. Data dapat berupa suhu, tekanan, percepatan, arus, voltase, perpindahan, atau berbagai parameter lainnya[9]. Gambar 2.10 menunjukan ilustrasi dari sistem *Logging* data.

Gambar 2.10. Komponen Sistem *Logging* Data

Proses ini bertujuan untuk mengarsipkan data-data yang telah diambil untuk dianalisa atau diproses lebih lanjut serta didasarkan pada komputer terhubung ke sejumlah sensor dan penyimpanan data[25]. Terdapat dua perbedaan fungsi mendasar data *logger*, yaitu bahwa satu jenis memungkinkan data disimpan dalam memori yang diambil di lain waktu, sementara jenis lainnya secara otomatis mencatat data untuk dilihat dan dianalisis secara langsung[24]. Memori merupakan piranti yang dapat menyimpan dan merekam informasi. *Logging* data umumnya memanfaatkan SD Card, EEPROM, atau jenis memori lainnya sebagai media penyimpanan[25]. SD Card salah satu dari banyak jenis perangkat penyimpanan memori yang dapat dilepas[26]. *Device ini* diperkenalkan untuk mengatasi kebutuhan penyimpanan data yang andal dalam bentuk kecil, seperti ponsel, kamera digital, dan lainnya[27]. Benda ini memiliki ukuran relatif kecil, yaitu 32x24 mm dan memiliki 9 pin untuk pertukaran data antara sistem yang terhubung dan kontroler. Pin-pin tersebut dapat dilihat pada Gambar 2.11[26], [28].



Gambar 2.11. Pin SD Card[27]

Komunikasi dengan kartu SD bisa dilakukan di salah satu dari dua mode, yaitu mode SD atau mode SPI[28]. Proses komunikasi SD Card adalah pengaturan *master* atau *slave* terdiri dari pertukaran *command* dan atau tanggapan *token* antara *host* dan kartu melalui jalur digital. Transfer data terjadi dalam paket yang terdiri dari blok data dan bit[26]. Perangkat tersebut beroperasi dalam mode serial *clock* dengan lebar bit 14, sedangkan pada mode SPI (*Serial Peripheral Interface*), SD Card hanya beroperasi dalam mode 1 bit. Terdapat sinyal pembawa dalam proses komunikasi ini, yaitu CLK, CMD, dan DAT. CLK, membawa sinyal *clock* dari *host*. CMD, membawa perintah dari *host* dan tanggapan dari kartu. DAT, membawa data dari *host* atau data dari kartu[27].

Penelitian nomor sepuluh ditemukan permasalahan dimana kebanyakan sistem DAQ (*Data Acquisition System*) gagal untuk penggunaan otomotif karena sifat spesifik aplikasi dan kurangnya kemampuan penyaringan dan pemrosesan data. Tujuan dari penelitian ini untuk mengembangkan sistem yang mampu menangkap data, menyimpannya dan kemudian membuatnya tersedia secara *offline* sebagai peninjau kinerja, keandalan dan penanganan kendaraan. Sistem ini menggunakan sensor suhu LM50 dan MPU-6050 yang berisi *accelerometer* MEMS (*Micro Electro Mechanical Systems*) dan sebuah MEMS *gyroscope* dalam satu *chip*. Dengan ini, Sushant Agrawal dan kawan-kawan mengusulkan tiga sub sistem yang berbeda untuk menyelesaikan permasalah tersebut. Pertama, serangkaian transduser dan sensor mencatat berbagai parameter, seperti suhu, percepatan, *pitch*, *roll*, tekanan, gaya dan perpindahan. Setelah itu unit mikrokontroler atau data *logger* mencatat data digital, mengubah data analog menjadi data digital 10 bit, memberikan pengkondisian sinyal dan pemetaan nilai sinyal yang diperlukan untuk mencatat data digital pada SD Card hingga 16 GB (*gigabyte*). Sistem ketiga, yaitu melalui bus komunikasi serial sehingga data ditransfer ke komputer untuk menjalankan analisis. Semua analisis dilakukan dengan menggunakan perangkat lunak MATLAB. Dengan sistem tersebut dapat menangkap sumbu x, y, dan z secara bersamaan.

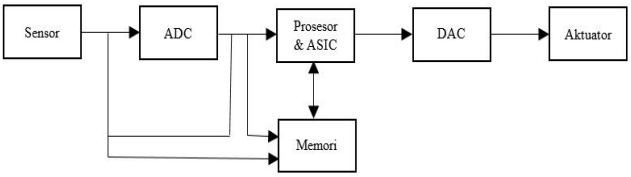
Penelitian di India ini mengusulkan menggunakan perangkat Arduino Uno sebagai jawaban dari proses data *logger*. Metode penyimpanan data dan format data menggunakan MATLAB dan Microsoft Excel agar dapat membaca file dalam format .csv (*comma-separated-value*). Format ini sangat baik dihasilkan oleh penyimpanan kartu SD Card yang kompatibel. Selain itu dapat dilampirkan sebagai modul diskrit ke *board* Arduino Uno dan kemudian data mengarah pada jalur yang dilalui dari sensor ke PC [10]. Literatur nomor sembilan melakukan penelitian mengenai perancang data akuisisi pH dan perangkat *logging* menggunakan mikrokontroler. Penelitian ini meliputi keberhasilan penerapan teknik akuisisi data dan data *logging* untuk keperluan analisis dan berbagi data. Perangkat yang digunakan adalah mikrokontroler Renesas M16C/62P. Dalam mikrokontroler tersebut terdapat EEPROM (*Electrically Erasable Programmable Read-Only Memory*) sebagai media *logging* data atau penyimpanan dan terdapat ADC sebagai konversi nilai analog ke digital. Akuisisi data meliputi proses penggalian, transformasi dan pengangkutan data dari sistem sumber dan sumber data eksternal ke mikrokontroler untuk penyimpanan. Data ini dapat diamati secara *real time* maupun *offline* dengan mengunduh memori EEPROM[9]. Tujuan dari penelitian nomor sebelas adalah penentuan posisi inertial berdasarkan pengukuran dan pengolahan data akselerasi untuk pengendalian panduan dan fungsi AUV. Metode dan implementasi yang digunakan meliputi pengembangan aplikasi Android dan GUI dengan menggunakan perangkat lunak Microsoft Visual Studio dan juga Arduino dengan bahasa pemrograman C. *Output* dari pembacaan sensor dapat dilihat melalui serial monitor pada perangkat lunak Arduino. Sistem ini dapat melakukan akuisisi dan pemrosesan sinyal digital yang dihasilkan dari unit penerima GPS dan unit pengukuran inersia mikro. Kemudian data dikirim ke stasiun induk dalam bentuk pesan secara berkala. Hasilnya integrasi antara komponen perangkat keras dan perangkat lunak yang telah dibuat berhasil mengkalibrasi semua data dan menampilkannya secara *real time* dalam bentuk kecepatan, suhu, *pitch*, *roll*, kompas, rekaman video, dan tampilan bujur dan lintang pada peta. Sensor GPS mendeteksi lokasi dimana AUV berada[13].

2.2.1. Sistem Embedded

Pengertian dari sistem *embedded* adalah sistem terintegrasi yang terdiri dari komponen perangkat keras dan perangkat lunak [29], [30].

Pada prosesnya, sistem ini dapat menjadi sistem yang independen atau menjadi bagian dari sistem yang besar. Sistem *embedded* berbasis mikroprosesor, dimana dapat dirancang untuk melakukan tugas tertentu. Prosessor *embedded* memiliki tiga komponen, yaitu perangkat keras, aplikasi perangkat lunak dan memiliki sistem operasi *real time* yang disebut RTOS. Proses RTOS bekerja mengawasi aplikasi perangkat lunak dan memberikan mekanisme prosesor dalam menjalankan proses sesuai penjadwalan yang telah direncanakan.

Beberapa sistem menyediakan *interface* pengguna dari jarak jauh dengan bantuan serial, contohya koneksi RS-232, USB, I²C, dan lain-lain atau jaringan, seperti *Ethernet*. Pendekatan ini memberikan beberapa keuntungan, yaitu memperluas kemampuan sistem *embedded* dan memungkinkan seseorang membangun *interface* di komputer[31]. Ilustrasi berikut, yaitu Gambar 2.12 menunjukan struktur dasar sistem *embedded*[32] :



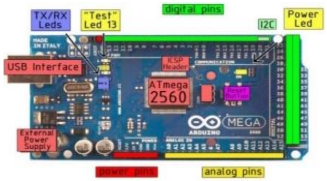
Gambar 2.12. Diagram Blok Struktur Sistem *Embedded*

Diagram blok pada Gambar 2.12 menunjukan hubungan dan masing-masing fungsi, seperti sensor untuk mengukur kuantitas fisik dan mengubahnya menjadi sinyal listrik yang dapat dibaca oleh pengamat atau instrumen elektronik. Sensor menyimpan jumlah yang terukur ke memori. Kemudian terdapat ADC atau konverter analog ke digital untuk mengubah sinyal analog yang dikirim oleh sensor menjadi sinyal digital. Prosesor & ASIC sebagai pemproses data untuk mengukur *output* dan menyimpannya ke memori. DAC atau konverter digital ke analog untuk mengubah data digital yang diumpankan oleh prosesor ke data analog. Aktuator sebagai pembanding *output* yang diberikan oleh DAC dengan *output* aktual yang tersimpan. Prosessor adalah inti dari sistem *embedded* karena berperan sebagai pengambil *input* dan menghasilkan *output* setelah pemrosesan data selesai[32].

Sistem *embedded* dapat dibagi menjadi dua kategori, yaitu Mikroprosesor (µP) yang menggunakan sirkuit terpadu terpisah untuk memori dan *peripheral*. Kategori berikutnya mikrokontroler (µC) yang

memiliki *peripheral on-chip* sehingga mengurangi konsumsi daya[31]. Mikroprosesor adalah unit pemroses komputer digital untuk membaca data, melakukan perhitungan ekstensif terhadap data, dan menyimpan hasilnya pada perangkat penyimpanan massal atau menampilkan hasilnya. Mikrokontroler merupakan pengendali operasi mesin dengan menggunakan program tetap yang tersimpan dalam ROM dan tidak berubah sepanjang masa sistem. Perangkat ini mendapatkan data dari dan ke pinnya sendiri. Arsitektur dioptimalkan untuk menangani data dalam ukuran bit dan *byte*[33]. Pada tahun 1978, National Engineering Manufacturers Association merilis "standar" untuk mikrokontroler yang dapat diprogram, termasuk pengendali berbasis komputer, seperti komputer *single board*, numerik, dan pengendali. Tahun 1980, komponen sistem memori, *input* dan *output* telah diintegrasikan ke dalam *chip* sama dengan prosesor yang membentuk mikrokontroler. Implementasi sistem *embedded*, kini telah maju sehingga mudah diterapkan dengan *board* berbasis *platform*. Contoh implementasi sistem *embedded*, yaitu pada Arduino dan Raspberry Pi[31].

Arduino adalah pengendali mikro *single board* yang bersifat *open source*, dimana dirancang untuk memudahkan penggunaan elektronik dalam berbagai bidang. Pada perangkat keras Arduino memiliki prosesor *Atmel AVR* dan perangkat lunaknya memiliki bahasa pemrograman sendiri, yaitu C+. Perangkat ini memiliki beberapa jenis, diantaranya Arduino Uno, Arduino Due, Arduino Mega, Arduino Leonardo dan masih banyak lagi[34]. Arduino Mega merupakan *board* mikrokontroler berdasarkan ATmega 2560. Jenis Arduino ini memiliki 54 pin *input*/*output* digital, dimana 14 pin dapat digunakan sebagai *output* PWM, 16 pin *input* analog, 4 UART (*Universal Asynchronous Receiver Transmitter*), osilator kristal 16 MHz, koneksi USB, colokan listrik, header ICSP, dan tombol reset. Ilustrasi dari Arduino Mega dapat dilihat pada Gambar 2.13.



Gambar 2.13. Arduino Mega[8]

Board ini berisi semua yang dibutuhkan dalam proses elektrik, dan untuk

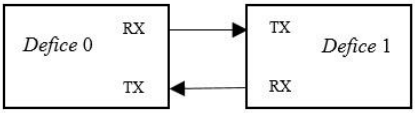
memulai mikrokontroler cukup dihubungkan ke komputer dengan kabel USB atau nyalakan dengan adaptor AC ke DC atau baterai. Arduino Mega dapat diproses melalui koneksi USB atau dengan catu daya eksternal. Pin daya Arduino Mega yaitu pin Vin merupakan tegangan *input* ke papan Arduino Mega saat menggunakan sumber daya eksternal, kemudian terdapat pin 5V (VCC), yaitu catu daya yang diatur untuk menyalakan mikrokontroler dan komponen lainnya pada *board*. Terdapat pula pin 3V3, pasokan 3,3 volt yang dihasilkan oleh regulator *on board*. Terakhir, pin GND (Ground) atau dikenal sebagai pin tanah. Masing-masing dari 54 pin digital pada Arduino Mega dapat digunakan sebagai *input* atau *output*, dengan menggunakan fungsi pemrograman bahasa C, seperti `pinMode ()`, `digitalWrite ()`, dan `digitalRead ()`. ATmega 2560 memiliki memori flash 256 KB untuk menyimpan kode, dimana 8 KB digunakan untuk *bootloader*, 8 KB SRAM dan 4 KB dari EEPROM yang dapat dibaca dan ditulis dengan perpustakaan EEPROM. Selain itu, beberapa pin memiliki fungsi khusus, yaitu serial RX dan TX yang digunakan untuk menerima dari jalur RX dan mengirimkan ke TX data serial TTL. Terdapat pula pin Interupsi Eksternal yang dapat dikonfigurasi untuk memicu interupsi pada nilai *low*, *high* atau perubahan nilai. Fungsi bahasa pemrograman tersebut `attachInterrupt ()`. Selanjutnya pin PWM, yaitu 0 sampai 13 dan disediakan *output* PWM 8 bit dengan fungsi `analogWrite ()`. Pin SPI berisi MISO, MOSI, SCK, SS yang mendukung komunikasi SPI dengan menggunakan *library* SPI. Pin LED *built-in* yang terhubung ke pin digital. Bila pin bernilai *high*, LED menyala, bila pin bernilai *low* artinya tidak menyala. Terakhir, I2C yang berisi SDA dan SCL, dimana komunikasinya menggunakan *library* Wire. Beberapa pin lainnya seperti AREF, yaitu tegangan referensi untuk *input* analog yang digunakan dengan fungsi `analogReference ()` dan RESET untuk mereset mikrokontroler[8].

Pada literatur nomor 7 melakukan pengembangan tentang sistem navigasi inersia menggunakan mikrokontroler Arduino, dimana pengukuran oleh sensor IR (inframerah) dan sensor MPU 6050 (*accelerometer* dan *gyroscope*). Ketika mendapatkan sinyal dari beberapa perangkat tersebut, kemudian dikirim ke unit pemrosesan sehingga bisa melacak perpindahan objek. Kemudian data ini dikirim ke PC dan hasilnya diplot dalam MATLAB. Sistem ini terus memantau gerak kendaraan dan memberikan hasil yang baik dan sangat sesuai untuk berbagai jenis aplikasi[7]. Literatur nomor 12 meneliti AUV (*Autonomus Underwater Vihacle*) berbasis INS[35], sedangkan pada nomor 13

melakukan riset mengenai integrasi data dari sistem navigasi GPS dan inersia untuk pejalan kaki[16]. Kedua penelitian tersebut memerlukan ketepatan informasi navigasi yang akurat, sementara INS mengalami kinerja terdegradasi di garis lintang tinggi akibat bangunan, infrastruktur kota dan keadaan ekologi laut. Dalam kasus ini, INS diintegrasikan dengan DVL, penerima GPS, dan sensor kedalaman. Serta sebuah EKF (*Extended Kalman Filter*) untuk mengintegrasikan data sistem GPS dan INS[16], [35]. Hasil yang diperoleh menunjukan bahwa EKF lebih akurat dan *robust*, dibandingkan algoritme yang menggunakan data dari sistem GPS dan INS secara terpisah[16].

2.2.2. Komunikasi Serial

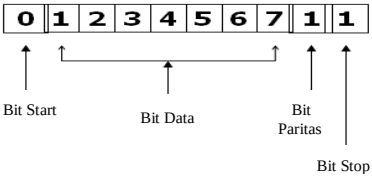
Suatu proses pengiriman data secara sekuensial atau satu persatu melalui sebuah kanal informasi disebut komunikasi serial. Jalur data serial pada komputer memungkinkan untuk melakukan komunikasi *half duplex* atau *full duplex*[36]. *Half duplex* merupakan bentuk komunikasi antara dua belah pihak dan dua arah, namun berkomunikasi secara bergantian. Dengan kata lain tidak dapat mengirim dan menerima secara bersama-sama. *Full duplex* adalah komunkasi antara dua belah pihak yang saling mengirim dan menerima informasi dalam waktu bersamaan. Pada umumnya mode ini memerlukan dua jalur komunikasi. Metode tersebut merupakan istilah dari komunikasi asinkron, dimana data dikirim satu *byte* setiap pengiriman. Jenis komunikasi asinkron tidak membutuhkan konfirmasi penerimaan data. Lain halnya dengan jenis komunikasi sinkron yang mengirimkan data menggunakan beberapa *byte* atau karakter sebelum meminta konfirmasi data sudah atau belum diterima dengan baik[37]. Pada Gambar 2.14 ditunjukan diagram blok komunikasi serial asinkron.



Gambar 2.14. Komunikasi Serial Asinkron

Komunikasi serial asinkron berbeda dengan serial sinkron, yaitu tidak adanya sinyal *clock* yang dapat menyinkronkan komunikasi data, tetapi hal tersebut dilakukan oleh *baud rate*. Dengan kata lain *Baudrate* adalah protokol komunikasi data dan kecepatan transfer yang dinyatakan dengan bit per *second* (bps). Apabila suatu mikrokontroler memiliki nilai *baud*

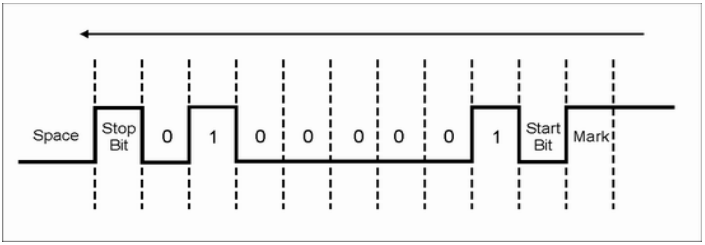
rate 9600 artinya mikrokontroler memiliki kecepatan transfer data 9600 bit per *second* maka 1 bit membutuhkan 1/9600 detik atau 0,000104 detik atau 0,104 ms (*milisecond*)[36]. Tingkat *baud rate* menentukan seberapa cepat data dikirim melalui jalur serial[38]. Proses komunikasi asinkron antar perangkat harus memiliki protokol yang sama antara perangkat satu dengan perangkat yang lain, apabila protokolnya berbeda maka terjadi kesalahan komunikasi data atau bahkan komunikasi data tidak dapat dilakukan. Protokol tersebut terdiri dari beberapa parameter yang dapat dilihat pada Gambar 2.15[36].



Gambar 2.15. Protokol Komunikasi Asinkron[36]

1. Bit *start* : Selalu bernilai 0, ketika komunkasi UART (serial asinkron) diberikan, tetapi terlebih dahulu dimulai dengan pemberian bit *start*. Fungsinya sebagai pemicu kepada penerima (RxD) bahwa terdapat data yang diberikan oleh pemancar (TxD) dan juga memicu *clock* pada *reciever* sehingga disinkronkan dengan *clock* pada *transmitter*.
2. Bit Data : Merupakan data yang dikirimkan secara UART dimulai dari LSB atau bit ke 0 hingga MSB atau bit terakhir. Dalam menentukan banyaknya bit tersebut haruslah sama antara pemancar dengan penerima. Banyaknya data bit pada AVR bisa bernilai 7,8, atau 9 data bit.
3. *Parity* (keseimbangan) : Berfungsi sebagai pengecekan eror data yang ditransfer. *Parity* dapat bernilai *odd* atau ganjil, *even* atau genap, dan *none* atau tidak bernilai.
4. Bit *Stop* : Selalu bernilai 1 berfungsi sebagai akhir dari komunikasi data dan kemudian masuk pada *Idle State*. Pengiriman data dapat dilakukan setelah bit *stop* diberikan.
5. *Idle state* : Merupakan kondisi tidak terjadinya komunikasi data dan jalur data berlogika 1 secara terus menerus[36].

Contohnya, pada Gambar 2.16, karakter A dengan biner 01000001 dikurung atau ditutup oleh bit *start* dan satu bit *stop*[37].



Gambar 2.16. Pembingkaian ASCII Karakter “A”[37]

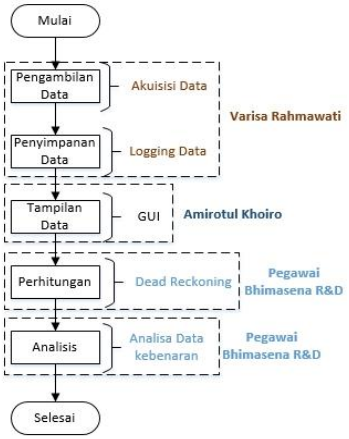
Komunikasi serial dapat menghubungkan mikrokontroler dengan peralatan lainnya. *Port* serial pada mikrokontroler terdiri atas dua pin, yaitu RxD dan TxD. RxD berfungsi untuk menerima data dari komputer atau peralatan lainya, sedangkan TxD berfungsi untuk mengirim data ke komputer atau peralatan lainya[36].[11], [36].

BAB III
PERANCANGAN AKUISISI DAN *LOGGING* DATA PADA INS

Dalam bab ini terdapat empat sub penjelasan mengenai navigasi Kapal Tempur Bawah Air, perancangan sistem Akuisisi dan *Logging* Data, yaitu blok fungsional sistem, perangkat keras dan lunak. Penjelasan diawali dengan penjelasan perangkat navigasi yang digunakan pada Kapal Tempur Bawah Air serta pembagian lingkup kerja, standar pengoperasian dan metode. Setelah itu dibahas mengenai blok fungsional sistem secara keseluran yang meliputi proses kerja alat dalam bentuk alur diagram. Terdapat perancangan konfigurasi pin *input*/*output* maupun perangkat-perangkat komunikasi yang mendukung cara kerja alat. Selain itu, terdapat pula langkah-langkah dalam pembuatan modul perangkat penyimpanan. Selanjutnya dibahas perancangan perangkat lunak yang terdiri dari *flowchart* program.

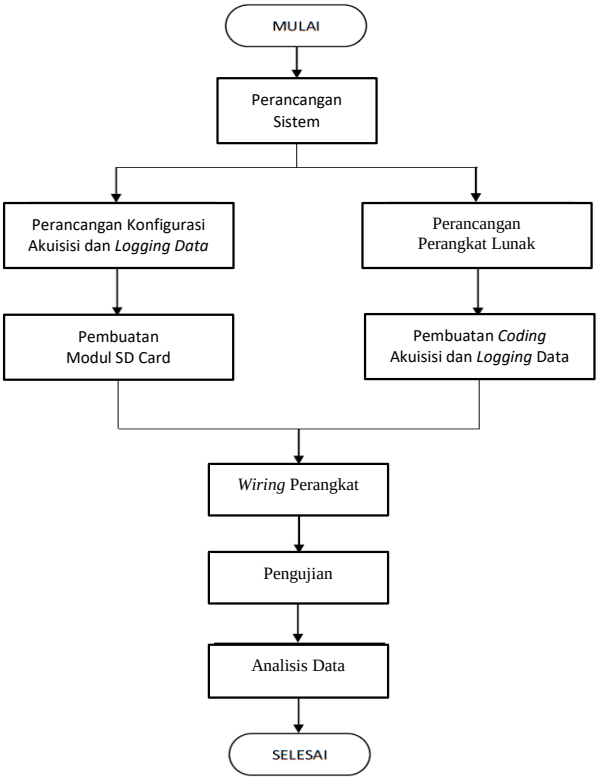
3.1. Metodologi Penelitian

Dalam proses penelitian ini, terdapat metode serta pembagian atau lingkup kerja pengolahan INS. Lingkup pengerjaan tersebut dimulai dari pengambilan data hingga analisis. Pengerjaan INS pada Kapal Tempur Bawah Air dapat dilihat pada *flowchart* Gambar 3.1.



Gambar 3.1. *Flowchart* Pengerjaan INS

Lingkup pengerjaan INS yang pertama adalah proses pengambilan dan penyimpanan data. Proses tersebut dikerjakan oleh peneliti yaitu Varisa Rahmawati. melalui perancangan akuisisi dan *logging* Data menggunakan Arduino dan SD Card. Tahap berikutnya adalah proses pengerjaan GUI (*Graphical User Interface*) sebagai tampilan monitor Kapal Tempur Bawah Air. Perancangan tersebut dikerjakan oleh Amirotul Khoiro. Berikutnya pengerjaan perhitungan atau *Dead-Reckoning* dan tahap analisis data oleh Pegawai PT Bhimasena Research and Development.Penelitiaan ini fokus pada akuisisi dan *logging* data INS (Sublocus) pada Kapal Tempur Bawah Air.Proses akuisisi dan *logging* data menggukakan sebuah sistem *embedded* sebagai pengendali utama dan pemroses data. Langkah pengerjaan dapat dilihat pada Gambar 3.2.

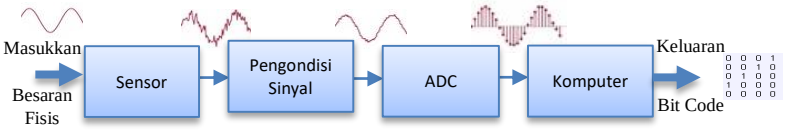


Gambar 3.2. Metodologi Penelitian

Langkah utama dalam pengerjaan penelitian ini adalah perancangan sistem, dimana hal tersebut dilakukan untuk memberikan gambaran secara umum kepada pembaca tentang sistem yang dibuat. Perancangan tersebut berisi tentang persyaratan proses akuisisi dan *logging* data. Berikutnya terdapat pengerjaan perangkat keras dan perangkat lunak. Untuk perancangan perangkat keras, dibuat konfigurasi sistem akuisisi dan *logging* data, adapun pembuatan modul SD Card. Perancangan perangkat lunak dilakukan untuk gambaran dasar pembuatan program. Kegiatan tersebut diimplementasikan dalam bentuk *flowchat*, dan berikutnya dilakukan *coding* akuisisi dan *logging* data. Setelah pengerjaan perangkat keras dan lunak, selanjutnya adalah proses *wiring* perangkat untuk kegiatan pengujian. Dari hasil pegujian didapatkan data yang berikutnya dianalisis ke dalam bentuk grafik.

3.2. Perancangan Sistem Akuisisi dan *Logging* Data

Tahap pertama merupakan perancangan blok fungsional sistem berupa diagram blok yang menjelaskan sistem utama penelitian. Dalam akuisisi dan *logging* data terdapat aturan atau persyaratan untuk menunjang pemrosesan. Persyaratan terjadinya akuisisi yaitu harus adanya Sensor sebagai objek pengambilan data. Setelah itu proses pengondisian sinyal dan ADC, serta komputer seperti pada Gambar 3.3.



Gambar 3.3. Proses Akuisisi Data

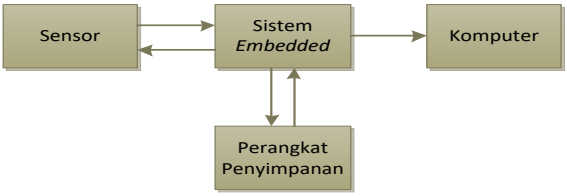
Pada Gambar 3.3 dapat diamati proses terjadinya sistem akuisisi, dimana masukkan sensor dalam adalah fenomena fisik atau sebuah besaran fisis. Masukan tersebut dalam bentuk sinyal analog, dimana setelah diproses pada sensor memiliki *noise*. Dengan demikian, dilakukan proses pengondisian sinyal agar *noise* dapat diminimalisir. Berikutnya dilakukan konversi sinyal analog ke digital menggunakan ADC. Sinyal analog disampling, sehingga menghasilkan sinyal dalam bentuk diskrit. Tahap terakhir yaitu sinyal diskrit diolah dalam komputer hingga menjadi bentuk kode bit. Dalam penelitian ini sensor yang digunakan adalah Sublocus yang memiliki frekuensi 50 Hz. Dari frekuensi tersebut dapat

menghasilkan 20 data/detik. Catu daya dan koneksi sinyal Sublocus dilakukan melalui konektor 12 pin. Konektor ini bernilai hingga 10.000 PSI dan dalam sistem pengerjaan dapat diberikan komunikasi standar RS232 atau RS422 pada jalur data utama. Sublocus menggunakan komunikasi serial RS 232 atau RS 422, dimana kecepatannya 1200 hingga 10M baud. Protokol yang digunakan dalam Sublocus adalah ANPP. Komunikasi data secara serial dilakukan dengan metode pengiriman data secara bit per bit atau satu per satu secara berurutan. Kecepatan transfer data RS 422 100 kbit/s hingga 10 Mbit/s. Pengiriman data dilakukan dua arah dengan menghubungkan 3 unit kabel, yaitu: kabel Tx, Rx dan GND. Keluaran dari RS 422 diteruskan ke Arduino untuk dilakukan proses parsing data yang kemudian dikirimkan ke komputer. Parsing data merupakan sebuah cara untuk memisahkan rangkaian *byte* ke dalam informasi yang diinginkan sesuai dengan referensi *datasheet* sensor. *Baudrate* yang telah ditetapkan sebesar 115200, sehingga dalam satu detik dapat mengirim sebanyak 115200 bit atau 115200/8 *byte*. Target yang telah ditentukan dari sistem akuisisi tersebut adalah dapat menampilkan 19 macam Data yaitu *Roll*, *Pitch*, *Heading*, *Latitude*, *Longitude*, *Microseconds*, *Gyroscope* X, *Gyroscope* Y, *Gyroscope* Z, *Accelerometer* X, *Accelerometer* Y, *Accelerometer* Z, *Magnetometer* X, *Magnetometer* Y, *Magnetometer* Z, GPS *Sattelite*, *Velocity North*, *Velocity East*, dan *Velocity Down*. Adapun persyaratan adanya proses *logging* data seperti yang ditunjukkan pada Gambar 3.4.

Gambar 3.4. Persyaratan Proses *Logging* Data

Pada Gambar 3.4 tahap paling utama yaitu kinerja sensor sebagai pembaca kondisi fisik. Sensor ini berikutnya diproses menggunakan perangkat keras *logging* data, dimana dalam penelitian ini adalah SD Card. Ketika data telah diproses, selanjutnya direkam pada perangkat lunak *logging* data atau dalam kata lain dapat berupa aplikasi Notepad. Target dari proses *logging* data adalah menyimpan 3 jenis data yaitu *Roll*, *Pitch*, *Yaw* atau *Heading*.

Dari persyaratan tersebut, maka dibuatlah rancangan blok sistem utama dapat dilihat pada Gambar 3.5.

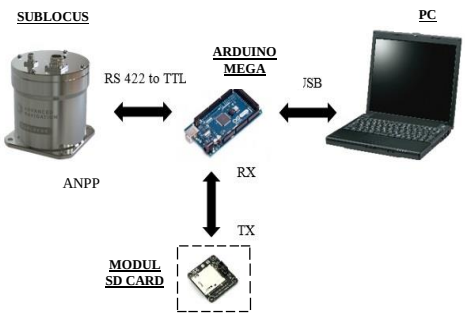


Gambar 3.5. Blok Sistem Akuisisi dan *Logging* Data

Diagram blok pada Gambar 3.5 menunjukan hubungan antara perangkat satu dan perangkat lainnya. Sensor dalam penelitian ini berguna untuk mengukur kuantitas fisik dan mengubahnya menjadi sinyal listrik yang dapat dibaca oleh pengamat atau instrumen elektronik lainnya. Sistem *Embedded* berperan sebagai mikrokontroler atau pengendali rangkaian elektronik, seperti sensor. Perangkat penyimpanan berperan sebagai media penyimpanan data untuk dianalisis secara *online* maupun *offline*. Dari hubungan komponen-komponen penunjang tersebut, hasil data yang diambil dapat ditampilkan pada layar komputer atau disimpan dalam perangkat penyimpanan.

3.3. Konfigurasi Sistem Keseluruhan

Tahap pada sub bab ini merupakan perancangan perangkat keras yang dilakukan dengan cara membuat konfigurasi antar perangkat melalui *input*/*output*. Hal ini bertujuan agar perangkat-perangkat tersebut dapat saling berkomunikasi antara satu dan lainnya. Perancangan ini meliputi proses akuisisi dan proses *logging* data serta pembuatan modul SD Card. Sistem tersebut dapat dilihat melalui diagram blok pada Gambar 3.6.

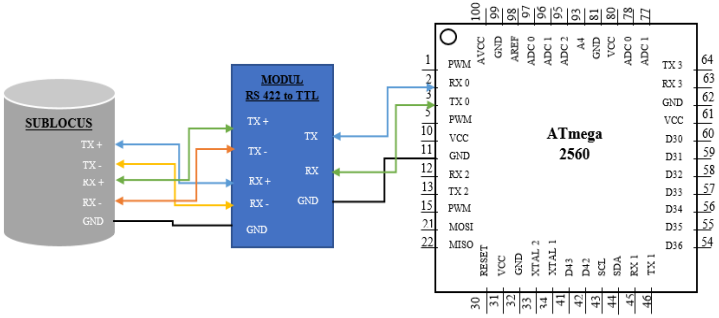


Gambar 3.6. Diagram Blok Perancangan Perangkat Keras

Dapat dilihat pada diagram blok yang ditunjukkan Gambar 3.6 bahwa sensor yang digunakan pada penelitian ini adalah Sublocus. Perangkat ini dapat menghitung posisi, kecepatan dan orientasi hingga kedalaman 3000 meter di bawah air. Data Sublocus dapat dikirim dalam format ANPP melalui jalur komunikasi RS 422 menuju mikrokontroler Arduino. ANPP (*Advanced Navigation Packet Protocol*) merupakan protokol biner yang dirancang dengan pengecekan eror dan efisiensi yang tinggi serta praktik perancangan yang aman. Protokol ini yang membantu dalam proses transmisi data atau paket data dari Sublocus. Hasil data yang telah diambil dan diproses oleh mikrokontroler Arduino, kemudian ditampilkan pada serial monitor Arduino pada PC (*Personal Computer*). Selain itu, data hasil pengukuran dapat disimpan pada modul SD Card dari waktu ke waktu secara kontinu. Hal tersebut, dikenal dengan istilah data *logger*.

3.5.1. Konfigurasi Akuisisi Data

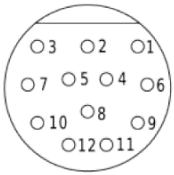
Dalam pengambilan dan pemrosesan data dari Sublocus, dibutuhkan metode untuk melakukan proses akuisisi. Metode ini merupakan ujung terdepan dari proses pengumpulan data mentah dari Sublocus. Sistem ini mengonversi sinyal fisik menjadi sinyal elektronik. Kemudian mendigitalisasi sinyal tersebut sehingga data dapat disimpan, ditransmisikan dan dapat ditampilkan pada komputer. Dengan ini dibutuhkan komponen atau perangkat untuk membantu dalam proses akuisisi data. Perangkat-perangkat penunjang metode untuk proses akuisisi dapat dilihat pada Gambar 3.7.



Gambar 3.7. Konfigurasi Sistem Akuisisi Data

Gambar 3.7 menunjukan konfigurasi atau komunikasi antar perangkat dari proses akuisisi data dalam penelitian ini. Dapat dilihat bahwa konfigurasi tersebut merupakan hubungan antar pin *input* dan *output* dari Sublocus ke perangkat-perangkat lainnya. Data dari Sublocus diproses menggunakan mikrokontroler Arduino yang menggunakan IC (*Integrated Circuit*) ATmega 2560. Setelah itu data ditampilkan melalui komunikasi serial, sedangkan data mentah dari sensor tersebut adalah *byte*. Dengan ini, dibutuhkan standar komunikasi serial untuk transmisi dan konversi data dari protokol Sublocus untuk dapat dibaca dan diolah dalam mikrokontroler. Proses akuisisi data ini memanfaatkan RS 422 sebagai standar komunikasi serial. Konfigurasi ini terdiri dari jalur komunikasi data TX, RX, dan GND. TX sebagai pin *transmitter* atau pengiriman, sedangkan RX adalah pin *receiver* atau penerima.

Sublocus merupakan elemen utama karena data bersumber dari sensor ini. Data keluaran dari Sublocus berupa *byte*, tetapi dapat ditransmisi dan dikonversikan menggunakan komunikasi standar RS 232 atau RS422 pada *port* komunikasi utama. Berikut ini Gambar 3.8 yang menunjukan penomoran pin pada Sublocus.

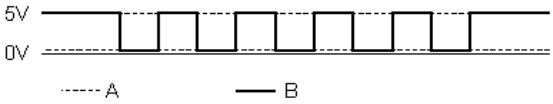


Gambar 3.8. Penomoran Pin Sublocus

Pada ilustrasi Gambar 3.8, pin yang dimiliki Sublocus sebanyak 12. Pin atau jalur komunikasi tersebut memiliki fungsi tersendiri, seperti yang diterangkan pada lampiran A-6. Jalur data yang digunakan pada proses ini adalah 5 pin, yaitu TX+, TX-, RX+,RX, dan GND. Pin tersebut berfungsi untuk jalur komunikasi Sublocus dengan tambahan *peripheral*, sensor dan format data. Dalam penelitian ini data yang diperoleh meliputi perhitungan *Latitude*, *Longitude*, *Velocity*, *Roll*, *Pitch*, *Yaw*, dan *Acceleration*. Perangkat Sublocus menggunakan paket protokol yang mengatur atau memberikan izin untuk terjadinya hubungan, komunikasi, dan perpindahan data antara dua atau lebih titik komputer. Protokol ini disebut ANPP (*Advanced Navigation Packet Protocol*) dengan kecepatan komunikasi berkisar 1200 *byte* hingga 10 *Megabyte*. Pada Gambar 17,

jalur data Sublocus dihubungkan pada jalur data modul RS 422 to TTL. Sensor ini dapat dijalankan dengan memberikan sumber tegangan sebesar 18 Volt hingga 50 Volt.

Modul RS 422 to TTL merupakan standar komunikasi serial, dimana sistem transmisi data tersebut menggunakan sinyal diferensial. Sinyal tersebut berupa pulsa *clock* yang mengomunikasikan informasi digital dari *transmitter* ke *receiver*. Jumlah pulsa per detik dalam standar kominikasi ini, menentukan banyaknya data yang dapat dikirim dalam periode tertentu. Sinyal diferensial RS 422 dapat dilihat pada Gambar 3.9.



Gambar 3.9. Sinyal Diferensial RS 422

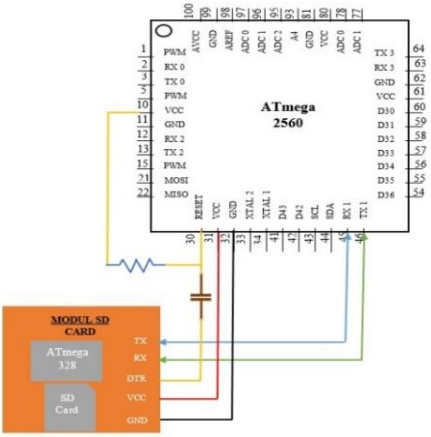
Gambar 3.9 merupakan bentuk dari kondisi sinyal diferensial RS 422 saat bekerja. Dalam hal ini, sinyal RS 422 didefinisikan dua garis sinyal diferensial sebagai garis A dan B. Keadaan sinyal, yaitu 0 atau 1 diartikan sebagai perbedaan tegangan antara dua garis. Ketika garis A lebih besar dari garis B maka bernilai 0 Volt. Sebaliknya, ketika garis B lebih besar dari garis A maka bernilai 5 Volt. RS 422 digunakan untuk aplikasi *multi drop* dimana hanya satu pemancar yang terhubung, dan dapat mengirimkan pada bus hingga 10 *receiver*. Selain itu, juga diperuntukan pada aplikasi berbasis *point to point* atau hanya satu pemancar dan satu penerima dan sebaliknya, seperti pada penelitian ini. Penelitian akuisisi dan *logging* data INS ini menggunakan sensor Sublocus yang membutuhkan 5 jalur data, sedangkan untuk transmisi ke Arduino membutuhkan 3 jalur data, yaitu TX, RX, dan GND. Oleh karena itu digunakan modul RS 422 to TTL untuk memudahkan konfigurasi *input* dan *output* antara Sublocus dan Arduino. Kecepatan pengiriman data komunikasi serial modul ini sebesar 10 Megabit per detik. Modul ini menentukan protokol, konektor maupun tugas pin dari fungsi.

Arduino Mega (ATmega 2560) dalam penelitian ini berperan sebagai pengendali dan pemroses data yang telah ditransmisikan dari RS 422. Dalam menjalankan tugasnya, perangkat ini mengombinasikan antara perangkat keras, bahasa pemrograman dan IDE (*Integrated Development Environment*). IDE merupakan perangkat lunak yang digunakan untuk menulis program, mengompilasi menjadi kode biner

dan mengunggah ke dalam memori mikrokontroler Arduino. Dengan ini, Sublocus membutuhkan mikrokontroler Arduino sebagai pengendali dan pemrosesan data melalui standar komunikasi serial RS 422, seperti pada Gambar 17. Dapat diamati bahwa pin RS 422 tersambung pada pin Arduino, yaitu pin TX, RX, dan GND. Jalur inilah yang membentuk suatu komunikasi dari Sublocus menuju mikrokontroler Arduino untuk proses akuisisi data.

3.5.2. Konfigurasi *Logging* Data

Hasil akuisisi atau pengambilan data dari sublocus, kemudian dianalisis lebih lanjut secara *offline*. Dengan demikian, dibutuhkan proses *logging* data agar data yang telah diambil dapat dicatatan dan disimpan pada sebuah perangkat penyimpanan. Proses tersebut memanfaatkan perangkat tambahan berupa SD Card yang memiliki kapasitas penyimpanan sebesar 16 GB. Perancangan *logging* data dapat dilihat pada Gambar 3.0.

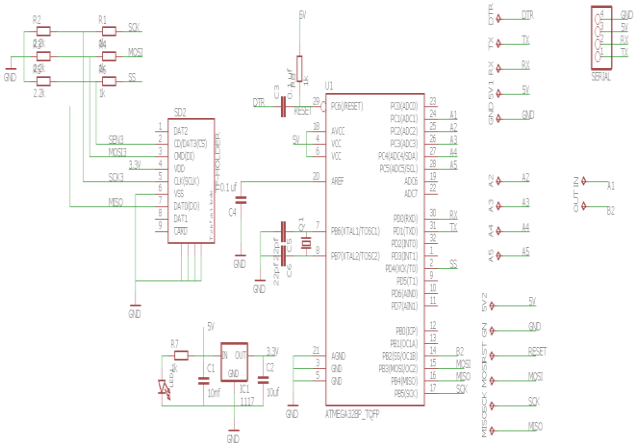


Gambar 3.10. Konfigurasi Sistem *Logging* Data

Pada konfigurasi yang terdapat pada Gambar 3.10, tampak Modul SD Card dihubungkan ke Arduino Mega (ATmega 2560). Hal tersebut merupakan perancangan *logging* data menggunakan media Modul SD Card, dimana pada modul tersebut terdiri dari ATmega 328 dan SD Card. Sistem data *logger* ini dibangun dari modul Arduino sebagai pengendali

disistem dan menggunakan SD Card sebagai media penyimpanan data. Saat proses *logging* data dijalankan maka data dari Sublocus yang telah diakuisisi oleh Arduino dikirimkan melalui komuikasi serial (RX TX), dan diterima oleh modul SD Card. Dapat diamati konfigurasi pada Gambar 20, jalur data Arduino, yaitu pin RX dan TX dihubungkan pada TX dan RX modul SD Card, sedangkan pin GND dihubungkan ke sesama pin GND dari modul SD Card. Lain halnya pada pin RESETarduino dihubungkan pada pin DTR dari modul SD Card, dimana diantara jalur tesebut terdapat dua komponen tambahan, yaitu Kapasitor dan Resistor. Kombinasi dua komponen ini berfungsi untuk menghindari lonjakan berlebih. Hubungan antar komponen dan jalur data tersebut berguna untuk *reset*, dengan kata lain untuk mengulang kembali proses atau data yang disimpan pada SD Card melalui Arduino.

Dalam penelitian ini dirancang pembuatan modul SD Card sehingga dapat membantu proses *logging* data. Skematik modul SD Card dapat dilihat pada Gambar 3.11.



Gambar 3.11. Skematik Modul SD Card

Terdapat tahap dalam pembuatan modul tersebut yang meliputi pembuatan skematik, penyablonan PCB, penyolderan komponen, dan *wiring*. Kabel ini berguna untuk proses transmisi atau sebagai penghubung antar komponen. Hasil dari tahapan tersebut dapat dilihat pada Gambar 3.12.

Gambar 3.12. Hasil Pemasangan Kabel pada Modul SD Card

Modul SD Card terdiri dari empat kabel untuk menghubungkan pin pada perangkat lainnya. Jalur data tersebut adalah TX, RX, GND dan DTR.

3.4. Perancangan Perangkat Lunak

Dalam penelitian ini, perangkat lunak digunakan untuk memproses data dengan cara memprogram atau dikenal dengan istilah *coding*. Hal ini bertujuan agar sistem akuisisi dan *logging* data dapat dikomunikasikan dan diolah menggunakan berbagai perintah. Perangkat lunak yang digunakan adalah Arduino IDE 1.6.7. Komunikasi *Full Duplex* antar *device* dan *flowchart* program dijelaskan pada sub bab ini. *Full Duplex* umumnya membutuhkan dua jalur komunikasi. Perancangan lunak ini memiliki dua sub sistem, yaitu sistem akuisisi data dan sistem *logging* data. Pada sistem akuisisi, komputer mengirimkan perintah yang telah diprogram oleh Arduino IDE ke mikrokontroler Arduino. Setelah itu, mikrokontroler tersebut mengirimkan perintah atau isyarat kepada Sublocus untuk melakukan proses pengambilan data. Sublocus mentransmisikan data dalam format ANPP melalui modul RS 422. Modul tersebut mengoversikan data *byte* dari Sublocus menjadi data serial. Data yang telah diproses dikirimkan kembali menuju Arduino. Kemudian data ditampilkan oleh serial monitor, sedangkan pada sistem *logging* data disimpan pada modul SD Card. Data-data ini meliputi perhitungan *Latitude*, *Longitude*, *Velocity*, *Roll*, *Pitch*, *Yaw*, dan *Acceleration* yang diolah dalam format ANPP. Format paket protokol tersebut dapat dilihat pada Tabel 3.1.

Tabel 3.1. Format ANPP

Header				
Header LRC	*Packet* ID	*Packet Length*	CRC16	Paket Data

Format pada Tabel 3.1 meliputi *Header* LRC, *Packet* ID, *Packet Lenght*, CRC, dan Paket Data. *Header* LRC (*Longitudinal Redundancy Check*)

adalah decoder untuk menemukan awal paket dengan memindai LRC yang valid dengan persamaan nomor (1) :

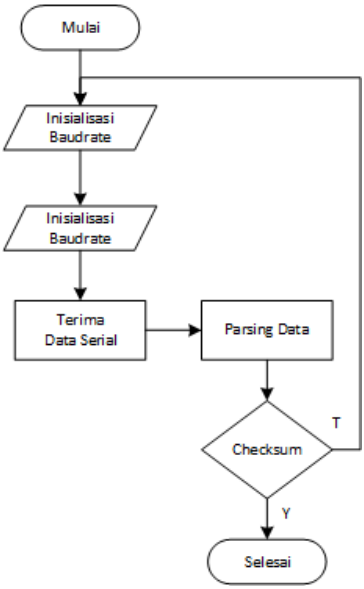
LRC=((packet_id+packet_length+crc[0]+crc[1])^0xFF)+1 (1)

LRC menyediakan pengecekan eror pada *header* paket. Paket ID digunakan untuk membedakan isi paket. ID paket berkisar dari 0 hingga 255. Dalam hal ini terdapat tiga sub paket yang berbeda antaralain *system packet* memiliki ID paket dalam rentang 0 hingga 19 *byte*. *State packet* merupakan paket yang berisi data yang berubah seiring waktu, misalnya suhu. Paket ini dapat diatur ke *output* pada tingkat tertentu. ID *state packet* dalam kisaran 20 hingga 179. Terakhir, paket konfigurasi yang digunakan untuk membaca dan menulis konfigurasi perangkat. Paket konfigurasi adalah ID paket dalam kisaran 180 hingga 255. Kemudian dalam format *header* juga terdapat *Packet Lenght* yang menunjukkan panjang data paket, yaitu dari indeks *byte* 5 dan seterusnya. *Packet Lenght* memiliki kisaran 0-255 *byte*. Berikutnya format CRC atau CRC16-CCITT. Format ini memiliki nilai awal 0xFFFF dan hanya mencakup data paket. Paket Data terdiri dari jenis-jenis subjek yang terdapat pada lampiran A-7. Tabel 7 merupakan kumpulan subjek data dari ANPP, dimana dalam protokol ini menyediakan kode-kode yang menyatakan suatu jenis data. Pada kolom tipe data terdapat kode seperti u8, fp32 dan sebagainya. Tipe data ini digunakan untuk menentukan jenis nilai yang dapat ditampung oleh suatu variabel. Nilai tersebut berupa rentang ukuran seperti halnya u8 memiliki *range* 0 hingga 255. Lain halnya dengan u16 memiliki *range* 0 hingga 65535 dan u32 sebesar 0 sampai 4294967295. Kode tersebut telah ditetapkan oleh pabrik dan dijadikan suatu petunjuk dalam mengakses komunikasi Sublocus menggunakan protokol.

3.6.1. Perancangan Perangkat Lunak Akuisisi Data

Pada proses akuisisi data, dihubungkan perangkat Sublocus ke mikrokontroler Arduino yang ditampilkan pada serial monitor. Dalam menjalankan komunikasi dari perangkat-perangkat tersebut, diperlukan suatu program untuk memberi perintah pada Arduino. Program ini yang dapat menjalankan proses akuisisi data sesuai keinginan pengguna. Dengan ini, dirancang beberapa *flowchart* untuk membantu dalam pembuatan program pada Arduino IDE. Perancangan *flowchart* meliputi

sistem program keseluruhan, *parsing* data, dan proses pemanggilan paket protokol. *Flowchart* program keseluruhan proses akuisisi dapat dilihat pada Gambar 3.13. Kecepatan *baud rate* yang ditetapkan pada program sebesar 115200. Setelah itu dilakukan inisialisasi atau penamaan objek yang diproses, yaitu Sublocus. Ketika data telah diterima secara serial maka data-data tersebut melaksanakan *parsing* menurut format *header* yang telah ditetapkan.



Gambar 3.13. *Flowchart* Sistem Program Keseluruhan

Hal ini bertujuan untuk mengolah suatu paket data masukkan dengan cara memisahkan paket data tersebut ke dalam uraian-uraian data yang diolah ke tahap selanjutnya. Ketika tahap itu telah terlaksana, langkah berikutnya, yaitu pengiriman data.

Penerapan pada kode Arduino dalam inisialisasi *baud rate* adalah sebagai berikut :

```
...
  Serial.begin(115200);
  Serial2.begin(9600);
  Serial3.begin(115200);
...
```

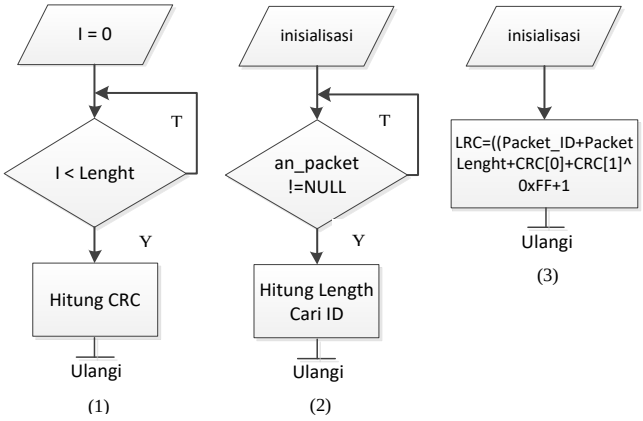
Terdapat tiga kode inisialisasi *baud rate*, yang pertama adalah jalur pembuka komunkasi dalam PC. *Baud rate* kedua merupakan jalur komunikasi perangkat Log, sedangkan ketika komunikasi ke dalam Sublocus. Berikutnya adalah inisialisasi objek yang ditunjukkan oleh kode sebagai berikut :

```
…
  system_state_packet_t system_state_packet;
  raw_sensors_packet_t raw_sensors_packet;
  unix_time_packet_t unix_time_packet;
  satellites_packet_t satellites_packet;
  external_velocity_packet_t external_velocity;
  an_decoder_initialise(&an_decoder);
...
```

Kode diatas menunjukkan bahwa perancangan dilakukan untuk memperoleh paket data dari *state packet*, *raw sensor*, *unix time packet*, *sattelites packet*, dan juga *external velocity packet*. Perintah untuk menerima data serial ditunjukkan oleh kode dibawah ini :

```
…
while((bytes_received=Serial3.readBytes(an_decoder_point
er(&an_decoder), an_decoder_size(&an_decoder))) > 0)
…
```

Arti dari kode di atas adalah setiap data yang ada dimasukkan ke dalam *pointer decoder* untuk dilakukan *parse* data. Pada Gambar 3.14 ditunjukan runtutan *flowchart parsing* data.



Gambar 3.14. *Flowchart Parsing* Data

Gambar 3.14 menunjukkan tiga diagram *flowchart*, dimana dalam proses *parsing* data dikelompokkan menjadi tiga bagian menurut jenis format *header* yang telah dijelaskan pada Tabel 1. Pada *flowchart* 1 *Lenght* yang menunjukkan panjang data paket kisaran 0-255, yaitu dari indeks *byte* 5 dan seterusnya. *Flowchart* 2 ID paket yang digunakan untuk membedakan isi paket, sedangkan nomor 3 adalah LRC bertugas sebagai pengecekan eror pada *header* paket. Runtutan pemanggilan paket-peket yang diolah dapat diamati pada lampiran A-8. *Flowchart* tersebut merupakan proses pemanggilan paket menggunakan fungsi `if`. Pertama, ketika data telah diterima dari Sublocus, data tersebut melakukan proses *parsing* menurut format *header*. Kemudian data-data di deteksi sesuai dengan kode ID dan *Lenght*. Kode-kode ini menentukan salah satu dari jenis-jenis subjek dalam paket antara *System State*, *Unix Time*, *Raw Sensor*, *Sattelit*, atau *External Velocity*. Implementasi dari *flowchart* A-8 pada lampiran ditunjukkan oleh kode sebagai berikut :

```
...
if (an_packet->id == packet_id_system_state) /* system state
packet */
        {
if(decode_system_state_packet(&system_state_packet,
an_packet) == 0)
          {
            DataKirim[0]                                        =
system_state_packet.orientation[0]*RADIANS_TO_DEGREES;
            DataKirim[1]                                        =
system_state_packet.orientation[1]*RADIANS_TO_DEGREES;
            DataKirim[2]                                        =
system_state_packet.orientation[2]*RADIANS_TO_DEGREES;
            DataKirim[3] = system_state_packet.latitude;
            DataKirim[4] = system_state_packet.longitude;
          }
        }
//--- ANPacket 21 Unix Time Packet
       if(an_packet->id == packet_id_unix_time)
        {
           if(decode_unix_time_packet(&unix_time_packet,
an_packet) == 0)
            {
              DataKirim[5] = unix_time_packet.microseconds;
          }
```

Setiap paket data yang masuk dilakukan *parse* sesuai dengan ID yang didapatkan. Data yang telah *parse* kemudian dimasukkan ke dalam

variable untuk dikirimkan pada perangkat penyimpanan. Kode selengkapnya dapat dilihat pada lampiran A-9.

3.6.2. Perancangan Perangkat Lunak *Logging* Data

Dalam proses *logging* data, dikonfigurasikan antar perangkat Arduino dan modul SD Card. Data yang telah diakuisisi oleh mikrokontroler disimpan atau dicatat pada modul SD Card. Pada Tabel 3.2 merupakan keterangan data yang disimpan dalam media penyimpanan.

Tabel 3.2. *Address* Protokol

Address	***Function***	***Unit***
0	*Roll*	radians
1	*Pitch*	radians
2	*Heading*	radians
3	*Latitude*	rad
4	*Longitude*	rad
5	*Microseconds*	-
6	*Gyroscope* X	rad/s
7	*Gyroscope* Y	rad/s
8	*Gyroscope* Z	rad/s
9	*Accelerometer* X	m/s/s
10	*Accelerometer* Y	m/s/s
11	*Accelerometer* Z	m/s/s
12	*Magnetometer* X	mG
13	*Magnetometer* Y	mG
14	*Magnetometer* Z	mG
15	GPS *Satellites*	-
16	*Velocity north*	m/s
17	*Velocity east*	m/s
18	*Velocity down*	m/s

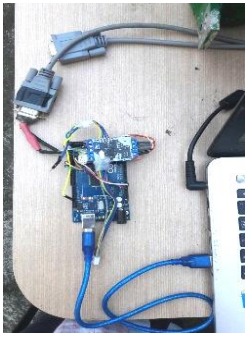
Data-data dalam Tabel 3.2 disimpan dalam SD Card dan dicatat atau direkam oleh perangkat lunak *Notepad*. Dengan ini data dapat diproses maupun dianalisis dalam keadaan *offline*.

BAB IV
PENGUJIAN DAN ANALISIS DATA

Dalam penelitian ini, pengujian sistem akuisisi data dan sistem *logging* data dilakukan untuk mengetahui performa sistem melalui data yang telah didapat. Bab ini merupakan realisasi dari perancangan sistem yang telah dibahas pada Bab 3. Dimana realisasi yang dilakukan adalah *wiring* antara *input* dan *output* perangkat. Tujuan dari *wiring* tersebut untuk proses komunikasi antara satu perangkat ke perangkat lain. Penelitian ini memanfaatkan jenis komunikasi *Full Duplex* dan *Half Duplex* yang terdiri dari hubungan antar pin RX dan TX.

4.1. Cara Kerja Sistem

Langkah pertama dilakukan dengan menghubungkan Sublocus sebagai objek data ke RS 422, Arduino, modul SD Card, PC dan sumber tegangan yaitu baterai. Proses *wiring* antar pin tersebut merupakan realisasi dari Gambar 16 yaitu diagram blok dalam Bab 3. Jalur data RX, TX, GND, dan VCC dari RS 422 dan Modul SD Card dihubungkan dengan jalur data yang terdapat pada Arduino. Setelah pin dari tiga komponen dihubungkan menggunakan kabel jumper, langkah berikutnya menambahkan pin DB 9 *male* pada jalur data RS 422. Pin DB 9 jenis *male* berguna untuk menghubungkan antara RS 422 dengan pin yang terdapat pada Sublocus. Jalur data yang dimiliki Sublocus adalah DB 9 jenis *female*. Hubungan antar perangkat tersebut dapat dilihat pada Gambar 4.1.



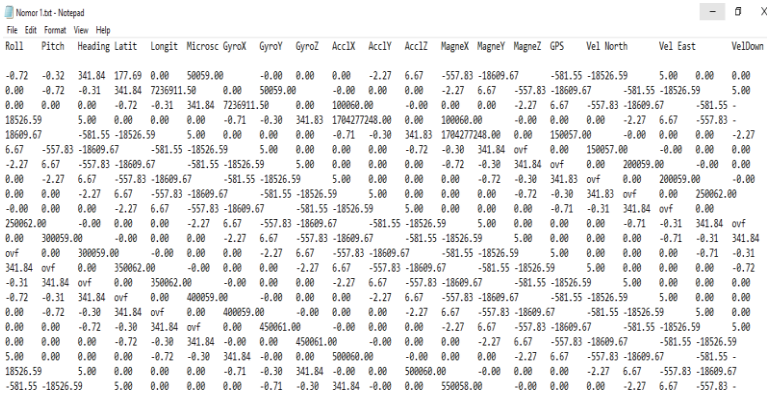
Gambar 4.1. Pemasangan Jalur Data DB 9 ke Sublocus

Hubungan antar perangkat pada Gambar 4.1. Dimana Sublocus dihubungkan pada Arduino menggunakan fasilitas jalur data dari konveter RS 422. Setelah itu pin USB Arduino dihubungkan pada PC, seperti pada Gambar 4.2.

Gambar 4.2. Ilustrasi *Wiring* antar Perangkat

Ketika sensor dihubungkan ke sumber tegangan, proses akuisisi dan *logging* data terjadi. Data didapat dari perhitungan sensor Sublocus yang kemudian dikonversi oleh RS 422. Setelah dikonversi, data disalurkan pada mikrokontroler Arduino untuk melakukan proses *parse*. Berbagai macam data ini kemudian direkam dan disimpan oleh modul SD Card. Selain itu, dipersiapkan beberapa perangkat lunak, seperti aplikasi Arduino IDE, CodeVision AVR, *Notepad*, dan Visual Studio sebagai penunjang proses pengujian. Pada PC diunggah program perintah dari Arduino IDE ke *board* Arduino. Cara menggunggah program perintah pada Arduino adalah dengan mengklik tombol "*Upload*". Proses unggah berjalan selama beberapa detik dan bila tidak terdapat kesalahan pada kode program, maka Arduino IDE memunculkan notifikasi "*Done uploading*". Hal ini menandakan bahwa program perintah berhasil diunggah ke *board* Arduino. Bila proses pengunggahan program perintah pada Arduino telah selesai, berikutnya menjalankan proses perekaman data dalam *Notepad* menggunakan bantuan perangkat lunak CodeVision AVR. Metode ini dapat dilakukan dengan cara membuka jendela RX *file* yang terdapat pada layar CodeVision AVR, kemudian dibuatlah folder untuk meletakan atau menyimpan data dalam *Notepad*. Klik tombol RX *start*, dan diakhiri dengan klik tombol RX *stop*. Dengan cara ini, data tersimpan secara otomatis pada *Notepad*. Hal ini berguna untuk proses

analisis lebih lanjut serta untuk memperoleh informasi yang dibutuhkan. Pada aplikasi CodeVision AVR atau Arduino IDE data yang ditampilkan tidak dapat disalin atau disimpan secara *offline* sehingga membutuhkan bantuan perangkat lunak *Notepad*.


Nomor1.txt - Notepad

File Edit Format View Help

Roll	Pitch	Heading	Latit	Longit	Microsc	GyroX	GyroY	GyroZ	AcclX	AcclY	AcclZ	MagneX	MagneY	MagneZ	GPS	Vel North	Vel East	VelDown
-0.72	-0.32	341.84	177.69	0.00	50059.00	-0.00	0.00	0.00	-2.27	6.67	-557.83	-18609.67	-581.55	-18526.59	5.00	0.00	0.00	
0.00	-0.72	-0.31	341.84	7236911.50	0.00	50059.00	-0.00	0.00	0.00	-2.27	6.67	-557.83	-18609.67	-581.55	-18526.59	5.00		
0.00	0.00	0.00	-0.72	-0.31	341.84	7236911.50	0.00	100060.00	-0.00	0.00	0.00	-2.27	6.67	-557.83	-18609.67	-581.55	-	
18526.59	5.00	0.00	0.00	0.00	-0.71	-0.30	341.83	1704277248.00	0.00	100060.00	-0.00	0.00	0.00	-2.27	6.67	-557.83	-	
18609.67	-581.55	-18526.59	5.00	0.00	0.00	0.00	-0.71	-0.30	341.83	1704277248.00	0.00	150057.00	-0.00	0.00	0.00	-2.27		
6.67	-557.83	-18609.67	-581.55	-18526.59	5.00	0.00	0.00	0.00	-0.72	-0.30	341.84	ovf	0.00	150057.00	-0.00	0.00	0.00	
-2.27	6.67	-557.83	-18609.67	-581.55	-18526.59	5.00	0.00	0.00	0.00	-0.72	-0.30	341.84	ovf	0.00	200059.00	-0.00	0.00	
0.00	-2.27	6.67	-557.83	-18609.67	-581.55	-18526.59	5.00	0.00	0.00	0.00	-0.72	-0.30	341.83	ovf	0.00	200059.00	-0.00	
0.00	0.00	-2.27	6.67	-557.83	-18609.67	-581.55	-18526.59	5.00	0.00	0.00	0.00	-0.72	-0.30	341.83	ovf	0.00	250062.00	
-0.00	0.00	0.00	-2.27	6.67	-557.83	-18609.67	-581.55	-18526.59	5.00	0.00	0.00	0.00	-0.71	-0.31	341.84	ovf	0.00	
250062.00	-0.00	0.00	0.00	-2.27	6.67	-557.83	-18609.67	-581.55	-18526.59	5.00	0.00	0.00	0.00	-0.71	-0.31	341.84	ovf	
0.00	300059.00	-0.00	0.00	0.00	-2.27	6.67	-557.83	-18609.67	-581.55	-18526.59	5.00	0.00	0.00	0.00	-0.71	-0.31	341.84	
ovf	0.00	300059.00	-0.00	0.00	0.00	-2.27	6.67	-557.83	-18609.67	-581.55	-18526.59	5.00	0.00	0.00	0.00	-0.71	-0.31	
341.84	ovf	0.00	350062.00	-0.00	0.00	0.00	-2.27	6.67	-557.83	-18609.67	-581.55	-18526.59	5.00	0.00	0.00	0.00	-0.72	
-0.31	341.84	ovf	0.00	350062.00	-0.00	0.00	0.00	-2.27	6.67	-557.83	-18609.67	-581.55	-18526.59	5.00	0.00	0.00	0.00	
-0.72	-0.31	341.84	ovf	0.00	400059.00	-0.00	0.00	0.00	-2.27	6.67	-557.83	-18609.67	-581.55	-18526.59	5.00	0.00	0.00	
0.00	-0.72	-0.30	341.84	ovf	0.00	400059.00	-0.00	0.00	0.00	-2.27	6.67	-557.83	-18609.67	-581.55	-18526.59	5.00	0.00	
0.00	0.00	-0.72	-0.30	341.84	ovf	0.00	450061.00	-0.00	0.00	0.00	-2.27	6.67	-557.83	-18609.67	-581.55	-18526.59	5.00	
0.00	0.00	0.00	-0.72	-0.30	341.84	-0.00	0.00	450061.00	-0.00	0.00	0.00	-2.27	6.67	-557.83	-18609.67	-581.55	-18526.59	
5.00	0.00	0.00	0.00	-0.72	-0.30	341.84	-0.00	0.00	500060.00	-0.00	0.00	0.00	-2.27	6.67	-557.83	-18609.67	-581.55	-
18526.59	5.00	0.00	0.00	0.00	-0.71	-0.30	341.84	-0.00	0.00	500060.00	-0.00	0.00	0.00	-2.27	6.67	-557.83	-18609.67	
-581.55	-18526.59	5.00	0.00	0.00	0.00	-0.71	-0.30	341.84	-0.00	0.00	550058.00	-0.00	0.00	0.00	-2.27	6.67	-557.83	-

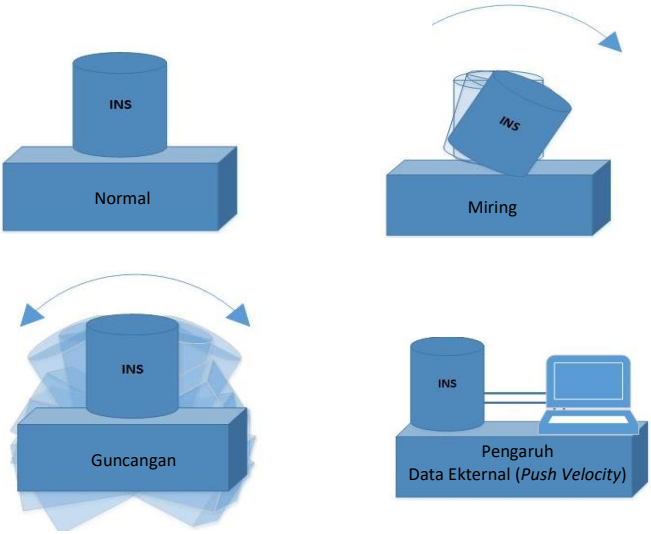
Gambar 4.3. Hasil Perekaman Data (*Data Logging*) dari Sublocus pada *Notepad*

Gambar 4.3 menunjukan data yang telah direkam dan kemudian ditampung pada aplikasi *Notepad*.

4.2. Hasil Akuisisi Data

Pengumpulan data atau akuisisi dilakukan untuk memperoleh informasi yang dibutuhkan dalam rangka mencapai tujuan penelitian. Dari data yang telah diambil, dapat diketahui bentuk respon keluaran dari perangkat navigasi Sublocus. Pengujian dilakukan dengan memberikan empat kondisi atau keadaan Sublocus yang berbeda. Kondisi-kondisi ini yaitu ketika Sublocus dalam keadaan normal, keadaan posisi miring, ketika keadaan gangguan atau guncangan, dan keadaan saat diberikan *push internal Dummy Data*. Data-data yang diambil adalah *Roll* (*radians*), *Pitch* (*radians*), *Heading* (*radians*), *Latitude* (*rad*), *Longitude* (*rad*), *Microsecond* (*ms*), *Gyroscope* X (*rad/s*), *Gyroscope* Y (*rad/s*), *Gyroscope* Z (*rad/s*), *Acceleration* X (m/s/s), *Acceleration* Y (m/s/s), *Acceleration* Z (m/s/s), *Magnetometer* X (mG), *Magnetometer* Y (mG), *Magnetometer* Z (mG), *Velocity North* (m/s), *Velocity East* (m/s), dan *Velocity Down* (m/s). Pengujian ini dilakukan dengan mengamati respon 19 macam data yang

telah didapat sesuai dengan pengaruh kondisi Sublocus. Berikut ini Gambar 4.4 yang merupakan ilustrasi pengujian 4 kondisi Sublocus.

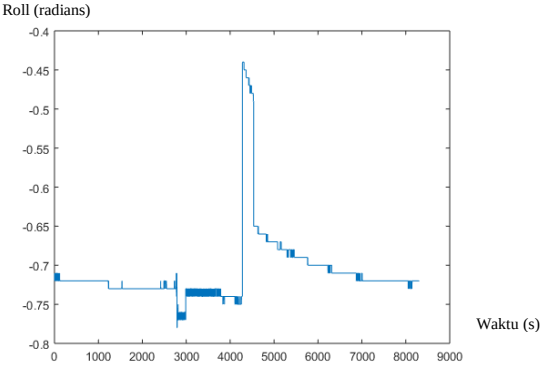


Gambar 4.4. Ilustrasi Pengujian Sistem Akuisisi dan *Logging* Data

Pengujian yang pertama yaitu ketika posisi Sublocus normal atau tidak diberikan pengaruh apapun. Sensor ini diletakan pada permukaan datar dalam keadaan diam. Kondisi Sublocus normal dapat diamati pada Gambar 4.5.

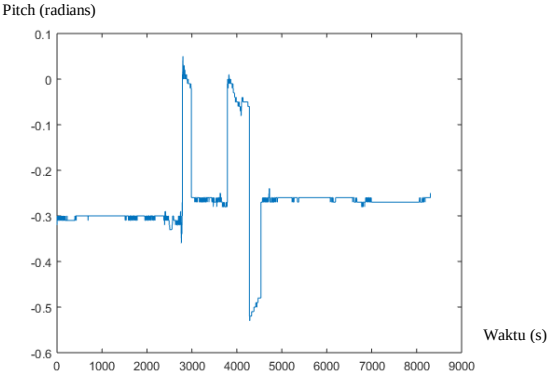
Gambar 4.5. Sublocus Kondisi Normal

Dari pengujian ini, didapatkan 19 macam data berbeda yang ditampilkan dalam bentuk grafik dari waktu ke waktu. Pada garis vertikal merupakan perhitungan hasil data yang ditetapkan, sedangkan garis horizontal menunjukkan perhitungan terhadap waktu. Masing-masing data memiliki pergerakan atau perubahan data yang berbeda-beda sesuai dengan perannya.



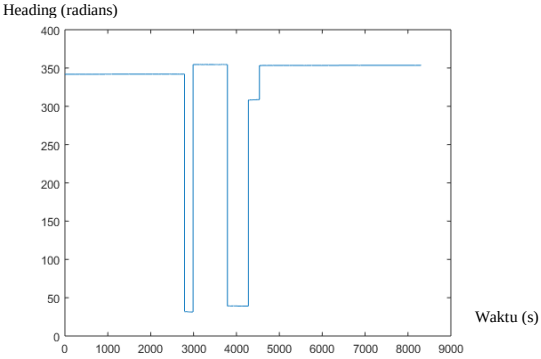
Gambar 4.6. Data Roll Normal

Nilai rata-rata pada data *Roll* pada Gambar 4.6 sebesar -0.70896 *radians*, sedangkan nilai rata-rata eror yang dimiliki adalah 0.011039 *radians*. Pada grafik dapat diamati nilai awal dan nilai akhir sebesar -0.72 *radians*. Selain itu, nilai yang sering muncul pada data *Roll* -0.72 *radians*. Data-data tersebut memiliki keragaman atau standar deviasi 0.049.



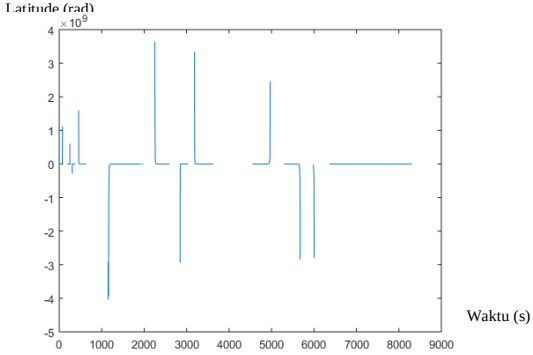
Gambar 4.7. Data Pitch Normal

Pada Gambar 4.7 data *Pitch*, nilai rata-rata sebesar -0.26533 *radians*, sedangkan nilai rata-rata eror yang dimiliki adalah 0.054674 *radians*. Pada grafik dapat diamati nilai awal yang muncul sebesar -0.32 *radians* dan nilai akhir -0.25 *radians*. Selain itu, nilai yang sering muncul pada data *Pitch* adalah -0.26 *radians*. Data-data tersebut memiliki keragaman atau standar deviasi 0.083.



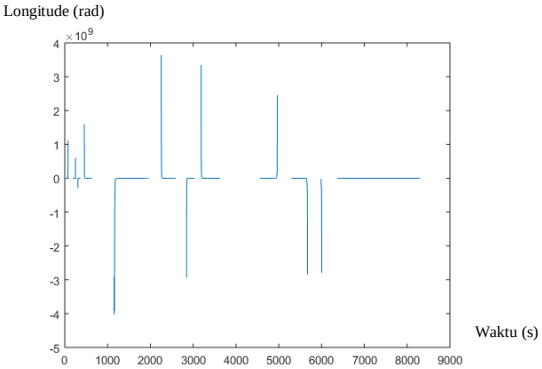
Gambar 4.8. Data Heading Normal

Data *Heading* dalam Gambar 4.8 menunjukkan rata-rata sebesar 321.9438 *radians*, sedangkan nilai standar deviasi yang dimiliki adalah 86.31. Pada grafik dapat diamati nilai awal yang muncul sebesar 341.84 *radians* dan nilai akhir 535.47 *radians*. Selain itu, nilai yang sering muncul pada data *Heading* adalah 353.44 *radians*. Respon grafik menunjukkan bahwa data tersebut mengalami fluktuasi.



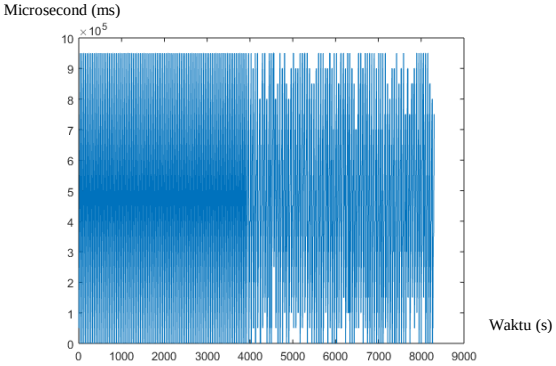
Gambar 4.9. Data Latitude Normal

Nilai rata-rata *Latitude* pada Gambar 4.9 sebesar 2759430.397 rad, sedangkan nilai standar deviasi yang dimiliki adalah 314399310. Pada grafik dapat diamati nilai awal yang muncul sebesar 177.69 rad dan nilai akhir 0 rad, sedangkan nilai terkecil adalah -4122917376 rad. Pergerakan nilai pada grafik menunjukkan bahwa data tersebut bersifat heterogen atau bervariasi.



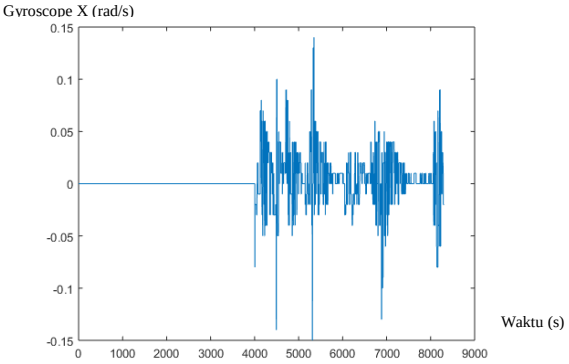
Gambar 4.10. Data Longitude Normal

Gambar 4.10 yaitu data *Latitude* memiliki rata-rata sebesar 314399310 rad, sedangkan nilai standar deviasi yang dimiliki adalah 252067975. Pada grafik dapat diamati nilai awal yang muncul sebesar 0 rad dan nilai akhir 0 rad, sedangkan nilai terkecil adalah -4033486080 rad. Pergerakan nilai dari waktu ke waktu pada grafik menunjukkan bahwa data tersebut bersifat heterogen atau bervariasi.



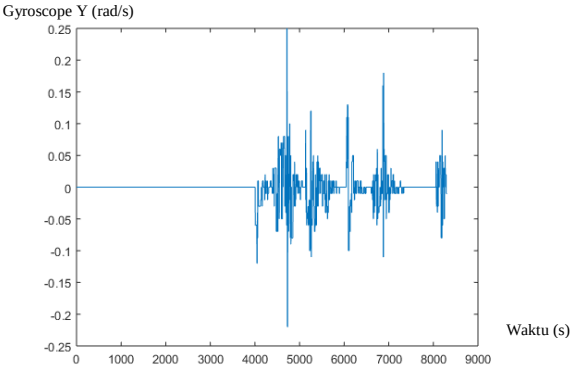
Gambar 4.11. Data Microseconds Normal

Nilai rata-rata pada Gambar 4.11 data Microsecond sebesar 470414.313, sedangkan nilai standar deviasi yang dimiliki adalah 288201. Pada grafik dapat diamati nilai awal yang muncul sebesar 50059 dan nilai akhir 750057, sedangkan nilai terkecil adalah 69.



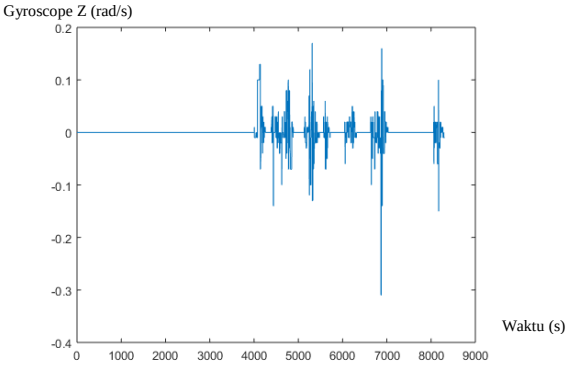
Gambar 4.12. Data Gyroscope X Normal

Gambar 4.12 menunjjukan data *Gyroscope* X dengan nilai rata-rata sebesar 0.002381 rad/s, sedangkan nilai standar deviasi adalah 0.02. Pada grafik dapat diamati nilai awal yang muncul sebesar 0 rad/s dan nilai akhir -0.02 rad/s, sedangkan nilai terkecil adalah -0.15 rad/s dan nilai terbesar 0.14 rad/s. Pergerakan nilai dari waktu ke waktu pada grafik menunjukkan bahwa data tersebut mengalami zona stabil hingga detik ke 3628 dan detik berikutnya mengalami fluktuasi.



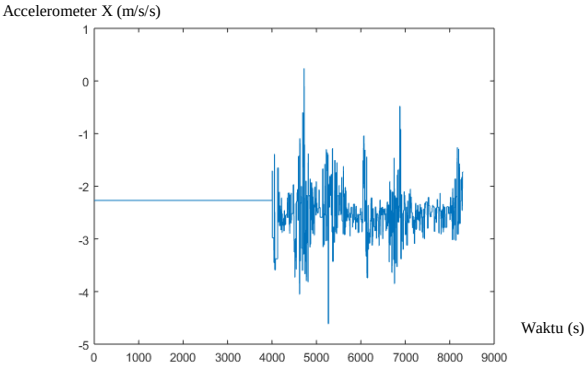
Gambar 4.13. Data Gyroscope Y Normal

Nilai rata-rata pada data *Gyroscope* Y pada Gambar 4.13 sebesar -0.0014 rad/s, sedangkan nilai standar deviasi yang dimiliki adalah 0.026. Pada grafik dapat diamati nilai awal yang muncul sebesar 0 rad/s dan nilai akhir -0.01 rad/s, sedangkan nilai terkecil adalah -0.22 rad/s dan nilai terbesar 0.25 rad/s. Dari kumpulan data ini di temukan rata-rata nilai eror sebesar 0.0014 rad/s.



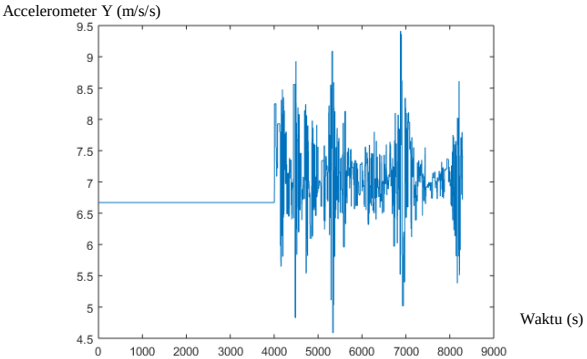
Gambar 4.14. Data Gyroscope Z Normal

Pada Gambar 4.14 nilai rata-rata data *Gyroscope* Z sebesar 9.27041E-05 rad/s, sedangkan nilai standar deviasi yang dimiliki adalah 0.023. Pada grafik dapat diamati nilai awal yang muncul sebesar 0 rad/s dan nilai akhir -0.01 rad/s, sedangkan nilai terkecil adalah -0.31 rad/s dan nilai terbesar 0.17 rad/s.



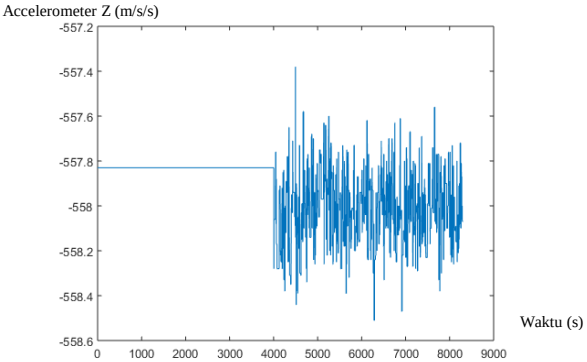
Gambar 4.15. Data Accelerometer X Normal

Data *Accelerometer* X pada Gambar 4.15 memiliki rata-rata sebesar -2.39807 m/s/s, sedangkan nilai standar deviasi yang dimiliki adalah 0.372. Nilai awal yang muncul sebesar -2.27 m/s/s dan nilai akhir -1.73 m/s/s, sedangkan nilai terkecil adalah -4.61 m/s/s dan nilai terbesar 0.24 m/s/s. Rata-rata nilai eror sebesar 0.1281 m/s/s.



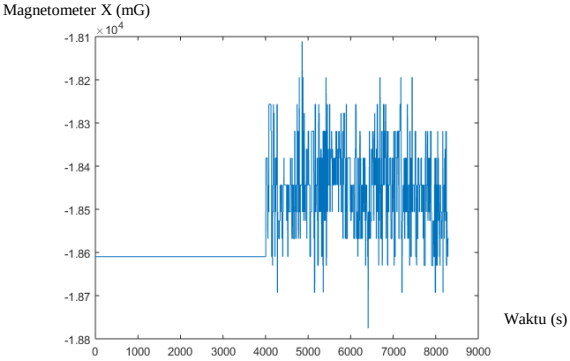
Gambar 4.16. Data Accelerometer Y Normal

Nilai rata-rata pada data *Accelerometer* Y pada Gambar 4.16 sebesar 6.87169 m/s/s, sedangkan nilai standar deviasi yang dimiliki adalah 0.499. Pada grafik dapat diamati nilai awal yang muncul sebesar 6.67 m/s/s dan nilai akhir 7.4 m/s/s, sedangkan nilai terkecil adalah 4.59 m/s/s dan nilai terbesar 9.41 m/s/s. Dari kumpulan data ini di temukan rata-rata nilai eror sebesar 0.2 m/s/s.



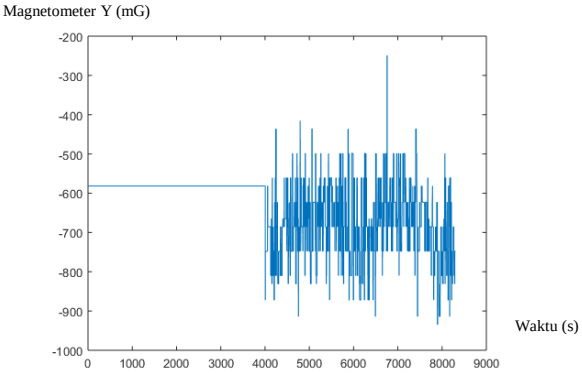
Gambar 4.17. Data Accelerometer Z Normal

Gambar 4.17 menunjukkan data *Accelerometer* Z dengan nilai rata-rata sebesar -557.926 m/s/s, sedangkan nilai standar deviasi adalah 0.153 m/s/s. Pada grafik dapat diamati nilai awal yang muncul sebesar -557.83 m/s/s dan nilai akhir -558.07 m/s/s, sedangkan nilai terkecil adalah -558.51 dan nilai terbesar -557.38 m/s/s.



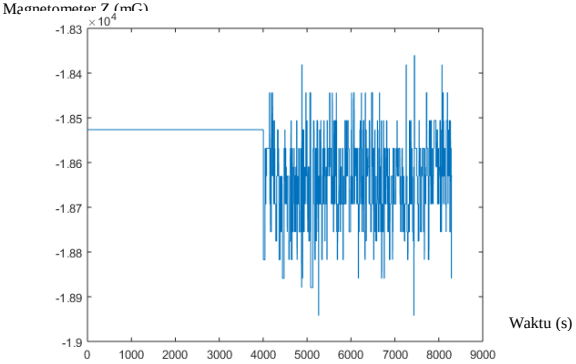
Gambar 4.18. Data Magnetometer X Normal

Nilai rata-rata pada Gambar 4.18 yaitu data *Magnetometer* X sebesar -18527.6 mG, sedangkan nilai standar deviasi yang dimiliki adalah 106.8. Pada grafik dapat diamati nilai awal yang muncul sebesar -18609.7 mG dan nilai akhir -18568.1 mG, sedangkan nilai terkecil adalah -18775.8 mG dan nilai terbesar -18111.2 mG.



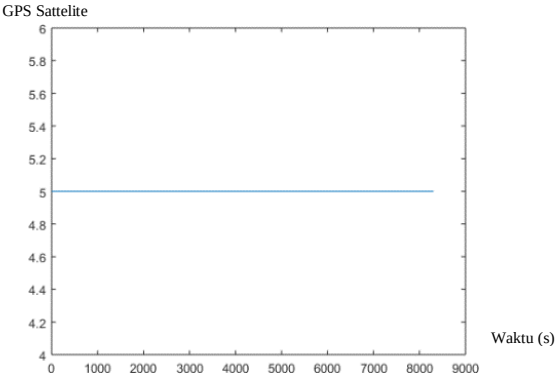
Gambar 4.19. Data Magnetometer Y Normal

Pada Gambar 4.19 data *Magnetometer* Y, nilai rata-rata sebesar -632.367 mG, sedangkan nilai standar deviasi yang dimiliki adalah 84.95. Pada grafik dapat diamati nilai awal yang muncul sebesar -581.55 mG dan nilai akhir -747.71 mG, sedangkan nilai terkecil adalah -934.64 mG dan nilai terbesar -249.24 mG.



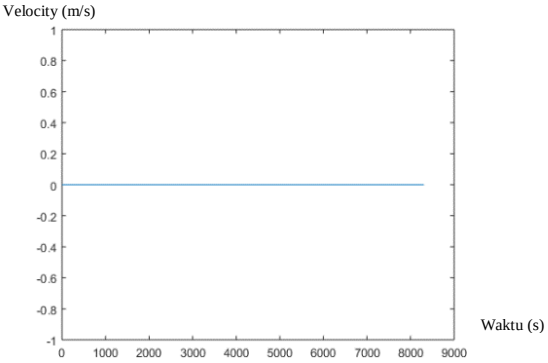
Gambar 40.20. Data Magnetometer Z Normal

Nilai rata-rata data *Magnetometer* Z pada Gambar 4.20 sebesar -1852.59 mG, sedangkan nilai standar deviasi yang dimiliki adalah 93.49. Pada grafik dapat diamati nilai awal yang muncul sebesar -18360.43 mG dan nilai akhir -18630.44 mG, sedangkan nilai terkecil adalah -18941.9 mG dan nilai terbesar -249.24 mG.



Gambar 4.21. Data GPS Sattelite Normal

Data GPS *Sattelite* pada Gambar 4.21 memiliki nilai rata-rata sebesar 5. Pergerakan nilai dari waktu ke waktu pada grafik menunjukkan bahwa data tersebut stabil karena angka yang dikeluarkan bernilai 5 sehingga standar deviasi yang dimiliki adalah 0.



Gambar 4.22. Data Velocity Kondisi Normal

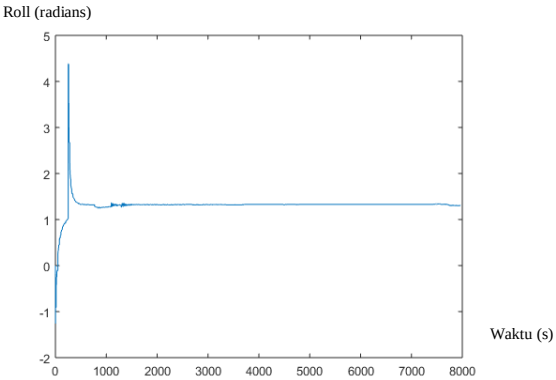
Dari grafik *Velocity* pada Gambar 4.22 diamati bahwa data bernilai 0. Hal ini dikarenakan posisi Sublocus tidak sedang berjalan, sehingga menunjukan nilai nol.

Perhitungan Sublocus ketika kondisi normal atau saat diam. Tidak semua jenis data menunjukkan nilai 0 di awal. Hal ini dikarenakan bidang permukaan tanah tidak datar atau tidak rata.

Pengujian berikutnya dilakukan dengan cara memiringkan posisi Sublocus. Ilustrasi pengujian ketika kondisi miring dapat dilihat pada Gambar 4.23.

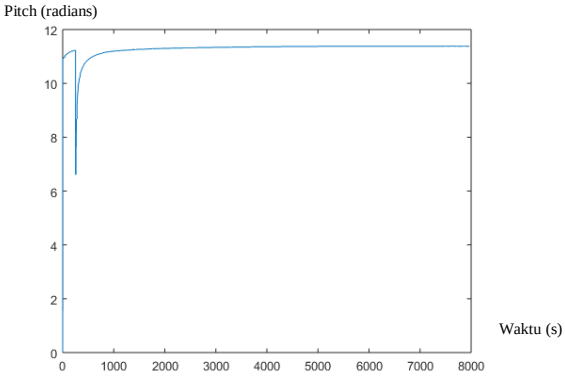
Gambar 4.23. Sublocus dalam Kondisi

Pengujian ini dilakukan dengan mengubah posisi *Yaw* atau biasa disebut *Heading* dan *roll* Sublocus, sehingga diperoleh salah satu sisi yang lebih tinggi.



Gambar 4.24. Data Roll Miring

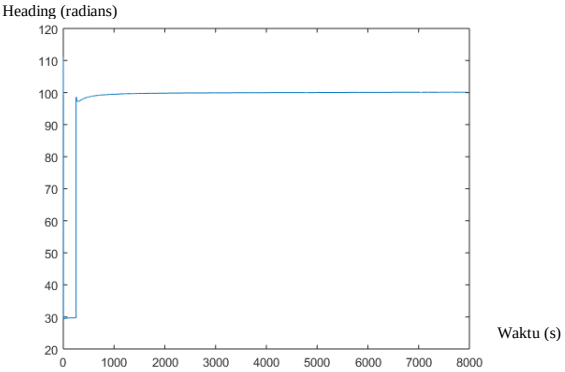
Nilai rata-rata pada data *Roll* pada Gambar 4.24 sebesar 1.311933 radians, sedangkan nilai rata-rata eror yang dimiliki adalah 2.5721 radians. Pada grafik dapat diamati nilai awal adalah -1.26 dan nilai akhir sebesar 1.31 radians. Data-data tersebut memiliki keragaman 0.216.



Gambar 4.25. Data Pitch Miring

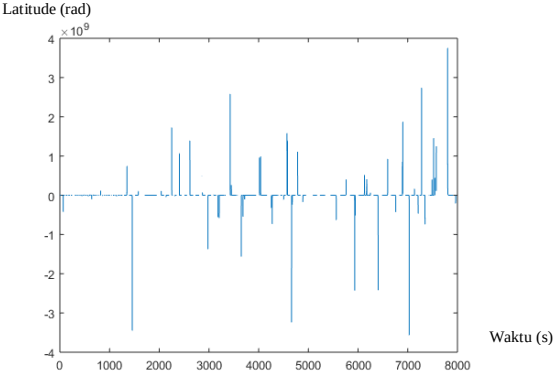
Pada Gambar 4.25 nilai rata-rata data *Pitch* sebesar 11.28525 radians, sedangkan nilai rata-rata eror yang dimiliki adalah 0.37668 radians. Pada

grafik dapat diamati nilai awal yang muncul sebesar 0.01 radians dan nilai akhir 11.38 radians. Data ini memiliki standar deviasi 0.285.



Gambar 4.26. Data Heading Miring

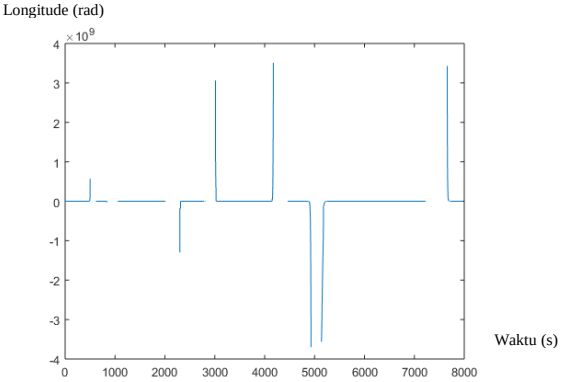
Nilai rata-rata pada Gambar 4.26 data *Heading* atau *Yaw* sebesar 97.6581 radians, sedangkan nilai standar deviasi yang dimiliki adalah 12.22. Pada grafik dapat diamati nilai awal yang muncul sebesar 111,11 radians dan nilai akhir 100.12 radians. Respon grafik menunjukkan bahwa sinyal berada pada kondisi stabil.



Gambar 4.27. Data Latitude Miring

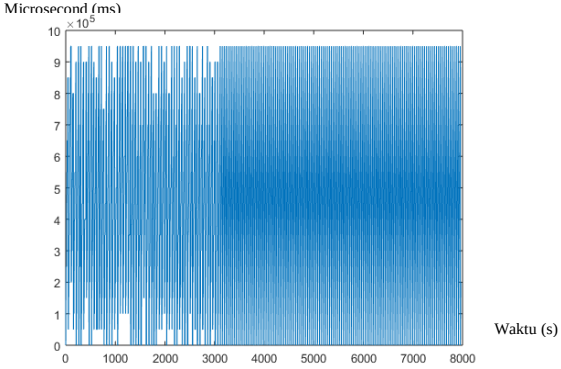
Pada Gambar 4.27 merupakan data *Latitude* yang memiliki nilai rata-rata sebesar 7110707.114 rad, sedangkan nilai standar deviasi atau keragaman data yang dimiliki adalah 231788099. Pergerakan nilai dari waktu ke

waktu pada grafik menunjukkan bahwa data tersebut bersifat heterogen atau bervariasi.



Gambar 4.28. Data Longitude Miring

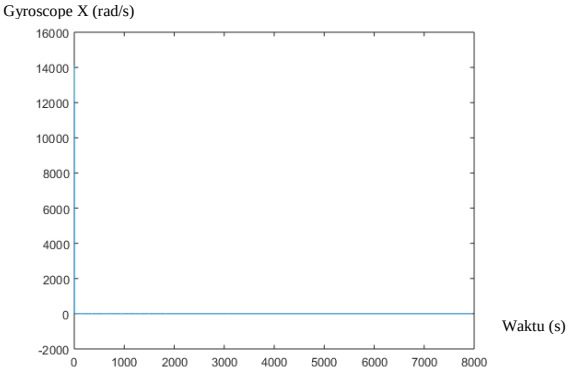
Nilai rata-rata pada data *Latitude* dalam Gambar 4.28 sebesar -10551913.9 rad, sedangkan nilai standar deviasi yang dimiliki adalah 263151942.4. Pada grafik dapat diamati nilai awal yang muncul sebesar -547,17 rad dan nilai akhir 2495.59 rad. Pergerakan nilai dari waktu ke waktu pada grafik menunjukkan bahwa data tersebut bersifat heterogen atau bervariasi.



Gambar 4.29. Data Microseconds Kondisi Miring

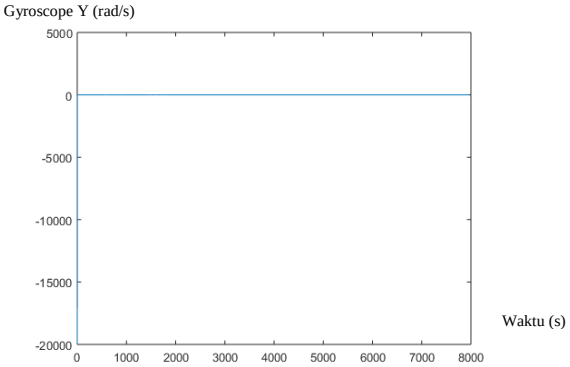
Gambar 4.29 menunjukkan data Microseconds, dimana nilai rata-rata sebesar 470746, sedangkan nilai standar deviasi yang dimiliki adalah

288516.0177. Pada grafik dapat diamati nilai awal yang muncul sebesar 747,71 dan nilai akhir 200065, sedangkan nilai terkecil adalah 68. Pergerakan nilai dari waktu ke waktu pada grafik menunjukkan bahwa data tersebut mengalami fluktuasi.



Gambar 4.30. Data Gyroscope X Kondisi Miring

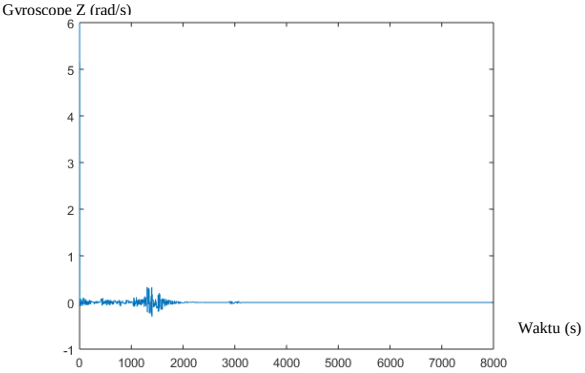
Nilai rata-rata pada data *Gyroscope* X pada Gambar 4.30 sebesar 1.775811 rad/s, sedangkan nilai standar deviasi yang dimiliki adalah 0.053. Nilai awal yang muncul sebesar 14081,87 rad/s dan nilai akhir 0.01 rad/s, sedangkan nilai terkecil adalah -0.51 rad/s dan nilai terbesar 0.68 rad/s. Pada grafik menunjukkan pergerakan nilai dari waktu ke waktu.



Gambar 4.31. Data Gyroscope Y Kondisi Miring

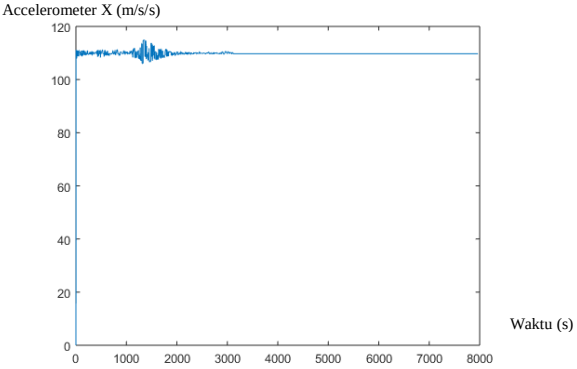
Gambar 4.31 data *Gyroscope* Y memiliki nilai rata-rata sebesar -2.50322 rad/s, sedangkan nilai standar deviasi yang dimiliki adalah 0.036. Pada

grafik dapat diamati nilai awal yang muncul sebesar -19938,93 rad/s dan nilai akhir juga 0 rad/s, sedangkan nilai terkecil adalah -0.24 rad/s dan nilai terbesar 0.27 rad/s.



Gambar 4.32. Data Gyroscope Z Kondisi Miring

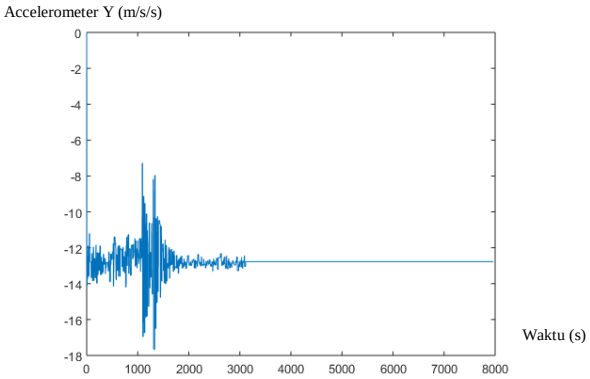
Nilai rata-rata pada Gambar 4.32 data *Gyroscope* Z sebesar 0.000984 rad/s, sedangkan nilai standar deviasi yang dimiliki adalah 0.034. Pada grafik dapat diamati nilai awal yang muncul sebesar 6 rad/s dan nilai akhir juga 0 rad/s, sedangkan nilai terkecil adalah -0.3 rad/s dan nilai terbesar 0.33 rad/s. Pada perubahan waktu sedikit berbeda dengan data *Gyroscope* X dan *Gyroscope* Y.



Gambar 4.33. Data Accelerometer X Kondisi Miring

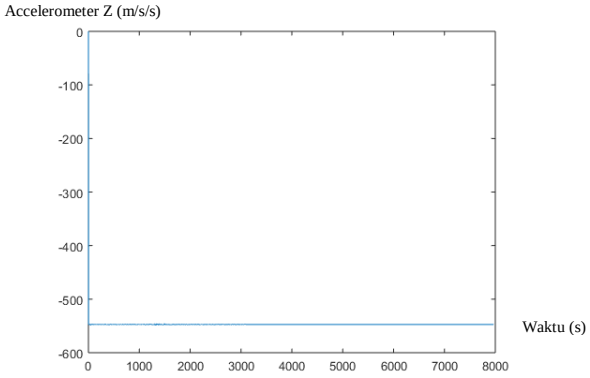
Pada Gambar 4.33 data *Accelerometer* X memiliki nilai rata-rata sebesar 109.7758, sedangkan nilai standar deviasi yang dimiliki adalah 0.698.

Pada grafik dapat diamati nilai awal yang muncul sebesar 111.11 dan nilai akhir 109.71, sedangkan nilai terkecil adalah 106.11 dan nilai terbesar 114.85.



Gambar 4.34. Data Accelerometer Y Kondisi Miring

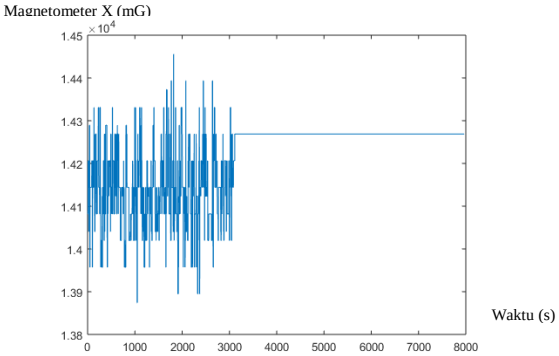
Nilai rata-rata pada Gambar 4.34 data *Accelerometer* Y sebesar -12.7755 m/s/s, sedangkan nilai standar deviasi yang dimiliki adalah 0.678. Pada grafik dapat diamati nilai awal yang muncul sebesar 6.67 m/s/s dan nilai akhir 7.4 m/s/s, sedangkan nilai terkecil adalah 4.59 m/s/s dan nilai terbesar 9.41 m/s/s.



Gambar 4.35. Data Accelerometer Z Kondisi Miring

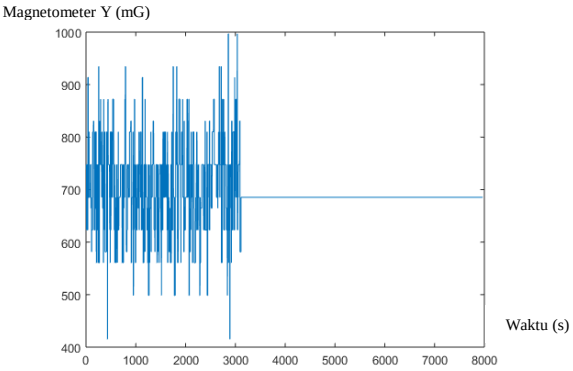
Pada Gambar 4.35 data *Accelerometer* Z nilai rata-rata sebesar -547.226 m/s/s, sedangkan nilai standar deviasi yang dimiliki adalah 0.166. Pada grafik dapat diamati nilai awal yang muncul sebesar -547.17 m/s/s dan

nilai akhir -547.31 m/s/s, sedangkan nilai terkecil adalah -548.25 m/s/s dan nilai terbesar -545.96 m/s/s. Pada perubahan waktu sedikit berbeda dengan data *Accelerometer* X dan Y.



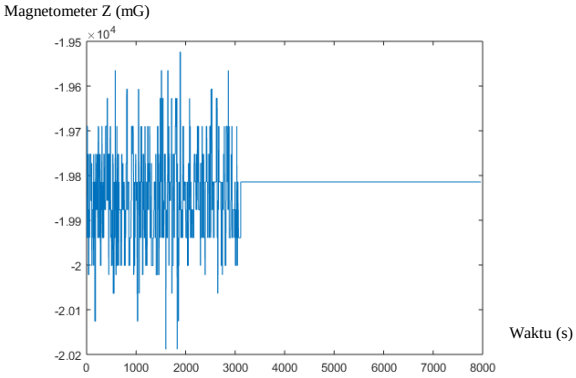
Gambar 4.36. Data Magnetometer X Kondisi Miring

Nilai rata-rata pada Gambar 4.36 data *Magnetometer* X sebesar 692.5354 mG, sedangkan nilai standar deviasi yang dimiliki adalah 60.29. Pada grafik dapat diamati nilai awal yang muncul sebesar 747.71 mG dan nilai akhir 685.4 mG, sedangkan nilai terkecil adalah 415.39 mG dan nilai terbesar 996.95 mG. Pergerakan nilai dari waktu ke waktu pada grafik menunjukkan bahwa data tersebut mengalami fluktuasi pada detik awal hingga detik ke 3189 dan detik berikutnya stabil.



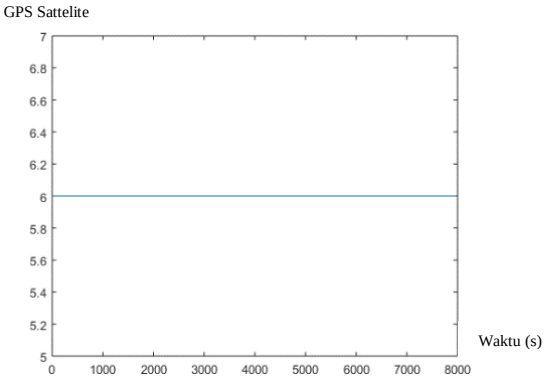
Gambar 4.37. Data Magnetometer Y Kondisi Miring

Pada Gambar 4.37 data *Magnetometer* Y, nilai rata-rata sebesar 14217.73 mG, sedangkan nilai standar deviasi yang dimiliki adalah 87.59. Nilai awal yang muncul sebesar 14081.87 mG dan nilai akhir 14268.8 mG, sedangkan nilai terkecil adalah 13874.17 mG dan nilai terbesar 14455.72 mG.



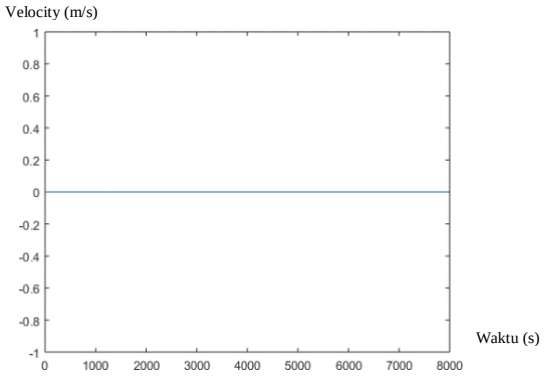
Gambar 4.38. Data Magnetometer Z Kondisi Miring

Nilai rata-rata pada Gambar 4.38 yaitu data *Magnetometer* Z sebesar -19828.6 mG, sedangkan nilai standar deviasi adalah 71.68. Pada grafik dapat diamati nilai awal yang muncul sebesar -19938.93 mG dan nilai akhir -19814.31 mG, sedangkan nilai terkecil adalah -20188.17 mG dan nilai terbesar -19523.53 mG.



Gambar 4.39. Data GPS Satellite Kondisi Miring

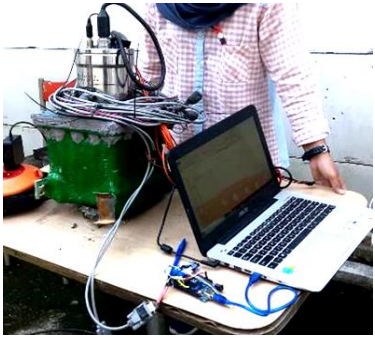
Pada Gambar 4.39 nilai rata-rata pada data GPS *Sattelite* sebesar 6 yang berarti Sublocus dapat menangkap enam satelit. Pergerakan nilai dari waktu ke waktu pada grafik menunjukkan bahwa data tersebut stabil karena angka yang dikeluarkan bernilai 6 sehingga standar deviasi yang dimiliki adalah 0.



Gambar 4.40. Data Velocity Kondisi Miring

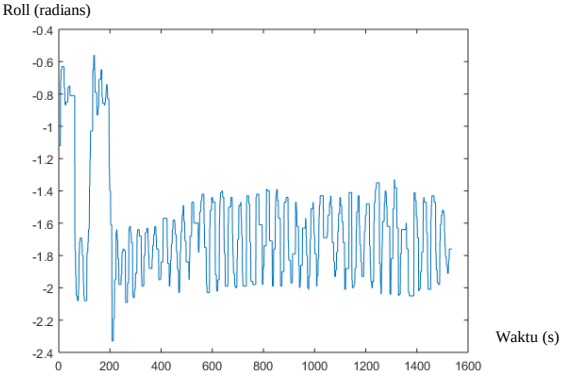
Dari grafik yang diilustrasikan pada Gambar 4.40 yaitu *Velocity* diamati bahwa data bernilai 0. Hal ini dikarenakan posisi Sublocus tidak sedang berjalan, sehingga menunjukan nilai nol.

Dilakukan pengujian ketika Sublocus berada di kondisi guncangan. Sensor Sublocus diletakkan diatas meja yang datar, setelah itu meja digerakkan dengan cara diayunkan seperti yang terlihat pada Gambar 4.41.



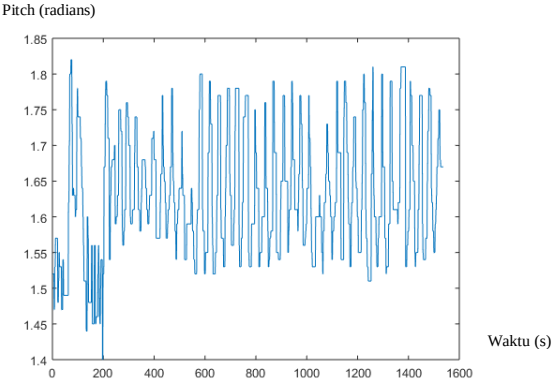
Gambar 4.41. Sublocus dalam Kondisi Guncangan

Pengujian dalam kondisi guncangan dilakukan bertujuan untuk mengetahui respon sensor ketika terjadi ombak di laut.



Gambar 4.42. Data Roll Kondisi Guncangan

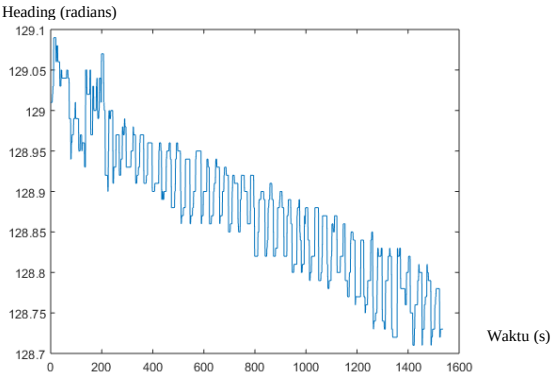
Nilai rata-rata pada data *Roll* pada Gambar 4.42 sebesar -1.63418 radians, sedangkan nilai rata-rata eror yang dimiliki adalah 0.474 radians. Nilai awal adalah -1.16 radians dan nilai akhir sebesar -1.76 radians. Nilai terbesar pada kumpulan data ini adalah -0.56, sedangkan yang terkecil -2.33. Data-data tersebut memiliki keragaman atau standar deviasi 0.333.



Gambar 4.43. Data *Pitch* Kondisi Guncangan

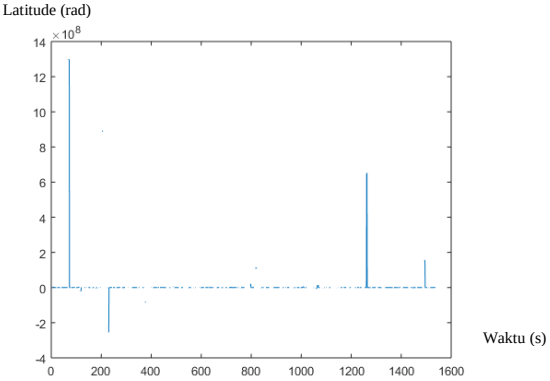
Gambar 4.43 merupakan data *Pitch* yang memiliki nilai rata-rata sebesar 1.634698 radians, sedangkan nilai rata-rata eror yang dimiliki adalah 0.117 radians. Pada grafik dapat diamati nilai awal adalah -1.16 radians

dan nilai akhir sebesar -1.76 radians. Nilai terbesar pada kumpulan data ini adalah -0.56, sedangkan yang terkecil -2.33. Data tersebut memiliki standar deviasi 0.089.



Gambar 4.44. Data *Heading* Kondisi Guncangan

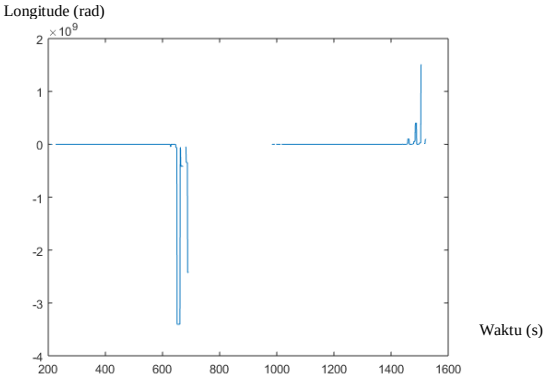
Nilai rata-rata pada data *Heading* pada Gambar 4.44 sebesar 128.8262 radians. Pada grafik dapat diamati nilai awal adalah 129.07 radians dan nilai akhir sebesar 128.73 radians. Nilai terbesar pada kumpulan data ini adalah 129.09, sedangkan yang terkecil 128.71. Data tersebut memiliki standar deviasi 0.094.



Gambar 4.45. Data *Latitude* Kondisi Guncangan

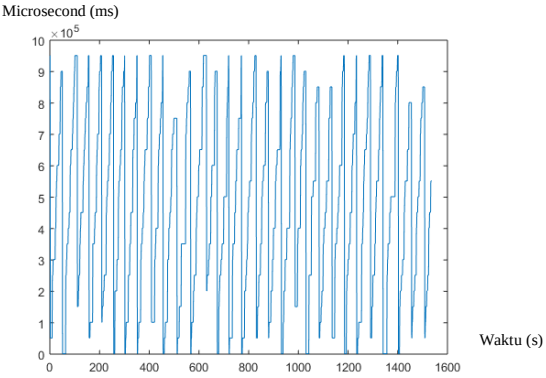
Gambar 4.45 merupakan data *Latitude* yang memiliki nilai rata-rata sebesar 12341233 rad, sedangkan nilai standar deviasi yang dimiliki

adalah 117953844.6. Pada grafik dapat diamati nilai awal yang muncul sebesar 0 rad dan nilai akhir 0 rad. Pergerakan nilai pada grafik menunjukkan bahwa data tersebut bersifat heterogen atau bervariasi.



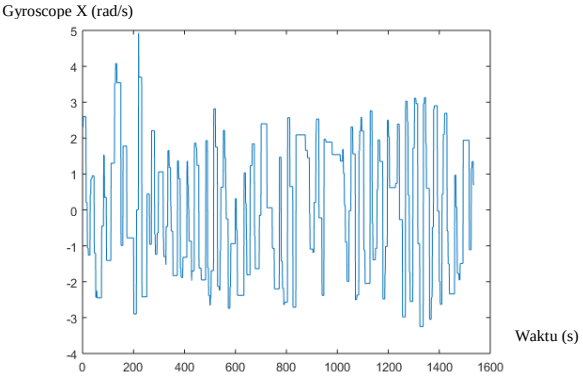
Gambar 4.46. Data *Logitude* Kondisi Guncangan

Nilai rata-rata pada Gambar 4.46 data *Longitude* sebesar -9938829.64 rad, sedangkan nilai standar deviasi yang dimiliki adalah 416734122.4. Pada grafik dapat diamati nilai awal yang muncul sebesar 0 rad.



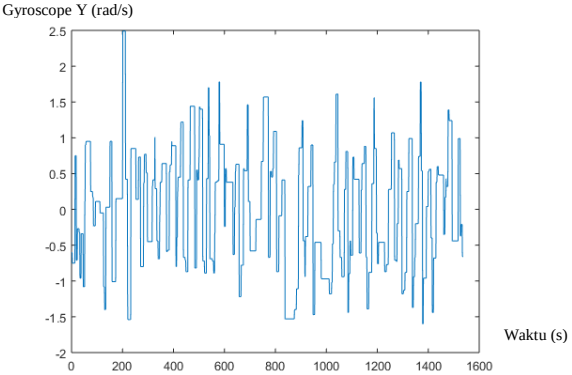
Gambar 4.47. Data *Microseconds* Kondisi Guncangan

Gambar 4.47 merupakan data *Microseconds* yang memiliki nilai rata-rata sebesar 473026.7, sedangkan nilai standar deviasi yang dimiliki adalah 291513.3921. Pada grafik dapat diamati nilai awal yang muncul sebesar 201560 dan nilai akhir 551431, sedangkan nilai terkecil adalah 1352.



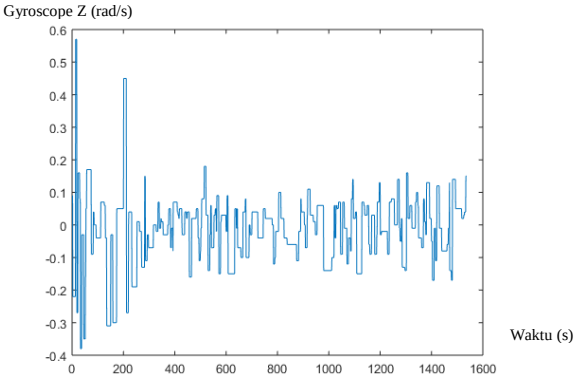
Gambar 4.48. Data *Gyroscope* X Kondisi Guncangan

Nilai rata-rata pada Gambar 4.48 data *Gyroscope* X sebesar -12.1144 rad/s, sedangkan nilai standar deviasi yang dimiliki adalah 1.801. Pada grafik dapat diamati nilai awal yang muncul sebesar 0.01 rad/s dan nilai akhir 0.7 rad/s, sedangkan nilai terkecil adalah -3.25 rad/s dan nilai terbesar 4.93 rad/s.



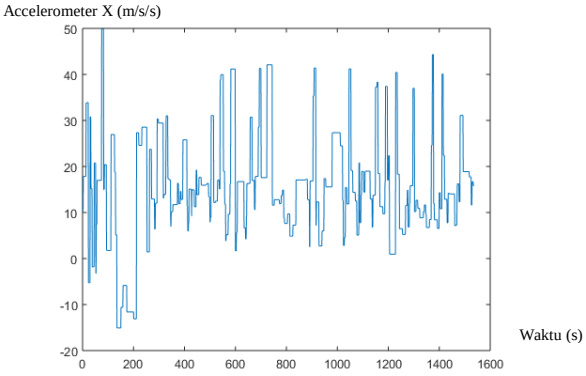
Gambar 4.49. Data *Gyroscope* Y Kondisi Guncangan

Gambar 4.49 merupakan data *Gyroscope* Y yang memiliki nilai rata-rata sebesar 0.013063 rad/s, sedangkan nilai standar deviasi yang dimiliki adalah 0.8. Pada grafik dapat diamati nilai awal yang muncul sebesar -0.05 rad/s dan nilai akhir -0.66 rad/s, sedangkan nilai terkecil adalah -1,16 rad/s dan nilai terbesar 2.49 rad/s.



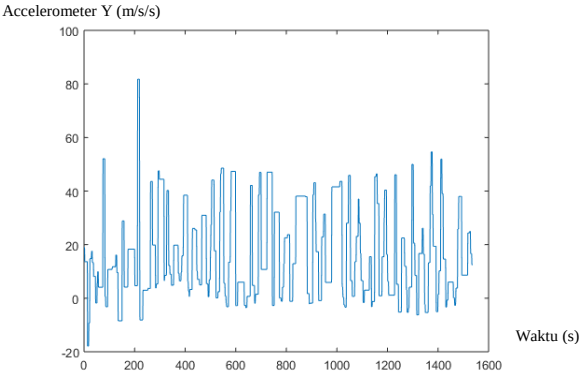
Gambar 4.50. Data *Gyroscope* Z Kondisi Guncangan

Nilai rata-rata pada Gambar 4.50 data *Gyroscope* Z sebesar -0.01219 rad/s, sedangkan nilai standar deviasi yang dimiliki adalah 0.107. Nilai awal yang muncul sebesar -0.05 rad/s dan nilai akhir -0.66 rad/s, sedangkan nilai terkecil adalah -0,38 rad/s dan nilai terbesar 0.57 rad/s.



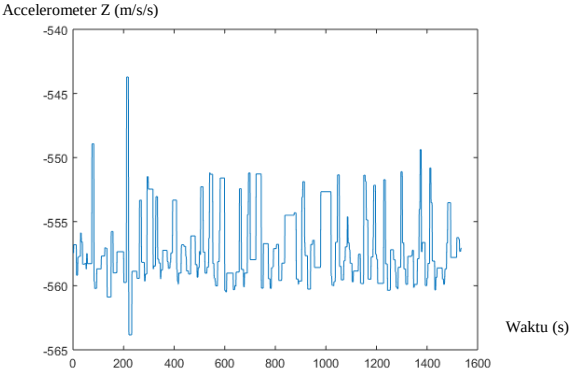
Gambar 4.51. Data *Accelerometer* X Kondisi Guncangan

Gambar 4.51 merupakan data *Accelerometer* X yang memiliki nilai rata-rata sebesar 15.619 m/s/s, sedangkan nilai standar deviasi yang dimiliki adalah 0.107. Pada grafik dapat diamati nilai awal yang muncul sebesar 13.79 m/s/s dan nilai akhir 15.82 m/s/s, sedangkan nilai terkecil adalah -15.08 m/s/s dan nilai terbesar 49.89 m/s/s.



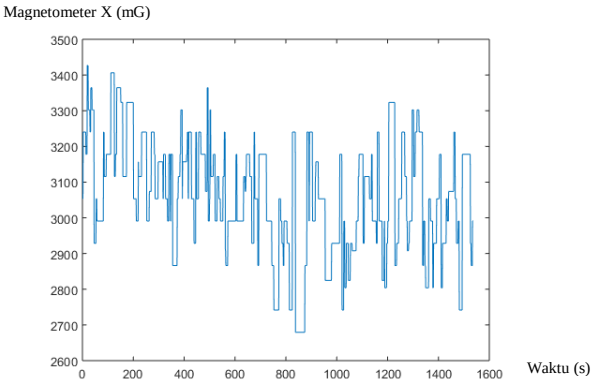
Gambar 4.52. Data *Accelerometer* Y Kondisi Guncangan

Nilai rata-rata pada Gambar 4.52 data *Accelerometer* Y sebesar 16.33314 m/s/s, sedangkan nilai standar deviasi yang dimiliki adalah 16.75. Nilai awal yang muncul sebesar 11.89 m/s/s dan nilai akhir 12.47 m/s/s, sedangkan nilai terkecil adalah -17.88 m/s/s dan nilai terbesar 81.76 m/s/s.



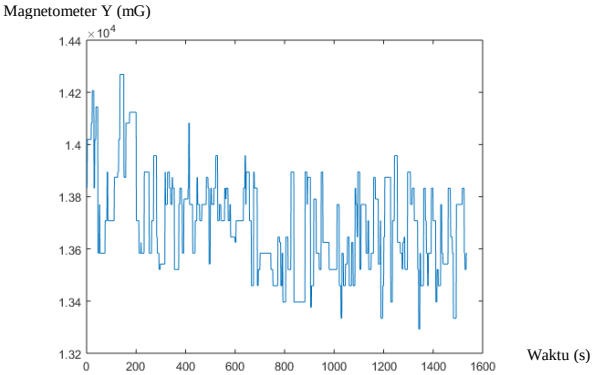
Gambar 4.53. Data *Accelerometer* Z Kondisi Guncangan

Gambar 4.53 merupakan data *Accelerometer* Z yang memiliki nilai rata-rata sebesar -557.15 m/s/s, sedangkan nilai standar deviasi yang dimiliki adalah 2.782. Pada grafik dapat diamati nilai awal yang muncul sebesar -558.05 m/s/s dan nilai akhir -557.08 m/s/s, nilai terkecil adalah -563.83 m/s/s dan nilai terbesar -543.72 m/s/s.



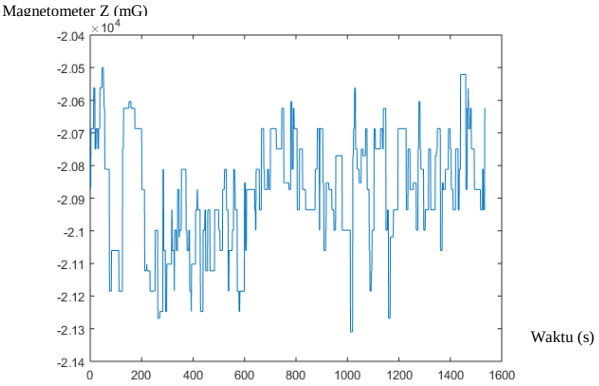
Gambar 4.54. Data *Magnetometer* X Kondisi Guncangan

Nilai rata-rata pada Gambar 4.54 data *Magnetometer* X sebesar 3069.573 mG, sedangkan nilai standar deviasi yang dimiliki adalah 167.3. Nilai awal yang muncul sebesar 2907.76 mG dan nilai akhir 2990.84 mG, sedangkan nilai terkecil adalah 2679.29 mG dan nilai terbesar 3427 mG.



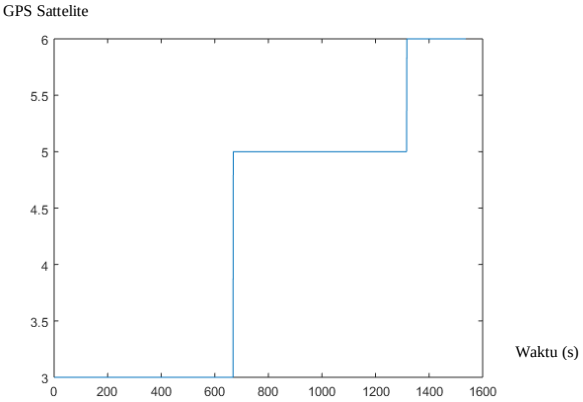
Gambar 4.55. Data *Magnetometer* Y Kondisi Guncangan

Gambar 4.55 merupakan data *Magnetometer* Y yang memiliki nilai rata-rata sebesar 3704.64 mG, sedangkan nilai standar deviasi yang dimiliki adalah 190.4. Nilai awal yang muncul sebesar 13541.86 mG dan nilai akhir 13583.4 mG, sedangkan nilai terkecil adalah 13292.62 mG dan nilai terbesar 14268.8 mG.



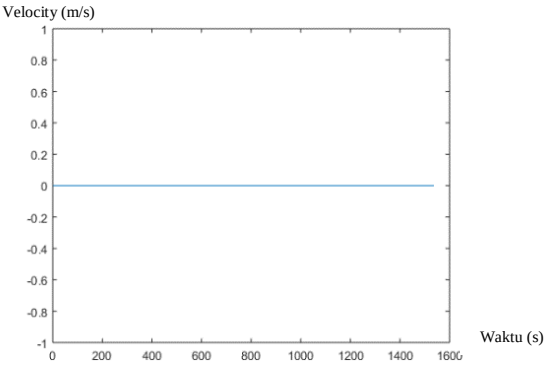
Gambar 4.56. Data *Magnetometer* Z Kondisi Guncangan

Nilai rata-rata pada Gambar 4.56 data *Magnetometer* Z sebesar -20884.4 mG, sedangkan nilai standar deviasi yang dimiliki adalah 178.5. Nilai awal yang muncul sebesar -20437.4 mG dan nilai akhir -20624.33 mG, sedangkan nilai terkecil adalah -21309.73 mG dan nilai terbesar -20437.4 mG.



Gambar 4.57. Data GPS *Satellite* Kondisi Guncangan

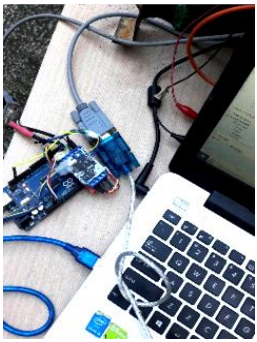
Gambar 4.57 merupakan data GPS *Sattelite* yang memiliki nilai rata-rata sebesar 4. Nilai terbesar adalah 6 dan terkecil 3. Standar deviasi atau keragaman data sebesar 1.169. Data tersebut menunjukkan bahwa Sublocus dapat menerima macam-macam satelit untuk diakses.



Gambar 4.58. Data *Velocity* Kondisi Guncangan

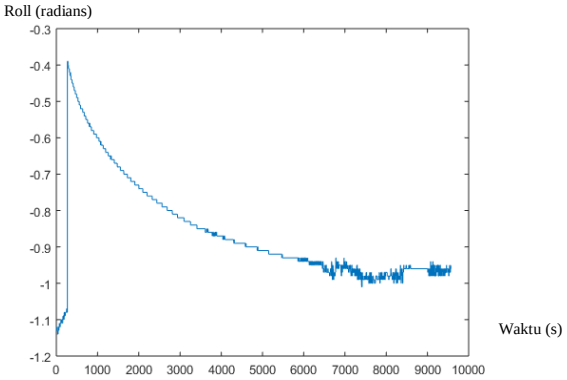
Dari Gambar 4.58 yaitu grafik *Velocity* diamati bahwa data bernilai 0. Hal ini dikarenakan posisi Sublocus tidak sedang berjalan, sehingga menunjukan nilai nol.

Terakhir dilakukan pengujian ketika Sublocus diberi pengaruh data eksternal dengan melakukan *push Velocity*. Pengujian bertujuan untuk mengetahui respon pembacaan *Velocity* pada Sublocus. Ilustrasi pengujian dengan pengaruh data *push* eksternal *Velocity* dapat dilihat pada Gambar 4.59.



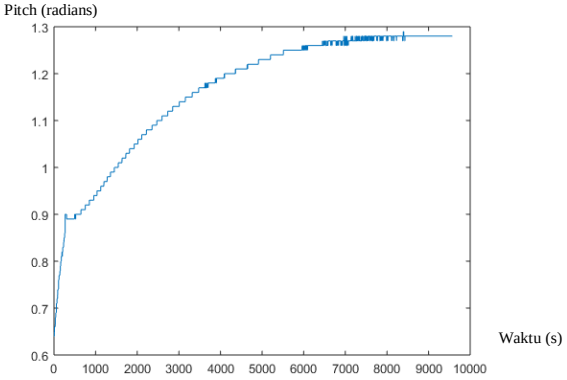
Gambar 4.59. Sublocus dalam Kondisi Pengaruh Data Eksternal

Pengujian terhadap perhitungan *Velocity*, dilakukan dengan memberikan data yang dibuat menggunakan perangkat lunak dan ditransmisikan menggunakan kabel RS 232 ke Sublocus seperti Gambar 4.59.



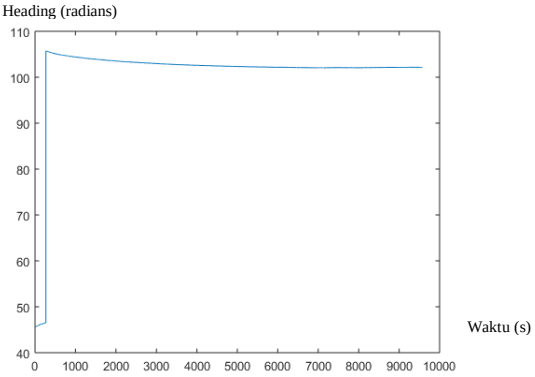
Gambar 4.60. Data *Roll* Kondisi Pengaruh *Push* Eksternal *Velocity*

Nilai rata-rata pada Gambar 4.60 data *Roll* sebesar -0.86094 radians, sedangkan nilai rata-rata eror 0.25906 radians. Nilai awal adalah -1.12 radians dan nilai akhir sebesar -0.96 radians. Nilai terbesar pada data ini adalah -0.39, dan yang terkecil -1.14. Data tersebut memiliki keragaman 0.142.



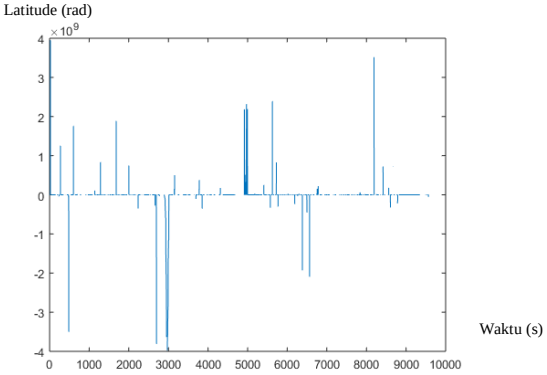
Gambar 4.61. Data *Pitch* Kondisi Pengaruh *Push* Eksternal *Velocity*

Gambar 4.61 yaitu data *Pitch* memiliki nilai rata-rata sebesar 1.163011 radians, sedangkan nilai rata-rata eror yang dimiliki adalah 0.52301 radians. Pada grafik dapat diamati nilai awal adalah 0.64 radians dan nilai akhir sebesar 1.28 radians. Nilai terbesar pada kumpulan data ini adalah 1.29, sedangkan terkecil 0.64. Data ini memiliki keragaman 0.136.



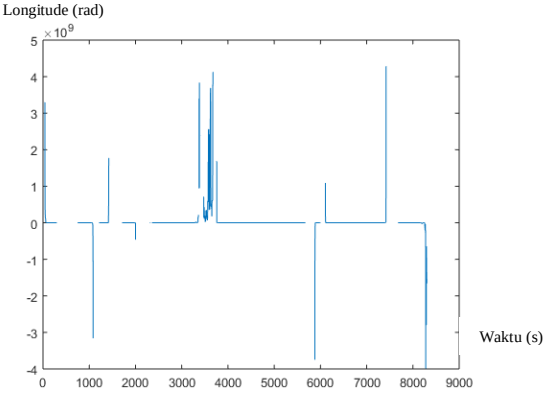
Gambar 4.62. Data *Heading* Kondisi Pengaruh *Push* Eksternal *Velocity*

Nilai rata-rata pada Gambar 4.62 data *Heading* sebesar 101.1917 radians. Nilai awal adalah 45.58 radians dan nilai akhir sebesar 102.13 radians. Nilai terbesar pada kumpulan data ini adalah 105.74, sedangkan yang terkecil 45.58. Data-data tersebut memiliki standar deviasi 9.369.



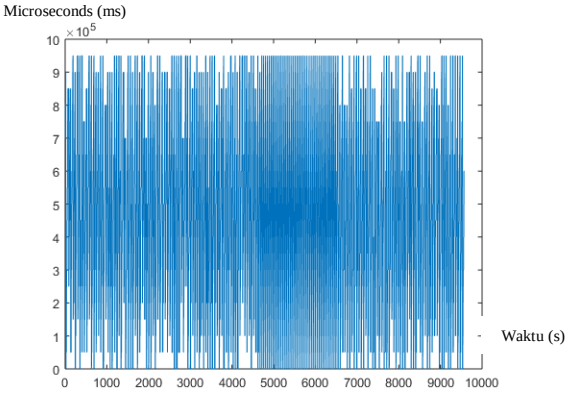
Gambar 4.63. Data *Latitude* Kondisi Pengaruh *Push* Eksternal *Velocity*

Gambar 4.63 yaitu data *Latitude* memiliki nilai rata-rata sebesar -21938178.19 rad, sedangkan nilai standar deviasi yang dimiliki adalah 391612792.8. Pada grafik dapat diamati nilai awal yang muncul sebesar 0 rad dan nilai akhir 0 rad. Pergerakan nilai pada grafik menunjukkan bahwa data tersebut bersifat heterogen atau bervariasi. *Lattitude* merupakan garis lintang antara kutub utara dan kutub selatan.



Gambar 4.64. Data *Longitude* Kondisi Pengaruh *Push* Eksternal *Velocity*

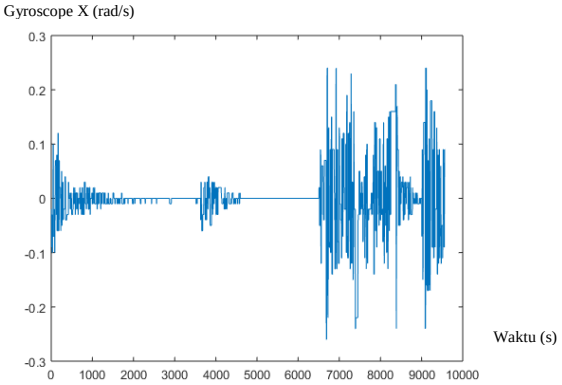
Nilai rata-rata pada Gambar 4.64 data *Longitude* sebesar 39843343 rad, sedangkan nilai standar deviasi yang dimiliki adalah 376000606.3. Data tersebut merupakan data mentah, sehingga terdapat nilai-nilai yang tidak bias dibaca atau melebihi batas maksimum. *Longitude* adalah garis membujur yang menghubungkan antara sisi utara dan sisi selatan bumi.



Gambar 4.65. Data *Microseconds* Kondisi Pengaruh *Push* Eksternal *Velocity*

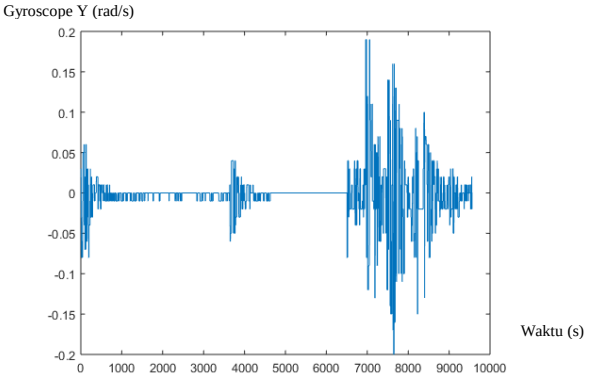
Gambar 4.65 yaitu data *Microseconds* memiliki nilai rata-rata sebesar 472311.9, sedangkan nilai standar deviasi yang dimiliki adalah 289261.6. Pada grafik dapat diamati nilai awal yang muncul sebesar 700060 dan nilai akhir 600060, sedangkan nilai terkecil adalah 69 dan terbesar

950321. *Microseconds* adalah waktu pengiriman data yang telah ditentukan oleh Sublocus.



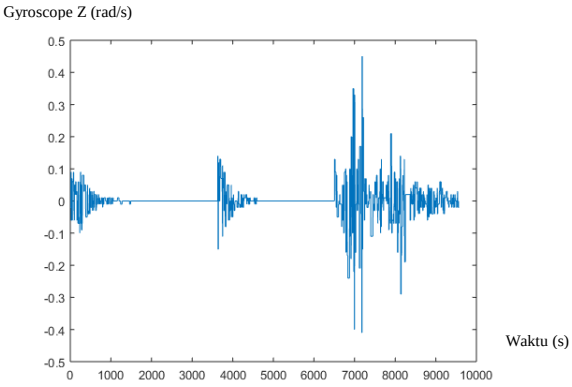
Gambar 4.66. Data *Gyroscope* X Kondisi Pengaruh *Push* Eksternal *Velocity*

Nilai rata-rata pada Gambar 4.66 data *Gyroscope* X sebesar 0.001085 rad/s, sedangkan nilai standar deviasi yang dimiliki adalah 0.056. Nilai awal yang muncul sebesar -0.02 rad/s dan nilai akhir 0.09 rad/s, sedangkan nilai terkecil adalah -0.26 rad/s dan nilai terbesar 0.24 rad/s.



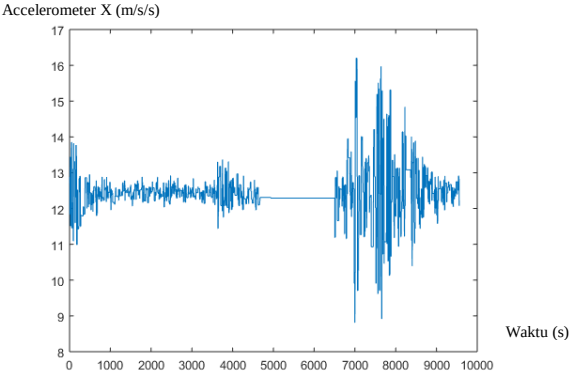
Gambar 4.67. Data *Gyroscope* Y Kondisi Pengaruh *Push* Eksternal *Velocity*

Gambar 4.67 yaitu data *Gyroscope* Y memiliki nilai rata-rata sebesar -0.00224 rad/s, sedangkan nilai standar deviasi adalah 0.033. Nilai awal yang muncul sebesar 0.03 rad/s dan nilai akhir 0 rad/s, sedangkan nilai terkecil adalah -0.2 rad/s dan nilai terbesar 0.19 rad/s.



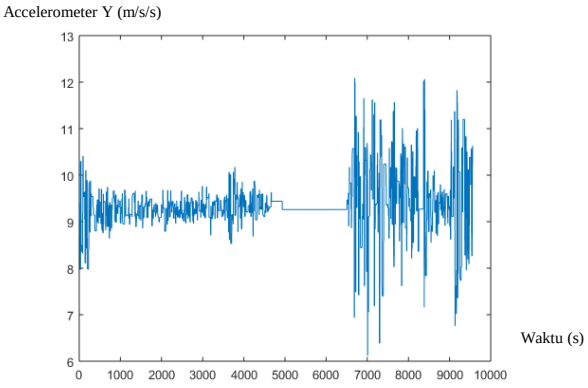
Gambar 4.68. Data *Gyroscope* Z Kondisi Pengaruh *Push* Eksternal *Velocity*

Nilai rata-rata pada Gambar 4.68 data *Gyroscope* Z sebesar 0.000329 rad/s, sedangkan nilai standar deviasi yang dimiliki adalah 0.054. Nilai awal yang muncul sebesar -0.02 rad/s dan nilai akhir -0.02 rad/s, sedangkan nilai terkecil adalah -0,41 rad/s dan nilai terbesar 0.45 rad/s.



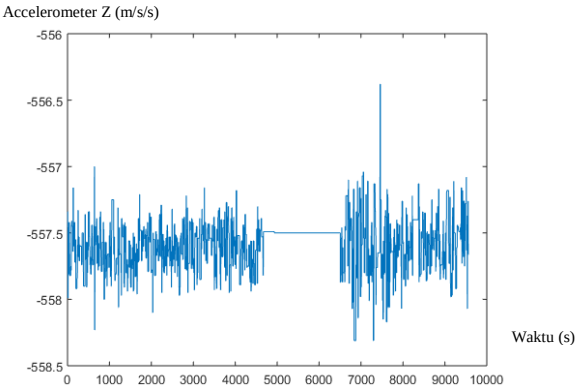
Gambar 4.69. Data *Accelerometer* X Kondisi Pengaruh *Push* Eksternal *Velocity*

Gambar 4.69 yaitu data *Accelerometer* X memiliki nilai rata-rata sebesar 12.41532 m/s/s, sedangkan keragaman data atau standar deviasi yang dimiliki adalah 0.639. Nilai awal sebesar 12.88 m/s/s dan nilai akhir 12.23 m/s/s, sedangkan nilai terkecil adalah 8.81 m/s/s dan nilai terbesar 16.2 m/s/s.



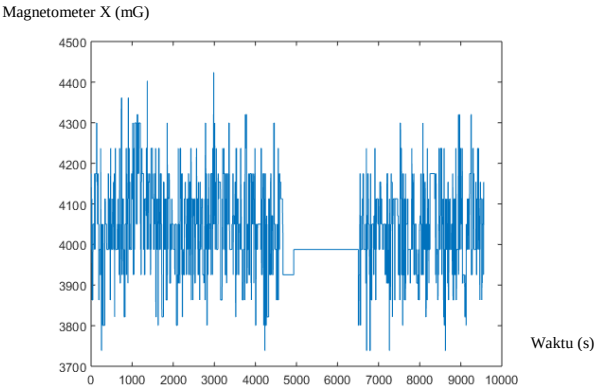
Gambar 4.70. Data *Accelerometer* Y Kondisi Pengaruh *Push* Eksternal *Velocity*

Nilai rata-rata pada Gambar 4.70 data *Accelerometer* Y sebesar 9.344673 m/s/s, sedangkan nilai standar deviasi atau keragaman data yang dimiliki adalah 0.597. Nilai awal sebesar 9.04 m/s/s dan nilai akhir 10.25 m/s/s, sedangkan nilai terkecil adalah 6.12 m/s/s dan nilai terbesar 12.09 m/s/s.



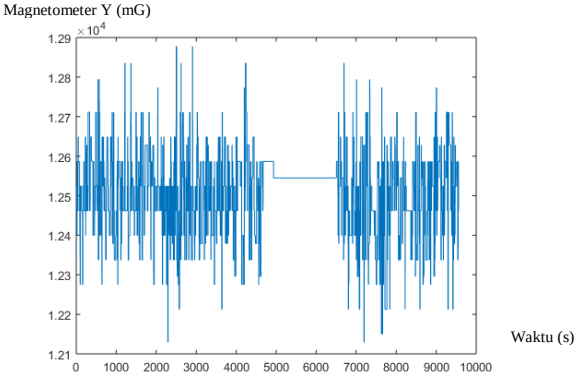
Gambar 4.71. Data *Accelerometer* Z Kondisi Pengaruh *Push* Eksternal *Velocity*

Gambar 4.71 yaitu data *Accelerometer* Z memiliki nilai rata-rata sebesar -557.584 m/s/s, sedangkan nilai standar deviasi atau keragaman data yang dimiliki sebesar 0.194. Nilai awal sebesar -557.34 m/s/s dan nilai akhir -557.66 m/s/s, sedangkan nilai terkecil -558.31 m/s/s dan nilai terbesar -556.38 m/s/s.



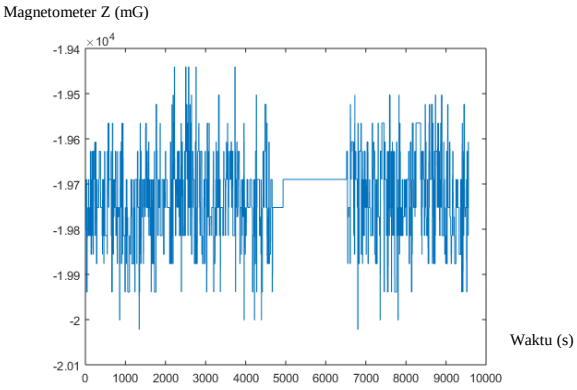
Gambar 4.72. Data *Magnetometer* X Kondisi Pengaruh *Push* Eksternal *Velocity*

Nilai rata-rata pada Gambar 4.72 data *Magnetometer* X sebesar 4032.432 mG, sedangkan keragaman data 109.4. Nilai awal yang muncul sebesar 4174.71 mG dan nilai akhir 4050.1 mG, sedangkan nilai terkecil 3738.55 mG dan nilai terbesar 4423.95 Mg.



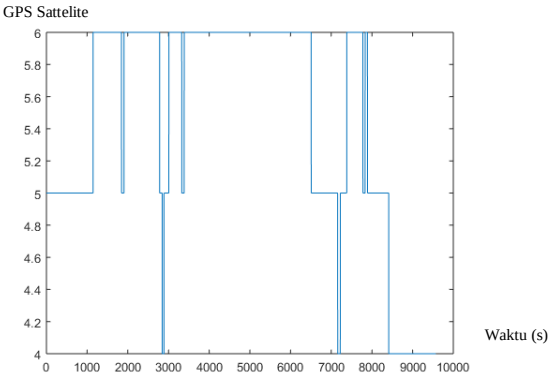
Gambar 4.73. Data *Magnetometer* Y Kondisi Pengaruh *Push* Eksternal *Velocity*

Gambar 4.73 yaitu data *Magnetometer* Y memiliki nilai rata-rata sebesar 12507.41 mG, sedangkan nilai standar deviasi adalah 107.4. Nilai awal yang muncul sebesar 12524.14 mG dan nilai akhir 12461.83 mG, sedangkan nilai terkecil adalah 12129.52 mG dan nilai terbesar 12877.2 mG.



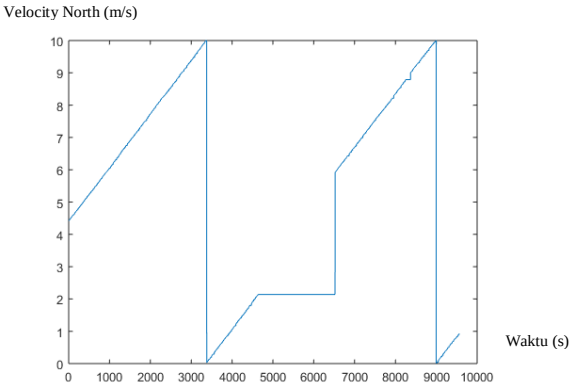
Gambar 4.74. Data *Magnetometer* Z Kondisi Pengaruh *Push* Eksternal *Velocity*

Nilai rata-rata pada Gambar 4.74 data *Magnetometer* Z sebesar -20884.4 mG, sedangkan nilai standar deviasi atau keragaman data yang dimiliki adalah 94.76. Nilai awal yang muncul sebesar -19814.31 mG dan nilai akhir -19752 mG, sedangkan nilai terkecil adalah -20022.01 mG dan nilai terbesar -19440.46 mG.



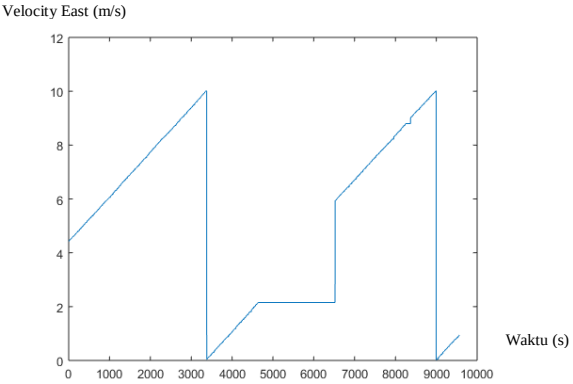
Gambar 4.75. Data GPS *Sattelite* Kondisi Pengaruh *Push* Eksternal *Velocity*

Gambar 4.75 yaitu data GPS *Sattelite* memiliki nilai rata-rata sebesar sebesar 5.4388626. Data ini memiliki standar deviasi atau keragaman data yang dimiliki sebesar 0.714. Nilai terkecil 4 dan terbesar adalah 6. Data tersebut menunjukkan bahwa Sublocus dapat menerima macam-macam satelit untuk diakses.



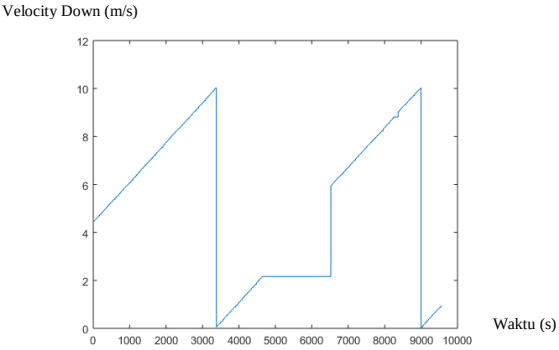
Gambar 4.76. Data *Velocity North* Kondisi Pengaruh *Push* Eksternal *Velocity*

Nilai rata-rata pada Gambar 4.76 data *Velocity North* sebesar 5.193883 m/s. Dari grafik ini diamati bahwa data awal yang muncul bernilai 4.42 m/s, sedangkan data akhir sebesar 0.92 m/s. Nilai terkecil adalah 0.01 m/s, dan terbesar 10 m/s. Data ini memiliki standar deviasi sebesar 3.186. Terlihat perubahan respon dari grafik *Velocity* ketika diberi data *push* eksternal.



Gambar 4.77. Data *Velocity East* Kondisi Pengaruh *Push* Eksternal *Velocity*

Gambar 101 yaitu data *Velocitye East* memiliki nilai rata-rata sebesar 5.203883 m/s. Data awal yang muncul bernilai 4.43 m/s, sedangkan data akhir sebesar 0.93 m/s. Nilai terkecil adalah 0.02 m/s, dan terbesar 10.01 m/s. Data ini memiliki standar deviasi sebesar 3.186.



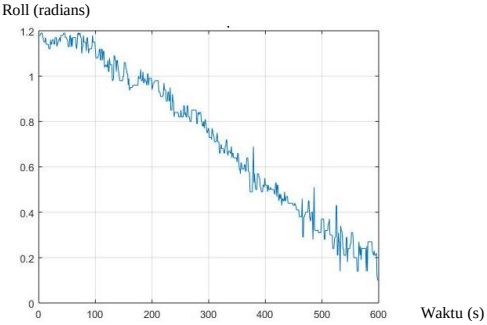
Gambar 4.78. Data *Velocity Down* Kondisi Pengaruh *Push* Eksternal *Velocity*

Nilai rata-rata pada Gambar 4.78 data *Velocity North* sebesar 5.213883 m/s. Dari grafik ini diamati bahwa data awal yang muncul bernilai 4.44 m/s, sedangkan data akhir sebesar 0.94 m/s. Nilai terkecil adalah 0.03 m/s, dan terbesar 10.02 m/s. Data ini memiliki standar deviasi sebesar 3.186.

Dengan adanya pengujian ini diperoleh hasil 76 data sesuai dengan respon keadaan Sublocus. Terdapat empat kondisi pengujian, dimana masing-masing berhasil mendapatkan 19 jenis data.

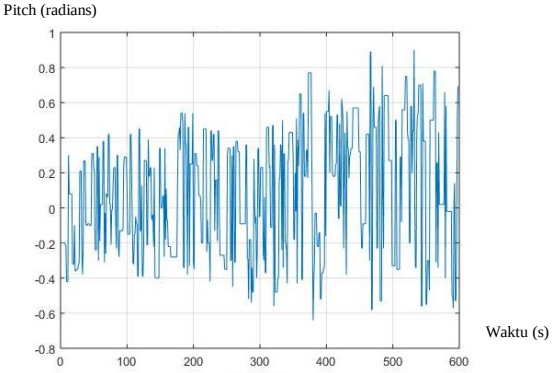
4.3. Hasil *Logging* Data

Perekaman dan penyimpanan data dilakukan untuk memperoleh informasi yang dibutuhkan, dimana dapat dianalisis secara *offline*. Data yang telah disimpan meliputi data *Roll*, *Pitch*, *Yaw* atau *Heading* dengan dua kondisi yang berbeda yaitu ketika kondisi normal dan dalam keadaan miring.



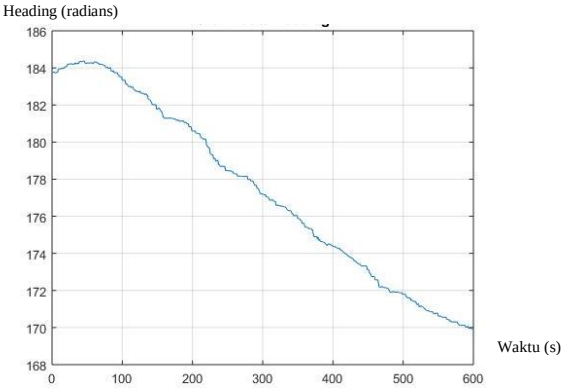
Gambar 4.79. Data *Logging Roll*

Pada Gambar 4.79, yaitu pada data *Roll* ketika keadaan miring menunjukkan bahwa perekaman dan penyimpanan menggunakan data *logger* berupa SD Card telah berhasil.



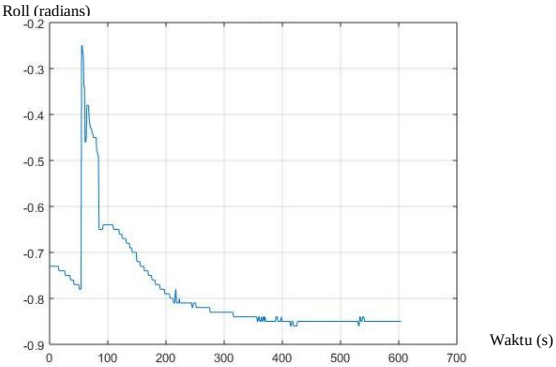
Gambar 4.80. Data *Logging Pitch*

Data *Pitch* yang telah ditampilkan pada Gambar 4.80 menunjukkan bahwa perekaman dan penyimpanan data berlangsung hingga 600 detik.



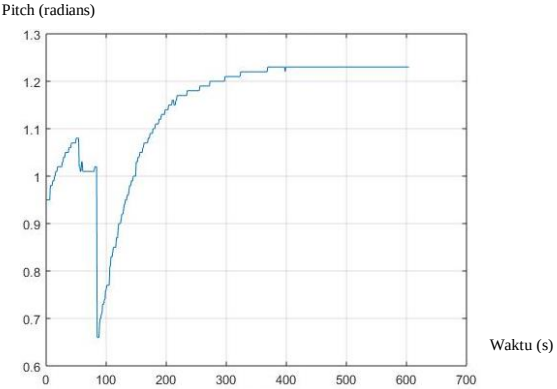
Gambar 4.81. Data *Logging Yaw/Heading*

Ilustrasi pada Gambar 4.81 menggambarkan respon grafik dari perekaman dan penyimpanan data *Yaw* atau *Heading*. Dimana dalam pengujiannya data ini didapat ketika Sublocus berada di posisi miring.



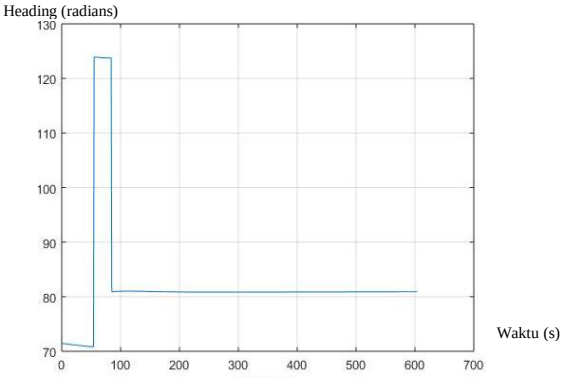
Gambar 4.82. Data *Logging Roll*

Pada Gambar 4.82, yaitu pada data *Roll* ketika keadaan normal menunjukkan bahwa perekaman dan penyimpanan menggunakan data *logger* berupa SD Card telah berhasil.



Gambar 4.83. Data *Logging Pitch*

Data *Pitch* yang telah ditampilkan pada Gambar 4.83 menunjukkan bahwa perekaman dan penyimpanan data berlangsung hingga 600 detik. Respon ini direkam ketika Sublocus berada di kondisi normal atau diam. Ilustrasi pada Gambar 105 menggambarkan respon grafik dari perekaman dan penyimpanan data *Yaw* atau *Heading*. Dimana dalam pengujiannya data ini didapat ketika Sublocus berada di posisi statis atau diam.



Gambar 4.84. Data *Logging Yaw/Heading*

Data yang disimpan dan akan dilakukan analisis berupa orientasi seperti *Roll*, *Pitch*, *Yaw* atau *Heading*. Parameter tersebut dipilih karena pada dasarnya gerak kapal mengikuti sumbu X, Y, dan Z yang dapat diterjemahkan dalam data tersebut.

Dari pengujian ini, didapatkan 12 data sesuai dengan respon keadaan Sublocus. Terdapat empat kondisi pengujian, dimana masing-masing berhasil merekam 3 jenis data pada SD Card. Penyimpanan data pada SD Card sebesar 259KB.

BAB V
PENUTUP

Dari penelitian ini diperoleh hasil pengambilan 19 macam data dari 76 data meliputi orientasi, posisi, dan kecepatan dari sensor inersia bernama Sublocus. Data tersebut berupa perhitungan *Roll*, *Pitch*, *Heading*, *Latitude*, *Longitude*, *Microseconds*, *Gyroscope* X, *Gyroscope* Y, *Gyroscope* Z, *Accelerometer* X, *Accelerometer* Y, *Accelerometer* Z, *Magnetometer* X, *Magnetometer* Y, *Magnetometer* Z, GPS *Sattelite*, *Velocity North*, *Velocity East*, dan *Velocity Down*. Urgensi pengambilan data-data ini untuk mengetahui Sublocus dapat bekerja. Sembilan belas macam data tersebut sudah mewakili dalam perhitungan orientasi, posisi, kecepatan, dan sensor. Selain itu, penyimpanan 3 jenis data dari 12 data yaitu *Roll*, *Pitch*, *Heading* atau *Yaw*. Data ini berhasil didapat melalui perancangan sistem akuisisi, dan dapat disimpan dengan proses data *logging* menggunakan perangkat Arduino dan modul SD Card. Arduino berperan sebagai mikrokontroler atau pengendali sistem, dimana perangkat ini didukung oleh sebuah perangkat lunak yaitu Arduino IDE yang berfungsi untuk memrogram perintah. Pemrograman bahasa C digunakan dalam Arduino IDE sebagai perintah untuk proses *parsing* data dan penampil data ke serial monitor. RS 422 adalah standar komunikasi antarmuka yang berperan sebagai penghubung antara Sublocus dan Arduino. Sublocus memerlukan lima jalur data, sedangkan Arduino menyediakan tiga jalur data maka digunakan modul RS 422 untuk menghubungkan dua perangkat tersebut. Modul SD Card berperan sebagai media penyimpan data. Dibutuhkan perangkat penyimpanan tambahan dikarenakan Arduino tidak dapat menampung atau menyimpan data terlalu banyak. Dengan ini digunakan perangkat modul SD Card karena menyediakan kapasitas penyimpanan sebesar 16 *gigabyte*. Peneliti melakukan percobaan mengunggah *code* pada Arduino versi Uno, tetapi perangkat tersebut tidak dapat menerima kapasitas data terlalu banyak. Fenomena tersebut dikarenakan Arduino versi Uno memiliki kapasitas memori sebesar 32 KB (*Kilo Byte*), maka digunakan Arduino versi Mega. Mikrokontroler Arduino Mega memiliki kapasitas memori sebesar 256 KB, sehingga *code* dapat terunggah secara maksimal. Hasil data menunjukkan bahwa sensor Sublocus memiliki sensitivitas tinggi. Hal ini dibuktikan ketika proses pengujian Sublocus dalam kondisi guncangan, data menunjukkan pergerakan yang fluktuatif. Ketika pengujian Sublocus

dalam kondisi *push velocity*, dimana data *velocity* menunjukkan pergerakan dari waktu ke waktu. Pengujian ketika kondisi normal atau diam dan kondisi miring, Sublocus menunjukan hasil sesuai dengan sikapnya.

Terdapat pula saran dalam pengembangan penelitian ini yaitu mengenai analisis data Sublocus belum sepenuhnya dilaksanakan secara maksimal. Dengan demikian, diperlukan analisis lebih lanjut dengan cara membandingkan hasil data yang diperoleh dari perbedaan empat kondisi Sublocus. Hal ini berguna untuk mengetahui respon kerja dari waktu ke waktu saat keadaan kapal berbeda-beda. Selain itu ketika pengujian dilaksanakan, Sublocus tidak dikalibrasi. Proses kalibrasi dapat mempengaruhi hasil data, maka perlunya dilakukan kalibrasi terlebih dahulu. Untuk menunjang keakuratan data di bawah air, perlunya dilakukan pengujian di laut. Penelitian dianggap kurang maksimal jika pengujian respon Sublocus hanya dilakukan di darat, sedangkan INS akan digunakan pada kapal di permukaan maupun di dalam laut.

DAFTAR PUSTAKA

[1] WIKIPEDIA, “Kapal.” 15 Februari 2018, 2018.

[2] C. Kerja *et al.*, “Kapal selam.”

[3] D. Kuil, “The implementation of an underwater navigation system, applied to Ortega’s submersibles,” pp. 1–41.

[4] IEEE, “Inertial Navigation Systems Information.” [Online]. Available: https://www.globalspec.com/learnmore/sensors_transducers_detectors/tilt_sensing/inertial_gyros. [Accessed: 06-Mar-2018].

[5] N. Instruments, “What Is Data Acquisition?” [Online]. Available: http://www.ni.com/data-acquisition/what-is/. [Accessed: 07-Mar-2018].

[6] “Sublocus Datasheet.pdf.” .

[7] S. P. Ambildhok and N. B. Hulle, “Development of Inertial Navigation System based on Accelerometer and Gyroscope,” vol. 7, no. 5, pp. 12149–12151, 2017.

[8] M. Windows *et al.*, “Aurduino mega 2560,” *Uma ética para quantos?*, vol. XXXIII, no. 2, pp. 81–87, 2014.

[9] C. S. Misal and J. M. Conrad, “Designing a pH data acquisition and logging device using an inexpensive microcontroller,” *Conf. Proc. - IEEE SOUTHEASTCON*, pp. 217–220, 2007.

[10] S. Agrawal, V. Kumar, N. Anand, V. K. Agarwal, and A. Islam, “Development of data acquisition system and data analysis technique for automotive applications,” *2016 World Conf. Futur. Trends Res. Innov. Soc. Welf. (Startup Conclave)*, vol. 50, no. Ic, pp. 1–4, 2016.

[11] O. P. Harahap, M. Pd, and M. P. Fis, “Cara kerja kapal selam.”

[12] G. A. Ramadass, “ing Tool for Sensor Selectio on for Inertial A Modeli Navigatio on Systems used in Underw water Vehicles 2 . Methodology of Position,” pp. 175–188.

[13] N. Syahroni, W. P. Yuniar Riska, P. I. Metha, H. W. Suparno, H. Budiman, and C. J. Weon, “Data acquisition and processing of movement and position for AUVs with experiment results,” *ICITACEE 2015 - 2nd Int. Conf. Inf. Technol. Comput. Electr. Eng. Green Technol. Strength. Inf. Technol. Electr. Comput. Eng. Implementation, Proc.*, pp. 97–101, 2016.

[14] “Reference Manual,” *Read*, pp. 2–2.

[15] A. D. King, “Inertial navigation - Forty years of evolution,” *Gec*

Rev., vol. 13, no. 3, pp. 140–149, 1998.
[16] K. Bikonis and J. Demkowicz, “Data Integration from GPS and Inertial Navigation Systems for Pedestrians in Urban Area,” *TransNav, Int. J. Mar. Navig. Saf. Sea Transp.*, vol. 7, no. 3, pp. 401–406, 2013.
[17] H. Martin, “Overcoming the Challenges of Indoor Navigation,” 2016.
[18] O. Ahmad, “SISTEM DATA AKUISISI UNTAI UJI TERMOHIDROLIKA KECELAKAAN (" BETA "),” 2000.
[19] Bruxton Corporation, “Data Acquisition : An Introduction,” *Signals*, pp. 1–7, 1998.
[20] B.-B. Texas Instruments (Inc., “Principles of data acquisition and conversion,” *BURR-BROWN Appl. Bull.*, no. 602, pp. 1–6, 1994.
[21] C. Measurement, “and Control Handbook.”
[22] WIKIPEDIA, “Data acquisition,” 2018. [Online]. Available: https://en.wikipedia.org/wiki/Data_acquisition. [Accessed: 08-Mar-2018].
[23] Slideshare, “Data Logging,” *5 December 2012*, 2012. [Online]. Available: https://www.slideshare.net/KesavartiniiBalaKrisnain/data-logging-15496444. [Accessed: 13-Mar-2018].
[24] “Introduction to Data Logging Systems,” pp. 3–5.
[25] T. A. Jesha and M. T. Iqbal, “Thermal simulation and energy consumption analysis of two houses in St. John’s, Newfoundland,” *Procedia Eng.*, vol. 105, pp. 607–612, 2015.
[26] F. Foust, “Secure Digital Card Interface for the MSP430,” *Comput. Eng.*, p. 23, 2004.
[27] S. D. C. Interface, “3 . The Basics CTAN009 : An Introduction to,” 2008.
[28] E. Systems, “Lecture 12 : SPI and SD cards,” *Electrical*, no. March, pp. 1–12, 2013.
[29] M. Daun, J. Höfflinger, and T. Weyer, “Function-centered engineering of embedded systems: Evaluating industry needs and possible solutions,” *9th Int. Conf. Eval. Nov. Approaches to Softw. Eng. ENASE 2014*, pp. 226–234, 2014.
[30] R. Feynman, “Programming Embedded Systems , Second Edition with C and GNU Development Tools,” pp. 1–288, 2007.
[31] WIKIPEDIA, “Embedded System,” *2018*. [Online]. Available: https://en.wikipedia.org/wiki/Embedded_system. [Accessed: 13-

Mar-2018].
[32] Tutorialspoint, “Embedded Systems,” p. 57, 2015.
[33] E. E. Iit, E. Systems, and E. E. Iit, “Lesson,” pp. 1–12.
[34] WIKIPEDIA, “Arduino,” *11 December 2017*, 2017. [Online]. Available: https://id.wikipedia.org/wiki/Arduino. [Accessed: 13-Mar-2018].
[35] R. Panish and M. Taylor, “Achieving high navigation accuracy using inertial navigation systems in autonomous underwater vehicles,” *Ocean. 2011 IEEE - Spain*, pp. 1–7, 2011.
[36] A. Bejo, *C&AVR Rahasia Kemudahan Bahasa C dalam Mikrokontroler ATMega8535*. Yogyakarta: Graha Ilmu, 2008.
[37] EE Herald, “Serial Communication Course,” pp. 1–14, 2010.
[38] S. P. Interface, “Serial Communication Standards.”

Halaman ini sengaja dikosongkan

LAMPIRAN

A-3 Spesifikasi Navigasi

Parameter	Nilai
Akurasi Posisi Horizontal (GPS)	0.8 m
Akurasi Posisi Horizontal (DVL)	0.08 % jarak tempuh
Akurasi Posisi Horizontal (Log)	0.4 % jarak tempuh
Akurasi Kedalaman	0.4 m
Akurasi *Roll* & *Pitch*	0.01 °
Akurasi *Heading*	0.05 ° garis lintang
Rentang Orientasi	Tak terbatas
Waktu *North Seeking*	< 60 s
Tingkat Filter Internal	1000 Hz
***Output* Data Rate**	Hingga 1000 Hz

A-4 Spesifikasi Sensor

Parameter	*Accelerometer*	*Gyroscope*	*Magnetometer*	*Pressure*
Jarak	10 g	490 °/s	8 G	400 bar
Kerapatan *Noise*	300 ug/√Hz	0.0002 °/s/√Hz	210 uG/√Hz	-
Non linearitas	< 0.03 %	< 0.005 %	< 0.05 %	-
Stabilitas Bias	50 ug	0.05 °/hr	-	-
Parameter	***Accelerometer***	***Gyroscope***	***Magnetometer***	***Pressure***
Stabilitas Faktor Skala	< 0.06 %	< 0.02 %	< 0.05 %	-
Kesalahan *Cross-axis Alignment*	< 0.05 °	< 0.02 °	0.05 °	-
Bandwidth	200 Hz	440 Hz	110 Hz	10 Hz

A-3 Spesifikasi Komunikasi

Parameter	Value
Interface	RS232 atau RS422
Kecepatan	1200 hingga 10M baud
Protokol	AN *Packet Protocol*
Peripheral Interface	2x GPIO dan 1x *Auxiliary* RS232
Tingkat GPIO	5 V atau RS232

A-4 Spesifikasi GPS

Parameter	Nilai
Sistem Navigasi yang didukung	GPS L1 GLONASS L1 GALILEO E1 BeiDou B1
Sistem SBAS yang didukung	WAAS EGNOS MSAS GAGAN QZSS
Update Rate	10 Hz
Hot Start First Fix	3 s
Cold Start First Fix	30 s
Akurasi Posisi Horizontal	1.2 m
Akurasi Posisi Horizontal (dengan SBAS)	0.5 m
Akurasi Kecepatan	0.03 m/s
Ketepatan waktu	25 ns

A-5. Spesifikasi Arduino Mega

Parameter	Nilai
Mikrokontroler	ATmega 2560
Tegangan Operasi	5V
Tegangan *Input*	7-12V
Tegangan *Input* (batas)	6-20V
Pin I/O Digital	54 (*output* PWM disediakan 14)
DD	16
Arus DC per Pin I/O	40 mA
Arus DC untuk 3.3V Pin	50 mA
Memori *Flash*	256 KB (8 KB digunakan oleh *bootloader*)
SRAM	SRAM 8 KB
EEPROM	4 KB
Kecepatan *Clock*	16 MHz

A-6. Alokasi Pin Sublocus

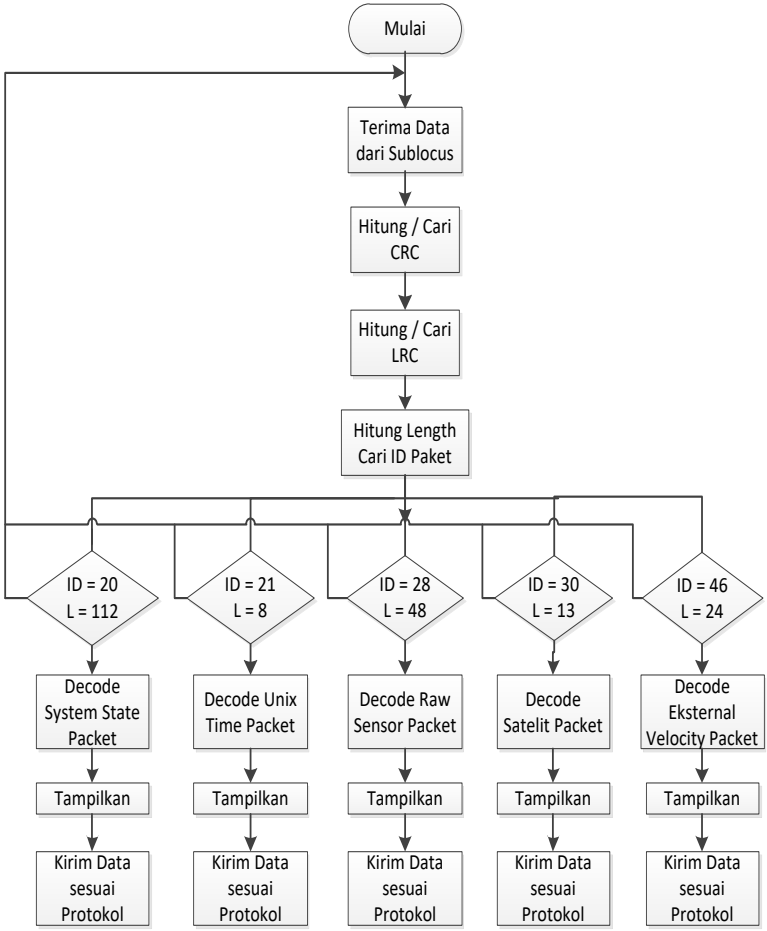
Pin	Function
1	Sumber Daya listrik
2	Daya *Ground*
3	Sinyal *Ground*
4	COM1 RS422 Tx(+) / COM1 RS232 Tx
5	COM1 RS422 Tx(-)
6	COM1 RS422 Rx(+) / COM1 RS232 Rx
7	COM1 RS422 Rx(-)
8	COM2 RS232 Tx
9	COM2 RS232 Rx
10	GPIO 1
11	GPIO 2
12	NC

A-7 *Advanced Navigation Packet Protocol*

System State Packet				
***Packet* ID**		20		
Length		112		
Field #	*Bytes Offset*	Tipe Data	Ukuran	Deskripsi
1	0	u16	2	*System status*
2	2	u16	2	*Filter status*
3	4	u32	4	*Unix time seconds*
4	8	u32	4	*Microseconds*
5	12	fp64	8	*Latitude* (rad)
6	20	fp64	8	*Longitude* (rad)
7	28	fp64	8	WGS84 *Height* (m)
8	36	fp32	4	*Velocity north* (m/s)
9	40	fp32	4	*Velocity east* (m/s)
10	44	fp32	4	*Velocity down* (m/s)
11	48	fp32	4	*Body acceleration* X (m/s/s)
12	52	fp32	4	*Body acceleration* Y (m/s/s)
13	56	fp32	4	*Body acceleration* Z (m/s/s)
14	60	fp32	4	G *force* (g)
15	64	fp32	4	*Roll* (radians)
16	68	fp32	4	*Pitch* (radians)
17	72	fp32	4	*Heading* (radians)
18	76	fp32	4	*Angular velocity* X (rad/s)
19	80	fp32	4	*Angular velocity* Y (rad/s)
20	84	fp32	4	*Angular velocity* Z (rad/s)
21	88	fp32	4	*Latitude standard deviation* (m)
22	92	fp32	4	*Longitude standard deviation* (m)
23	96	fp32	4	*Height standard deviation* (m)
24	100	fp32	4	*Depth* (m)
25	104	fp32	4	*Altitude* (m)
26	108	fp32	4	*Heave* (m)

Unix Time Packet				
***Packet* ID**		21		
Length		8		
Field #	*Bytes Offset*	Tipe Data	Ukuran	Deskripsi
1	0	u32	4	*Unix time seconds*
2	4	u32	4	*Microseconds*
Raw Sensors Packet				
Packet ID		28		
Length		48		
Field #	*Bytes Offset*	Tipe Data	Ukuran	Deskripsi
1	0	fp32	4	*Accelerometer* X (m/s/s)
2	4	fp32	4	*Accelerometer* Y (m/s/s)
3	8	fp32	4	*Accelerometer* Z (m/s/s)
4	12	fp32	4	*Gyroscope* X (rad/s)
5	16	fp32	4	*Gyroscope* Y (rad/s)
6	20	fp32	4	*Gyroscope* Z (rad/s)
7	24	fp32	4	*Magnetometer* X (mG)
8	28	fp32	4	*Magnetometer* Y (mG)
9	32	fp32	4	*Magnetometer* Z (mG)
10	36	fp32	4	IMU *Temperature* (deg C)
11	40	fp32	4	*Pressure* (Pascals)
12	44	fp32	4	*Pressure Temperature* (deg C)
Satellites Packet				
Packet ID		30		
Length		13		
Field #	*Bytes Offset*	Tipe Data	Ukuran	Deskripsi
1	0	fp32	4	HDOP
2	4	fp32	4	VDOP
3	8	u8	1	GPS *satellites*
4	9	u8	1	GLONASS *satellites*
5	10	u8	1	BeiDou *satellites*
6	11	u8	1	GALILEO *satellites*
7	12	u8	1	SBAS *satellites*
External Velocity Packet				
Packet ID		46		
Length		24		
Field #	*Bytes Offset*	Tipe Data	Ukuran	Deskripsi
1	0	fp32	4	*Velocity north* (m/s)
2	4	fp32	4	*Velocity east* (m/s)
3	8	fp32	4	*Velocity down* (m/s)
4	12	fp32	4	*Velocity north standard deviation* (m/s)
5	16	fp32	4	*Velocity east standard deviation* (m/s)
6	20	fp32	4	*Velocity down standard deviation* (m/s)

A-8 *Flowchart* Pemanggilan Paket



A-9 *Listing* Program Akuisisi Sublocus

```
#include <stdlib.h>
#include <stdint.h>
#include <string.h>
#include <math.h>
#include <stdio.h>
```

```
#define RADIANS_TO_DEGREES (180.0/M_PI)
#ifdef __cplusplus
extern "C" {
#endif
#define AN_PACKET_HEADER_SIZE 5
#define AN_MAXIMUM_PACKET_SIZE 255
#define AN_DECODE_BUFFER_SIZE
2*(AN_MAXIMUM_PACKET_SIZE+AN_PACKET_HEADER_SIZE)
#define an_packet_pointer(packet) packet->header
#define an_packet_size(packet) (packet->length +
AN_PACKET_HEADER_SIZE)*sizeof(uint8_t)
#define an_packet_crc(packet) ((packet->header[4]<<8) |
packet->header[3])
#define an_decoder_pointer(an_decoder) &(an_decoder)-
>buffer[(an_decoder)->buffer_length]
#define an_decoder_size(an_decoder) (sizeof((an_decoder)-
>buffer) - (an_decoder)->buffer_length)
#define an_decoder_increment(an_decoder, bytes_received)
(an_decoder)->buffer_length += bytes_received
#ifndef FALSE
#define FALSE 0
#define TRUE 1
#endif
typedef struct
{
  uint8_t buffer[AN_DECODE_BUFFER_SIZE];
  uint16_t buffer_length;
  uint32_t crc_errors;
} an_decoder_t;
typedef struct
{
    uint8_t id;
    uint8_t length;
    uint8_t header[AN_PACKET_HEADER_SIZE];
    uint8_t data[1];
} an_packet_t;
an_packet_t *an_packet_allocate(uint8_t length, uint8_t id);
void an_packet_free(an_packet_t **an_packet);
void an_decoder_initialise(an_decoder_t *an_decoder);
an_packet_t *an_packet_decode(an_decoder_t *an_decoder);
void an_packet_encode(an_packet_t *an_packet);
//------------------------------------------------------------
uint16_t calculate_crc16(const void *data, uint16_t length)
{
  uint8_t *bytes = (uint8_t *) data;
  uint16_t crc = 0xFFFF, i;
  for (i = 0; i < length; i++)
  {
    crc = (uint16_t)((crc << 8) ^ crc16_table[(crc >> 8) ^
bytes[i]]);
```

```
  }
  return crc;
}
//-------------------------------------------------------------
uint8_t calculate_header_lrc(uint8_t *data)
{
  return ((data[0] + data[1] + data[2] + data[3]) ^ 0xFF) +
1;
}

//-------------------------------------------------------------
an_packet_t *an_packet_allocate(uint8_t length, uint8_t id)
{
  an_packet_t *an_packet = (an_packet_t *)
malloc(sizeof(an_packet_t) + length * sizeof(uint8_t));
  if (an_packet != NULL)
  {
    an_packet->id = id;
    an_packet->length = length;
  }
  return an_packet;
}
//-------------------------------------------------------------
an_packet_t *an_packet_decode(an_decoder_t *an_decoder)
{
  uint16_t decode_iterator = 0;
  an_packet_t *an_packet = NULL;
  uint8_t header_lrc, id, length;
  uint16_t crc;
  while (decode_iterator + AN_PACKET_HEADER_SIZE <=
an_decoder->buffer_length)
  {
    header_lrc = an_decoder->buffer[decode_iterator++];
    if (header_lrc == calculate_header_lrc(&an_decoder-
>buffer[decode_iterator]))
    {
      id = an_decoder->buffer[decode_iterator++];
      length = an_decoder->buffer[decode_iterator++];
      crc = an_decoder->buffer[decode_iterator++];
      crc |= an_decoder->buffer[decode_iterator++] << 8;
      if (decode_iterator + length > an_decoder-
>buffer_length)
      {
        decode_iterator -= AN_PACKET_HEADER_SIZE;
        break;
      }
      if (crc == calculate_crc16(&an_decoder-
>buffer[decode_iterator], length))
      {
        an_packet = an_packet_allocate(length, id);
```

```
        if (an_packet != NULL)
        {
          memcpy(an_packet->header, &an_decoder-
>buffer[decode_iterator - AN_PACKET_HEADER_SIZE],
AN_PACKET_HEADER_SIZE * sizeof(uint8_t));
          memcpy(an_packet->data, &an_decoder-
>buffer[decode_iterator], length * sizeof(uint8_t));
        }
        decode_iterator += length;
        break;
      }
      else
      {
        decode_iterator -= (AN_PACKET_HEADER_SIZE - 1);
        an_decoder->crc_errors++;
      }
    }
  }
  if (decode_iterator < an_decoder->buffer_length)
  {
    if (decode_iterator > 0)
    {
      memmove(&an_decoder->buffer[0], &an_decoder-
>buffer[decode_iterator], (an_decoder->buffer_length -
decode_iterator) * sizeof(uint8_t));
      an_decoder->buffer_length -= decode_iterator;
    }
  }
  else an_decoder->buffer_length = 0;
  return an_packet;
}
//-------------------------------------------------------------
void an_packet_encode(an_packet_t *an_packet)
{
  uint16_t crc;
  an_packet->header[1] = an_packet->id;
  an_packet->header[2] = an_packet->length;
  crc = calculate_crc16(an_packet->data, an_packet->length);
  memcpy(&an_packet->header[3], &crc, sizeof(uint16_t));
  an_packet->header[0] = calculate_header_lrc(&an_packet-
>header[1]);
}
//---------------------------- BATAS ------------------------
#define MAXIMUM_PACKET_PERIODS 50
#define MAXIMUM_DETAILED_SATELLITES 32
#define MAXIMUM_DEPTH_CELLS 15
#define START_SYSTEM_PACKETS 0
#define START_STATE_PACKETS 20
#define START_CONFIGURATION_PACKETS 180
typedef enum
```

```
{

  packet_id_unix_time,
  packet_id_raw_sensors,
  packet_id_satellites,
  packet_id_external_velocity,
  packet_id_state_packet,

//------------------------------------------------------------
 *system_state_packet, an_packet_t *an_packet);
int decode_unix_time_packet(unix_time_packet_t
*unix_time_packet, an_packet_t *an_packet);
int decode_formatted_time_packet(formatted_time_packet_t
*raw_sensors_packet, an_packet_t *an_packet);
int decode_raw_gnss_packet(raw_gnss_packet_t
*satellites_packet, an_packet_t *an_packet);
int
decode_detailed_satellites_packet(detailed_satellites_packet
_t *detailed_satellites_packet, an_packet_t *an_packet);
int
decode_geodetic_position_packet(geodetic_position_packet_t
*external_velocity_packet);
int
decode_external_body_velocity_packet(external_body_velocity_
packet_t *external_body_velocity_packet, an_packet_t

int decode_unix_time_packet(unix_time_packet_t
*unix_time_packet, an_packet_t *an_packet)
{
  if(an_packet->id == packet_id_unix_time && an_packet-
>length == 8)
  {
    memcpy(&unix_time_packet->unix_time_seconds, &an_packet-
>data[0], sizeof(uint32_t));
    memcpy(&unix_time_packet->microseconds, &an_packet-
>data[4], sizeof(uint32_t));
    return 0;
  }
  else return 1;
}

int decode_raw_sensors_packet(raw_sensors_packet_t
*raw_sensors_packet, an_packet_t *an_packet)
{
  if(an_packet->id == packet_id_raw_sensors && an_packet-
>length == 48)
  {
    memcpy(&raw_sensors_packet->accelerometers[0],
&an_packet->data[0], 3 * sizeof(float));
```

```
    memcpy(&raw_sensors_packet->gyroscopes[0], &an_packet-
>data[12], 3 * sizeof(float));
    memcpy(&raw_sensors_packet->magnetometers[0],
&an_packet->data[24], 3 * sizeof(float));
    memcpy(&raw_sensors_packet->imu_temperature, &an_packet-
>data[36], sizeof(float));
    memcpy(&raw_sensors_packet->pressure, &an_packet-
>data[40], sizeof(float));
    memcpy(&raw_sensors_packet->pressure_temperature,
&an_packet->data[44], sizeof(float));
    return 0;
  }
  else return 1;
}

int decode_satellites_packet(satellites_packet_t
*satellites_packet, an_packet_t *an_packet)
{
  if(an_packet->id == packet_id_satellites && an_packet-
>length == 13)
  {
    memcpy(&satellites_packet->hdop, &an_packet->data[0],
sizeof(float));
    memcpy(&satellites_packet->vdop, &an_packet->data[4],
sizeof(float));
    memcpy(&satellites_packet->gps_satellites, &an_packet-
>data[8], 5*sizeof(uint8_t));
    return 0;
  }
  else return 1;
}

an_packet_t
*encode_external_velocity_packet(external_velocity_packet_t
*external_velocity_packet)
{
  an_packet_t *an_packet = an_packet_allocate(24,
packet_id_external_velocity);
  if(an_packet != NULL)
  {
    memcpy(&an_packet->data[0], &external_velocity_packet-
>velocity[0], 3*sizeof(float));
    memcpy(&an_packet->data[12], &external_velocity_packet-
>standard_deviation[0], 3*sizeof(float));
  }
  return an_packet;
}

//------------------------------------------------------------
void set_sensor_ranges()
```

```
{
  an_packet_t *an_packet;
  sensor_ranges_packet_t sensor_ranges_packet;
  sensor_ranges_packet.permanent = TRUE;
  sensor_ranges_packet.accelerometers_range =
accelerometer_range_4g;
  sensor_ranges_packet.gyroscopes_range =
gyroscope_range_500dps;
  sensor_ranges_packet.magnetometers_range =
magnetometer_range_2g;
  an_packet =
encode_sensor_ranges_packet(&sensor_ranges_packet);
  an_packet_transmit(an_packet);
  an_packet_free(&an_packet);
 }
 //--------------------------------------------------------
void setup()
{
  Serial.begin(115200);
  Serial1.begin(115200);
  Serial2.begin(9600);
  Serial3.begin(115200);
//--------------------------------------------------------
set_sensor_ranges();
an_decoder_t an_decoder;
an_decoder_initialise(&an_decoder);
//--------------------------------------------------------
  }
int bytes_received;
char buff[100], buff2[100];
double DataKirim[100];
void loop()
{
  an_decoder_t an_decoder;
  an_packet_t *an_packet;
  system_state_packet_t system_state_packet;
  raw_sensors_packet_t raw_sensors_packet;
  unix_time_packet_t unix_time_packet;
  satellites_packet_t satellites_packet;
  external_velocity_packet_t external_velocity;
  an_decoder_initialise(&an_decoder);
  while ((bytes_received = Serial3.readBytes(
an_decoder_pointer(&an_decoder),
an_decoder_size(&an_decoder))) > 0)
    {
       an_decoder_increment(&an_decoder, bytes_received);
       an_packet = an_packet_decode(&an_decoder);
//---------------------- ANPAcket 20 System State Packet
        if (an_packet->id == packet_id_system_state) /*
system state packet */
```

```
        {
         if(decode_system_state_packet(&system_state_packet,
an_packet) == 0)
          {
            DataKirim[0] =
system_state_packet.orientation[0]*RADIANS_TO_DEGREES; //
Roll
            DataKirim[1] =
system_state_packet.orientation[1]*RADIANS_TO_DEGREES; //
Pitch
            DataKirim[2] =
system_state_packet.orientation[2]*RADIANS_TO_DEGREES; //
Yaw
            DataKirim[3] =
system_state_packet.latitude*RADIANS_TO_DEGREES;       //
Latitude
            DataKirim[4] =
system_state_packet.longitude*RADIANS_TO_DEGREES;      //
Longitude
          }
        }
//------------------ ANPacket 21 Unix Time Packet
        if(an_packet->id == packet_id_unix_time)
        {
           if(decode_unix_time_packet(&unix_time_packet,
an_packet) == 0)
            {
              DataKirim[5] = unix_time_packet.microseconds;
              Serial.println(unix_time_packet.microseconds);
              }
          }
//---------------------- ANPacket 28 Raw Sensor Packet
        if (an_packet->id == packet_id_raw_sensors) /* raw
sensors packet */
        {
          if(decode_raw_sensors_packet(&raw_sensors_packet,
an_packet) == 0)
          {

            DataKirim[6] = raw_sensors_packet.gyroscopes[0]
* RADIANS_TO_DEGREES;
            DataKirim[7] = raw_sensors_packet.gyroscopes[1]
* RADIANS_TO_DEGREES;
            DataKirim[8] = raw_sensors_packet.gyroscopes[2]
* RADIANS_TO_DEGREES;

            DataKirim[9]  =
raw_sensors_packet.accelerometers[0] * RADIANS_TO_DEGREES;
            DataKirim[10] =
raw_sensors_packet.accelerometers[1] * RADIANS_TO_DEGREES;
```

```
            DataKirim[11] =
raw_sensors_packet.accelerometers[2] * RADIANS_TO_DEGREES;

            DataKirim[12] =
raw_sensors_packet.magnetometers[0] * RADIANS_TO_DEGREES;
            DataKirim[13] =
raw_sensors_packet.magnetometers[1] * RADIANS_TO_DEGREES;
            DataKirim[14] =
raw_sensors_packet.magnetometers[2] * RADIANS_TO_DEGREES;
          }
        }
//---------------------------- ANPacket 30 Sattelite Packet

 if (an_packet->id == packet_id_satellites) /* raw sensors
packet */
        {
          if(decode_satellites_packet(&satellites_packet,
an_packet) == 0)
          {
           DataKirim[15]= satellites_packet.gps_satellites;
          }
        }
//---------- ANPacket 46 External Packet
        if (an_packet->id == packet_id_external_velocity) /*
raw sensors packet */
        {
if(decode_external_velocity_packet(&external_velocity,
an_packet) == 0)
          {
           DataKirim[16]= external_velocity.velocity[0];
           DataKirim[17]= external_velocity.velocity[1];
           DataKirim[18]= external_velocity.velocity[2];
          }
        }
        else
        {
        }
        an_packet_free(&an_packet);
      }
      for(int i=0; i<19; i++)
      {
        Serial2.print(DataKirim[i]); Serial2.print("\t");
        }
        Serial2.print("\n");
    delay(1000);
  }
#ifdef __cplusplus }
#endif
```

Halaman ini sengaja dikosongkan

RIWAYAT PENULIS

Varisa Rahmawati lahir di Surabaya pada tanggal 10 April tahun 1997. Nama panggilan penulis adalah Varisa yang merupakan anak pertama dari Novan Julianto dan Prima Sari. Tinggal di Perumahan Diamond Park Residence B6/27 Jalan Raya Juanda, Sedati, Sidoarjo, sedangkan alamat Surabaya terletak di Jalan Kupang Praupan Pasar 1 No.5. Pada tahun 2003, penulis bersekolah di SDN Dr.Sutomo VIII Surabaya, dan di tahun 2009 melanjutkan ke jenjang Sekolah Menengah Pertama di SMPN 10 Surabaya. Setelah lulus SMP, kemudian tahun 2012 penulis belajar di bangku Sekolah Menengah Atas di SMAN 10 Surabaya. Penulis melanjutkan pendidikan perkuliahan pada tahun 2015 di Institut Teknologi Sepuluh Nopember Jurusan Teknik Elektro Otomasi. Dalam masa perkuliahan, penulis bergabung di Laboratorium Sistem Komputer dan Otomasi.

E-mail : varisarahmawati@gmail.com