



UNIVERSITAS DIPONEGORO

**PERANCANGAN *TELECONTROLLING* DAN
TELEMETERING PADA *GROUND CONTROL STATION*
UNTUK PURWARUPA *AUTONOMOUS SURFACE VEHICLE***

TUGAS AKHIR

IBRAHIM

21060116130099

**FAKULTAS TEKNIK
DEPARTEMEN TEKNIK ELEKTRO
PROGRAM STUDI SARJANA**

**SEMARANG
AGUSTUS 2020**



UNIVERSITAS DIPONEGORO

**PERANCANGAN *TELECONTROLLING* DAN
TELEMETERING PADA *GROUND CONTROL STATION*
UNTUK PURWARUPA *AUTONOMOUS SURFACE VEHICLE***

TUGAS AKHIR

Diajukan sebagai salah satu syarat untuk memperoleh gelar Sarjana Teknik

IBRAHIM

21060116130099

**FAKULTAS TEKNIK
DEPARTEMEN TEKNIK ELEKTRO
PROGRAM STUDI SARJANA**

**SEMARANG
AGUSTUS 2020**

HALAMAN PERNYATAAN ORISINALITAS

**Tugas Akhir ini adalah hasil karya saya sendiri,
dan semua sumber baik yang dikutip maupun yang dirujuk
telah saya nyatakan dengan benar.**

NAMA : Ibrahim
NIM : 21060116130099
Tanda Tangan :
Tanggal : 17 Agustus 2020

HALAMAN PENGESAHAN

Tugas Akhir ini diajukan oleh

NAMA : IBRAHIM
NIM : 21060116130099
Departemen/Program Studi : TEKNIK ELEKTRO / SARJANA (S1)
Judul Skripsi : PERANCANGAN *TELECONTROLLING* DAN
TELEMETERING PADA *GROUND CONTROL*
STATION UNTUK PURWARUPA
AUTONOMOUS SURFACE VEHICLE

Telah berhasil dipertahankan di hadapan Tim Penguji dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Sarjana Teknik pada Program Studi Sarjana, Departemen Teknik Elektro, Fakultas Teknik, Universitas Diponegoro.

TIM PENGUJI

Pembimbing I : Dr. Wahyudi, S.T., M.T. (.....)
Pembimbing II : Eko Handoyo, S.T., M.T. (.....)
Penguji I : Dr. Maman Somantri, S.T., M.T. (.....)
Penguji II : Achmad Hidayatno, S.T., M.T. (.....)

Semarang, 27 Agustus 2020
Ketua Departemen Teknik Elektro,

Dr. Wahyudi, S.T., M.T
NIP. 196906121994031001

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

Sebagai sivitas akademika Universitas Diponegoro, saya yang bertanda tangan di bawah ini :

Nama : IBRAHIM
NIM : 21060116130099
Program Studi : SARJANA (S1)
Departemen : TEKNIK ELEKTRO
Fakultas : TEKNIK
Jenis Karya : TUGAS AKHIR

demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Diponegoro **Hak Bebas Royalti Noneksklusif** (*None-exclusive Royalty Free Right*) atas karya ilmiah saya yang berjudul:

*PERANCANGAN TELECONTROLLING DAN TELEMETERING PADA
GROUND CONTROL STATION UNTUK PURWARUPA AUTONOMOUS
SURFACE VEHICLE*

beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti/Noneksklusif ini Universitas Diponegoro berhak menyimpan, mengalihmedia/formatkan, mengelola dalam bentuk pangkalan data (*database*), merawat dan memublikasikan Tugas Akhir saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Semarang
Pada Tanggal : 17 Agustus 2020

Yang menyatakan,

(Ibrahim)

ABSTRAK

Observasi perairan di Indonesia masih dilakukan secara manual menggunakan alat survei yang dioperasikan oleh manusia. Perairan beracun dapat berbahaya bagi kesehatan operator, sehingga diperlukan sebuah kendaraan tanpa awak yang dapat melakukan misi observasi. Ground Control Station (GCS) merupakan sebuah sistem yang digunakan untuk mengendalikan dan memantau sebuah kendaraan tanpa awak secara nirkabel. GCS dirancang agar mampu memberikan informasi dan kendali yang cukup, sehingga misi yang diberikan kepada Autonomous Surface Vehicle (ASV) dapat terlaksana secara autonomous dengan intervensi telecontrolling yang minimal dari operator. Tugas Akhir ini membahas mengenai perancangan GCS dalam mengoperasikan sebuah ASV. Microsoft Visual Studio Express 2012 digunakan dalam pengembangan perangkat lunak GCS. Modul telemetri radio 3DR 433Mhz digunakan sebagai media komunikasi antara GCS dan ASV. Berdasarkan pengujian, jarak optimal penggunaan modul radio 3DR 433Mhz untuk mengirim dan menerima data adalah 100 meter. Fitur telecontrolling pada GCS yang dirancang berupa parameter kecepatan, start/stop ASV, tuning PID, waypoint, dan failsafe. Fitur telemetering pada GCS yang dirancang berupa data latitude, longitude, heading, bearing, error hadap, jarak ASV ke waypoint selanjutnya, juga umpan balik PWM, state misi, nilai K_p , K_i , K_d , S_p , dan failsafe. Fitur lain pada GCS adalah menampilkan peta hybrid dan menyimpan data ASV dalam dokumen csv.

Kata Kunci: GCS, ASV, waypoint, telemetering, dan telecontrolling

ABSTRACT

Waters observation in Indonesia are still done manually using a survey tool operated by humans. Toxic waters can be hazardous to operator health, so it's required an autonomous surface vehicle to do that mission. Ground Control Station (GCS) is a system that used to control and monitoring an unmanned vehicle wirelessly. GCS is designed to provide sufficient information and control, so that the mission given to Autonomous Surface Vehicle (ASV) can be completed autonomously with minimal telecontrolling intervention from the operator. This final project is about design of GCS to operate an ASV. Microsoft Visual Studio Express 2012 is used to develop GCS software. The 3DR 433Mhz radio telemetry module is used as a communication medium between GCS and ASV. Based on the experiment, the optimal range of 3DR 433Mhz radio module is 100 meters for transmit and receive data. The telecontrolling features in the GCS that was designed are speed parameter, start/stop ASV, PID tuning, waypoint, and failsafe. The telemetering features in the GCS that was designed are latitude data, longitude data, heading data, bearing data, error heading data, distance to next waypoint, also PWM feedback, mission feedback, Kp, Ki, Kd, Sp value and failsafe. The other features in GCS that was designed are showing hybrid map and save the ASV data to csv file.

Keywords: *GCS, ASV, waypoint, telemetering, and telecontrolling*

KATA PENGANTAR

Alhamdulillah, segala puji dan syukur penulis panjatkan kepada Allah *Subhanahu wa Ta'ala* atas rahmat, karunia, taufiq, dan hidayah-Nya, sehingga penulis dapat menyelesaikan Tugas Akhir dan penyusunan laporan ini. Tugas Akhir dengan judul “*PERANCANGAN TELECONTROLLING DAN TELEMETERING PADA GROUND CONTROL STATION UNTUK PURWARUPA AUTONOMOUS SURFACE VEHICLE*” ini diajukan sebagai salah satu syarat akhir untuk menyelesaikan program Sarjana di Departemen Teknik Elektro, Fakultas Teknik, Universitas Diponegoro. Penyusunan dan penyelesaian Tugas Akhir ini tidak lepas dari dukungan berbagai pihak, baik secara langsung ataupun tidak langsung. Oleh karena itu, pada kesempatan ini penulis mengucapkan terimakasih kepada:

1. Bapak Dr. Wahyudi, S.T., M.T., selaku Ketua Departemen Teknik Elektro Fakultas Teknik, Universitas Diponegoro sekaligus Dosen Pembimbing I.
2. Bapak Yuli Christiyono, S.T., M.T., selaku Ketua Program Studi S1 Teknik Elektro, Fakultas Teknik, Universitas Diponegoro.
3. Bapak Eko Handoyo, S.T., M.T., selaku Dosen Pembimbing II.
4. Bapak M. Arfan, S.Kom., M.Eng., selaku Dosen Wali.
5. Kedua orang tua, adik-adik, dan seluruh keluarga besar penulis yang selalu mendoakan dan memotivasi penulis.
6. Tim Robotik Undip URDC yang memotivasi dan mendukung penulis dalam mengambil topik Tugas Akhir ini.
7. Alif Ihza Ahmada sebagai teman satu topik dalam menyelesaikan Tugas Akhir.
8. Keluarga besar Gugus Radikal Teknik Elektro Undip 2016, yang selalu memberi dukungan dan nasehat dalam pengerjaan Tugas Akhir.
9. Keluarga besar S1 Teknik Elektro Undip.
10. Semua pihak yang telah membantu dan mendoakan penulis selama proses pengerjaan Tugas Akhir yang tidak dapat disebutkan satu persatu.

Penulis menyadari masih terdapat kekurangan pada Tugas Akhir ini. Namun demikian, penulis berharap Tugas Akhir ini dapat bermanfaat bagi pembaca khususnya di bidang teknik kendali.

Semarang, 17 Agustus 2020

Penulis

DAFTAR ISI

HALAMAN JUDUL	i
HALAMAN PERNYATAAN.....	ii
HALAMAN PENGESAHAN	iii
HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI.....	iv
ABSTRAK	v
ABSTRACT	vi
KATA PENGANTAR.....	vii
DAFTAR ISI.....	ix
DAFTAR GAMBAR.....	xi
DAFTAR TABEL	xiii
BAB I PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Tujuan	2
1.3 Batasan Masalah	2
1.4 Sistematika Penulisan	2
BAB II DASAR TEORI.....	4
2.1 <i>Autonomous Surface Vehicle</i>	4
2.2 <i>Ground Control Station</i>	5
2.3 Microsoft Visual Studio	5
2.4 Arduino IDE	6
2.5 Modul Radio 3DR 433Mhz	7
2.6 Komunikasi Serial Asinkron	9
BAB III PERANCANGAN SISTEM.....	11
3.1 Deskripsi Sistem	11
3.2 Analisis Kebutuhan	11
3.3 Perancangan Perangkat Keras.....	12
3.3.1 Penggunaan Modul Radio 3DR 433Mhz.....	12
3.3.2 Penggunaan Laptop.....	13

3.4	Perancangan Perangkat Lunak	13
3.4.1	Fitur Koneksi GCS dengan ASV	15
3.4.2	<i>Telecontrolling</i> Kecepatan ASV	16
3.4.3	<i>Telecontrolling Start/Stop</i> ASV	17
3.4.4	<i>Telecontrolling Tuning</i> Parameter Kendali.....	18
3.4.5	Fitur Menampilkan Peta.....	20
3.4.6	<i>Telecontrolling Waypoint</i> ASV	21
3.4.7	<i>Telecontrolling Failsafe</i> ASV	26
3.4.8	<i>Telemetry</i> pada GCS	27
3.4.9	Fitur <i>Save Data</i>	30
3.4.10	Program Terima dan Kirim Data antara ASV dan GCS	32
BAB IV	PENGUJIAN DAN ANALISIS	38
4.1	Pengujian Perangkat Keras	38
4.2	Pengujian Perangkat Lunak	39
4.2.1	Pengujian Koneksi GCS dan ASV	39
4.2.2	Pengujian <i>Telecontrolling</i> Kecepatan ASV	40
4.2.3	Pengujian <i>Telecontrolling Start/Stop</i> ASV	42
4.2.4	Pengujian <i>Telecontrolling Tuning</i> Parameter Kendali.....	43
4.2.5	Pengujian <i>Telecontrolling Waypoint</i> ASV	43
4.2.6	Pengujian <i>Telecontrolling Failsafe</i> ASV	45
4.2.7	Pengujian <i>Telemetry</i> pada GCS	46
4.2.8	Pengujian <i>Save Data</i>	47
BAB V	PENUTUP	49
5.1	Kesimpulan	49
5.2	Saran	49
DAFTAR PUSTAKA	50
BIODATA	52
LAMPIRAN	53

DAFTAR GAMBAR

Gambar 2.1	Contoh kapal tanpa awak	4
Gambar 2.2	Diagram representatif fitur GCS.....	5
Gambar 2.3	Tampilan Visual Studio Express 2012.....	6
Gambar 2.4	Tampilan Arduino IDE	7
Gambar 2.5	Pola radiasi antena <i>directional</i>	8
Gambar 2.6	Pola radiasi antena <i>omnidirectional</i>	8
Gambar 2.7	Modul radio 3DR 433Mhz	9
Gambar 2.8	Bit data serial asinkron	9
Gambar 3.1	Konfigurasi perangkat keras	12
Gambar 3.2	Penggunaan modul radio 3DR 433Mhz.....	12
Gambar 3.3	Penggunaan laptop	13
Gambar 3.4	Tampilan GCS	14
Gambar 3.5	Fitur GCS.....	14
Gambar 3.6	Tampilan koneksi GCS	15
Gambar 3.7	Tampilan <i>telecontrolling</i> kecepatan ASV.....	16
Gambar 3.8	Tampilan umpan balik <i>telecontrolling</i> kecepatan ASV.....	17
Gambar 3.9	Tampilan <i>telecontrolling start/stop</i> ASV	17
Gambar 3.10	Tampilan umpan balik <i>telecontrolling start/stop</i> ASV.....	18
Gambar 3.11	Tampilan <i>telecontrolling tuning</i> parameter kendali.....	18
Gambar 3.12	Tampilan umpan balik <i>telecontrolling tuning</i> parameter kendali	19
Gambar 3.13	Tampilan peta GCS.....	20
Gambar 3.14	Tampilan <i>telecontrolling waypoint</i> pada GCS.....	21
Gambar 3.15	Tampilan <i>telecontrolling failsafe</i>	26
Gambar 3.16	Tampilan umpan balik <i>telecontrolling failsafe</i>	27
Gambar 3.17	Tampilan <i>telemetry</i> pada GCS.....	27
Gambar 3.18	Tampilan <i>save data</i>	30
Gambar 4.1	Pengujian jangkauan sinyal modul radio 3DR 433Mhz	38
Gambar 4.2	Tampilan pengujian GCS.....	39

Gambar 4.3 Pengujian tombol “CONNECT HERE”	40
Gambar 4.4 Pengujian tombol “CLOSE”	40
Gambar 4.5 Pengujian <i>telecontrolling</i> kecepatan ASV variasi 1	40
Gambar 4.6 Pengujian <i>telecontrolling</i> kecepatan ASV variasi 2	41
Gambar 4.7 Pengujian <i>telecontrolling</i> kecepatan ASV variasi 3	41
Gambar 4.8 Pengujian umpan balik <i>telecontrolling</i> kecepatan ASV variasi 1	41
Gambar 4.9 Pengujian umpan balik <i>telecontrolling</i> kecepatan ASV variasi 2	41
Gambar 4.10 Pengujian umpan balik <i>telecontrolling</i> kecepatan ASV variasi 3	42
Gambar 4.11 Pengujian <i>telecontrolling start</i> ASV	42
Gambar 4.12 Pengujian <i>telecontrolling stop</i> ASV	42
Gambar 4.13 Pengujian <i>telecontrolling tuning</i> parameter kendali variasi 1	43
Gambar 4.14 Pengujian <i>telecontrolling tuning</i> parameter kendali variasi 2	43
Gambar 4.15 Pengujian <i>telecontrolling tuning</i> parameter kendali variasi 3	43
Gambar 4.16 Pengujian <i>telecontrolling waypoint</i> GCS variasi 1	44
Gambar 4.17 Pengujian <i>telecontrolling waypoint</i> GCS variasi 2	44
Gambar 4.18 Pengujian <i>telecontrolling waypoint</i> GCS variasi 3	45
Gambar 4.19 Dokumen Data_WP.csv	45
Gambar 4.20 Pengujian <i>telecontrolling failsafe</i>	46
Gambar 4.21 Pengujian ke-1 <i>telemetry</i> pada GCS	46
Gambar 4.22 Pengujian ke-2 <i>telemetry</i> pada GCS	46
Gambar 4.23 Pengujian ke-3 <i>telemetry</i> pada GCS	47
Gambar 4.24 Tampilan tabel data ASV	47
Gambar 4.25 Dokumen Data_ASV.csv	48
Gambar 4.26 Pengujian <i>clear data</i> ASV	48

DAFTAR TABEL

Tabel 3.1 Data <i>telemetry</i>	29
Tabel 3.2 Paket data yang dikirim GCS.....	33
Tabel 4.1 Pengujian jangkauan sinyal modul radio 3DR 433Mhz	38

BAB I

PENDAHULUAN

1.1 Latar Belakang

Indonesia merupakan negara yang memiliki wilayah perairan lebih luas dari daratannya dengan luas sekitar 5,9 juta km² [1]. Seiring perkembangan waktu, sektor maritim Indonesia sudah semestinya ikut berkembang, namun pada kenyataannya Indonesia belum memanfaatkan secara maksimal sumber daya kelautan yang dimilikinya [2]. Disisi lain, eksploitasi sumber daya secara berlebihan tanpa memperhatikan lingkungan dapat berdampak buruk bagi kelanjutan ekosistem perairan, begitu juga dengan tingginya pertumbuhan penduduk. Dibutuhkan suatu moda transportasi atau sebuah alat yang mampu menjaga dan mengeksplorasi perairan Indonesia dengan tetap memperhatikan dampak terhadap lingkungan.

Autonomous Surface Vehicle (ASV) telah dikembangkan oleh beberapa negara untuk menunjang aktifitas penelitian wilayah perairan. Perkembangan ASV di dunia sudah pesat. Saat ini, survei batimetri dan oseanografi dapat dilakukan dengan menggunakan ASV, seperti Delfim, Sesamo, IRIS, SCOUT, dan ROAZ. USV memiliki manfaat yang cukup besar dalam observasi perairan. Perkembangan ini belum diikuti dengan baik di Indonesia, sehingga perlu dilakukan pengembangan lebih lanjut pada teknologi ASV [3].

Menurut observatorium oseanografi pengukuran variabel fisika, kimia dan biologis di perairan pantai secara *realtime* dapat membantu pengambilan keputusan dalam pengelolaan seperti perubahan iklim, bencana alam dan kondisi ekosistem [4]. Dalam pengembangannya, ASV tidak hanya untuk observasi tetapi juga digunakan sebagai moda transportasi air [5]. Untuk mengoperasikan sebuah kendaraan *autonomous* seperti ASV, digunakan sebuah *Ground Control Station* (GCS) yang memudahkan operator untuk melakukan pengaturan dan *monitoring* [6]. Dalam Tugas Akhir ini dirancang sebuah GCS yang memiliki beberapa fitur mendasar yaitu *telemetry* dan *telecontrolling* yang diperlukan agar sebuah ASV

dapat menyelesaikan misi yang diberikan. GCS dirancang dengan sebuah perangkat komputer dan menggunakan modul radio 3DR 433Mhz untuk berkomunikasi dengan ASV.

1.2 Tujuan

Tugas Akhir ini bertujuan untuk merancang sebuah sistem *telecontrolling* dan *telemetry* pada GCS yang digunakan untuk mengoperasikan sebuah purwarupa ASV.

1.3 Batasan Masalah

Dalam penyusunan Tugas Akhir ini, telah ditentukan batasan-batasan masalah sebagai berikut:

1. GCS dirancang dengan sebuah perangkat komputer.
2. Komunikasi dilakukan secara *wireless* menggunakan protokol serial asinkron.
3. Pengujian dilakukan pada area *outdoor* dengan kondisi cerah.
4. Perancangan perangkat lunak GCS pada laptop menggunakan bahasa pemrograman C# dan perangkat lunak komunikasi data pada ASV menggunakan bahasa pemrograman C.

1.4 Sistematika Penulisan

Sistematika penulisan laporan dalam Tugas Akhir ini adalah sebagai berikut:

BAB I PENDAHULUAN

Bab ini berisi latar belakang, tujuan, batasan masalah, dan sistematika penulisan.

BAB II DASAR TEORI

Bab ini berisi teori yang melandasi Tugas Akhir tentang ASV, GCS, Microsoft Visual Studio, Arduino IDE, modul radio 3DR 433Mhz, dan komunikasi serial asinkron.

BAB III PERANCANGAN SISTEM

Bab ini berisi tentang perancangan perangkat keras dan perangkat lunak dari sistem GCS.

BAB IV PENGUJIAN DAN ANALISIS

Bab ini berisi tentang pengujian perangkat keras dan perangkat lunak yang telah dibuat serta pengujian sistem secara keseluruhan.

BAB V PENUTUP

Bab ini berisi kesimpulan dan saran terhadap sistem GCS yang telah dibuat.

BAB II

DASAR TEORI

2.1 Autonomous Surface Vehicle

ASV merupakan sebuah kapal tanpa awak yang mampu menyusuri perairan secara otomatis. ASV mampu bergerak di permukaan air secara otomatis dari suatu lokasi ke lokasi lain dengan bantuan sebuah sistem navigasi berupa *waypoint* di mana titik-titiknya telah ditentukan sebelumnya [7]. Prinsip kerja *waypoint* adalah dengan membuat titik-titik lokasi tujuan yang akan dilalui ASV, kemudian ASV dapat mengikuti jalur yang telah terbentuk menuju titik lokasi yang telah ditentukan tersebut. Contoh kapal tanpa awak seperti pada Gambar 2.1.

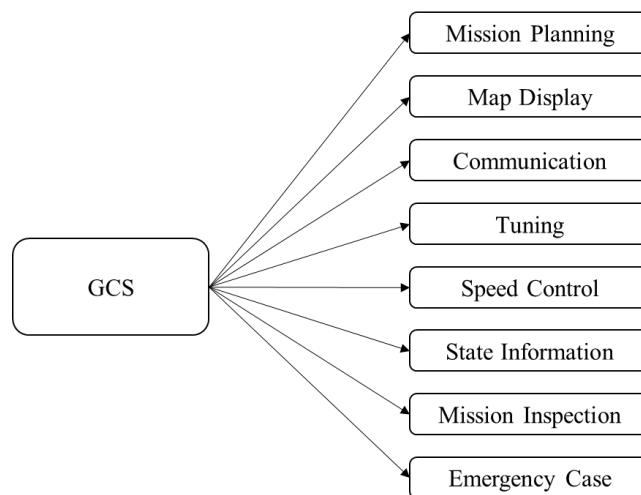


Gambar 2.1 Contoh kapal tanpa awak.

Pada umumnya ASV dilengkapi dengan GPS, kompas, telemetri, serta sensor pengukur parameter lingkungan perairan bila dibutuhkan [8]. Ketika lokasi telah ditentukan, kapal akan bergerak otomatis dan menjalankan misi yang diberikan. Data ASV akan dikirimkan secara *realtime* ke GCS. Operator dapat melakukan intervensi ketika ASV akan memulai misi ataupun keadaan darurat.

2.2 Ground Control Station

GCS adalah sebuah sistem yang digunakan untuk mengendalikan dan memantau sebuah kendaraan tanpa awak secara nirkabel [9]. GCS dapat berupa sistem pada perangkat komputer atau perangkat *mobile* seperti *smartphone* [6]. GCS diperlukan dalam pengoperasian sebuah ASV untuk menjalankan misi. GCS dirancang agar mampu memberikan informasi dan kendali yang cukup, sehingga misi yang diberikan kepada ASV dapat terlaksana secara *autonomous* dengan intervensi yang minimal dari operator [9]. Gambar 2.2 merupakan diagram representatif fitur yang terdapat pada GCS.



Gambar 2.2 Diagram representatif fitur GCS.

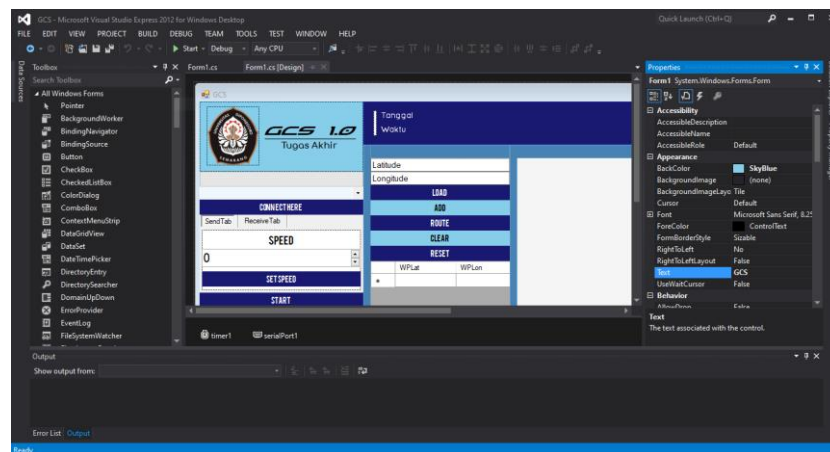
Dalam mengembangkan sebuah GCS untuk sebuah ASV banyak faktor yang harus dipertimbangkan, mulai dari tata letak, skema tampilan, metode komunikasi dan metode pengoperasiannya. Pemilihan warna tampilan HMI pada GCS juga perlu diperhatikan agar sesuai dengan standar yang ada [10].

2.3 Microsoft Visual Studio

Microsoft Visual Studio merupakan sebuah perangkat lunak lengkap yang digunakan untuk pengembangan aplikasi komputer, web, ataupun *android*. Sebagai sebuah *Integrated Development Environment* (IDE), Microsoft Visual Studio menyediakan berbagai fasilitas yang dibutuhkan untuk membangun perangkat

lunak. Pada umumnya, IDE memiliki *source code editor*, *built automation tools* dan *debugger*. IDE dirancang untuk meminimalkan *error* dengan adanya *error correction* dan *execution monitoring* [11].

Tugas Akhir ini menggunakan Microsoft Visual Studio Express 2012 for Windows Desktop untuk membangun perangkat lunak GCS pada komputer. Microsoft Visual Studio Express merupakan versi ringan dan gratis dari Visual Studio. Bahasa pemrograman yang digunakan adalah bahasa pemrograman C#. *C sharp* (C#) merupakan bahasa pemrograman yang diciptakan oleh Microsoft untuk mengembangkan aplikasi Microsoft.NET [12]. C# dapat digunakan untuk membangun berbagai aplikasi baik itu aplikasi *desktop* maupun *web services*. Gambar 2.3 merupakan tampilan Visual Studio Express 2012.



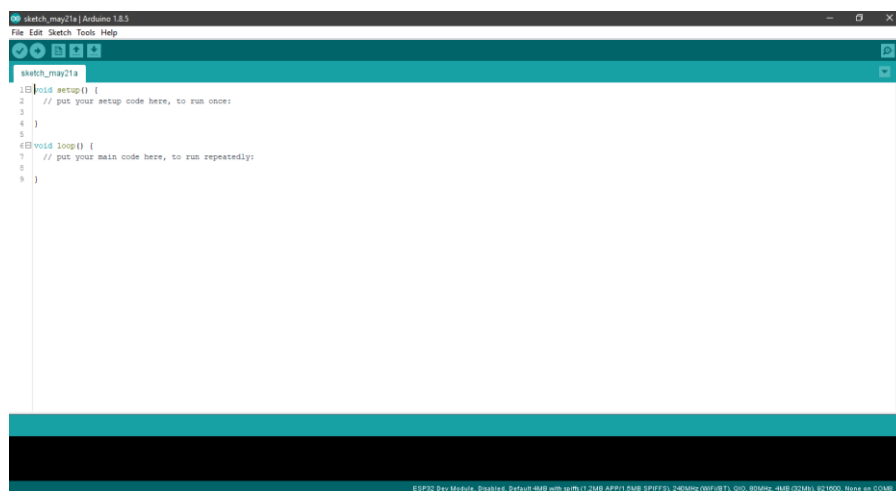
Gambar 2.3 Tampilan Visual Studio Express 2012.

2.4 Arduino IDE

Arduino IDE adalah sebuah perangkat lunak *open source* yang dikembangkan oleh Arduino.CC. Pada antarmuka pengguna, arduino IDE memberikan struktur program sederhana yang disebut *sketch* dengan ekstensi .ino yang di dalamnya terdapat 2 fungsi yaitu fungsi *setup* yang dijalankan sekali di awal dan fungsi *loop* yang dijalankan terus-menerus sampai aliran dayanya diputus. Arduino IDE juga memberikan fitur *compile* dan *upload* program untuk memudahkan penggunaannya [13]. Perangkat lunak ini berjalan dalam *platform* Java

dan tersedia dalam berbagai sistem operasi mulai dari Windows, MacOS serta Linux.

Tugas Akhir ini menggunakan Arduino IDE untuk mengembangkan perangkat lunak mikrokontroler STM32F1 pada ASV yang akan mengirim dan menerima data dari GCS melalui modul radio 3DR 433Mhz. Bahasa pemrograman yang digunakan adalah C dan C++. Gambar 2.4 merupakan tampilan dari Arduino IDE.

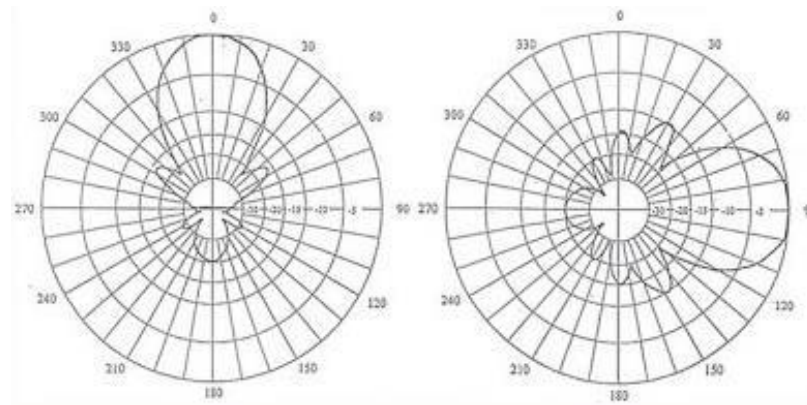


Gambar 2.4 Tampilan Arduino IDE.

2.5 Modul Radio 3DR 433Mhz

Antena adalah suatu piranti yang digunakan untuk memancarkan atau meneruskan gelombang elektromagnetik menuju ruang bebas atau menangkap gelombang elektromagnetik dari ruang bebas [14]. Apabila antena digunakan sebagai penerima (*receiver*) maka antena akan menangkap jejak gelombang elektromagnetik, sedangkan antena sebagai pemancar (*transceiver*) akan memancarkan gelombang elektromagnetik. Dalam penerapannya, antena digunakan pada berbagai perangkat elektronik nirkabel seperti radio, *handphone*, radar, GPS, dan sebagainya. Berdasarkan pola radiasinya antena secara umum dapat dibagi menjadi 2, yaitu antena *directional* dan *omnidirectional*. Pola radiasi antena *directional* berupa sinyal yang terarah (pada satu arah), sedangkan antena *omnidirectional* memiliki pola radiasi sinyal ke segala arah dalam sudut 360°

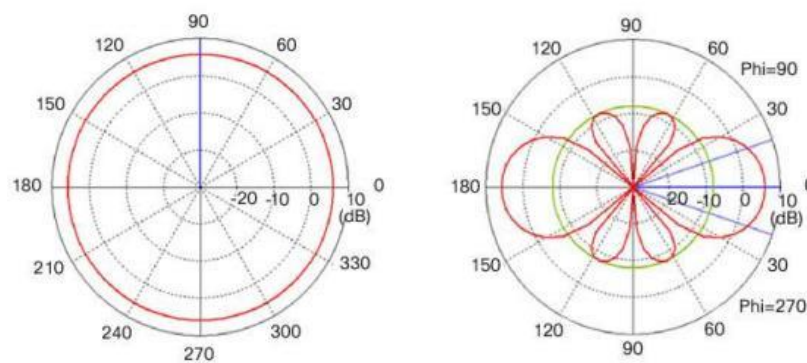
dengan daya yang sama [14]. Gambar 2.5 dan Gambar 2.6 merupakan pola radiasi antenna *directional* dan *omnidirectional* secara horizontal dan vertikal.



(a) Horizontal.

(b) Vertikal

Gambar 2.5 Pola radiasi antenna *directional*.



(a) Horizontal

(b) Vertikal

Gambar 2.6 Pola radiasi antenna *omnidirectional*.

Modul Radio 3DR 433Mhz merupakan salah satu modul telemetri yang dapat mengirim dan menerima sinyal informasi secara nirkabel pada frekuensi 433Mhz. Antena pada modul Radio 3DR 433Mhz berupa *transceiver* yang merupakan gabungan dari *transmitter* dan *receiver*. Antena yang digunakan oleh modul Radio 3DR 433Mhz adalah antenna *omnidirectional*. Tugas Akhir ini menggunakan Modul Radio 3DR 433Mhz sebagai media komunikasi antara komputer GCS dengan ASV. Modul Radio 3DR 433Mhz seperti pada Gambar 2.7 memiliki antarmuka UART dengan spesifikasi sensitivitas dalam menerima

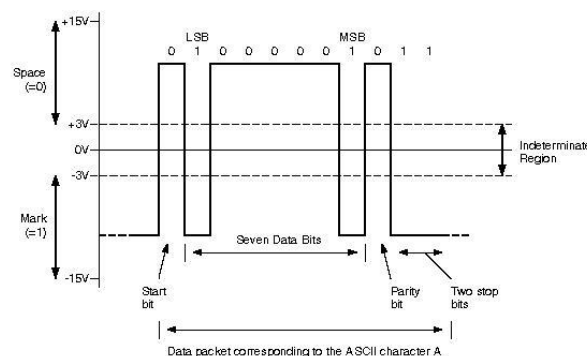
informasi sebesar -117 dBm, tegangan kerjanya 5V, daya keluaran maksimal sebesar 100mW, *transmit current* sebesar 100mA pada 30 dBm, *receive current* 25mA, kecepatan transfer di udara mencapai 250 kbps dan memiliki dimensi 26,7 x 55,5 x 13,3 cm tanpa antena [15].



Gambar 2.7 Modul radio 3DR 433Mhz.

2.6 Komunikasi Serial Asinkron

Secara umum komunikasi data dibagi menjadi 2 jenis yaitu komunikasi paralel dan komunikasi serial. Komunikasi serial adalah proses pengiriman data suatu bit pada 1 waktu secara sekuensial melalui 1 kanal komunikasi. Komunikasi serial juga terbagi menjadi 2 jenis yaitu sinkron dan asinkron [16]. Komunikasi serial asinkron adalah komunikasi data yang memerlukan *start bit* untuk menunjukkan mulainya data dan *stop bit* untuk menunjukkan selesainya data. Contoh pengiriman karakter A secara serial asinkron, dimulai dengan 1 *start bit*, tujuh bit data, 1 *parity bit* sebagai data *check* dan diakhiri dengan 1 atau 2 *stop bit* seperti pada Gambar 2.8.



Gambar 2.8 Bit data serial asinkron.

Komunikasi serial asinkron, mensyaratkan pengirim dan penerima harus memiliki protokol komunikasi yang sama. Berikut ini adalah beberapa istilah dalam protokol komunikasi serial:

1. *Baud rate*

Kecepatan transmisi dalam satuan bit per detik. *Clock* dari pengirim dan penerima harus memiliki *baud rate* yang sama. Apabila pengiriman data dilakukan setiap detik untuk mengetahui nilai *baudrate* yang diperlukan dapat dicari menggunakan persamaan 2.1.

$$baudrate = \frac{\Sigma(char * (8 + 2))}{1 s} \quad (2.1)$$

$\Sigma char$ merupakan jumlah karakter yang ingin dikirimkan, dengan menggunakan konfigurasi SERIAL_8N1 maka nilai *char* dikalikan 10 karena 1 variabel *char* berisi 8 bit kemudian ditambah 1 *start bit* dan 1 *stop bit*.

2. *Start bit*

Logika *low* menunjukkan transmisi data dimulai. Kondisi *low* yang terjadi pada *start bit* dinamakan *spacing state*.

3. *Data bit*

Berisi data dengan jumlah 5, 6, 7, atau 8 bit. Bit pertama yang dikirim adalah *Least Significant Bit (LSB)*.

4. *Parity bit*

Berfungsi untuk mengecek adanya *error* data yang ditransfer. *Parity* dapat bernilai *odd*(ganjil), *even*(genap), dan *none*.

5. *Stop bit*

Satu atau dua bit berlogika *high* akan dikirim setelah *data bit* atau *parity bit* jika ada *parity bit*. Dengan adanya *stop bit* dapat dipastikan bahwa penerima memiliki waktu yang cukup untuk menerima karakter berikutnya.

BAB III

PERANCANGAN SISTEM

3.1 Deskripsi Sistem

Sistem GCS merupakan perantara antara operator dan ASV yang digunakan untuk menyelesaikan sebuah misi. GCS yang dirancang memiliki fitur *telecontrolling* dan *telemetry* sehingga misi yang diberikan kepada ASV dapat terlaksana secara *autonomous* dengan intervensi yang minimal dari operator. GCS dirancang agar dapat berkomunikasi 2 arah dengan ASV secara *wireless*. GCS dirancang untuk dioperasikan pada sebuah laptop dengan sistem operasi *windows*.

3.2 Analisis Kebutuhan

Analisis kebutuhan dilakukan untuk mengetahui kebutuhan dari sistem yang dikembangkan baik kebutuhan fungsional maupun non-fungsional.

1. Kebutuhan Fungsional

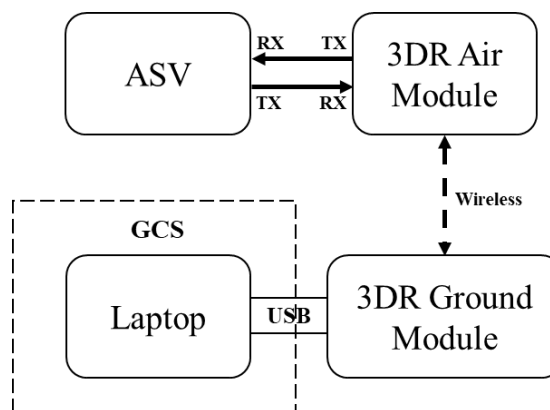
Operator mampu menghubungkan GCS dengan ASV secara *wireless*, mampu melakukan pengaturan kecepatan ASV melalui GCS, mampu menjalankan dan menghentikan ASV melalui GCS, mampu melakukan *tuning* parameter kendali menggunakan GCS, mampu membuat *waypoint* menggunakan GCS dan mengirimkannya ke ASV, mampu melakukan *failsafe mode*, operator mampu memantau data-data yang dikirimkan oleh ASV dan GCS dapat menyimpan data pengujian ke dalam dokumen csv.

2. Kebutuhan Non-Fungsional

Tampilan peta pada GCS bertipe *hybrid*, GCS dijalankan menggunakan laptop dengan sistem operasi windows, terdapat petunjuk penggunaan GCS bagi operator, terdapat tampilan waktu juga tanggal pada GCS dan antarmuka GCS dapat terlihat jelas pada area *outdoor*.

3.3 Perancangan Perangkat Keras

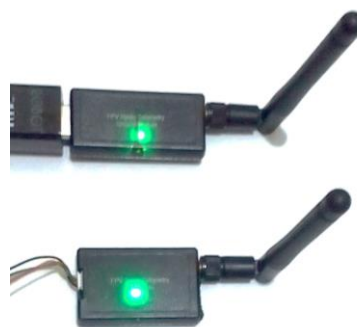
Perancangan perangkat keras terdiri dari modul Radio 3DR 433Mhz dan laptop. Modul radio 3DR 433Mhz sebagai media komunikasi, sedangkan laptop yang akan menjalankan program GCS. Hubungan komponen penyusun sistem dapat dilihat pada Gambar 3.1.



Gambar 3.1 Konfigurasi perangkat keras.

3.3.1 Penggunaan Modul Radio 3DR 433Mhz

Modul radio 3DR 433Mhz digunakan sebagai media komunikasi dua arah antara GCS dan ASV. Pemasangan antenna bertipe *omnidirectional* pada *ground module* dan *air module* radio 3DR 433Mhz bertujuan untuk memperluas jangkauan sinyal seperti pada Gambar 3.2. Istilah *air module* dan *ground module* mengacu kepada penamaan di kemasan modul radio 3DR 433Mhz.



Gambar 3.2 Penggunaan modul radio 3DR 433Mhz.

Nyala lampu berwarna hijau pada modul radio 433Mhz menandakan bahwa *ground module* dan *air module* sudah saling terhubung dan siap digunakan. Lampu hijau yang berkedip menandakan modul radio 433Mhz tidak saling terhubung dan belum siap digunakan. *Ground module* dihubungkan dengan laptop, sedangkan *air module* dihubungkan dengan mikrokontroler pada ASV. Data dikirim dan diterima melalui protokol komunikasi serial asinkron.

3.3.2 Penggunaan Laptop

Laptop dengan sistem operasi *windows* diperlukan untuk menjalankan perangkat lunak GCS yang dirancang. Laptop yang digunakan menggunakan prosesor AMD A8-2,2Ghz dengan 4 *giga byte random access memory*. *Ground module* radio 3DR 433Mhz dihubungkan ke laptop melalui *port Universal Serial Bus* (USB) seperti pada Gambar 3.3.

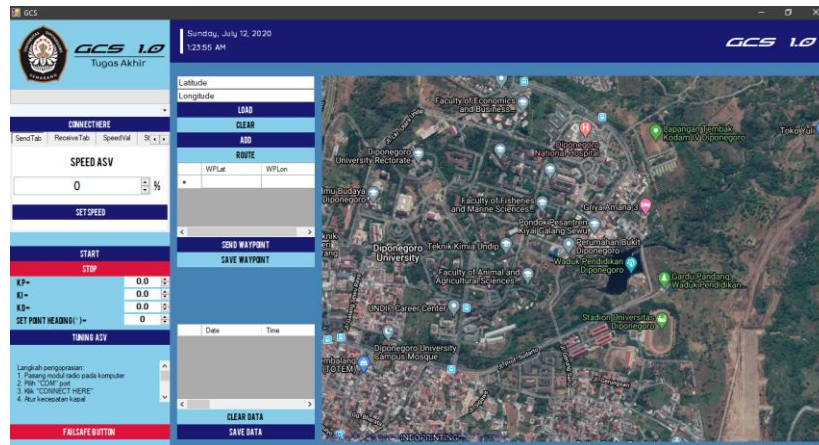


Gambar 3.3 Penggunaan laptop.

3.4 Perancangan Perangkat Lunak

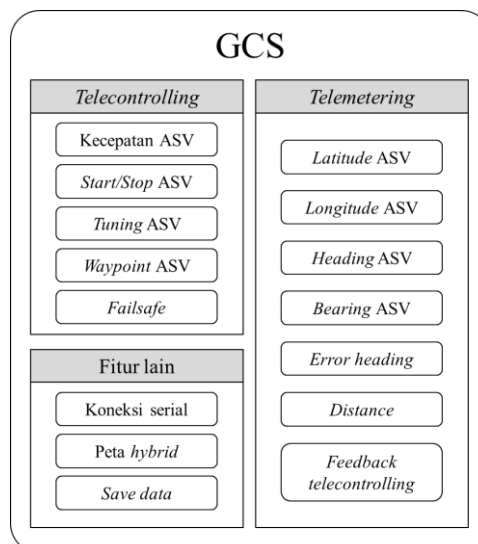
Perancangan perangkat lunak GCS pada laptop dilakukan menggunakan Microsoft Visual Studio Express 2012 dengan bahasa pemrograman C#, sedangkan perangkat lunak untuk mengirim dan menerima data pada ASV dilakukan menggunakan bahasa pemrograman C. GCS yang dirancang memiliki kemampuan *telecontrolling* dan *telemetry* untuk memudahkan operator dalam mengoperasikan ASV. Fitur-fitur yang terdapat pada GCS adalah *Connect/Disconnect* komunikasi, tombol *start/stop* mengoperasikan ASV, mengatur kecepatan, *tuning* parameter kendali, menampilkan peta, membuat *waypoint*,

menyimpan data selama pengoperasian ASV, dan *failsafe button*. Tampilan awal GCS dapat dilihat pada Gambar 3.4.



Gambar 3.4 Tampilan GCS.

Perancangan perangkat lunak GCS pada laptop dilakukan mulai dari memasukkan *library* GMap.NET ke dalam Visual Studio, menginisialisasi *library* yang digunakan, menginisialisasi variabel yang digunakan, membuat *layout* GCS, serta membuat fungsi komunikasi, pengontrol kecepatan, fitur *start/stop*, fitur *tuning* parameter kendali, fitur *waypoint*, fitur *save data*, dan *failsafe*. Fitur pada GCS dapat dilihat pada Gambar 3.5.



Gambar 3.5 Fitur GCS.

3.4.1 Fitur Koneksi GCS dengan ASV

Komunikasi dilakukan secara serial melalui modul telemetri. Terdapat tombol “CONNECT HERE” untuk menghubungkan GCS dengan ASV seperti pada Gambar 3.6.



(a) Tombol penghubung koneksi.

(b) Tombol pemutus koneksi.

Gambar 3.6 Tampilan koneksi GCS.

Konfigurasi komunikasi serial asinkron yang digunakan adalah Serial_8N1, dengan *baudrate* 57600 bps, *none parity bit* dan *stop bit* bernilai *high*. Nilai baudrate 57600 bps sudah mencukupi untuk koneksi antara GCS dan ASV karena data yang dikirim tidak melebihi 5760 byte. Pemrograman yang dilakukan agar dapat menghubungkan/memutus komunikasi dengan ASV seperti *listing* program berikut.

```
private void OpenClose_Click(object sender, EventArgs e)
{
    if (OpenClose.Text == "CONNECT HERE")
    {
        if (PortNumber.Text.Length > 1)
        {
            port = new SerialPort(PortNumber.Text, 57600,
                Parity.None, 8, StopBits.One);
            port.Open();
            port.DataReceived += DataReceived;
            progressBar1.Value = 100;
            OpenClose.Text = "Close";
        }
    }
    else
    {
        port.Close();
        progressBar1.Value = 0;
        timer1.Stop();
        OpenClose.Text = "CONNECT HERE";
    }
}
```

Agar program tersebut berjalan perlu ditambahkan `"using System.IO.Ports;"` untuk mengakses *port* serial yang tersedia pada laptop. Pilih *port* yang tersedia pada *ComboBox*, setelah memilih *port* tekan tombol "CONNECT HERE" maka *ProgressBar* akan *full* dan GCS akan terhubung dengan ASV. Apabila ingin memutus koneksi, cukup tekan tombol "CLOSE".

3.4.2 Telecontrolling Kecepatan ASV

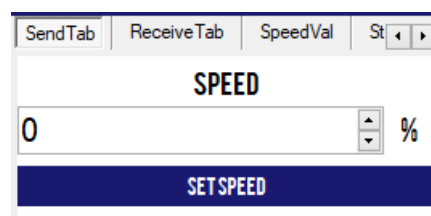
Kecepatan ASV dapat dikendalikan melalui GCS. Nilai masukan kecepatan berupa persentase dari nilai PWM motor, nilai tersebut kemudian diolah untuk menghasilkan nilai PWM sesuai dengan batasan yang telah ditentukan sebelumnya. Nilai kecepatan dapat diberikan dengan mengatur nilai *speed* pada GCS, kemudian mengirimkannya ke ASV dengan menekan tombol "SET SPEED" seperti Gambar 3.7. Persentase nilai kecepatan diubah ke dalam nilai PWM menggunakan persamaan 3.1.

$$s = ((speed.Value) \times 5) + 1000 \quad (3.1)$$

dengan

s = variabel kecepatan yang dikirim GCS

$speed.Value$ = nilai dari *numeric.UpDown* pada GCS



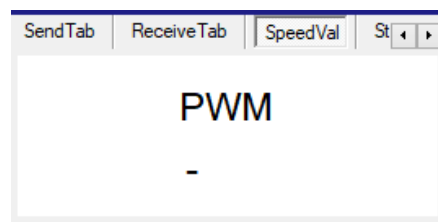
Gambar 3.7 Tampilan *telecontrolling* kecepatan ASV.

Listing program yang dibuat untuk pengaturan kecepatan ASV adalah sebagai berikut.

```
private void speed_ValueChanged(object sender, EventArgs e)
{
    s = ((speed.Value)*5) + 1000;
}
```

```
private void btnSend(object sender, EventArgs e)
{
    if (port.IsOpen)
    {
        String ok = s.ToString();
        port.Write("@" + "speed" + "|" + ok + ">");
    }
}
```

Nilai PWM aktual pada ASV dapat dilihat melalui umpan balik kecepatan ASV pada *tab* “SpeedVal” seperti Gambar 3.8.



Gambar 3.8 Tampilan umpan balik *telecontrolling* kecepatan ASV.

3.4.3 *Telecontrolling Start/Stop ASV*

Start dan *stop* pada GCS dirancang untuk memberi perintah memulai atau menghentikan misi pada ASV. Tombol “START” apabila ditekan akan memberikan nilai 1 pada variabel *start* dan mengirimkannya ke ASV, sedangkan apabila tombol “STOP” ditekan akan memberikan nilai 0 pada variabel *start* dan mengirimkannya ke ASV. Tampilan *start/stop* pada GCS seperti pada Gambar 3.9.



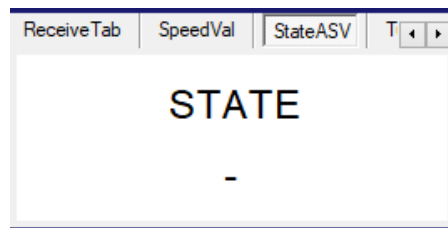
Gambar 3.9 Tampilan *telecontrolling start/stop* ASV.

Listing program yang dibuat untuk fitur *start* dan *stop* ASV adalah sebagai berikut.

```
private void btnStart_Click(object sender, EventArgs e)
{
    if (port.IsOpen)
    {
        String start = "1";
        port.Write("#" + "start" + "|" + start + ">");
    }
}
```

```
private void btnStop_Click(object sender, EventArgs e)
{
    if (port.IsOpen)
    {
        String start = "0";
        port.Write("#" + "start" + "|" + start + ">");
    }
}
```

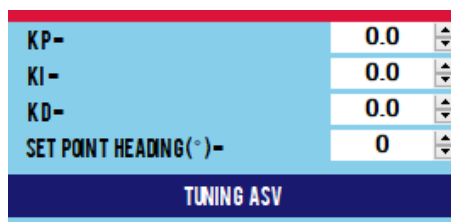
Kondisi atau *state* aktual misi ASV dapat dilihat melalui umpan balik kecepatan ASV pada *tab* “SpeedVal” seperti Gambar 3.10.



Gambar 3.10 Tampilan umpan balik *telecontrolling start/stop* ASV.

3.4.4 *Telecontrolling Tuning Parameter Kendali*

Telecontrolling tuning parameter kendali dirancang untuk memberikan nilai parameter kendali PID berupa Kp, Ki, Kd. *Setpoint* ditentukan oleh operator dengan memberi nilai parameter SP. Parameter tersebut diatur menggunakan *numericUpDown* yang nantinya akan dikirimkan kepada ASV setelah menekan tombol “TUNING ASV” seperti pada Gambar 3.11.



Gambar 3.11 Tampilan *telecontrolling tuning* parameter kendali.

Listing program yang dibuat agar parameter *tuning* PID dapat dikirimkan ke ASV adalah sebagai berikut.


```

private void Kp_ValueChanged(object sender, EventArgs e)
{
    vKp = Kp.Value;
}

private void Ki_ValueChanged(object sender, EventArgs e)
{
    vKi = Ki.Value;
}

private void Kd_ValueChanged(object sender, EventArgs e)
{
    vKd = Kd.Value;
}

private void Sp_ValueChanged(object sender, EventArgs e)
{
    vSp = Sp.Value;
}

private void btnTuning_Click(object sender, EventArgs e)
{
    if (port.IsOpen)
    {
        String kpValue = vKp.ToString();
        String kiValue = vKi.ToString();
        String kdValue = vKd.ToString();
        String spValue = vSp.ToString();
        port.Write("^" + "tuning" + "|" + kpValue + "|" +
            kiValue + "|" + kdValue + "|" + spValue + ">");
    }
}

```

Nilai aktual parameter kendali PID pada ASV dapat dilihat melalui umpan balik nilai *tuning* parameter kendali pada *tab* “TuningVal” seperti Gambar 3.12.



Gambar 3.12 Tampilan umpan balik *telecontrolling tuning* parameter kendali.

3.4.5 Fitur Menampilkan Peta

Peta digunakan sebagai area kerja membuat *waypoint* dan memantau pergerakan ASV. Pada Tugas Akhir ini agar GCS dapat menampilkan peta, digunakan *library* GMap.NET yang tersedia untuk Visual Studio seperti pada Gambar 3.13.



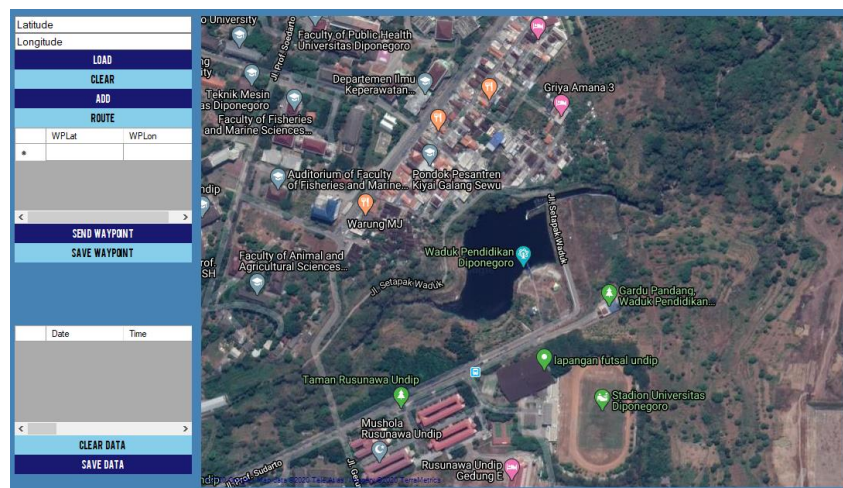
Gambar 3.13 Tampilan peta GCS.

GMap.NET merupakan sebuah *library* berarsitektur .NET Framework yang menyediakan berbagai fitur penggunaan peta, mulai dari menentukan posisi, membuat rute, membuat bentuk area dan fitur lainnya. GMap.NET tersedia secara gratis, pengguna hanya perlu men-*download* dan meng-*compile*-nya kemudian menambahkannya pada Reference Manager Visual Studio. Tipe peta yang digunakan pada Tugas Akhir ini adalah *Google Map Hybrid View*. Tipe peta ini memungkinkan untuk menampilkan citra satelit dan tetap memberikan informasi jalan. *Listing* program yang dibuat untuk menampilkan peta pada GCS adalah sebagai berikut.

```
public Form1()
{
    gmap.MapProvider = GoogleHybridMapProvider.Instance;
    GMaps.Instance.Mode = AccessMode.ServerOnly;
}
private void gmap_Load(object sender, EventArgs e)
{
    gmap.Position = new PointLatLng(-7.05250157725237,
    110.445610284805);
    gmap.ShowCenter = false;
    gmap.Zoom = 17;
}
```

3.4.6 Telecontrolling Waypoint ASV

Waypoint yang dibuat ditampilkan pada peta, juga pada tabel WPLat dan WPLon. Waypoint yang dibuat akan menghasilkan rute dengan garis berwarna merah, sedangkan umpan balik yang diterima GCS berupa rute yang dilalui ASV adalah garis berwarna kuning. Antarmuka fitur *waypoint* pada GCS dapat dilihat pada Gambar 3.14.



Gambar 3.14. Tampilan *telecontrolling waypoint* pada GCS.

Waypoint dapat dibuat dengan cara menekan titik tertentu pada peta atau mengetik manual pada *form Latitude* dan *Longitude*, kemudian tekan tombol “LOAD”, apabila ingin membatalkan titik tersebut tekan tombol “CLEAR” bila sudah yakin dapat menekan tombol “ADD”, kemudian untuk membuat rute dapat menekan tombol “ROUTE”. Apabila telah selesai membuat rute, *waypoint* dapat dikirim ke ASV dengan menekan tombol “SEND WAYPOINT”. Waypoint juga dapat disimpan ke dalam dokumen dengan format *comma separated value* (csv). Listing program agar *waypoint* dapat dibuat dan dikirimkan ke ASV adalah sebagai berikut.

```
private void gmap_MouseClick(object sender, MouseEventArgs e)
{
    if (e.Button == MouseButtons.Left)
    {
        var points = gmap.FromLocalToLatLng(e.X, e.Y);
        lat_gps = points.Lat;
        lon_gps = points.Lng;
    }
}
```

```

        txtLat.Text = lat_gps + "";
        txtLon.Text = lon_gps + "";
    }
}

private void loadGPS_Click(object sender, EventArgs e)
{
    try
    {
        GMapOverlay markers = new GMapOverlay("markers");
        lat_gps = Convert.ToDouble(txtLat.Text);
        lon_gps = Convert.ToDouble(txtLon.Text);
        gmap.Position = new PointLatLng(lat_gps, lon_gps);
        gmap.Zoom = 20;

        GMapMarker marker = new GMarkerGoogle(
            new PointLatLng(lat_gps, lon_gps),
            GMarkerGoogleType.red);
        markers.Markers.Add(marker);
        gmap.Overlays.Add(markers);
    }

    catch
    {
        MessageBox.Show("Please Input Number", "Try Again");
    }
}

public void addPos_Click(object sender, EventArgs e)
{
    try
    {
        points.Add(new
            PointLatLng(Convert.ToDouble(txtLat.Text),
                Convert.ToDouble(txtLon.Text)));
        dataGridView1.Rows.Add();
        dataGridView1.Rows[i].Cells[0].Value =
            Convert.ToDouble(txtLat.Text);
        dataGridView1.Rows[i].Cells[1].Value =
            Convert.ToDouble(txtLon.Text);
        i++;
    }

    catch
    {
        MessageBox.Show("Load First");
    }
}

public void routBtn_Click(object sender, EventArgs e)
{
    GMapOverlay routes = new GMapOverlay("routes");
    GMapRoute route = new GMapRoute(points, "rute");
    route.Stroke = new Pen(Color.Red, 3);
}

```

```

        routes.Routes.Add(route);
        gmap.Overlays.Add(routes);
        gmap.Refresh();
    }

    public void clrBtn_Click(object sender, EventArgs e)
    {
        for (int m = 0; m < gmap.Overlays.Count; m = 0)
        {
            gmap.Overlays.RemoveAt(m);
            gmap.Refresh();
        }
    }

    private void sendWP_Click(object sender, EventArgs e)
    {
        if (port.IsOpen)
        {
            if (dataGridView1.Rows[0].Cells[0].Value != null)
            {
                WP1Lat =
                    dataGridView1.Rows[0].Cells[0].Value.ToString();
                WP1Lon =
                    dataGridView1.Rows[0].Cells[1].Value.ToString();
            }
            if (dataGridView1.Rows[1].Cells[0].Value != null)
            {
                WP2Lat =
                    dataGridView1.Rows[1].Cells[0].Value.ToString();
                WP2Lon =
                    dataGridView1.Rows[1].Cells[1].Value.ToString();
            }
            if (dataGridView1.Rows[2].Cells[0].Value != null)
            {
                WP3Lat =
                    dataGridView1.Rows[2].Cells[0].Value.ToString();
                WP3Lon =
                    dataGridView1.Rows[2].Cells[1].Value.ToString();
            }
            if (dataGridView1.Rows[3].Cells[0].Value != null)
            {
                WP4Lat =
                    dataGridView1.Rows[3].Cells[0].Value.ToString();
                WP4Lon =
                    dataGridView1.Rows[3].Cells[1].Value.ToString();
            }
            if (dataGridView1.Rows[4].Cells[0].Value != null)
            {
                WP5Lat =
                    dataGridView1.Rows[4].Cells[0].Value.ToString();
                WP5Lon =
                    dataGridView1.Rows[4].Cells[1].Value.ToString();
            }
            if (dataGridView1.Rows[5].Cells[0].Value != null)
            {
                WP6Lat =
                    dataGridView1.Rows[5].Cells[0].Value.ToString();
                WP6Lon =
                    dataGridView1.Rows[5].Cells[1].Value.ToString();
            }
            if (dataGridView1.Rows[6].Cells[0].Value != null)

```

```

{
    WP7Lat =
        dataGridView1.Rows[6].Cells[0].Value.ToString();
    WP7Lon =
        dataGridView1.Rows[6].Cells[1].Value.ToString();
}
else { WP7Lat = "0"; WP7Lon = "0"; }
}
else { WP6Lat = "0"; WP6Lon = "0";
        WP7Lat = "0"; WP7Lon = "0"; }
}
else { WP5Lat = "0"; WP5Lon = "0";
        WP6Lat = "0"; WP6Lon = "0";
        WP7Lat = "0"; WP7Lon = "0"; }
}
else { WP4Lat = "0"; WP4Lon = "0";
        WP5Lat = "0"; WP5Lon = "0";
        WP6Lat = "0"; WP6Lon = "0";
        WP7Lat = "0"; WP7Lon = "0"; }
}
else { WP3Lat = "0"; WP3Lon = "0";
        WP4Lat = "0"; WP4Lon = "0";
        WP5Lat = "0"; WP5Lon = "0";
        WP6Lat = "0"; WP6Lon = "0";
        WP7Lat = "0"; WP7Lon = "0"; }
}

else { WP2Lat = "0"; WP2Lon = "0";
        WP3Lat = "0"; WP3Lon = "0";
        WP4Lat = "0"; WP4Lon = "0";
        WP5Lat = "0"; WP5Lon = "0";
        WP6Lat = "0"; WP6Lon = "0";
        WP7Lat = "0"; WP7Lon = "0"; }
}
else
{
    WP1Lat = "0"; WP1Lon = "0";
    WP2Lat = "0"; WP2Lon = "0";
    WP3Lat = "0"; WP3Lon = "0";
    WP4Lat = "0"; WP4Lon = "0";
    WP5Lat = "0"; WP5Lon = "0";
    WP6Lat = "0"; WP6Lon = "0";
    WP7Lat = "0"; WP7Lon = "0";
}
port.Write("!" + "wp" + "|"
    + WP1Lat + "|" + WP1Lon + "|"
    + WP2Lat + "|" + WP2Lon + "|"
    + WP3Lat + "|" + WP3Lon + "|"
    + WP4Lat + "|" + WP4Lon + "|"
    + WP5Lat + "|" + WP5Lon + "|"
    + WP6Lat + "|" + WP6Lon + "|"
    + WP7Lat + "|" + WP7Lon + ">");
MessageBox.Show("!" + "wp" + "|"
    + WP1Lat + "|" + WP1Lon + "|"
    + WP2Lat + "|" + WP2Lon + "|"
    + WP3Lat + "|" + WP3Lon + "|"

```

```

        + WP4Lat + "|" + WP4Lon + "|"
        + WP5Lat + "|" + WP5Lon + "|"
        + WP6Lat + "|" + WP6Lon + "|"
        + WP7Lat + "|" + WP7Lon + ">");
    }
}

private void saveWP_Click(object sender, EventArgs e)
{
    SaveWPtoCSV();
}

private void SaveWPtoCSV()
{
    string filename = "";
    SaveFileDialog sfd = new SaveFileDialog();
    sfd.Filter = "CSV (*.csv) | *.csv";
    sfd.FileName = "Data_WP.csv";
    if (sfd.ShowDialog() == DialogResult.OK)
    {
        MessageBox.Show("Data akan disimpan.");
        if (File.Exists(filename))
        {
            try
            {
                File.Delete(filename);
            }
            catch (IOException ex)
            {
                MessageBox.Show("Error" + ex.Message);
            }
        }
        int columnCount = dataGridView1.ColumnCount;
        string columnNames = "";
        string[] output = new string[dataGridView1.RowCount
            + 1];
        for (int i = 0; i < columnCount; i++)
        {
            columnNames +=
                dataGridView1.Columns[i].Name.ToString() + ",";
        }
        output[0] += columnNames;
        for (int i = 1; (i - 1) < dataGridView1.RowCount;
            i++)
        {
            for (int j = 0; j < columnCount; j++)
            {
                if (dataGridView1.Rows[i - 1].Cells[j].Value !=
                    null)
                {
                    output[i] +=
                        dataGridView1.Rows[i - 1].Cells[j].Value.ToString()
                            + ",";
                }
            }
        }
        System.IO.File.WriteAllLines(sfd.FileName, output,
            System.Text.Encoding.UTF8);
    }
}

```

```

        MessageBox.Show("Data berhasil disimpan.");
    }
}

```

3.4.7 Telecontrolling Failsafe ASV

Telecontrolling failsafe yang dimaksud adalah ketika terjadi ketidaknormalan pada ASV, operator dapat menekan tombol “FAILSAFE BUTTON” sehingga GCS akan mengirimkan sinyal darurat kepada ASV. Tipe *failsafe* yang digunakan adalah *Return To Launch* (RTL), sehingga ASV akan kembali ke titik *waypoint* awal dijalankan. Tampilan *failsafe button* pada GCS seperti pada Gambar 3.15.



Gambar 3.15 Tampilan *telecontrolling failsafe*.

Listing program yang dibuat agar ASV dapat sinyal bahwa dalam keadaan darurat adalah sebagai berikut.

```

private void btnEmg_Click(object sender, EventArgs e)
{
    if (port.IsOpen)
    {
        String danger = "1";
        port.Write("$" + "failsafe" + "|" + danger + ">");
    }
}

private void saveBtn_Click(object sender, EventArgs e)
{
    SaveToCSV();
}

```

Kondisi atau *state* aktual keadaan darurat ASV dapat dilihat melalui umpan balik pada *tab* “Emergency” seperti Gambar 3.16.



Gambar 3.16 Tampilan umpan balik *telecontrolling failsafe*.

3.4.8 Telemetry pada GCS

Telemetry atau pengukuran jarak jauh pada GCS dilakukan untuk mengetahui hasil pembacaan dan pengolahan data sensor oleh ASV secara *realtime*. Selain parameter yang telah disebutkan pada sub bab sebelumnya, nilai-nilai yang dikirim ASV untuk ditampilkan pada *telemetry* GCS adalah data *latitude*, *longitude*, *heading*, *bearing*, *error* hadap, dan *distance*. *Latitude* dan *longitude* merupakan hasil pembacaan GPS. *Heading* merupakan hasil pembacaan sensor kompas. *Bearing* merupakan hasil perhitungan sebagai *setpoint* hadap ASV. *Error* hadap merupakan selisih antara *heading* dan *bearing*. *Distance* merupakan jarak antara ASV dengan *waypoint* selanjutnya. Data-data tersebut ditampilkan pada *tab* “ReceiveTab” seperti pada Gambar 3.17.

 A screenshot of a software window with four tabs: 'SendTab', 'ReceiveTab', 'SpeedVal', and 'St'. The 'ReceiveTab' is selected. The main area displays a table of telemetry data.

Latitude	-	(°)
Longitude	-	(°)
Heading	-	(°)
Bearing	-	(°)
ErrorK	-	(°)
Distance	-	(m)

Gambar 3.17 Tampilan *telemetry* pada GCS.

Listing program yang dibuat untuk agar *telemetry* pada GCS dapat berfungsi adalah sebagai berikut.

```
void DataReceived(object sender,
    System.IO.Ports.SerialDataReceivedEventArgs e)
```

```

{
    dataIn = port.ReadLine();
    this.Invoke(new EventHandler(get_data));
    k = k + 100;
}

private void get_data(object sender, EventArgs e)
{
    try
    {
        System.Globalization.CultureInfo customCulture =
            (System.Globalization.CultureInfo)System.Threading.
            Thread.CurrentThread.CurrentCulture.Clone();
        customCulture.NumberFormat.NumberDecimalSeparator = ".";
        System.Threading.Thread.CurrentThread.CurrentCulture =
            customCulture;//format penulisan desimal

        if (dataIn.StartsWith("#"))
        {
            string[] dataArray = dataIn.Split(',');
            latN = double.Parse(dataArray[1], customCulture);
            lonN = double.Parse(dataArray[2], customCulture);
            latB = double.Parse(dataArray[3], customCulture);
            lonB = double.Parse(dataArray[4], customCulture);
            head = double.Parse(dataArray[5], customCulture);
            spK = double.Parse(dataArray[6], customCulture);
            error = double.Parse(dataArray[7], customCulture);
            jarak = double.Parse(dataArray[8], customCulture);
            pwm = double.Parse(dataArray[9], customCulture);
            rSt = double.Parse(dataArray[10], customCulture);
            rKp = double.Parse(dataArray[11], customCulture);
            rKi = double.Parse(dataArray[12], customCulture);
            rKd = double.Parse(dataArray[13], customCulture);
            rSp = double.Parse(dataArray[14], customCulture);
            rEm = double.Parse(dataArray[15], customCulture);
        }

        dataGridView2.Rows.Add();
        dataGridView2.Rows[j].Cells[0].Value =
            DateTime.Now.Month + "/" + DateTime.Now.Day + "/"
            + DateTime.Now.Year;
        dataGridView2.Rows[j].Cells[1].Value = Waktu.Text;
        dataGridView2.Rows[j].Cells[2].Value = head;
        dataGridView2.Rows[j].Cells[3].Value = latN;
        dataGridView2.Rows[j].Cells[4].Value = lonN;
        dataGridView2.Rows[j].Cells[5].Value = spK;
        dataGridView2.Rows[j].Cells[6].Value = error;
        dataGridView2.Rows[j].Cells[7].Value = jarak;
        dataGridView2.Rows[j].Cells[8].Value = pwm;
        dataGridView2.Rows[j].Cells[9].Value = start;
        j++;

        label4.Text = Convert.ToString(latN);
        label5.Text = Convert.ToString(lonN);
        label7.Text = Convert.ToString(head);
        label17.Text = Convert.ToString(spK);
    }
    catch { }
}

```

```

label12.Text = Convert.ToString(error);
label13.Text = Convert.ToString(jarak);
label19.Text = Convert.ToString(pwm);
label31.Text = Convert.ToString(rSt);
label27.Text = Convert.ToString(rKp);
label28.Text = Convert.ToString(rKi);
label29.Text = Convert.ToString(rKd);
label30.Text = Convert.ToString(rSp);
label22.Text = Convert.ToString(rEm);
if (latN != 0 && lonN != 0 && latB != 0 && lonB != 0)
{
    GMapOverlay jejak = new GMapOverlay("jejak");
    List<PointLatLng> jalur =
    new List<PointLatLng>();
    jalur.Add(new PointLatLng(latN, lonN));
    jalur.Add(new PointLatLng(latB, lonB));
    GMapRoute rute = new GMapRoute(jalur, "rute");
    rute.Stroke = new Pen(Color.Yellow, 5);
    rute.Stroke.StartCap = LineCap.Round;
    rute.Stroke.LineJoin = LineJoin.Round;
    rute.Stroke.EndCap = LineCap.Round;
    jejak.Routes.Add(rute);
    gmap.Overlays.Add(jejak);
}
}
catch { }
}

```

Data yang dikirim ASV dan diterima oleh GCS adalah sebuah paket data yang diawali oleh *header* berupa “#” dan diakhiri oleh *endline* berupa karakter “!”. Diantara *header* dan *endline* terdapat 15 data ASV berupa variabel *string* yang dipisahkan oleh *comma* “,”. Apabila *header* data yang diterima GCS adalah “#” maka akan dijalankan fungsi *parsing* atau diurai untuk memisahkan isi paket data tersebut. Data yang dikirim ASV dan diterima oleh GCS dapat dilihat pada Tabel 3.1.

Tabel 3.1 Data *telemetering*.

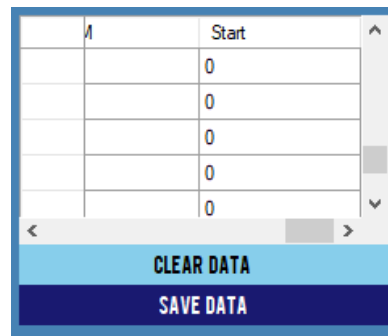
No	Isi Data	Keterangan
1	LatN	<i>Latitude</i> ASV sekarang
2	LonN	<i>Longitude</i> ASV sekarang
3	LatB	<i>Latitude</i> ASV sebelumnya
4	LonB	<i>Longitude</i> ASV sebelumnya
5	head	Sudut hadap ASV
6	spK	Sudut target ASV
7	<i>error</i>	<i>Error</i> sudut ASV
8	jarak	Jarak antara ASV dengan <i>waypoint</i> selanjutnya

Tabel 3.1 Lanjutan.

9	pwm	PWM motor BLDC
10	rSt	Umpan balik nilai <i>start</i>
11	rKp	Umpan balik nilai Kp
12	rKi	Umpan balik nilai Ki
13	rKd	Umpan balik nilai Kd
14	rSp	Umpan balik <i>setpoint</i>
15	rEm	Umpan balik <i>failsafe</i>

3.4.9 Fitur Save Data

Fitur *save data* dirancang untuk menyimpan data yang diterima oleh GCS. Data akan disimpan ke dalam dokumen dengan format csv. Untuk menampung data dibuat tabel dengan kolom berjumlah 10 buah seperti Gambar 3.18. Data yang disimpan adalah tanggal, waktu, *heading*, *latitude*, *longitude*, *bearing*, *error* kompas, jarak ke *waypoint* selanjutnya, PWM motor, dan kondisi *start*.

Gambar 3.18 Tampilan *save data*.

Listing program yang dibuat agar data yang diterima GCS dapat disimpan ke dalam dokumen csv ataupun dihapus datanya adalah sebagai berikut.

```
private void SaveToCSV()
{
    string filename = "";
    SaveFileDialog sfd = new SaveFileDialog();
    sfd.Filter = "CSV (*.csv) | *.csv";
    sfd.FileName = "Data_ASV.csv";
    if (sfd.ShowDialog() == DialogResult.OK)
    {
        MessageBox.Show("Data akan disimpan.");
        if (File.Exists(filename))
        {
            try
```

```

        {
            File.Delete(filename);
        }
        catch (IOException ex)
        {
            MessageBox.Show("Error" + ex.Message);
        }
    }
    int columnCount = dataGridView2.ColumnCount;
    string columnNames = "";
    string[] output = new string[dataGridView2.RowCount
        + 1];
    for (int i = 0; i < columnCount; i++)
    {
        columnNames +=
            dataGridView2.Columns[i].Name.ToString() + ",";
    }
    output[0] += columnNames;
    for (int i = 1; (i - 1) < dataGridView2.RowCount;
        i++)
    {
        for (int j = 0; j < columnCount; j++)
        {
            if (dataGridView2.Rows[i - 1].Cells[j].Value !=
                null)
                output[i] +=
                    dataGridView2.Rows[i - 1].Cells[j].Value.ToString()
                    + ",";
        }
    }
    System.IO.File.WriteAllLines(sfd.FileName, output,
        System.Text.Encoding.UTF8);
    MessageBox.Show("Data berhasil disimpan.");
}
private void clrData_Click(object sender, EventArgs e)
{
    reset();
}
private void reset()
{
    int numRows = dataGridView2.Rows.Count;
    for (int l = 0; l < numRows; l++)
    {
        try
        {
            int max = dataGridView2.Rows.Count - 1;
            dataGridView2.Rows.Remove(dataGridView2.Rows[max]);
        }
        catch (Exception exe)
        {
            MessageBox.Show("try again" + exe, "deleted",
                MessageBoxButtons.OK, MessageBoxIcon.Information);
        }
    }
}
}

```

3.4.10 Program Terima dan Kirim Data antara ASV dan GCS

Pengiriman dan penerimaan paket data pada ASV membutuhkan fungsi-fungsi khusus agar dapat berjalan dengan baik. Data yang dikirim oleh ASV adalah paket data yang nantinya digunakan untuk *telemetry* pada GCS, sedangkan data yang diterima ASV adalah data-data perintah *telecontrolling* dari GCS.

Data yang dikirim ASV merupakan sebuah paket data yang berisi beberapa variabel *string* dan dikirim setiap 200 ms. Penjelasan isi data yang dikirim oleh ASV dapat dilihat pada sub bab sebelumnya tentang *telemetry*. Pemrograman pada mikrokontroler STM32F1 yang terletak di ASV untuk mengirim paket data seperti *listing* program berikut:

```
void sendData() {
    GCS.print("#DAT");
    GCS.print(",");
    GCS.print(latN, 8);
    GCS.print(",");
    GCS.print(lonN, 8);
    GCS.print(",");
    GCS.print(latB, 8);
    GCS.print(",");
    GCS.print(lonB, 8);
    GCS.print(",");
    GCS.print(head);
    GCS.print(",");
    GCS.print(bearing);
    GCS.print(",");
    GCS.print(error);
    GCS.print(",");
    GCS.print(jarak);
    GCS.print(",");
    GCS.print(rSpeed);
    GCS.print(",");
    GCS.print(Start);
    GCS.print(",");
    GCS.print(tuning[0]);
    GCS.print(",");
    GCS.print(tuning[1]);
    GCS.print(",");
    GCS.print(tuning[2]);
    GCS.print(",");
    GCS.print(tuning[3]);
    GCS.print(",");
    GCS.print(Failsafe);
    GCS.print(",");
    GCS.print("!");
    GCS.println();
}
```

Paket data yang dikirim GCS agar dapat digunakan oleh ASV perlu diolah terlebih dahulu. Pengiriman paket data dari GCS bersifat insidental yaitu ketika operator menekan tombol *telecontrolling* tertentu yang dijelaskan pada sub bab sebelumnya. Paket data yang dikirim GCS adalah *waypoint*, *kecepatan*, *start*, *tuning*, dan *failsafe*. Setiap paket data yang dikirim dari GCS memiliki format yang berbeda seperti pada Tabel 3.2.

Tabel 3.2 Paket data yang dikirim GCS.

No	Paket	Header	Isi	Endline
1	<i>Waypoint</i>	“!wp”	<i>Latitude 1, longitude 1,</i> <i>latitude 2, longitude 2,</i> <i>latitude 3, longitude 3,</i> <i>latitude 4, longitude 4,</i> <i>latitude 5, longitude 5,</i> <i>latitude 6, longitude 6,</i> <i>latitude 7, longitude 7</i>	“>”
2	<i>Kecepatan</i>	“@speed”	Nilai kecepatan	“>”
3	<i>Start</i>	“#start”	Nilai <i>start</i>	“>”
4	<i>Tuning</i>	“^tuning”	Nilai Kp, Ki, Kd, <i>Setpoint Arah</i>	“>”
5	<i>Failsafe</i>	“\$failsafe”	Nilai sinyal <i>failsafe</i>	“>”

Paket data tersebut perlu diklasifikasi sesuai dengan *header*-nya, sehingga ASV dapat menentukan fungsi *parsing* mana yang akan dijalankan. Pemrograman yang dibuat untuk menentukan fungsi *parsing* seperti *listing* program berikut:

```
void rec() {
    static bool recData = false;
    static byte ndx;
    char endlne = '>';
    char c;

    while (GCS.available() > 0 && newData == false) {
        c = GCS.read();

        if (recData == true) {
            if (c != endlne) {
                dataRec[ndx] = c;
                ndx++;
            }
        }
    }
}
```

```

        if (ndx >= dataByte) {
            ndx = dataByte - 1;
        }
    }
    else {
        dataRec[ndx] = '\\0';
        recData = false;
        ndx = 0;
        newData = true;
    }
}
else if (c == '!') {
    recData = true;
    parsing = 1;
    //Parse Waypoint
}
else if (c == '@') {
    recData = true;
    parsing = 2;
    //Parse Speed
}
else if (c == '#') {
    recData = true;
    parsing = 3;
    //Parse Start
}
else if (c == '$') {
    recData = true;
    parsing = 4;
    //Parse Failsafe
}
else if (c == '^') {
    recData = true;
    parsing = 5;
    //Parse Tuning
}
}
}
void recData() {
    rec();
    if (newData == true) {
        strcpy(tempChar, dataRec);
        if (parsing == 1) {
            recWaypoint();
        }
        else if (parsing == 2) {
            recSpeed();
        }
        else if (parsing == 3) {
            recStart();
        }
        else if (parsing == 4) {
            recFailsafe();
        }
        else if (parsing == 5) {
            recTuning();
        }
    }
}

```



```

    }
    newData = false;
}
}

```

Paket data *waypoint* yang dikirim GCS adalah sekumpulan data *string* yang dipisahkan oleh karakter “|”. Isi paket data *waypoint* adalah *header* berupa “!wp”, isi data 10 titik *waypoint* dan diakhiri dengan karakter “>”. Apabila titik *waypoint* tidak ditentukan maka akan bernilai 0. Paket data tersebut perlu dilakukan *parsing*, data *string* yang telah diurai perlu dikonversi agar dapat diolah menjadi titik *waypoint*. Pada paket data *waypoint*, variabel data *string* dikonversi menjadi *float* dengan perintah `atof`. Pemrograman dilakukan dengan membuat fungsi `recWaypoint` seperti *listing* program berikut:

```

void recWaypoint() {
    char * strtokIndx;
    strtokIndx = strtok(tempChar, "|");
    strcpy(dataGCS, strtokIndx);
    strtokIndx = strtok(NULL, "|");
    WP[0] = atof(strtokIndx);
    strtokIndx = strtok(NULL, "|");
    WP[1] = atof(strtokIndx);
    strtokIndx = strtok(NULL, "|");
    WP[2] = atof(strtokIndx);
    strtokIndx = strtok(NULL, "|");
    WP[3] = atof(strtokIndx);
    strtokIndx = strtok(NULL, "|");
    WP[4] = atof(strtokIndx);
    strtokIndx = strtok(NULL, "|");
    WP[5] = atof(strtokIndx);
    strtokIndx = strtok(NULL, "|");
    WP[6] = atof(strtokIndx);
    strtokIndx = strtok(NULL, "|");
    WP[7] = atof(strtokIndx);
    strtokIndx = strtok(NULL, "|");
    WP[8] = atof(strtokIndx);
    strtokIndx = strtok(NULL, "|");
    WP[9] = atof(strtokIndx);
    strtokIndx = strtok(NULL, "|");
    WP[10] = atof(strtokIndx);
    strtokIndx = strtok(NULL, "|");
    WP[11] = atof(strtokIndx);
    strtokIndx = strtok(NULL, "|");
    WP[12] = atof(strtokIndx);
    strtokIndx = strtok(NULL, "|");
    WP[13] = atof(strtokIndx);
    strtokIndx = strtok(NULL, "|");
}

```

Paket data kecepatan yang dikirim GCS adalah sekumpulan data *string* yang dipisahkan oleh karakter “|”. Isi paket data kecepatan adalah *header* berupa “@speed”, data kecepatan dan diakhiri dengan karakter “>”. Paket data tersebut perlu dilakukan *parsing*, data *string* yang telah diurai perlu dikonversi agar dapat diolah menjadi nilai PWM motor BLDC pada ASV. Pada paket data kecepatan, variabel data *string* dikonversi menjadi *integer* dengan perintah `atoi`. Pemrograman dilakukan dengan membuat fungsi `recSpeed` seperti *listing* program berikut:

```
void recSpeed() {
    char * strtokIndx;
    strtokIndx = strtok(tempChar, "|");
    strcpy(dataGCS, strtokIndx);
    strtokIndx = strtok(NULL, "|");
    rSpeed = atof(strtokIndx);
}
```

Paket data *start* yang dikirim GCS adalah sekumpulan data *string* yang dipisahkan oleh karakter “|”. Isi paket data *start* adalah *header* berupa “#start”, data *start* dan diakhiri dengan karakter “>”. Paket data tersebut perlu dilakukan *parsing*, data *string* yang telah diurai perlu dikonversi agar dapat diolah menjadi parameter *start* ASV. Pada paket data *start*, variabel data *string* dikonversi menjadi *integer* dengan perintah `atoi`. Pemrograman dilakukan dengan membuat fungsi `recStart` seperti *listing* program berikut:

```
void recStart() {
    char * strtokIndx;
    strtokIndx = strtok(tempChar, "|");
    strcpy(dataGCS, strtokIndx);
    strtokIndx = strtok(NULL, "|");
    Start = atof(strtokIndx);
}
```

Paket data *tuning* yang dikirim GCS adalah sekumpulan data *string* yang dipisahkan oleh karakter “|”. Isi paket data *tuning* adalah *header* berupa karakter “^tuning”, data nilai Kp, Ki, Kd, *Setpoint* hadap dan diakhiri dengan karakter “>”. Paket data tersebut perlu dilakukan *parsing*, data *string* yang telah diurai perlu dikonversi agar dapat diolah menjadi parameter *tuning* parameter kendali. Pada

paket data *tuning*, variabel data *string* dikonversi menjadi *float* dengan perintah *atof*. Pemrograman dilakukan dengan membuat fungsi *recTuning* seperti *listing* program berikut:

```
void recTuning() {
    char * strtokIndx;
    strtokIndx = strtok(tempChar, "|");
    strcpy(dataGCS, strtokIndx);
    strtokIndx = strtok(NULL, "|");
    tuning[0] = atof(strtokIndx); //kp
    strtokIndx = strtok(NULL, "|");
    tuning[1] = atof(strtokIndx); //ki
    strtokIndx = strtok(NULL, "|");
    tuning[2] = atof(strtokIndx); //kd
    strtokIndx = strtok(NULL, "|");
    tuning[3] = atof(strtokIndx); //sp
}
```

Paket data *failsafe* yang dikirim GCS adalah sekumpulan data *string* yang dipisahkan oleh karakter “|”. Isi paket data *failsafe* adalah *header* berupa karakter “\$failsafe”, sinyal *failsafe* dan diakhiri dengan karakter “>”. Paket data tersebut perlu dilakukan *parsing*, data *string* yang telah diurai perlu dikonversi agar dapat diolah menjadi parameter *failsafe* ASV. Pada paket data *failsafe*, variabel data *string* dikonversi menjadi *integer* dengan perintah *atoi*. Pemrograman dilakukan dengan membuat fungsi *recFailsafe* seperti *listing* program berikut:

```
void recFailsafe() {
    char * strtokIndx;
    strtokIndx = strtok(tempChar, "|");
    strcpy(dataGCS, strtokIndx);
    strtokIndx = strtok(NULL, "|");
    Failsafe = atoi(strtokIndx);
}
```

BAB IV

PENGUJIAN DAN ANALISIS

4.1 Pengujian Perangkat Keras

Pengujian perangkat keras dilakukan untuk melihat kemampuan jangkauan sinyal dari modul telemetri yang digunakan. Setelah kemampuan modul telemetri diketahui, akan dijadikan sebagai batasan dalam mengoperasikan ASV agar mampu menjalankan misi dengan optimal. Pengujian jangkauan sinyal dilakukan untuk mengetahui seberapa jauh data dapat dikirim atau diterima dengan baik. Pengujian dilakukan dengan mengirim data posisi dan arah ke GCS. Data yang diterima komputer ditampilkan secara *realtime* pada GCS. Hasil pengujian jangkauan sinyal modul radio 3DR 433Mhz dapat dilihat pada Tabel 4.1.

Tabel 4.1 Pengujian jangkauan sinyal modul radio 3DR 433Mhz.

No	Jarak antara <i>receiver</i> dan <i>transmitter</i> (m)	Status data
1	20	Data diterima
2	40	Data diterima
3	60	Data diterima
4	80	Data diterima
5	100	Data diterima
6	115,5	Data tidak diterima

Pengujian dilakukan di Jl. Setapak Waduk mulai dari koordinat -7,0530157 dan 110,4463016 sampai -7,0520131 dan 110,4460270 seperti pada Gambar 4.1.

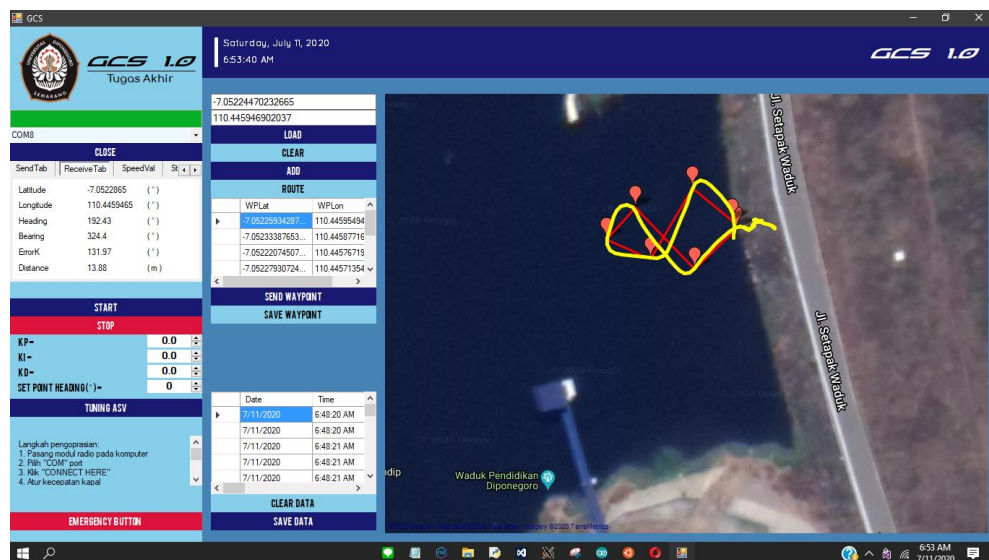


Gambar 4.1 Pengujian jangkauan sinyal modul radio 3DR 433Mhz.

Modul radio 3DR 433Mhz memiliki antena bertipe *omnidirectional* yang memiliki radiasi berbentuk lingkaran pada sumbu horizontal maka nilai 100m merupakan jari-jari lingkaran tersebut.

4.2 Pengujian Perangkat Lunak

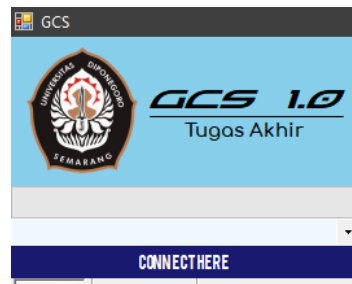
Pengujian perangkat lunak dilakukan untuk melihat apakah GCS yang dirancang dapat berfungsi dengan baik dalam mengoperasikan sebuah ASV. Pengujian dilakukan dengan melakukan pengoperasian sistem navigasi ASV secara *autonomous*, mulai dari fitur koneksi GCS dan ASV, fitur pengaturan *speed* ASV, fitur *start/stop* ASV, fitur *tuning* parameter kendali, fitur *waypoint*, fitur *save data*, dan fitur *failsafe button*. Tampilan GCS ketika pengujian dapat dilihat pada Gambar 4.2.



Gambar 4.2 Tampilan pengujian GCS.

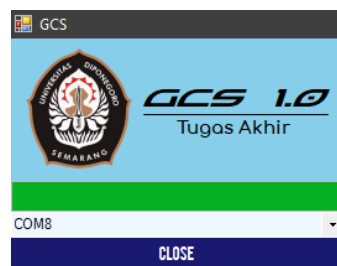
4.2.1 Pengujian Koneksi GCS dan ASV

Pengujian koneksi GCS dan ASV dilakukan untuk mengetahui apakah fitur tersebut dapat berfungsi dengan baik tanpa adanya *error*. Pengujian dilakukan dengan menghubungkan *ground module* radio 3DR 433Mhz pada laptop dan *air module* radio 3DR 433Mhz pada ASV, setelah itu pilih *port* COM yang tersedia dan menekan tombol "CONNECT HERE" pada GCS seperti pada Gambar 4.3.



Gambar 4.3 Pengujian tombol “CONNECT HERE”.

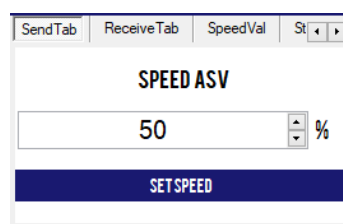
Setelah tombol ditekan, koneksi berhasil dilakukan, *progressbar* penuh berwarna hijau dan tombol berubah bertuliskan “CLOSE” seperti pada Gambar 4.4. Apabila ingin memutuskan koneksi GCS dengan ASV dapat dilakukan dengan menekan tombol “CLOSE”.



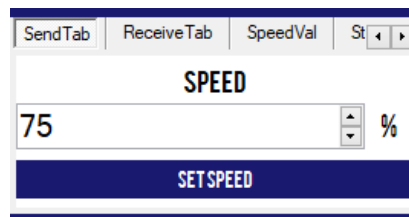
Gambar 4.4 Pengujian tombol “CLOSE”.

4.2.2 Pengujian *Telecontrolling* Kecepatan ASV

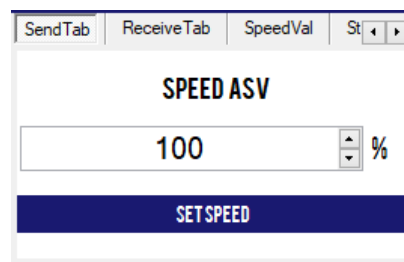
Pengujian *telecontrolling* kecepatan ASV dilakukan untuk mengetahui apakah nilai kecepatan yang diatur melalui GCS dapat diolah dan digunakan oleh ASV. Pengujian dilakukan dengan memberi nilai kecepatan melalui GCS kemudian dikirimkan ke ASV dengan menekan tombol “SET SPEED” seperti Gambar 4.5, Gambar 4.6 dan Gambar 4.7.



Gambar 4.5 Pengujian *telecontrolling* kecepatan ASV variasi 1.



Gambar 4.6 Pengujian *telecontrolling* kecepatan ASV variasi 2.



Gambar 4.7 Pengujian *telecontrolling* kecepatan ASV variasi 3.

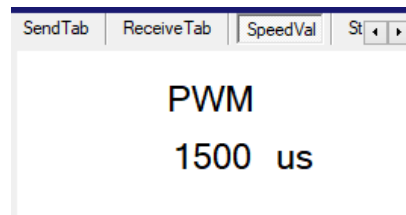
Pengujian *telecontrolling* kecepatan dengan 3 variasi presentase kecepatan menunjukkan tingkat keberhasilan 100% dibuktikan dengan nilai umpan balik PWM secara *realtime* pada tab “SpeedVal” seperti pada Gambar 4.8, Gambar 4.9 dan Gambar 4.10.



Gambar 4.8 Pengujian umpan balik *telecontrolling* kecepatan ASV variasi 1.



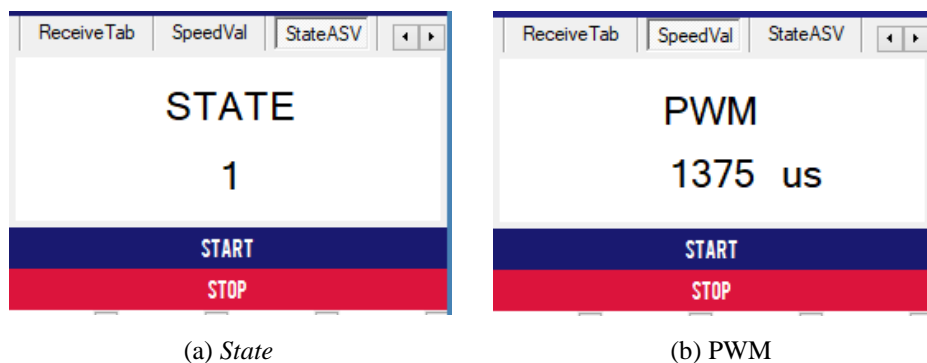
Gambar 4.9 Pengujian umpan balik *telecontrolling* kecepatan ASV variasi 2.



Gambar 4.10 Pengujian umpan balik *telecontrolling* kecepatan ASV variasi 3.

4.2.3 Pengujian *Telecontrolling Start/Stop ASV*

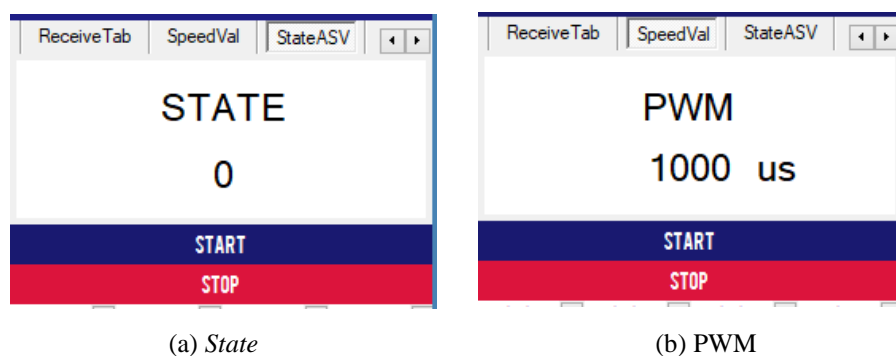
Pengujian *start* dan *stop* ASV dilakukan untuk mengetahui apakah ASV dapat diperintahkan untuk memulai dan menghentikan misi melalui GCS. Ketika tombol “START” ditekan akan memberikan nilai 1 dan nilai PWM sesuai yang ditentukan, sedangkan saat tombol “STOP” ditekan akan memberikan nilai 0 dan nilai PWM 1000 seperti pada Gambar 4.11 dan Gambar 4.12.



(a) *State*

(b) *PWM*

Gambar 4.11 Pengujian *telecontrolling start* ASV.



(a) *State*

(b) *PWM*

Gambar 4.12 Pengujian *telecontrolling stop* ASV.

4.2.4 Pengujian *Telecontrolling Tuning Parameter Kendali*

Pengujian *telecontrolling tuning* parameter kendali dilakukan untuk mengetahui apakah ASV dapat di-*tuning* menggunakan GCS secara *wireless*. Pengujian dilakukan dengan mengirimkan nilai 3 variasi nilai Kp, Ki, Kd dan SP ke ASV. Nilai parameter *tuning* berhasil dikirim ke ASV dengan tingkat keberhasilan 100% dibuktikan dengan umpan balik dari ASV pada tab “TuningVal” seperti pada Gambar 4.13, Gambar 4.14 dan Gambar 4.15.

KP-	1.5
KI-	0.3
KD-	0.2
SET POINT HEADING(°)-	45

TUNING ASV

(a) *Tuning* parameter kendali.

SpeedVal	StateASV	TuningVal	Emergency
Kp: 1.5 Kd: 0.2			
Ki: 0.3 Sp: 45 °			

(b) Umpan balik *tuning* parameter kendali

Gambar 4.13 Pengujian *telecontrolling tuning* parameter kendali variasi 1.

KP-	1.5
KI-	0.6
KD-	0.2
SET POINT HEADING(°)-	90

TUNING ASV

(a) *Tuning* parameter kendali.

SpeedVal	StateASV	TuningVal	Emergency
Kp: 1.5 Kd: 0.2			
Ki: 0.6 Sp: 90 °			

(b) Umpan balik *tuning* parameter kendali

Gambar 4.14 Pengujian *telecontrolling tuning* parameter kendali variasi 2.

KP-	1.0
KI-	0.6
KD-	0.1
SET POINT HEADING(°)-	75

TUNING ASV

(a) *Tuning* parameter kendali.

SpeedVal	StateASV	TuningVal	Emergency
Kp: 1 Kd: 0.1			
Ki: 0.6 Sp: 75 °			

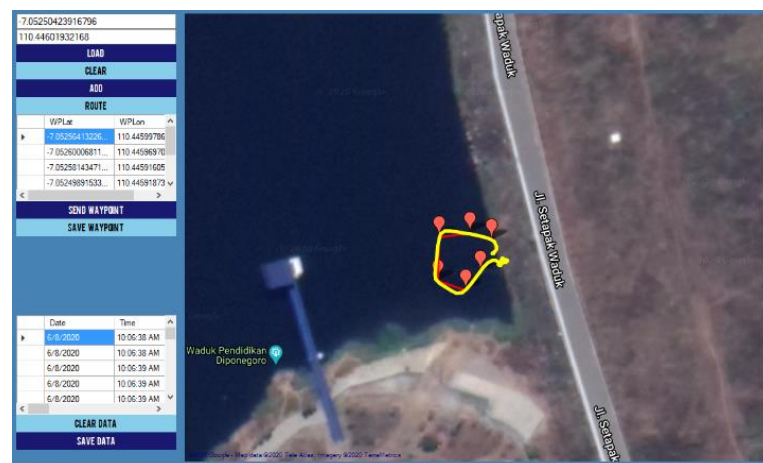
(b) Umpan balik *tuning* parameter kendali

Gambar 4.15 Pengujian *telecontrolling tuning* parameter kendali variasi 3.

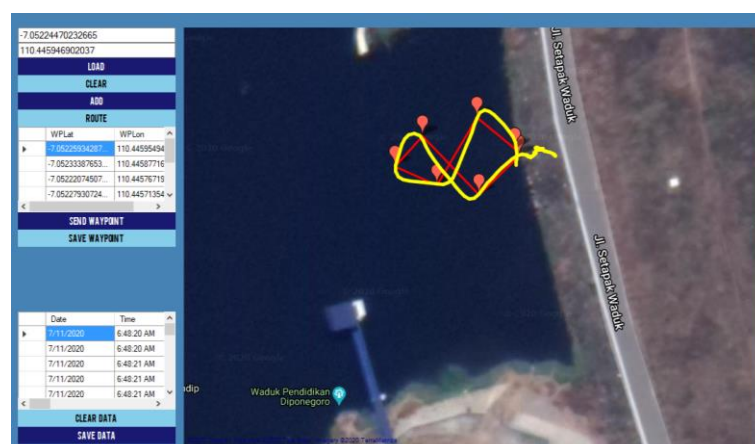
4.2.5 Pengujian *Telecontrolling Waypoint ASV*

Pengujian *telecontrolling waypoint* dilakukan untuk mengetahui apakah GCS yang dirancang dapat membuat *waypoint* dengan benar, mengirim nilai

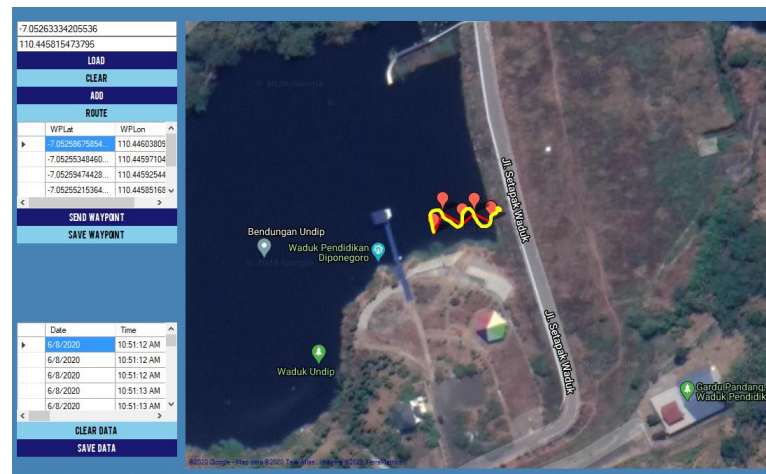
waypoint ke ASV dan menyimpannya dalam dokumen csv. Pengujian dilakukan dengan cara membuat *waypoint* target yang akan dilalui oleh ASV, kemudian dilihat umpan balik berupa jalur yang telah dilalui oleh ASV. Jumlah maksimal titik *waypoint* yang dapat dibuat adalah 7 buah karena dikirimkan secara bersamaan menggunakan komunikasi serial sehingga jumlah data yang dapat dikirimkan terbatas. *Waypoint* dengan 3 variasi berhasil dibuat dan ASV dapat berjalan menuju target *waypoint*, garis berwarna merah merupakan target rute yang akan dilalui oleh ASV dan garis berwarna kuning merupakan lintasan yang telah dilalui ASV seperti pada Gambar 4.16, Gambar 4.17 dan Gambar 4.18.



Gambar 4.16 Pengujian *telecontrolling waypoint* GCS variasi 1.



Gambar 4.17 Pengujian *telecontrolling waypoint* GCS variasi 2.



Gambar 4.18 Pengujian *telecontrolling waypoint* GCS variasi 3.

Data *waypoint* dapat disimpan dalam bentuk dokumen csv dan dibuka di Microsoft Excel seperti pada Gambar 4.19.

	A	B	C	D	E
1	WPLat	WPLon			
2	-7.052564132	110.4459979			
3	-7.052600068	110.4459697			
4	-7.052581435	110.4459161			
5	-7.052498915	110.4459187			
6	-7.05249093	110.4459777			
7	-7.052504239	110.4460193			

Gambar 4.19 Dokumen Data_WP.csv.

4.2.6 Pengujian *Telecontrolling Failsafe ASV*

Pengujian *failsafe button* dilakukan untuk mengetahui apakah dalam keadaan tertentu, GCS dapat memberikan sinyal darurat kepada ASV. Keadaan darurat disimulasikan dengan mematikan *remote* secara manual. Ketika *failsafe button* ditekan, GCS memberikan sinyal darurat berupa variabel bertipe *integer* dengan nilai 1, umpan balik dari ASV dapat dilihat pada *tab* “Emergency” seperti pada Gambar 4.20.



Gambar 4.20 Pengujian *telecontrolling failsafe*.

4.2.7 Pengujian *Telemetry* pada GCS

Pengujian *telemetry* dilakukan untuk mengetahui hasil pembacaan dan pengolahan data sensor oleh ASV secara *realtime*, dari 3 kali pengujian, data *latitude*, *longitude*, *heading*, *bearing*, *error* hadap, dan *distance* berhasil ditampilkan melalui tab “ReceiveTab” seperti pada Gambar 4.21, Gambar 4.22 dan Gambar 4.23.

SendTab	ReceiveTab	SpeedVal	St
Latitude	-7.0522865	(°)	
Longitude	110.4459465	(°)	
Heading	192.43	(°)	
Bearing	324.4	(°)	
ErrorK	131.97	(°)	
Distance	13.88	(m)	

Gambar 4.21 Pengujian ke-1 *telemetry* pada GCS.

SendTab	ReceiveTab	SpeedVal	St
Latitude	-7.05224417	(°)	
Longitude	110.44599	(°)	
Heading	10.14	(°)	
Bearing	246.39	(°)	
ErrorK	-123.76	(°)	
Distance	4.21	(m)	

Gambar 4.22 Pengujian ke-2 *telemetry* pada GCS.

SendTab	ReceiveTab	SpeedVal	St
Latitude	-7.05249917	(°)	
Longitude	110.446062	(°)	
Heading	355.45	(°)	
Bearing	277.5	(°)	
ErrorK	-77.96	(°)	
Distance	12236185	(m)	

Gambar 4.23 Pengujian ke-3 *telemetering* pada GCS.

Data *telemetering* juga dapat dilihat pada tampilan tabel data ASV seperti Gambar 4.24.

	PWM	St
4	1000	0
4	1000	0
4	1000	0
4	1000	0
4	1000	0

CLEAR DATA
SAVE DATA

Gambar 4.24 Tampilan tabel data ASV.

4.2.8 Pengujian *Save Data*

Pengujian fitur *save* data dilakukan untuk mengetahui apakah GCS dapat menyimpan data yang diterima ke dalam dokumen berbentuk csv. Pengujian berhasil dilakukan setelah ASV terhubung dengan GCS kemudian menyimpan data yang diterima ke dalam dokumen Data_ASV.csv dan dibuka di Microsoft Excel seperti pada Gambar 4.25. Pengujian *save* data juga membuktikan bahwa sistem *parsing* data yang dirancang dapat berfungsi, sehingga masing-masing data tersimpan sesuai pada tempatnya.

BAB V

PENUTUP

5.1 Kesimpulan

1. Jangkauan transmisi maksimal antara antena *transmitter* dan *receiver* modul radio 3DR 433Mhz adalah 100 meter.
2. Komunikasi antara GCS dan ASV dapat dilakukan secara *wireless* menggunakan modul radio 3DR 433Mhz.
3. GCS dapat menggerakkan dan menghentikan ASV melalui fitur *start* dan *stop*.
4. GCS dapat mengirim sinyal darurat melalui *failsafe button*.
5. *Telecontrolling* kecepatan ASV dengan 3 variasi nilai kecepatan 50%, 75% dan 100% memiliki tingkat keberhasilan 100%.
6. *Telecontrolling tuning* parameter kendali dengan 3 variasi nilai Kp, Ki, Kd dan *Setpoint* memiliki tingkat keberhasilan 100%.
7. *Telecontrolling waypoint* berhasil dilakukan dengan 3 variasi *waypoint* berjumlah maksimal 7 titik dan memiliki tingkat keberhasilan 100% dalam pengiriman dan umpan balik rute ASV.
8. *Telemetry* ASV dari 3 kali pengujian memiliki tingkat keberhasilan 100% dalam menampilkan data sesuai dengan *parsing* yang ditentukan.

5.2 Saran

1. Perancangan GCS dapat dilakukan menggunakan IDE yang mendukung pengembangan perangkat lunak *multi device* agar mendukung lebih banyak perangkat.
2. Dilakukan pengembangan algoritma pemrograman GCS agar jumlah *waypoint* yang diterima ASV tidak terbatas.
3. Antena *omnidirectional* dapat diganti dengan antena bertipe *directional* apabila menginginkan transmisi jarak jauh, atau mengganti media komunikasi menggunakan *internet/satelit*.

DAFTAR PUSTAKA

- [1] R. Lasabuda, "Pembangunan Wilayah Pesisir dan Lautan dalam Perspektif Negara Kepulauan Republik Indonesia," *Jurnal Ilmiah Platax*, vol. 1-2, no. 1, hal. 92-101, 2013.
- [2] S. P. K. Soedarmo, *Mengelola Laut untuk Kesejahteraan Rakyat*, Semarang, Indonesia: Undip Press, 2018.
- [3] K. Kobatake, T. Okazaki, dan M. Arima, "Study on Optimal Tuning of PID Autopilot for Autonomous Surface Vehicle," *International Federation of Automatic Control*, vol. 52, no. 21 hal. 335-340, 2019.
- [4] I. Gonzalez-Reolid, J. Carlos Molina-Molina, A. Guerrero-Gonzalez, F. J. Ortiz, dan D. Alonso. "An Autonomous Solar-Powered Marine Robotic Observatory for Permanent Monitoring of Large Areas of Shallow Water," *Sensors*. vol. 18, no. 10, hal. 1-24, 2018.
- [5] Z. H. Munim, "Autonomous Ships: A Review, Innovate Applications and Future Maritime Bussines Models," *Supply Chain Forum: An International Journal*, vol. 20, no. 4, hal. 266-279, 2019.
- [6] P. Vasile, C. Ciaoca, D. Luculeseu, A. Luchian, dan S. Pop, "Consideration about UAV command and control. Ground Control Station," dalam *5th International Scientific Conference SEA-CONF*, 2019, hal 1-9.
- [7] M. Rivai, "Autonomous Surface Vehicle sebagai Alat Pemantau Lingkungan Menggunakan Metode Navigasi Waypoint," *Jurnal Teknik ITS*, vol. 7, no. 1, hal. 76-81, 2018.
- [8] R Dhanasingaraja, S. Kalaimagal, dan G. Muralidharan, "Autonomous Vehicle Navigation and Mapping System," *International Journal of Innovation Research in Science, Engineering and Technology*, vol. 3, no. 3, hal. 1347-1350, 2014.
- [9] Y. Kwon, J. Heo, S. Jeong, S. Yu, S. Kim, "Analysis of Design Direction for Ground Control Station (GCS)," *Journal of Computer and Communication*, vol. 4, no. 15, hal 1-7, 2016.
- [10] *Electrical, Instrumentation and SCADA System Design*, ACWWA Standard 3.0, 2010.

- [11] S. Kavitha, S. Shindu, “Comparisson of Integrated Development Environment (IDE) Debugging Tools: Eclipse vs Netbeans,” *International Research Journal of Engineering and Technology*, vol. 2, no. 4, hal. 432-437, 2015.
- [12] Suprpto, K. T. Yuwono, T. Sukardiyono, dan A. Dewanto, Bahasa Pemrograman untuk Sekolah Menengah Kejuruan, Jakarta, Indonesia: Direktorat Pembinaan Sekolah Menengah Kejuruan, 2008.
- [13] S. Kurniawan, D. K. Halim, H. Dicky, dan C. M. Tang, “Multicore development environment for embedded processor in Arduino IDE,” *TELKOMNIKA Telecommunication, Computing, Electronics and Control*, vol. 18, no.2, hal. 870-878, 2019.
- [14] N. Rachmad, R. Subagja, “Rancang Bangun Antena Omni Collinear Sebagai Antena Wireless Penguat Modem Wireless,” *Jurnal ICT*, vol. 5, no. 9, hal. 8-17, 2014.
- [15] 3D Robotics Inc., “3DR Radio V2 Quick Start Guide,” *Datasheet*, hal.1-4, 2013.
- [16] D. Hariyanto, A. C. Nugraha, dan A. Asmara, “Design and Development of an Asynchronus Serial Communication Learning Media to Visualize the Bit Data,” dalam *International Conference on Electrical, Electronic, Informatics and Vocational Education (ICE-ELINVO 2018)*, Yogyakarta, Indonesia, Nov. 13, 2018, hal. 1-9.

BIODATA MAHASISWA



Nama Mahasiswa : Ibrahim
NIM : 21060116130099
Konsentrasi : Teknik Kendali
Tempat/Tgl. Lahir : Jakarta, 22 September 1998
Alamat Sekarang : Kp. Balimatraman Rt.006/09,
Manggarai, Tebet, Jakarta Selatan
No. Telepon/HP : 083807602198
Alamat e-mail : ibrohimalbayyan@yahoo.co.id
Nama Orang Tua : Supriyanto
Alamat Orang Tua : Kp. Balimatraman Rt.006/09,
Manggarai, Tebet, Jakarta Selatan
IP Kumulatif : 3.29

Pengalaman dan prestasi yang pernah diraih:

1. Staff Muda Syi'ar Biro Al-Muhandis 2016
2. Ketua Departemen Syi'ar Biro Al-Muhandis 2017
3. Ketua Pelaksana Electrical Line Follower Competition 2017
4. Staff Muda KMI Himpunan Mahasiswa Elektro 2018
5. Juara 2 Kontes Robot Sepakbola Beroda Indonesia Regional II 2018
6. Asisten Praktikum Mikroprosesor 2018
7. Kerja Praktik Lembaga Penerbangan dan Antariksa Nasional 2019
8. 8th Position International Roboboat Competition 2019
9. Asisten Praktikum Kontrol Proses Manufaktur 2020
10. Koordinator Praktikum Kontrol Digital 2020

Semarang, 15 Agustus 2020

Ibrahim

21060116130099

LAMPIRAN A

SENARAI PROGRAM C#

```

/* Undergraduate Final Project
 * Electrical Engineering Diponegoro University
 * By : Ibrahim (21060116130099)
 * My repo: github.com/ibmgrx
 */

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Drawing.Drawing2D;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.IO.Ports;
using GMap.NET;
using GMap.NET.WindowsForms;
using GMap.NET.WindowsForms.Markers;
using GMap.NET.MapProviders;
using System.IO;

namespace GCS
{
    public partial class Form1 : Form
    {
        static SerialPort port;
        string dataIn, start="0";
        double lon_gps;
        decimal s, vKp, vKi, vKd, vSp;
        int i,j,k;
        double lat_gps;
        List<PointLatLng> points;
        double latN, lonN, latB, lonB, head, spK, error,
            lim, jarak, pwm, rSt, rKp, rKi, rKd, rSp, rEm;
        String WP1Lat, WP1Lon, WP2Lat, WP2Lon,
            WP3Lat, WP3Lon, WP4Lat, WP4Lon,
            WP5Lat, WP5Lon, WP6Lat, WP6Lon,
            WP7Lat, WP7Lon;

        public Form1()
        {
            InitializeComponent();
            port = new SerialPort();
            gmap.MapProvider = GoogleHybridMapProvider.Instance;
            GMaps.Instance.Mode = AccessMode.ServerOnly;
            points = new List<PointLatLng>();

```

```

        timer1.Start();
        Tanggal.Text = DateTime.Now.ToLongDateString();
        Waktu.Text = DateTime.Now.ToLongTimeString();
    }

    private void timer1_Tick(object sender, EventArgs e)
    {
        timer1.Start();
        Tanggal.Text = DateTime.Now.ToLongDateString();
        Waktu.Text = DateTime.Now.ToLongTimeString();
    }

    private void ComboBox1_Click(object sender, EventArgs e)
    {
        string[] ports = SerialPort.GetPortNames();
        PortNumber.Items.Clear();
        PortNumber.Items.AddRange(ports);
    }

    private void OpenClose_Click(object sender, EventArgs e)
    {
        if (OpenClose.Text == "CONNECT HERE")
        {
            if (PortNumber.Text.Length > 1)
            {
                port = new SerialPort(PortNumber.Text, 57600,
Parity.None, 8, StopBits.One);

                port.Open();
                port.DataReceived += DataReceived;

                progressBar1.Value = 100;
                OpenClose.Text = "Close";
            }
        }
        else
        {
            port.Close();
            progressBar1.Value = 0;
            timer1.Stop();
            OpenClose.Text = "CONNECT HERE";
        }
    }

    void DataReceived(object sender,
System.IO.Ports.SerialDataReceivedEventArgs e)
    {
        dataIn = port.ReadLine();
        this.Invoke(new EventHandler(get_data));
        k = k + 100;
    }

```

```

private void get_data(object sender, EventArgs e)
{
    try
    {
        System.Globalization.CultureInfo customCulture =
        (System.Globalization.CultureInfo)System.Threading.Thread.CurrentT
hread.CurrentCulture.Clone();
        customCulture.NumberFormat.NumberDecimalSeparator
        = ".";

        System.Threading.Thread.CurrentThread.CurrentCulture =
        customCulture;//format penulisan desimal

        if (dataIn.StartsWith("#"))
        {
            string[] dataArray = dataIn.Split(',');
            latN = double.Parse(dataArray[1],
customCulture);
            lonN = double.Parse(dataArray[2],
customCulture);
            latB = double.Parse(dataArray[3],
customCulture);
            lonB = double.Parse(dataArray[4],
customCulture);
            head = double.Parse(dataArray[5],
customCulture);
            spK = double.Parse(dataArray[6],
customCulture);
            error = double.Parse(dataArray[7],
customCulture);
            jarak = double.Parse(dataArray[8],
customCulture);
            pwm = double.Parse(dataArray[9],
customCulture);
            rSt = double.Parse(dataArray[10],
customCulture);
            rKp = double.Parse(dataArray[11],
customCulture);
            rKi = double.Parse(dataArray[12],
customCulture);
            rKd = double.Parse(dataArray[13],
customCulture);
            rSp = double.Parse(dataArray[14],
customCulture);
            rEm = double.Parse(dataArray[15],
customCulture);
        }

        dataGridView2.Rows.Add();
        dataGridView2.Rows[j].Cells[0].Value =
DateTime.Now.Month + "/" + DateTime.Now.Day + "/" +
DateTime.Now.Year;
        dataGridView2.Rows[j].Cells[1].Value = Waktu.Text;
        dataGridView2.Rows[j].Cells[2].Value = head;
        dataGridView2.Rows[j].Cells[3].Value = latN;
        dataGridView2.Rows[j].Cells[4].Value = lonN;
    }
}

```

```

dataGridView2.Rows[j].Cells[5].Value = spK;
dataGridView2.Rows[j].Cells[6].Value = error;
dataGridView2.Rows[j].Cells[7].Value = jarak;
dataGridView2.Rows[j].Cells[8].Value = pwm;
dataGridView2.Rows[j].Cells[9].Value = start;
j++;

label14.Text = Convert.ToString(latN);
label15.Text = Convert.ToString(lonN);
label17.Text = Convert.ToString(head);
label17.Text = Convert.ToString(spK);
label12.Text = Convert.ToString(error);
label13.Text = Convert.ToString(jarak);
label19.Text = Convert.ToString(pwm);
label31.Text = Convert.ToString(rSt);
label27.Text = Convert.ToString(rKp);
label28.Text = Convert.ToString(rKi);
label29.Text = Convert.ToString(rKd);
label30.Text = Convert.ToString(rSp);
label22.Text = Convert.ToString(rEm);

lim = latN - lonB;
if (latN != 0 && lonN != 0 && latB != 0 && lonB !=
0)
{
    GMapOverlay jejak = new GMapOverlay("jejak");
    List<PointLatLng> jalur = new
List<PointLatLng>();
    jalur.Add(new PointLatLng(latN, lonN));
    jalur.Add(new PointLatLng(latB, lonB));
    GMapRoute rute = new GMapRoute(jalur, "rute");
    rute.Stroke = new Pen(Color.Yellow, 5);
    rute.Stroke.StartCap = LineCap.Round;
    rute.Stroke.LineJoin = LineJoin.Round;
    rute.Stroke.EndCap = LineCap.Round;
    jejak.Routes.Add(rute);
    gmap.Overlays.Add(jejak);
}
}
catch { }

}

//===== M A P
=====//
private void gmap_Load(object sender, EventArgs e)
{
    gmap.Position = new PointLatLng(-7.05250157725237,
110.445610284805);
    gmap.ShowCenter = false;
    gmap.Zoom = 17;
}

```

```

private void gmap_MouseClick(object sender, MouseEventArgs
e)
{
    if (e.Button == MouseButton.Left)
    {
        var points = gmap.FromLocalToLatLng(e.X, e.Y);
        lat_gps = points.Lat;
        lon_gps = points.Lng;

        txtLat.Text = lat_gps + "";
        txtLon.Text = lon_gps + "";
    }
}

private void loadGPS_Click(object sender, EventArgs e)
{
    try
    {
        GMapOverlay markers = new GMapOverlay("markers");
        lat_gps = Convert.ToDouble(txtLat.Text);
        lon_gps = Convert.ToDouble(txtLon.Text);
        gmap.Position = new PointLatLng(lat_gps, lon_gps);
        gmap.Zoom = 20;

        GMapMarker marker = new GMarkerGoogle(
            new PointLatLng(lat_gps, lon_gps),
            GMarkerGoogleType.red);
        markers.Markers.Add(marker);
        gmap.Overlays.Add(markers);
    }

    catch
    {
        MessageBox.Show("Please    Input    Number",    "Try
Again");
    }
}

public void addPos_Click(object sender, EventArgs e)
{
    try
    {
        points.Add(new
PointLatLng(Convert.ToDouble(txtLat.Text),
            Convert.ToDouble(txtLon.Text)));
        dataGridView1.Rows.Add();
        dataGridView1.Rows[i].Cells[0].Value           =
Convert.ToDouble(txtLat.Text);
        dataGridView1.Rows[i].Cells[1].Value           =
Convert.ToDouble(txtLon.Text);
        i++;
    }
}

```

```

        catch
        {
            MessageBox.Show("Load First");
        }
    }

    public void routBtn_Click(object sender, EventArgs e)
    {
        GMapOverlay routes = new GMapOverlay("routes");
        GMapRoute route = new GMapRoute(points, "rute");
        route.Stroke = new Pen(Color.Red, 3);
        routes.Routes.Add(route);
        gmap.Overlays.Add(routes);
        gmap.Refresh();
    }

    public void clrBtn_Click(object sender, EventArgs e)
    {
        for (int m = 0; m < gmap.Overlays.Count; m = 0)
        {
            gmap.Overlays.RemoveAt(m);
            gmap.Refresh();
        }
    }

    private void sendWP_Click(object sender, EventArgs e)
    {
        if (port.IsOpen)
        {
            if (dataGridView1.Rows[0].Cells[0].Value != null)
            {
                WP1Lat =
                dataGridView1.Rows[0].Cells[0].Value.ToString();
                WP1Lon =
                dataGridView1.Rows[0].Cells[1].Value.ToString();
                if (dataGridView1.Rows[1].Cells[0].Value !=
                null)
                {
                    WP2Lat =
                    dataGridView1.Rows[1].Cells[0].Value.ToString();
                    WP2Lon =
                    dataGridView1.Rows[1].Cells[1].Value.ToString();
                    if (dataGridView1.Rows[2].Cells[0].Value
                    != null)
                    {
                        WP3Lat =
                        dataGridView1.Rows[2].Cells[0].Value.ToString();
                        WP3Lon =
                        dataGridView1.Rows[2].Cells[1].Value.ToString();
                        if
                        (dataGridView1.Rows[3].Cells[0].Value != null)
                        {
                            WP4Lat =
                            dataGridView1.Rows[3].Cells[0].Value.ToString();

```



```

                                WP4Lon           =
dataGridView1.Rows[3].Cells[1].Value.ToString();
                                if
(dataGridView1.Rows[4].Cells[0].Value != null)
                                {
                                    WP5Lat           =
dataGridView1.Rows[4].Cells[0].Value.ToString();
                                    WP5Lon           =
dataGridView1.Rows[4].Cells[1].Value.ToString();
                                    if
(dataGridView1.Rows[5].Cells[0].Value != null)
                                    {
                                        WP6Lat           =
dataGridView1.Rows[5].Cells[0].Value.ToString();
                                        WP6Lon           =
dataGridView1.Rows[5].Cells[1].Value.ToString();
                                        if
(dataGridView1.Rows[6].Cells[0].Value != null)
                                        {
                                            WP7Lat           =
dataGridView1.Rows[6].Cells[0].Value.ToString();
                                            WP7Lon           =
dataGridView1.Rows[6].Cells[1].Value.ToString();
                                        }
                                        else { WP7Lat = "0"; WP7Lon
= "0"; }
                                    }
                                    else { WP6Lat = "0"; WP6Lon =
"0";
                                        WP7Lat = "0"; WP7Lon =
"0"; }
                                }
                                else { WP5Lat = "0"; WP5Lon = "0";
WP6Lat = "0"; WP6Lon = "0";
WP7Lat = "0"; WP7Lon = "0";
}
                                }
                                else { WP4Lat = "0"; WP4Lon = "0";
WP5Lat = "0"; WP5Lon = "0";
WP6Lat = "0"; WP6Lon = "0";
WP7Lat = "0"; WP7Lon = "0"; }
                                }
                                else { WP3Lat = "0"; WP3Lon = "0";
WP4Lat = "0"; WP4Lon = "0";
WP5Lat = "0"; WP5Lon = "0";
WP6Lat = "0"; WP6Lon = "0";
WP7Lat = "0"; WP7Lon = "0"; }
                                }
                                else { WP2Lat = "0"; WP2Lon = "0";
WP3Lat = "0"; WP3Lon = "0";
WP4Lat = "0"; WP4Lon = "0";
WP5Lat = "0"; WP5Lon = "0";
WP6Lat = "0"; WP6Lon = "0";
WP7Lat = "0"; WP7Lon = "0"; }
                                }
}

```

```

else
{
    WP1Lat = "0"; WP1Lon = "0";
    WP2Lat = "0"; WP2Lon = "0";
    WP3Lat = "0"; WP3Lon = "0";
    WP4Lat = "0"; WP4Lon = "0";
    WP5Lat = "0"; WP5Lon = "0";
    WP6Lat = "0"; WP6Lon = "0";
    WP7Lat = "0"; WP7Lon = "0";
}
port.Write("!" + "wp" + "|"
    + WP1Lat + "|" + WP1Lon + "|"
    + WP2Lat + "|" + WP2Lon + "|"
    + WP3Lat + "|" + WP3Lon + "|"
    + WP4Lat + "|" + WP4Lon + "|"
    + WP5Lat + "|" + WP5Lon + "|"
    + WP6Lat + "|" + WP6Lon + "|"
    + WP7Lat + "|" + WP7Lon + ">");
MessageBox.Show("!" + "wp" + "|"
    + WP1Lat + "|" + WP1Lon + "|"
    + WP2Lat + "|" + WP2Lon + "|"
    + WP3Lat + "|" + WP3Lon + "|"
    + WP4Lat + "|" + WP4Lon + "|"
    + WP5Lat + "|" + WP5Lon + "|"
    + WP6Lat + "|" + WP6Lon + "|"
    + WP7Lat + "|" + WP7Lon + ">");
}
}

private void saveWP_Click(object sender, EventArgs e)
{
    SaveWPtoCSV();
}

private void SaveWPtoCSV()
{
    string filename = "";
    SaveFileDialog sfd = new SaveFileDialog();
    sfd.Filter = "CSV (*.csv)|*.csv";
    sfd.FileName = "Data_WP.csv";
    if (sfd.ShowDialog() == DialogResult.OK)
    {
        MessageBox.Show("Data akan disimpan.");
        if (File.Exists(filename))
        {
            try
            {
                File.Delete(filename);
            }
            catch (IOException ex)
            {
                MessageBox.Show("Error" + ex.Message);
            }
        }
        int columnCount = dataGridView1.ColumnCount;
        string columnNames = "";
    }
}

```

```

        string[] output = new string[dataGridView1.RowCount
+ 1];
        for (int i = 0; i < columnCount; i++)
        {
            columnNames                                     +=
dataGridView1.Columns[i].Name.ToString() + ",";
        }
        output[0] += columnNames;
        for (int i = 1; (i - 1) < dataGridView1.RowCount;
i++)
        {
            for (int j = 0; j < columnCount; j++)
            {
                if (dataGridView1.Rows[i -
1].Cells[j].Value != null) output[i] += dataGridView1.Rows[i -
1].Cells[j].Value.ToString() + ",";
            }
        }
        System.IO.File.WriteAllLines(sfd.FileName, output,
System.Text.Encoding.UTF8);
        MessageBox.Show("Data berhasil disimpan.");
    }
}

//===== S P E E D
=====//
private void speed_ValueChanged(object sender, EventArgs e)
{
    s = ((speed.Value)*5) + 1000;
}

private void btnSend(object sender, EventArgs e)
{
    if (port.IsOpen)
    {
        String ok = s.ToString();
        port.Write("@" + "speed" + "|" + ok + ">");
    }
}

//===== T U N I N G
=====//
private void Kp_ValueChanged(object sender, EventArgs e)
{
    vKp = Kp.Value;
}

private void Ki_ValueChanged(object sender, EventArgs e)
{
    vKi = Ki.Value;
}

private void Kd_ValueChanged(object sender, EventArgs e)
{
    vKd = Kd.Value;
}

```

```

private void Sp_ValueChanged(object sender, EventArgs e)
{
    vSp = Sp.Value;
}

private void btnTuning_Click(object sender, EventArgs e)
{
    if (port.IsOpen)
    {
        String kpValue = vKp.ToString();
        String kiValue = vKi.ToString();
        String kdValue = vKd.ToString();
        String spValue = vSp.ToString();
        port.Write("^" + "tuning" + "|" + kpValue + "|" +
kiValue + "|" + kdValue + "|" + spValue + ">");
    }
}

//===== S T A R T / S T O P
=====//
private void btnStart_Click(object sender, EventArgs e)
{
    if (port.IsOpen)
    {
        start = "1";
        port.Write("#" + "start" + "|" + start + ">");
    }
}

private void btnStop_Click(object sender, EventArgs e)
{
    if (port.IsOpen)
    {
        start = "0";
        port.Write("#" + "start" + "|" + start + ">");
    }
}

//===== F A I L S A F E
=====//
private void btnEmg_Click(object sender, EventArgs e)
{
    if (port.IsOpen)
    {
        String danger = "1";
        port.Write("$" + "emergency" + "|" + danger + ">");
    }
}

private void saveBtn_Click(object sender, EventArgs e)
{
    SaveToCSV();
}

```

```

//===== S A V E F I L E
=====//
private void SaveToCSV()
{
    string filename = "";
    SaveFileDialog sfd = new SaveFileDialog();
    sfd.Filter = "CSV (*.csv)|*.csv";
    sfd.FileName = "Data_ASV.csv";
    if (sfd.ShowDialog() == DialogResult.OK)
    {
        MessageBox.Show("Data akan disimpan.");
        if (File.Exists(filename))
        {
            try
            {
                File.Delete(filename);
            }
            catch (IOException ex)
            {
                MessageBox.Show("Error" + ex.Message);
            }
        }
        int columnCount = dataGridView2.ColumnCount;
        string columnNames = "";
        string[] output = new string[dataGridView2.RowCount
+ 1];

        for (int i = 0; i < columnCount; i++)
        {
            columnNames +=
dataGridView2.Columns[i].Name.ToString() + ",";
        }
        output[0] += columnNames;
        for (int i = 1; (i - 1) < dataGridView2.RowCount;
i++)
        {
            for (int j = 0; j < columnCount; j++)
            {
                if (dataGridView2.Rows[i -
1].Cells[j].Value != null) output[i] += dataGridView2.Rows[i -
1].Cells[j].Value.ToString() + ",";
            }
        }
        System.IO.File.WriteAllLines(sfd.FileName, output,
System.Text.Encoding.UTF8);
        MessageBox.Show("Data berhasil disimpan.");
    }
}

private void clrData_Click(object sender, EventArgs e)
{
    reset();
    j = 0;
}

```

```
private void reset()
{
    int numRows = dataGridView2.Rows.Count;
    for (int l = 0; l < numRows; l++)
    {
        try
        {
            int max = dataGridView2.Rows.Count - 1;

dataGridView2.Rows.Remove(dataGridView2.Rows[max]);
        }
        catch (Exception exe)
        {
            MessageBox.Show("try again" + exe, "deleted",
                MessageBoxButtons.OK, MessageBoxIcon.Information);
        }
    }
}
```

LAMPIRAN B

SENARAI PROGRAM C

```

void sendData() {
    GCS.print("#DAT");
    GCS.print(",");
    GCS.print(latN, 8);
    GCS.print(",");
    GCS.print(lonN, 8);
    GCS.print(",");
    GCS.print(latB, 8);
    GCS.print(",");
    GCS.print(lonB, 8);
    GCS.print(",");
    GCS.print(head);
    GCS.print(",");
    GCS.print(bearing);
    GCS.print(",");
    GCS.print(error);
    GCS.print(",");
    GCS.print(jarak);
    GCS.print(",");
    GCS.print(rspeed);
    GCS.print(",");
    GCS.print(Start);
    GCS.print(",");
    GCS.print(tuning[0]);
    GCS.print(",");
    GCS.print(tuning[1]);
    GCS.print(",");
    GCS.print(tuning[2]);
    GCS.print(",");
    GCS.print(tuning[3]);
    GCS.print(",");
    GCS.print(Failsafe);
    GCS.print(",");
    GCS.print("!");
    GCS.println();
}

void rec() {
    static bool recData = false;
    static byte ndx;
    char endl = '>';
    char c;

    while (GCS.available() > 0 && newData == false) {
        c = GCS.read();

        if (recData == true) {
            if (c != endl) {
                dataRec[ndx] = c;
                ndx++;
            }
        }
    }
}

```

```

        if (ndx >= dataByte) {
            ndx = dataByte - 1;
        }
    }
    else {
        dataRec[ndx] = '\\0';
        recData = false;
        ndx = 0;
        newData = true;
    }
}
else if (c == '!') {
    recData = true;
    parsing = 1;
    //Parse Waypoint
}
else if (c == '@') {
    recData = true;
    parsing = 2;
    //Parse Speed
}
else if (c == '#') {
    recData = true;
    parsing = 3;
    //Parse Start
}
else if (c == '$') {
    recData = true;
    parsing = 4;
    //Parse Failsafe
}
else if (c == '^') {
    recData = true;
    parsing = 5;
    //Parse Tuning
}
}
}

void recData() {
    rec();
    if (newData == true) {
        strcpy(tempChar, dataRec);
        if (parsing == 1) {
            recWaypoint();
        }
        else if (parsing == 2) {
            recSpeed();
        }
        else if (parsing == 3) {
            recStart();
        }
        else if (parsing == 4) {
            recFailsafe();
        }
        else if (parsing == 5) {

```



```

        recTuning();
    }
    newData = false;
}

void recWaypoint() {
    char * strtokIndx;
    strtokIndx = strtok(tempChar, "|");
    strcpy(dataGCS, strtokIndx);
    strtokIndx = strtok(NULL, "|");
    WP[0] = atof(strtokIndx);
    strtokIndx = strtok(NULL, "|");
    WP[1] = atof(strtokIndx);
    strtokIndx = strtok(NULL, "|");
    WP[2] = atof(strtokIndx);
    strtokIndx = strtok(NULL, "|");
    WP[3] = atof(strtokIndx);
    strtokIndx = strtok(NULL, "|");
    WP[4] = atof(strtokIndx);
    strtokIndx = strtok(NULL, "|");
    WP[5] = atof(strtokIndx);
    strtokIndx = strtok(NULL, "|");
    WP[6] = atof(strtokIndx);
    strtokIndx = strtok(NULL, "|");
    WP[7] = atof(strtokIndx);
    strtokIndx = strtok(NULL, "|");
    WP[8] = atof(strtokIndx);
    strtokIndx = strtok(NULL, "|");
    WP[9] = atof(strtokIndx);
    strtokIndx = strtok(NULL, "|");
    WP[10] = atof(strtokIndx);
    strtokIndx = strtok(NULL, "|");
    WP[11] = atof(strtokIndx);
    strtokIndx = strtok(NULL, "|");
    WP[12] = atof(strtokIndx);
    strtokIndx = strtok(NULL, "|");
    WP[13] = atof(strtokIndx);
    strtokIndx = strtok(NULL, "|");
}

void recSpeed() {
    char * strtokIndx;
    strtokIndx = strtok(tempChar, "|");
    strcpy(dataGCS, strtokIndx);
    strtokIndx = strtok(NULL, "|");
    rSpeed = atof(strtokIndx);
}

void recStart() {
    char * strtokIndx;
    strtokIndx = strtok(tempChar, "|");
    strcpy(dataGCS, strtokIndx);
    strtokIndx = strtok(NULL, "|");
    Start = atof(strtokIndx);
}

```

```

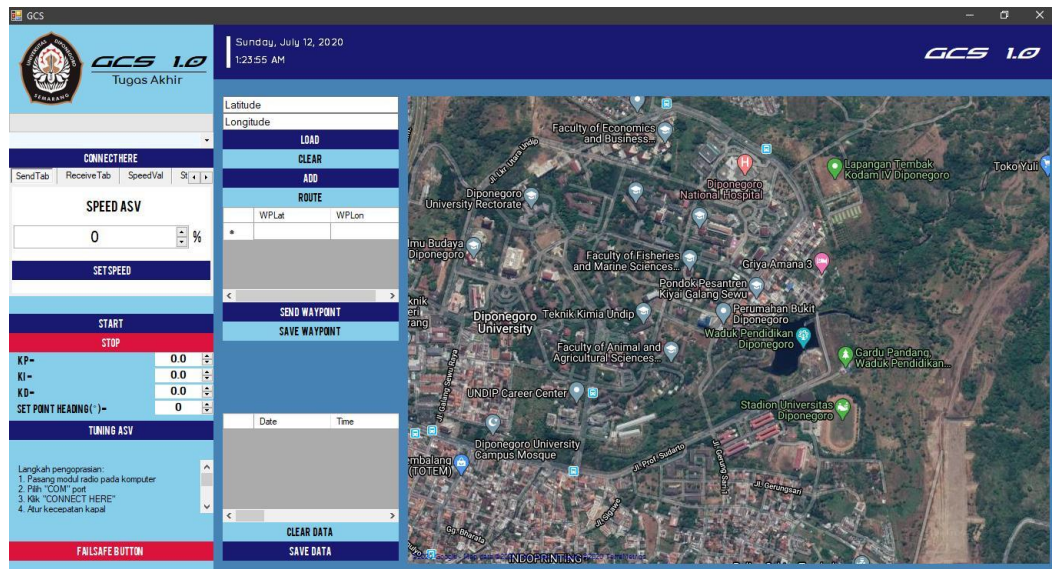
void recTuning() {
    char * strtokIndx;
    strtokIndx = strtok(tempChar, "|");
    strcpy(dataGCS, strtokIndx);
    strtokIndx = strtok(NULL, "|");
    tuning[0] = atof(strtokIndx); //kp
    strtokIndx = strtok(NULL, "|");
    tuning[1] = atof(strtokIndx); //ki
    strtokIndx = strtok(NULL, "|");
    tuning[2] = atof(strtokIndx); //kd
    strtokIndx = strtok(NULL, "|");
    tuning[3] = atof(strtokIndx); //sp
}

void recFailsafe() {
    char * strtokIndx;
    strtokIndx = strtok(tempChar, "|");
    strcpy(dataGCS, strtokIndx);
    strtokIndx = strtok(NULL, "|");
    Failsafe = atof(strtokIndx);
}

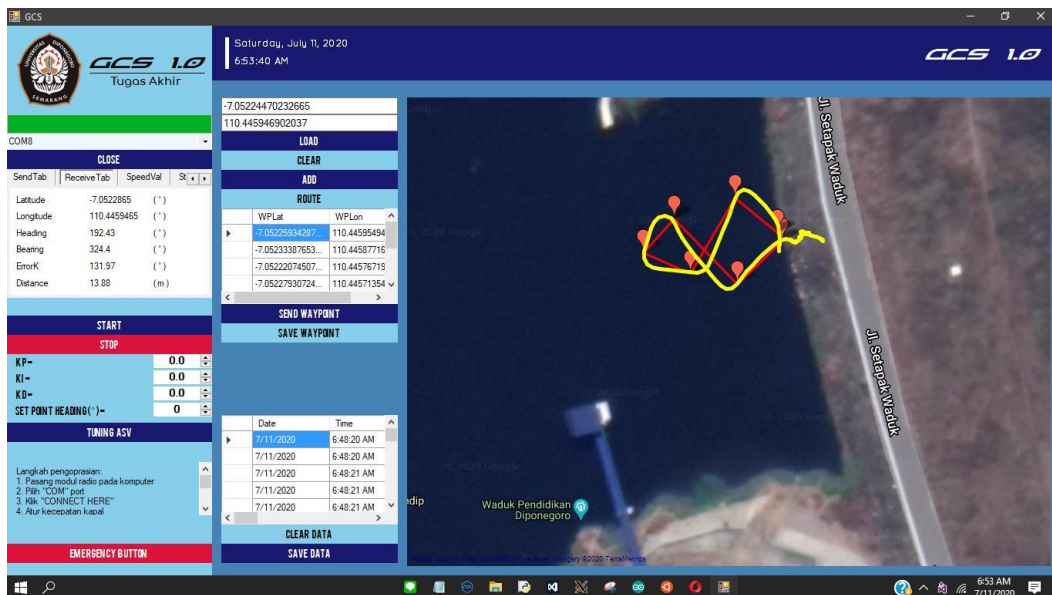
```

LAMPIRAN C TAMPILAN GCS

1. Antarmuka GCS



2. Antarmuka pengujian GCS



LAMPIRAN D

DATASHEET MODUL RADIO 3DR 433MHZ



3DRobotics

3DR RADIO V2

QUICK START GUIDE

GETTING STARTED

3DR Radios provide an air-to-ground data link between the autopilot and your ground station laptop or tablet. Follow this guide to install the radios on your plane, copter, or rover. 3DR Radios arrive ready to use. Just mount and connect to view real-time data from your drone.

PARTS



Two 3DR Radios in 915 or 433 mHz

Attach the antennas, and use either radio as an air or ground module.



Android adapter cable



Micro-USB cable



6-wire Pixhawk connector cable



6-to-5-position APM and PX4 connector cable

LED MEANINGS

	Blinking green Searching for paired radio		Blinking red Transmitting data
	Solid green Link established with paired radio		Solid red Firmware update mode

RADIO DESCRIPTION



SPECIFICATIONS

Processing

100 mW maximum output power (adjustable)
-117 dBm receive sensitivity
Based on HopeRF's HM-TRP module
RP-SMA connector
2-way full-duplex communication through adaptive TDM
UART interface
Transparent serial link
MAVLink protocol framing
Frequency Hopping Spread Spectrum (FHSS)
Configurable duty cycle
Error correction corrects up to 25% of bit errors
Open-source SIK firmware
Configurable through Mission Planner & APM Planner

Features

Interchangeable air and ground modules
915 or 433 MHz
Micro-USB port
6-position DF13 connector

Dimensions

26.7 cm x 55.5 cm x 13.3 cm
(without antenna)

Power

Supply voltage: 3.7-6 VDC (from USB or DF13)
Transmit current: 100 mA at 30 dBm
Receive current: 25 mA
Serial interface: 3.3 V UART

SUPPORT

For more information about mission planner applications and APM firmware, visit ardupilot.com. For online documentation of 3DR Radios, visit goo.gl/Tsrkst.

For customer support, contact us at help@3drobotics.com or call our support line at +1 (858) 225-1414.

Wireless : 3DR Radio Telemetry Kit - 433 Mhz - RoboticsPK

3DR Radio Telemetry Kit - 433 Mhz



Rating: Not Rated Yet

Price:

Variant price modifier:

Price with discount: 12,500.00 PKR

Salesprice with discount:

Sales price: 12,500.00 PKR

Discount:

Tax amount:

[Ask a question about this product](#)

Description

The 3DR Radio telemetry system is designed as an open source Xbee replacement radio set, offering a lower price, longer range (approx 1 mile) and superior performance to Xbee radios. The system provides a full-duplex link using HopeRF's HM-TRP modules running custom, open source firmware. Interface to the module is via standard 5V-tolerant TTL serial / FTDI USB serial. The SiK firmware includes a bootloader that permits radio firmware upgrades over the serial interface, and radio firmware with configurable parameters. Firmware upgrades and configuration are fully supported in the APM Mission Planner. Configuration is also possible through the 3DR Radio Configurator or AT commands.

Hardware features and specifications

- Very small size
- Light weight (under 4 grams without antenna)
- Available in 900MHz or 433MHz variants
- Receiver sensitivity to -121 dBm
- Transmit power up to 20dBm (100mW)
- Transparent serial link
- Air data rates up to 250kbps
- Range of approx 1 mile
- MAVLink protocol framing and status reporting
- Frequency hopping spread spectrum (FHSS)
- Adaptive time division multiplexing (TDM)
- Support for LBT and AFA
- Configurable duty cycle

Wireless : 3DR Radio Telemetry Kit - 433 Mhz - RoboticsPK

- Built in error correcting code (can correct up to 25% data bit errors)
- Demonstrated range of several kilometers with a small omni antenna
- Can be used with a bi-directional amplifier for even more range
- Open source firmware
- AT commands for radio configuration
- RT commands for remote radio configuration
- Adaptive flow control when used with APM
- Based on the HopeRF HM-TRP radio module, featuring an SiLabs Si1000 RF microcontroller.

The following items are included with this product:

- 2 x "3DR Radio 433Mhz "Air" module (Europe)"
- 1 x "Telemetry Cable for APM 1.X and APM 2.0"
- 2 x "6 Pin Right Angle Male Header"
- 1 x "FTDI Cable 3.3V"
- 2 x "Antenna 433MHz RP-SMA 2dBi"

[3DR Radio Configurator](#)

[Documentation and Setup Guide](#)

Reviews

There are yet no reviews for this product.

LAMPIRAN E
MAKALAH TUGAS AKHIR

PERANCANGAN *TELECONTROLLING* DAN *TELEMETERING* PADA *GROUND CONTROL STATION* UNTUK PURWARUPA *AUTONOMOUS SURFACE VEHICLE*

Ibrahim^{*)}, Dr. Wahyudi, S.T., M.T., Eko Handoyo S.T., M.T.

Program Studi Sarjana, Departemen Teknik Elektro, Universitas Diponegoro
Jl. Prof. Sudharto, SH, Kampus UNDIP Tembalang, Semarang 50275, Indonesia

^{*)}E-mail: ibrahim@students.undip.ac.id

Abstrak

Ground Control Station (GCS) digunakan untuk mengendalikan dan memantau sebuah kendaraan tanpa awak secara nirkabel. GCS dirancang agar mampu memberikan informasi dan kendali yang cukup, sehingga misi yang diberikan kepada Autonomous Surface Vehicle (ASV) dapat terlaksana secara autonomus dengan intervensi telecontrolling yang minimal dari operator. Tugas Akhir ini membahas mengenai perancangan GCS dalam mengoperasikan sebuah ASV. Microsoft Visual Studio Express 2012 digunakan dalam pengembangan perangkat lunak GCS. Modul telemetri radio 3DR 433Mhz digunakan sebagai media komunikasi antara GCS dan ASV. Berdasarkan pengujian, jarak optimal penggunaan modul radio 3DR 433Mhz untuk mengirim dan menerima data adalah 100 meter. Fitur telecontrolling pada GCS yang dirancang berupa parameter kecepatan, start/stop ASV, tuning PID, waypoint, dan failsafe. Fitur telemetering pada GCS yang dirancang berupa data latitude, longitude, heading, bearing, error hadap, jarak ASV ke waypoint selanjutnya, juga umpan balik PWM, state misi, nilai Kp, Ki, Kd, Sp, dan failsafe. Fitur lain pada GCS adalah menampilkan peta hybrid dan menyimpan data ASV dalam dokumen csv.

Kata Kunci: GCS, ASV, waypoint, telemetering, dan telecontrolling

Abstract

Ground Control Station (GCS) is used to control and monitoring an unmanned vehicle wirelessly. GCS is designed to provide sufficient information and control, so that the mission given to Autonomous Surface Vehicle (ASV) can be completed autonomously with minimal telecontrolling intervention from the operator. This final project is about design of GCS to operate an ASV. Microsoft Visual Studio Express 2012 is used to develop GCS software. The 3DR 433Mhz radio telemetry module is used as a communication medium between GCS and ASV. Based on the experiment, the optimal range of 3DR 433Mhz radio module is 100 meters for transmit and receive data. The telecontrolling features in the GCS that was designed are speed parameter, start/stop ASV, PID tuning, waypoint, and failsafe. The telemetering features in the GCS that was designed are latitude data, longitude data, heading data, bearing data, error heading data, distance to next waypoint, also PWM feedback, mission feedback, Kp, Ki, Kd, Sp value and failsafe. The other features in GCS that was designed are showing hybrid map and save the ASV data to csv file.

Keywords: GCS, ASV, waypoint, telemetering, and telecontrolling

1. Pendahuluan

1.1 Latar Belakang

Indonesia merupakan negara yang memiliki wilayah perairan lebih luas dari daratannya dengan luas sekitar 5,9 juta km² [1]. Seiring perkembangan waktu, sektor maritim Indonesia sudah semestinya ikut berkembang. Namun pada kenyataannya Indonesia belum memanfaatkan secara maksimal sumber daya kelautan yang dimilikinya [2]. *Autonomous Surface Vehicle* (ASV) baru-baru ini dikembangkan oleh beberapa negara untuk menunjang aktifitas penelitian wilayah

perairan. ASV merupakan sebuah kapal tanpa awak yang mampu menyusuri perairan secara otomatis. ASV mampu bergerak di permukaan air secara otomatis dari suatu lokasi ke lokasi lain dengan bantuan sebuah sistem navigasi berupa *waypoint* di mana titik-titiknya telah ditentukan sebelumnya [3]. Perkembangan ASV di dunia sudah pesat. Saat ini survei batimetri dan oseanografi dapat dilakukan dengan menggunakan ASV, seperti Delfim, Sesamo, IRIS, SCOUT, dan ROAZ, yang dapat digunakan pada perairan tawar maupun laut. USV memiliki manfaat yang cukup besar dalam observasi perairan [4]. Perkembangan ini belum diikuti dengan baik di Indonesia, sehingga perlu

dilakukan pengembangan lebih lanjut pada teknologi ASV.

Pada umumnya ASV dilengkapi dengan GPS, kompas, telemetri, serta sensor pengukur parameter lingkungan perairan bila dibutuhkan [5]. Menurut observatorium oseanografi pengukuran variabel fisika, kimia dan biologis di perairan pantai secara *realtime* dapat membantu pengambilan keputusan dalam pengelolaan seperti perubahan iklim, bencana alam dan kondisi ekosistem [6]. Dalam pengembangannya ASV tidak hanya digunakan untuk observasi melainkan sebagai moda transportasi air [7]. Untuk mengoperasikan sebuah kendaraan *autonomous* seperti ASV, digunakan sebuah *Ground Control Station* (GCS) yang memudahkan operator untuk melakukan pengaturan dan *monitoring* [8].

GCS adalah sebuah sistem yang digunakan untuk mengendalikan dan memantau sebuah kendaraan tanpa awak secara nirkabel [9]. GCS dapat berupa sistem pada perangkat komputer atau perangkat *mobile* seperti *smartphone* [8]. GCS dirancang agar mampu memberikan informasi dan kendali yang cukup, sehingga misi yang diberikan kepada ASV dapat terlaksana secara *autonomous* dengan intervensi yang minimal dari operator [9]. Dalam Tugas Akhir ini dirancang sebuah GCS yang memiliki beberapa fitur mendasar *telemetry* dan *telecontrolling* yang diperlukan agar sebuah ASV dapat menyelesaikan misi yang diberikan. GCS dirancang pada sebuah perangkat komputer dan menggunakan modul radio 3DR 433Mhz untuk berkomunikasi dengan ASV.

1.2 Batasan Masalah

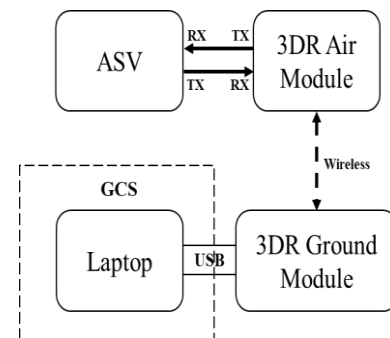
Dalam penyusunan Tugas Akhir ini, telah ditentukan batasan-batasan masalah sebagai berikut:

1. GCS dirancang dengan sebuah perangkat komputer.
2. Modul komunikasi yang digunakan adalah radio 3D Robotics 433Mhz dengan antenna bertipe *omnidirectional*.
3. Komunikasi yang digunakan adalah serial asinkron dengan *baudrate* 57600 *bit per seconds*.
4. Jumlah *waypoint* maksimal 7 titik.
5. Jangkauan operasi ASV melalui GCS adalah 100 meter.
6. Pengujian dilakukan pada area *outdoor* dengan kondisi cerah.
7. Perancangan perangkat lunak GCS pada laptop menggunakan bahasa pemrograman C# dan perangkat lunak komunikasi data pada ASV menggunakan bahasa pemrograman C.

2. Perancangan

2.1. Perancangan Perangkat Keras

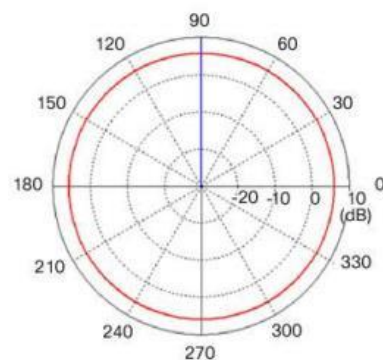
Perancangan perangkat keras terdiri dari modul Radio 3DR 433Mhz dan laptop. Modul radio 3DR 433Mhz sebagai media komunikasi, sedangkan laptop yang akan menjalankan program GCS. Hubungan komponen penyusun sistem dapat dilihat pada Gambar 1.



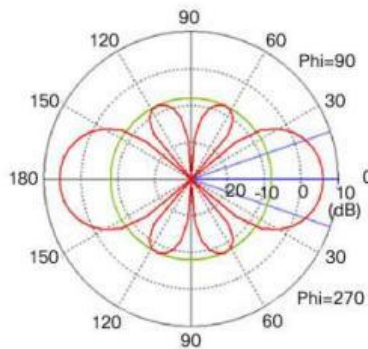
Gambar 1. Konfigurasi perangkat keras

2.1.1 Penggunaan Modul Radio 3DR 433Mhz

Modul Radio 3DR 433Mhz digunakan untuk mengirim dan menerima sinyal informasi secara nirkabel pada rentang frekuensi 433Mhz dari ASV ke GCS ataupun sebaliknya. Untuk memperkuat dan memperluas jangkauan sinyal ditambahkan antenna pada modul telemetri. Antena adalah suatu piranti yang digunakan untuk memancarkan atau meneruskan gelombang elektromagnetik menuju ruang bebas atau menangkap gelombang elektromagnetik dari ruang bebas [10]. Antena yang digunakan oleh modul Radio 3DR 433Mhz adalah antena *omnidirectional*. Antena *omnidirectional* memiliki pola radiasi sinyal ke segala arah dalam sudut 360° dengan daya yang sama [10]. Gambar 2. merupakan radiasi pada sumbu horizontal dan Gambar 3. merupakan radiasi pada sumbu vertikal [10].



Gambar 2. Pola radiasi horizontal antena *omnidirectional*



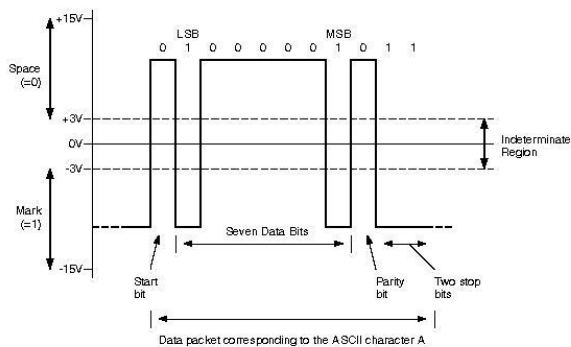
Gambar 3. Pola radiasi vertikal antenna omnidirectional

Modul Radio 3DR 433Mhz seperti pada Gambar 4. memiliki antarmuka UART dengan spesifikasi sensitivitas dalam menerima informasi sebesar -117 dBm, tegangan kerjanya 5V, daya keluaran maksimal sebesar 100mW, *transmit current* sebesar 100mA pada 30 dBm, receive current 25mA, kecepatan transfer di udara mencapai 250 kbps dan memiliki dimensi 26,7 x 55,5 x 13,3 cm tanpa antenna [11].



Gambar 4. Modul radio 3DR 433Mhz

Sistem komunikasi yang digunakan adalah komunikasi serial. Secara umum komunikasi data dibagi menjadi 2 jenis yaitu komunikasi paralel dan komunikasi serial. Komunikasi serial adalah proses pengiriman data suatu bit pada 1 waktu secara sekuensial melalui 1 kanal komunikasi. Komunikasi serial juga terbagi menjadi 2 jenis yaitu sinkron dan asinkron [12]. Komunikasi serial asinkron adalah komunikasi data yang memerlukan start bit untuk menunjukkan mulainya data dan *stop bit* untuk menunjukkan selesainya data.



Gambar 5. Bit data serial asinkron

Pada Gambar 5. merupakan contoh pengiriman karakter A secara serial asinkron, dimulai dengan 1 *start bit*, tujuh bit data, 1 *parity bit* sebagai *data check* dan diakhiri dengan 1 atau 2 *stop bit* [12]. Berikut ini adalah beberapa istilah dalam protokol komunikasi serial:

1. *Baud rate*

Kecepatan transmisi dalam satuan bit per detik. *Clock* dari pengirim dan penerima harus memiliki *baud rate* yang sama. Pada Tugas Akhir ini digunakan *baud rate* 57600 bps yang artinya *transfer* 1 bit membutuhkan 1/57600 detik atau 0.017361 *milisecond* (ms).

2. *Start bit*

Logika *low* menunjukkan transmisi data dimulai. Kondisi *low* yang terjadi pada *start bit* dinamakan *spacing state*.

3. *Data bit*

Berisi data dengan jumlah 5, 6, 7, atau 8 bit. Bit pertama yang dikirim adalah *LSB* (*Least Significant Bit*).

4. *Parity bit*

Berfungsi untuk mengecek adanya *error* data yang ditransfer. *Parity* dapat bernilai *odd*(ganjil), *even*(genap), dan *none*.

5. *Stop bit*

Satu atau dua bit berlogika *high* akan dikirim setelah data bit atau *parity bit* jika ada *parity bit*. Dengan adanya *stop bit* dapat dipastikan bahwa penerima mempunyai waktu yang cukup untuk menerima karakter berikutnya.

2.1.2 Penggunaan Laptop

Laptop dengan sistem operasi *windows* diperlukan untuk menjalankan perangkat lunak GCS yang dirancang. Laptop yang digunakan menggunakan prosesor AMD A8-2,2Ghz dengan 4 *giga byte random access memory*. *Ground module* radio 3DR 433Mhz dihubungkan ke laptop melalui *port Universal Serial Bus* (USB) seperti pada Gambar 6.

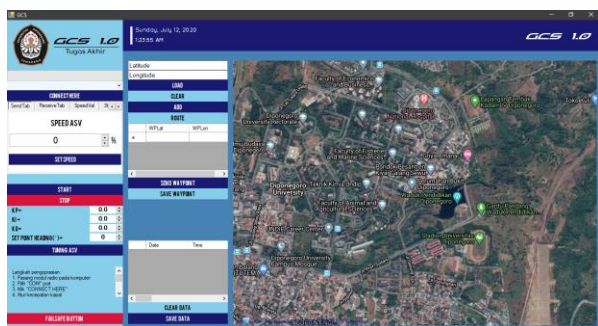


Gambar 6. Penggunaan laptop

2.2 Perancangan Perangkat Lunak

Perancangan perangkat lunak GCS pada laptop dilakukan menggunakan Microsoft Visual Studio Express 2012. Sebagai sebuah *Integrated Development Environment* (IDE), Microsoft Visual Studio menyediakan berbagai fasilitas yang dibutuhkan untuk membangun perangkat lunak. Pada umumnya, IDE memiliki *source code editor*, *built automation tools* dan *debugger*. IDE dirancang untuk meminimalkan *error* dengan adanya *error correction* dan *execution monitoring* [13]. Bahasa pemrograman yang digunakan adalah bahasa pemrograman C#. *C sharp* (C#) merupakan bahasa pemrograman yang diciptakan oleh Microsoft untuk mengembangkan aplikasi Microsoft.NET [14].

GCS yang dirancang memiliki kemampuan *telecontrolling* dan *telemetry* untuk memudahkan operator dalam mengoperasikan ASV. Fitur-fitur yang terdapat pada GCS adalah *Connect/Disconnect* komunikasi, tombol *start/stop* mengoperasikan ASV, mengatur kecepatan, *tuning* PID, menampilkan peta, membuat *waypoint*, menyimpan data selama pengoperasian ASV, dan *failsafe button*. Dalam mengembangkan sebuah GCS untuk sebuah ASV banyak faktor yang harus dipertimbangkan, mulai dari tata letak, skema tampilan, metode komunikasi dan metode pengoperasiannya. Pemilihan warna tampilan HMI pada GCS juga perlu diperhatikan agar sesuai dengan standar yang ada [15]. Tampilan awal GCS dapat dilihat pada Gambar 7.



Gambar 7. Tampilan GCS

Data yang dikirim ASV dan diterima oleh GCS adalah sebuah paket data yang diawali oleh *header* berupa “#” dan diakhiri oleh *endline* berupa karakter “!”. Diantara *header* dan *endline* terdapat 15 data ASV berupa variabel *string* yang dipisahkan oleh *comma* “,”. Apabila *header* data yang diterima GCS adalah “#” maka akan dijalankan fungsi *parsing* untuk memisahkan isi paket data tersebut. Data yang dikirim ASV dan diterima oleh GCS dapat dilihat pada Tabel 1.

Tabel 1. Data *telemetry*

No	Isi Data	Keterangan
1	LatN	Latitude ASV sekarang
2	LonN	Longitude ASV sekarang
3	LatB	Latitude ASV sebelumnya
4	LonB	Longitude ASV sebelumnya
5	head	Sudut hadap ASV
6	spK	Sudut target ASV
7	error	Error sudut ASV
8	jarak	Jarak antara ASV dengan <i>waypoint</i> selanjutnya
9	pwm	PWM motor BLDC
10	rSt	Umpan balik nilai <i>start</i>
11	rKp	Umpan balik nilai Kp
12	rKi	Umpan balik nilai Ki
13	rKd	Umpan balik nilai Kd
14	rSp	Umpan balik <i>setpoint</i>
15	rEm	Umpan balik <i>failsafe</i>

Paket data yang dikirim GCS agar dapat digunakan oleh ASV perlu diolah terlebih dahulu. Pengiriman paket data dari GCS bersifat insidental yaitu ketika operator menekan tombol *telecontrolling* tertentu. Paket data yang dikirim GCS adalah *waypoint*, kecepatan, *start*, *tuning*, dan *failsafe*. Setiap paket data yang dikirim dari GCS memiliki format yang berbeda seperti pada Tabel 2.

Tabel 2. Paket data yang dikirim GCS

No	Paket	Header	Isi	Endline
1	Waypoint	“!wp”	Latitude 1, longitude 1, latitude 2, longitude 2, latitude 3, longitude 3, latitude 4, longitude 4, latitude 5, longitude 5, latitude 6, longitude 6, latitude 7, longitude 7	“>”
2	Kecepatan	“@speed”	Nilai kecepatan	“>”
3	Start	“#start”	Nilai <i>start</i>	“>”
4	Tuning	“^tuning”	Nilai Kp, Ki, Kd, <i>Setpoint</i> Arah	“>”
5	Failsafe	“\$failsafe”	Nilai sinyal <i>failsafe</i>	“>”

3. Hasil dan Pembahasan

3.1 Pengujian Perangkat Keras

Pengujian perangkat keras dilakukan untuk melihat kemampuan jangkauan sinyal dari modul telemetri yang digunakan. Setelah kemampuan modul telemetri diketahui, akan dijadikan sebagai batasan dalam mengoperasikan ASV agar mampu menjalankan misi dengan optimal. Pengujian jangkauan sinyal dilakukan untuk mengetahui seberapa jauh data dapat dikirim atau diterima dengan baik. Pengujian dilakukan dengan mengirim data posisi dan arah ke GCS. Data yang diterima komputer ditampilkan secara *realtime* pada GCS. Hasil pengujian jangkauan sinyal modul radio 3DR 433Mhz dapat dilihat pada Tabel 1.

Tabel 3. Hasil pengujian jangkauan sinyal modul radio 3DR 433Mhz

No	Jarak antara <i>receiver</i> dan <i>transmitter</i> (m)	Status data
1	20	Data diterima
2	40	Data diterima
3	60	Data diterima
4	80	Data diterima
5	100	Data diterima
6	115,5	Data tidak diterima

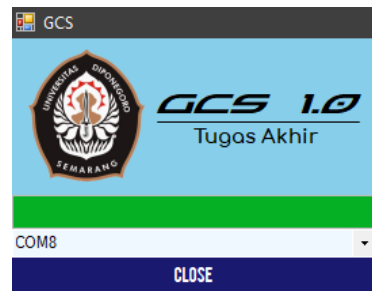
Modul radio 3DR 433Mhz memiliki antena bertipe *omnidirectional* yang memiliki radiasi berbentuk lingkaran pada sumbu horizontal maka nilai 100 meter merupakan jari-jari lingkaran tersebut.

3.2 Pengujian Perangkat Lunak

Pengujian perangkat lunak dilakukan untuk melihat apakah GCS yang dirancang dapat berfungsi dengan baik dalam mengoperasikan sebuah ASV. Pengujian dilakukan dengan melakukan pengoperasian sistem navigasi ASV secara *autonomous*, mulai dari fitur koneksi GCS dan ASV, fitur pengaturan speed ASV, fitur *start/stop* ASV, fitur *tuning* PID, fitur *waypoint*, fitur *save data*, dan fitur *failsafe button*.

3.2.1 Pengujian Koneksi GCS dan ASV

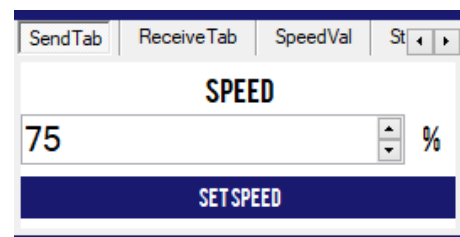
Pengujian dilakukan dengan menghubungkan *ground modul* radio 3DR 433Mhz pada laptop dan *air modul* radio 3DR 433Mhz pada ASV, setelah itu pilih *port COM* yang tersedia dan menekan tombol “CONNECT HERE” pada GCS seperti pada Gambar 8.



Gambar 8. Pengujian koneksi

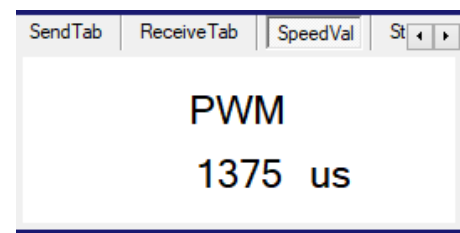
3.2.2 Pengujian *Telecontrolling* Kecepatan ASV

Pengujian *telecontrolling* kecepatan ASV dilakukan untuk mengetahui apakah nilai kecepatan yang diatur melalui GCS dapat diolah dan digunakan oleh ASV. Pengujian dilakukan dengan memberi nilai kecepatan melalui GCS kemudian dikirimkan ke ASV dengan menekan tombol “SET SPEED” seperti Gambar 9.



Gambar 9. Pengujian *telecontrolling* kecepatan ASV

Nilai kecepatan yang diberikan adalah 75%, yang menghasilkan nilai PWM sebesar 1375 *microseconds*. Nilai umpan balik PWM dapat dilihat secara *realtime* pada tab “SpeedVal” seperti pada Gambar 10.

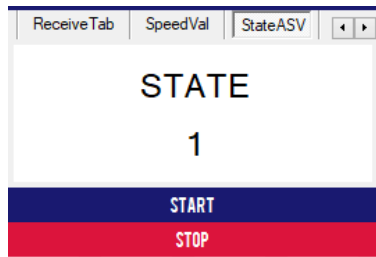


Gambar 10. Pengujian umpan balik *telecontrolling* kecepatan ASV

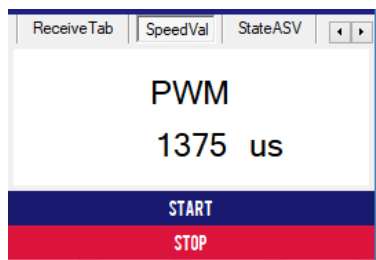
3.2.3 Pengujian *Telecontrolling* Start/Stop ASV

Pengujian start dan stop ASV dilakukan untuk mengetahui apakah ASV dapat diperintahkan untuk memulai dan menghentikan misi melalui GCS. Ketika tombol “START” ditekan akan memberikan nilai 1 dan nilai PWM sesuai yang ditentukan, sedangkan saat

tombol “STOP” ditekan akan memberikan nilai 0 dan nilai PWM 1000 seperti pada Gambar 11. dan Gambar 12.

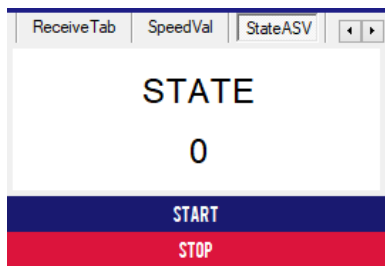


(a) State



(b) PWM

Gambar 11 Pengujian *telecontrolling start ASV*



(a) State



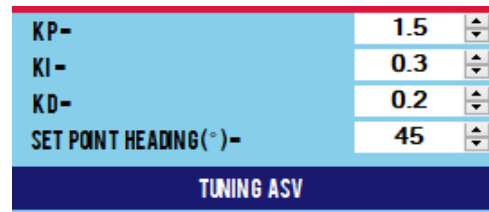
(b) PWM

Gambar 12. Pengujian *telecontrolling stop ASV*

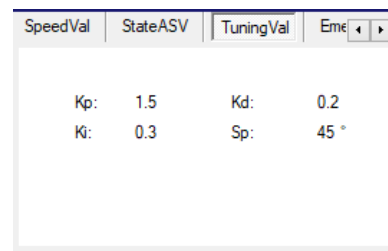
3.2.4 Pengujian *Telecontrolling Tuning Parameter Kendali*

Pengujian *telecontrolling tuning* parameter kendali ASV dilakukan untuk mengetahui apakah ASV dapat di-*tuning* menggunakan GCS secara *wireless*. Pengujian dilakukan dengan mengirimkan nilai Kp, Ki, Kd dan SP ke ASV. Parameter *tuning* yang diberikan adalah Kp=1,5, Ki=0,3, Kd=0,2 dan Sp=45°. Nilai parameter

tuning berhasil dikirim ke ASV dibuktikan dengan umpan balik dari ASV pada tab “TuningVal” seperti pada Gambar 13.



(a) *Tuning parameter kendali*

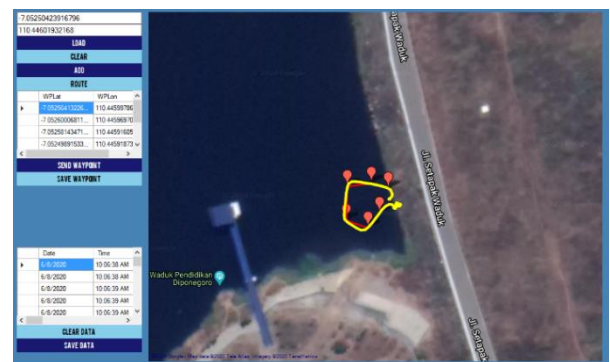


(b) Umpan balik *tuning parameter kendali*

Gambar 13. Pengujian *telecontrolling tuning parameter kendali*

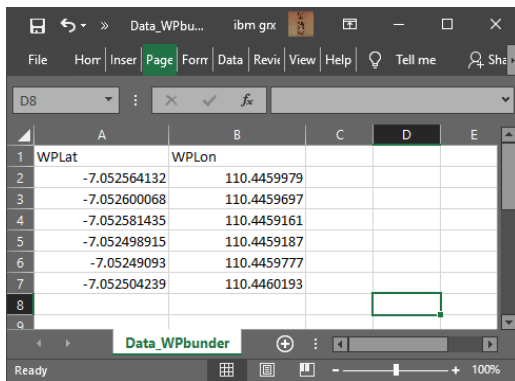
3.2.5 Pengujian *Telecontrolling Waypoint ASV*

Pengujian *telecontrolling waypoint* dilakukan untuk mengetahui apakah GCS yang dirancang dapat membuat *waypoint* dengan benar, mengirim nilai *waypoint* ke ASV dan menyimpannya dalam dokumen csv. Pengujian dilakukan dengan cara membuat *waypoint* target yang akan dilalui oleh ASV, kemudian dilihat umpan balik berupa jalur yang telah dilalui oleh ASV. *Waypoint* berhasil dibuat dan ASV dapat berjalan menuju target *waypoint*, garis berwarna merah merupakan target rute yang akan dilalui oleh ASV dan garis berwarna kuning merupakan lintasan yang telah dilalui ASV seperti pada Gambar 14.



Gambar 14. Pengujian *telecontrolling waypoint GCS*

Data *waypoint* dapat disimpan dalam bentuk dokumen csv dan dibuka di Microsoft Excel seperti pada Gambar 15.

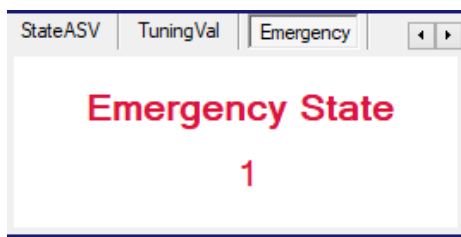


	WPLat	WPLon
1	-7.052564132	110.4459979
2	-7.052600068	110.4459697
3	-7.052581435	110.4459161
4	-7.052498915	110.4459187
5	-7.05249093	110.4459777
6	-7.052504239	110.4460193

Gambar 15. Dokumen Data_WP.csv

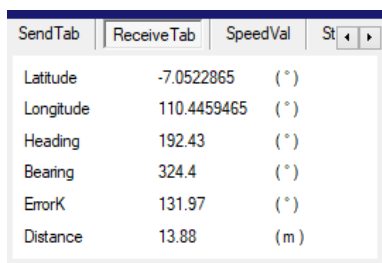
3.2.6 Pengujian *Telecontrolling Failsafe* ASV

Pengujian *failsafe button* dilakukan untuk mengetahui apakah dalam keadaan tertentu, GCS dapat memberikan sinyal darurat kepada ASV. Keadaan darurat disimulasikan dengan mematikan *remote* secara manual. Ketika *failsafe button* ditekan, GCS memberikan sinyal darurat berupa variabel bertipe *integer* dengan nilai 1, umpan balik dari ASV dapat dilihat pada tab “Emergency” seperti pada Gambar 16.

Gambar 16. Pengujian *telecontrolling failsafe*

3.2.7 Pengujian *Telemetry* ASV

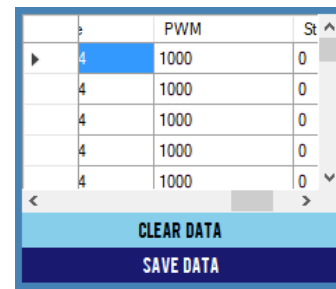
Pengujian *telemetry* dilakukan untuk mengetahui hasil pembacaan dan pengolahan data sensor oleh ASV secara realtime. Data *latitude*, *longitude*, *heading*, *bearing*, *error* hadap, dan *distance* berhasil ditampilkan melalui tab “ReceiveTab” seperti pada Gambar 17.



	SpeedVal	St
Latitude	-7.0522865	(°)
Longitude	110.4459465	(°)
Heading	192.43	(°)
Bearing	324.4	(°)
ErrorK	131.97	(°)
Distance	13.88	(m)

Gambar 17. Pengujian *telemetry* pada GCS

Data *telemetry* juga dapat dilihat pada tampilan tabel data ASV seperti Gambar 18.

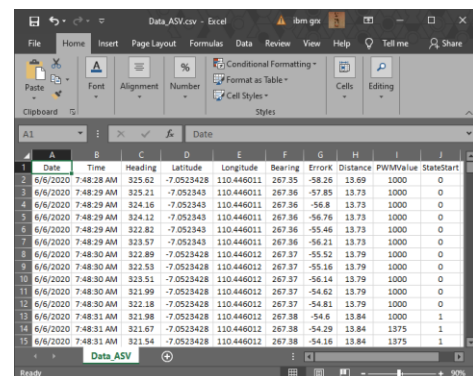


	PWM	St
4	1000	0
4	1000	0
4	1000	0
4	1000	0
4	1000	0

Gambar 18. Tampilan tabel data ASV

3.2.8 Pengujian *Save Data*

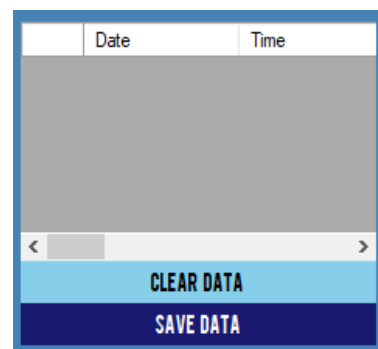
Pengujian fitur *save data* dilakukan untuk mengetahui apakah GCS dapat menyimpan data yang diterima ke dalam dokumen berbentuk csv. Pengujian berhasil dilakukan setelah ASV terhubung dengan GCS kemudian menyimpan data yang diterima ke dalam dokumen Data_ASV.csv dan dibuka di Microsoft Excel seperti pada Gambar 19. Pengujian *save data* juga membuktikan bahwa sistem *parsing* data yang dirancang dapat berfungsi, sehingga masing-masing data tersimpan sesuai pada tempatnya.



	Date	Time	Heading	Latitude	Longitude	Bearing	ErrorK	Distance	Path/Value	StateStart
1	6/6/2020	7:48:28 AM	325.62	-7.0523428	110.446011	267.35	-58.26	13.69	1000	0
2	6/6/2020	7:48:29 AM	325.21	-7.052343	110.446011	267.36	-57.85	13.73	1000	0
3	6/6/2020	7:48:29 AM	324.16	-7.052343	110.446011	267.36	-58.8	13.73	1000	0
4	6/6/2020	7:48:29 AM	324.12	-7.052343	110.446011	267.36	-56.76	13.73	1000	0
5	6/6/2020	7:48:29 AM	322.82	-7.052343	110.446011	267.36	-55.46	13.73	1000	0
6	6/6/2020	7:48:29 AM	323.57	-7.052343	110.446011	267.36	-56.21	13.73	1000	0
7	6/6/2020	7:48:30 AM	322.89	-7.0523428	110.446012	267.37	-55.52	13.79	1000	0
8	6/6/2020	7:48:30 AM	322.53	-7.0523428	110.446012	267.37	-55.16	13.79	1000	0
9	6/6/2020	7:48:30 AM	323.51	-7.0523428	110.446012	267.37	-56.14	13.79	1000	0
10	6/6/2020	7:48:30 AM	321.99	-7.0523428	110.446012	267.37	-54.62	13.79	1000	0
11	6/6/2020	7:48:30 AM	322.18	-7.0523428	110.446012	267.37	-54.81	13.79	1000	0
12	6/6/2020	7:48:31 AM	321.98	-7.0523428	110.446012	267.38	-54.6	13.84	1000	1
13	6/6/2020	7:48:31 AM	321.67	-7.0523428	110.446012	267.38	-54.29	13.84	1375	1
14	6/6/2020	7:48:31 AM	321.54	-7.0523428	110.446012	267.38	-54.16	13.84	1375	1

Gambar 19. Dokumen Data_ASV.csv

Pengujian juga dilakukan dengan menghapus data yang diterima dengan tombol “CLEAR DATA” seperti pada Gambar 20. Pengujian *clear data* berfungsi sehingga data ASV dapat dihapus, apabila GCS menerima data baru maka tabel data akan mulai menyimpan data kembali dari baris pertama.



	Date	Time

Gambar 20. Pengujian *clear data* ASV

4. Kesimpulan

Jangkauan transmisi maksimal antara antenna *transmitter* dan *receiver* modul radio 3DR 433Mhz adalah 100 meter. *Telecontrolling* pada GCS berupa parameter kecepatan, *start/stop* ASV, *tuning* PID, *waypoint*, dan *failsafe* ASV. *Telemetry* pada GCS berupa data *latitude*, *longitude*, *heading*, *bearing*, *error*hadap, jarak ASV ke *waypoint* selanjutnya, juga umpan balik PWM, *state* misi, nilai Kp, Ki, Kd, *Setpoint*, dan *failsafe*. Tampilan peta pada GCS *hybrid* dan menyimpan data ASV dalam dokumen csv.

Referensi

- [1]. R. Lasabuda, "Pembangunan Wilayah Pesisir dan Lautan dalam Perspektif Negara Kepulauan Republik Indonesia," *Jurnal Ilmiah Platax*, vol. 1-2, no. 1, hal. 92-101, 2013.
- [2]. S. P. K. Soedarmo, *Mengelola Laut untuk Kesejahteraan Rakyat*, Semarang, Indonesia: Undip Press, 2018.
- [3]. M. Rivai, "Autonomous Surface Vehicle sebagai Alat Pemantau Lingkungan Menggunakan Metode Navigasi Waypoint," *Jurnal Teknik ITS*, vol. 7, no. 1, hal. 76-81, 2018.
- [4]. K. Kobatake, T. Okazaki, dan M. Arima, "Study on Optimal Tuning of PID Autopilot for Autonomous Surface Vehicle," *International Federation of Automatic Control*, vol. 52, no. 21 hal. 335-340, 2019.
- [5]. R. Dhanasingaraja, S. Kalaimagal, dan G. Muralidharan, "Autonomous Vehicle Navigation and Mapping System," *International Journal of Innovation Research in Science, Engineering and Technology*, vol. 3, no. 3, hal. 1347-1350, 2014.
- [6]. I. Gonzalez-Reolid, J. Carlos Molina-Molina, A. Guerrero-Gonzalez, F. J. Ortiz, dan D. Alonso. "An Autonomous Solar-Powered Marine Robotic Observatory for Permanent Monitoring of Large Areas of Shallow Water," *Sensors*, vol. 18, no. 10, hal. 1-24, 2018.
- [7]. Z. H. Munim, "Autonomous Ships: A Review, Innovate Applications and Future Maritime Business Models," *Supply Chain Forum: An International Journal*, vol. 20, no. 4, hal. 266-279, 2019.
- [8]. P. Vasile, C. Ciaoca, D. Luculescu, A. Luchian, dan S. Pop, "Consideration about UAV command and control. Ground Control Station," dalam *5th International Scientific Conference SEA-CONF*, 2019, hal 1-9.
- [9]. Y. Kwon, J. Heo, S. Jeong, S. Yu, S. Kim, "Analysis of Design Direction for Ground Control Station (GCS)," *Journal of Computer and Communication*, vol. 4, no. 15, hal 1-7, 2016.
- [10]. N. Rachmad, R. Subagja, "Rancang Bangun Antena Omni Collinear Sebagai Antena Wireless Penguat Modem Wireless," *Jurnal ICT*, vol. 5, no. 9, hal. 8-17, 2014.
- [11]. D. Hariyanto, A. C. Nugraha, dan A. Asmara, "Design and Development of an Asynchronous Serial Communication Learning Media to Visualize the Bit Data," dalam *International Conference on Electrical, Electronic, Informatics and Vocational Education (ICE-ELINVO 2018)*, Yogyakarta, Indonesia, Nov. 13, 2018, hal. 1-9.
- [12]. 3D Robotics Inc., "3DR Radio V2 Quick Start Guide," *Datasheet*, hal.1-4, 2013.
- [13]. S. Kavitha, S. Shindu, "Comparisson of Integrated Development Environment (IDE) Debugging Tools: Eclipse vs Netbeans," *International Research Journal of Engineering and Technology*, vol. 2, no. 4, hal. 432-437, 2015.
- [14]. Suprpto, K. T. Yuwono, T. Sukardiyono, dan A. Dewanto, *Bahasa Pemrograman untuk Sekolah Menengah Kejuruan*, Jakarta, Indonesia: Direktorat Pembinaan Sekolah Menengah Kejuruan, 2008.
- [15]. *Electrical, Instrumentation and SCADA System Design*, ACWWA Standard 3.0, 2010.

Biodata



Ibrahim lahir di Jakarta pada tanggal 22 September 1998. Pernah menempuh Pendidikan di SD N Manggarai 05, SMP N 3 Jakarta, dan SMA N 43 Jakarta. Saat ini sedang menempuh Pendidikan S1 Teknik Elektro Universitas Diponegoro bidang Teknik Kendali.

Saya menyatakan bahwa segala informasi yang tersedia di makalah ini adalah benar, merupakan hasil karya sendiri, bebas dari plagiat, dan semua karya orang lain telah dikutip dengan benar.

Ibrahim

21060116130099

Pengesahan

Telah disetujui untuk diajukan pada sidang Tugas Akhir.

Semarang, 27 Agustus 2020

Pembimbing I

Pembimbing II

Dr. Wahyudi, S.T., M.T.
196906121994031001

Eko Handoyo, S.T., M.T
197506082005011001