Majalah Ilmiah Teknologi Elektro, Vol.24, No.2, Jul-Des 2025
DOI: https://doi.org/10.24843/MITE.2025.v24i02.P03

Rancang Bangun Sistem Monitoring Kecepatan Pada Kapal Trimaran Menggunakan Pixhawk

Mochammad Zaqi Arif[1], Edi Kurniawan[2], Shofa Dai Robbi[3]

[Submission: 09-09-2025, Accepted: 14-10-2025]

***Abstract*— Indonesia as a maritime nation with vast and strategic oceanic territory, continues to develop efficient and innovative ship technologies. This research focuses on the design and construction of a *speed monitoring system* for a trimaran vessel using the Pixhawk controller. The system integrates a Pixhawk autopilot unit as the main controller, an STM32 microcontroller for processing, dual BLDC motors, an internal combustion engine (ICE), and a remote control system. The Internet of Things (IoT) platform is employed for real-time data logging and monitoring. Experimental tests were conducted on five reference speed levels slow patrol, reconnaissance, medium patrol, cruising, and chase modes. Quantitative results show that the system achieved rapid response times at medium speeds, with stable operation and an average delay below 0.5 seconds. At high speeds, however, stability decreased by 15–20% due to environmental and load variations. Remote control communication maintained stable performance up to 210 meters with obstructions and 280 meters without obstacles. The results demonstrate that the system effectively supports real-time speed monitoring for trimaran vessels. The novelty of this study lies in the integration of hybrid propulsion and adaptive control within an IoT-based Pixhawk monitoring system, contributing to advancements in smart maritime technology.**

Keywords*— *Pixhawk; Trimaran Vessel; STM3; IoT; Hybrid Propulsion.

***Intisari*— Indonesia sebagai negara maritim dengan wilayah laut yang luas dan strategis terus mendorong pengembangan teknologi kapal yang efisien dan inovatif. Penelitian ini berfokus pada *rancang bangun sistem monitoring kecepatan* pada kapal trimaran menggunakan pengendali Pixhawk. Sistem ini mengintegrasikan unit autopilot Pixhawk sebagai pengendali utama, mikrokontroler STM32 sebagai pemroses data, dua motor BLDC, mesin pembakaran dalam (ICE), serta sistem kendali jarak jauh. Platform *Internet of Things* (IoT) digunakan untuk pencatatan dan pemantauan data secara *real-time*. Pengujian dilakukan pada lima tingkat kecepatan referensi mode patroli lambat, pengintaian, patroli cepat, jelajah, dan pengejaran. Hasil kuantitatif menunjukkan sistem mampu merespons cepat pada kecepatan menengah dengan rata-rata keterlambatan di bawah 0,5 detik serta kestabilan yang baik. Namun, pada kecepatan tinggi terjadi penurunan stabilitas sebesar 15–20% akibat perubahan beban dan kondisi lingkungan. Uji jangkauan kendali jarak jauh menunjukkan sistem tetap responsif hingga jarak 210 meter dengan hambatan dan 280 meter tanpa hambatan. Penelitian ini membuktikan efektivitas sistem monitoring kecepatan kapal berbasis Pixhawk secara *real-time*. Kebaruan penelitian terletak pada integrasi sistem propulsi hibrida dan kontrol adaptif dalam sistem monitoring berbasis IoT yang berkontribusi terhadap kemajuan teknologi maritim cerdas.**

***Kata Kunci*— Pixhawk; Kapal Trimaran; STM32; IoT; Propulsi Hibrida.**

I. PENDAHULUAN

Indonesia merupakan negara maritim dengan posisi geografis yang strategis, terletak di persilangan dua benua dan dua samudera yang menjadi jalur penting perdagangan dunia. Dalam beberapa tahun terakhir, sektor transportasi laut mengalami perkembangan pesat terutama pada desain kapal modern yang menekankan efisiensi, keselamatan, dan pengurangan dampak lingkungan [1]. Salah satu inovasi terkini adalah penggunaan desain kapal *trimaran*, yaitu kapal dengan tiga lambung yang menawarkan keunggulan dalam hal stabilitas, kecepatan, serta ruang interior [2]. *Trimaran* sendiri terbagi menjadi dua tipe, yakni simetris dan asimetris, dengan penerapan yang beragam seperti pada *Autonomous Surface Vehicle* (ASV) dan *Platform Supply Vessel* (PSV) [3], [4].

Pada khususnya ASV, kemajuan teknologi memungkinkan kapal beroperasi secara otomatis menggunakan sistem *waypoint*, sehingga mampu bergerak optimal tanpa keterlibatan awak kapal secara langsung [5]. Teknologi ini sangat penting untuk mendukung kebutuhan pengawasan, logistik, maupun pemantauan di perairan. Salah satu fokus pengembangan ASV adalah integrasi sistem *monitoring* kecepatan kapal yang presisi karena berpengaruh langsung pada efisiensi bahan bakar dan keselamatan pelayaran [6]. Dalam hal ini, perangkat keras Pixhawk telah menjadi pilihan utama dalam berbagai kendaraan nirawak, termasuk kapal, pesawat, dan *rover* [7]. Penelitian terdahulu menunjukkan bahwa Pixhawk memiliki keandalan tinggi sebagai *autopilot* dan dapat mengintegrasikan berbagai sensor seperti IMU, barometer, dan magnetometer untuk menghasilkan kontrol stabil dan data akurat [8].

Berbagai studi juga telah mengkaji integrasi sistem kendali pada kapal otonom. Misalnya, desain *trimaran* terbukti mampu meningkatkan stabilitas secara signifikan dibanding *monohull* [9]. Selain itu, penggunaan Pixhawk dalam sistem *autopilot* kapal nirawak terbukti mampu meningkatkan akurasi navigasi

[1]*Mahasiswa, Jurusan Teknologi Rekayasa Kelistrikan Kapal Politeknik Pelayaran Surabaya, Jln. Gunung Anyar Boulevard No.1 Surabaya (telp: 031-8714643; fax: 031-8714652; e-mail: mochammadzaqiarif@gmail.com*
[2]*Jurusan Teknologi Rekayasa Kelistrikan Kapal Politeknik Pelayaran Surabaya, Jln. Gunung Anyar Boulevard No.1 Surabaya (telp: 031-8714643; fax: 031-8714652; e-mail: edi.kurniawan@poltekpel-sby.ac.id*
[3]*Jurusan Teknologi Rekayasa Permesinan Kapal Politekik Pelayaran Surabaya, Jln. Gunung Anyar Boulevard No.1 Surabaya (telp:031-8714643; fax: 031-8714652; e-mail: shofa_dai@kemenhub.go.id*

dan efisiensi kontrol kecepatan [10]. Namun, penelitian yang secara khusus merancang dan membangun sistem *monitoring* kecepatan pada kapal *trimaran* berbasis Pixhawk masih sangat terbatas. Oleh karena itu, penelitian ini bertujuan untuk mengisi kekosongan tersebut sekaligus memperkaya literatur di bidang teknologi maritim otonom.

Penelitian ini difokuskan pada perancangan dan pembangunan sistem *monitoring* kecepatan kapal *trimaran* dengan menggunakan Pixhawk. Permasalahan yang diangkat mencakup bagaimana merancang sistem *monitoring* kecepatan yang andal serta bagaimana kinerja pengendali kecepatan ketika diaplikasikan pada kapal *trimaran*. Pendekatan penelitian menggunakan prototipe dengan lima mode kecepatan, yaitu pengintaian, patroli lambat, patroli cepat, jelajah, dan pengejaran, yang diintegrasikan dengan penggerak mesin *internal combustion engine* (ICE) dan motor BLDC.

Tujuan penelitian ini adalah merancang sistem *monitoring* kecepatan yang efektif dengan teknologi Pixhawk serta mengevaluasi keandalan pengendalian kecepatan kapal *trimaran*. Hasil penelitian diharapkan dapat memperluas pemahaman teknis mengenai sistem *monitoring* kapal nirawak sekaligus mendukung pengembangan teknologi *autopilot* maritim di Indonesia. Selain itu, penelitian ini memberikan kontribusi berupa literatur baru terkait integrasi Pixhawk pada sistem *monitoring* kapal dan menjadi motivasi pengembangan sistem kontrol serta *autopilot* berbasis teknologi modern di sektor maritim. Dengan demikian, penelitian ini diharapkan menjadi solusi inovatif dalam menghadapi tantangan operasional di perairan serta mendukung visi Indonesia sebagai poros maritim dunia.

II. STUDI PUSTAKA

A. Kapal Trimaran Jenis V

Kapal trimaran merupakan kapal multilambung (multihull) yang terdiri atas satu lambung utama dan dua lambung pendamping yang berfungsi sebagai penyeimbang [11]. Konfigurasi ini memberikan peningkatan stabilitas, daya angkut, dan efisiensi hidrodinamika yang lebih baik dibandingkan monohull. Salah satu bentuk lambung yang banyak digunakan adalah lambung berbentuk huruf *V* (*V-shaped hull*), karena mampu memecah gelombang dengan lebih efektif dan mengurangi hambatan air selama pelayaran [12]. Desain tersebut juga meningkatkan efisiensi bahan bakar serta kenyamanan berlayar pada kecepatan tinggi, menjadikannya ideal untuk kapal cepat seperti trimaran [13].

Gambar 1. Kapal Trimaran Jenis V
(Sumber: https://id.wikipedia.org/wiki/KRI_Klewang)

B. Autonomous Surface Vehicle (ASV)

Autonomous Surface Vehicle (ASV) merupakan kapal tanpa awak yang mampu bergerak secara otomatis menggunakan sistem navigasi berbasis waypoint [14]. Jalur pelayaran ditentukan berdasarkan koordinat GPS yang telah diprogram, dan kapal dapat menyesuaikan arah serta kecepatan secara mandiri. Dalam praktiknya, ASV dilengkapi dengan berbagai sensor seperti GPS, sensor kualitas air, sensor gas, dan sistem telemetri untuk mendukung misi pemantauan, penelitian, maupun patroli laut. Keunggulan utama ASV adalah efisiensi tinggi serta kemampuan operasi jangka panjang tanpa kendali manual.

C. Sistem Propulsi Hibrida

Sistem propulsi hibrida mengombinasikan mesin pembakaran dalam (*Internal Combustion Engine* / ICE) dengan motor listrik yang digerakkan baterai [15]. Kombinasi ini meningkatkan efisiensi energi serta mengurangi emisi gas buang. Dalam aplikasi kelautan, sistem ini memungkinkan kapal beralih antara mode konvensional dan listrik sesuai kebutuhan daya. Teknologi ini juga mendukung konsep kapal ramah lingkungan, khususnya untuk kapal penelitian dan otonom.

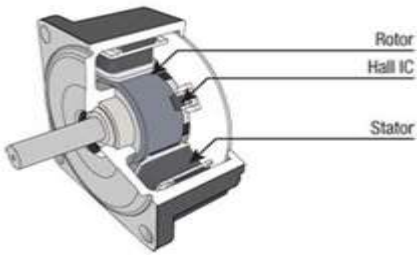
D. Motor Internal Combustion Engine (ICE)

Mesin ICE bekerja dengan prinsip pembakaran bahan bakar di ruang tertutup untuk menghasilkan energi mekanik [16]. Terdapat dua jenis utama, yaitu mesin bensin (*Otto engine*) yang menggunakan busi, dan mesin diesel yang mengandalkan tekanan kompresi tinggi. Mesin ini masih menjadi penggerak utama pada kapal karena daya besar, efisiensi tinggi, dan ketahanan terhadap beban berat.

Gambar 2. *Motor Internal Combustion Engine* (ICE)
(Sumber: https://www.tokopedia.com/almirosa/)

E. Motor Brushless DC (BLDC)

Motor *Brushless DC* (BLDC) menggunakan rotor bermagnet permanen dengan sistem kontrol elektronik tanpa sikat, menghasilkan efisiensi tinggi dan minim perawatan [6]. Motor ini banyak digunakan dalam sistem propulsi kapal modern, terutama pada kapal otonom, karena kemampuannya menjaga kecepatan konstan serta mendukung kontrol kecepatan yang presisi.



Gambar 3. Motor BLDC
(Sumber: http://surl.li/iejuao)

F. Pixhawk

Pixhawk merupakan sistem *autopilot* berbasis mikrokontroler 32-bit yang umum digunakan pada kendaraan nirawak seperti drone dan kapal ASV [17]. Perangkat ini mengintegrasikan berbagai sensor seperti giroskop, akselerometer, kompas, dan GPS, yang bekerja dalam sistem kontrol *feedback loop* tertutup untuk menjaga kestabilan arah dan kecepatan. Fleksibilitas Pixhawk dalam integrasi sensor tambahan menjadikannya pilihan utama dalam pengembangan kapal otonom.

G. Sensor Global Positioning System (GPS)

Global Positioning System (GPS) adalah sistem navigasi berbasis satelit yang menyediakan informasi posisi dan kecepatan kapal secara real-time [18]. Dengan prinsip trilaterasi sinyal dari beberapa satelit, GPS memungkinkan kapal menentukan lintasan *waypoint* secara akurat. Dalam sistem kapal nirawak, GPS juga digunakan untuk evaluasi performa navigasi dan efisiensi jalur tempuh.

H. STM32

STM32 merupakan mikrokontroler berbasis ARM Cortex-M yang menawarkan kinerja tinggi dengan konsumsi daya rendah [1]. Mikrokontroler ini banyak digunakan pada sistem tertanam (*embedded systems*) karena mendukung perhitungan real-time, kontrol motor, dan komunikasi data. Dalam penelitian ini, STM32 berperan penting dalam pengendalian sistem monitoring kecepatan kapal secara otomatis.

I. Electronic Speed Controller (ESC)

Electronic Speed Controller (ESC) berfungsi mengatur kecepatan motor BLDC melalui teknik *Pulse Width Modulation* (PWM) [19]. Alat ini menjadi penghubung antara sumber daya dan motor, mengontrol arus listrik agar perubahan kecepatan berlangsung halus dan efisien. Standar IEEE menyebutkan bahwa PWM merupakan metode paling efektif dalam sistem kendali kecepatan motor kelautan [19].

J. Transistor-Transistor Logic (TTL)

Gerbang logika (*logic gates*) merupakan komponen dasar sistem komputer yang memungkinkan realisasi sistem digital melalui kombinasi bit 0 dan 1. Salah satu bentuk yang umum digunakan adalah *integrated circuit* (IC) *Transistor-Transistor Logic* (TTL) yang tersusun dari transistor BJT dan resistor, berfungsi menjalankan operasi logika sekaligus penguatan [20] Dalam aplikasinya, kabel USB to TTL mendukung komunikasi mikrokontroler dengan perangkat lain, memfasilitasi transfer data cepat, pemrograman, serta integrasi mudah dalam proyek DIY maupun prototipe.

K. Sensor dan Baterai Pendukung Sistem

Untuk memantau kecepatan, digunakan *sensor photodiode* yang membaca putaran motor melalui sinyal inframerah dengan mendeteksi keberadaan objek, gerakan, atau perubahan kondisi. Energi sistem disuplai oleh baterai *Lithium Polymer* (LiPo) yang menggunakan senyawa berbasis lithium sebagai elektroda material, sehingga baterai ini dikenal memiliki densitas energi tinggi serta bobot ringan [20].

L. Remote Control, Receiver Dan ESP32

Remote control Taranis adalah perangkat pengendali jarak jauh yang populer untuk drone, RC, dan pesawat model. Perangkat ini terdiri dari transmitter dan receiver dengan protokol komunikasi andal. Sinyal kendali dari pemancar dimodulasi oleh sinyal carrier berfrekuensi lebih tinggi dibanding sinyal informasi, sehingga transmisi berjalan stabil [21].

ESP32 merupakan mikrokontroler populer dalam pengembangan aplikasi IoT berbiaya rendah namun bertenaga. Dengan prosesor ganda, Wi-Fi, Bluetooth, GPIO, serta konsumsi daya rendah, ESP32 mendukung otomasi rumah, wearable, hingga industri. Kinerjanya ditunjang kecepatan prosesor 240–400 MHz, memori 520 Kb, dan eksternal flash hingga 16 MB [21].

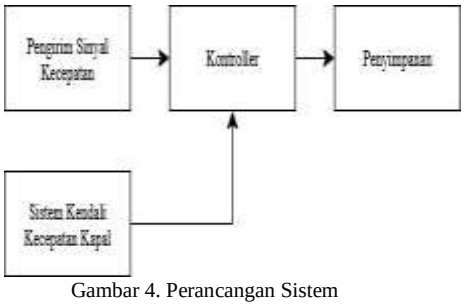
III. *METODOLOGI*

Bab ini menjelaskan metode penelitian eksperimen yang digunakan dalam merancang, mengimplementasikan, dan menguji sistem kendali kecepatan pada kapal trimaran berbasis Pixhawk. Tahapan penelitian meliputi perancangan sistem, perancangan alat, rencana pengujian, analisis data hasil uji, serta algoritma kontrol yang digunakan [20].

A. Perancangan Sistem

Perancangan sistem bertujuan mendesain dan menyusun sistem kendali kapal trimaran yang mampu memenuhi kebutuhan fungsi operasional. Desain sistem mengacu pada prinsip kontrol *closed-loop* yang umum digunakan dalam *autonomous surface vessel* (ASV) [5].

Sistem monitoring kecepatan dirancang dengan alur sinyal dari *remote control* berbasis aplikasi smartphone atau joystick menuju Pixhawk sebagai *main controller*, yang kemudian mengolah data sensor IMU dan GPS untuk menjaga kestabilan kecepatan [16].



Gambar 4. Perancangan Sistem
Sumber: Dokumen Penelitian

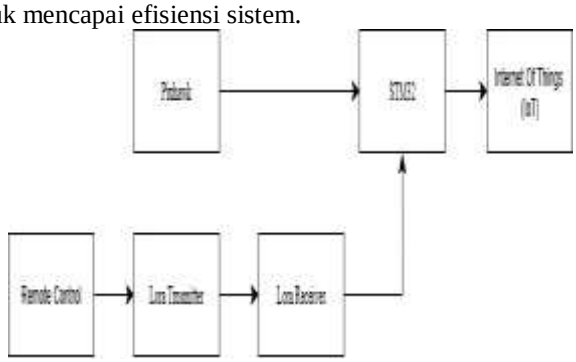
Blok diagram pada Gambar 4 menunjukkan rancangan sistem *monitoring* kecepatan kapal trimaran berbasis Pixhawk. Sistem ini dirancang untuk mengintegrasikan komponen utama seperti *remote control*, mikrokontroler STM32, motor BLDC, serta sensor GPS dan IMU dalam satu kesatuan kendali otomatis. Alur kerja dimulai dari pengirim sinyal kecepatan, yaitu *remote control*, aplikasi *smartphone*, atau *joystick*, yang mengirimkan perintah kecepatan ke Pixhawk sebagai kontroler utama.

Pixhawk kemudian memproses sinyal masukan dan membandingkannya dengan data sensor untuk menghasilkan perintah kontrol yang sesuai [22]. Setelah itu, sistem menggerakkan aktuator berupa motor atau *propeller* untuk menyesuaikan kecepatan kapal sesuai nilai referensi. Sensor GPS dan IMU memberikan umpan balik berupa data kecepatan dan orientasi aktual yang digunakan untuk memastikan kestabilan sistem dan ketepatan respons terhadap perubahan kecepatan.

Seluruh data yang diperoleh dari hasil pembacaan sensor direkam secara otomatis dalam modul penyimpanan berbasis *Internet of Things* (IoT) untuk pencatatan *log* perjalanan, pengujian sistem, serta analisis performa. Sistem ini bekerja dalam arsitektur *closed-loop* yang memungkinkan setiap perubahan kondisi lingkungan direspons secara cepat dan akurat oleh kontroler. Dengan demikian, Pixhawk berfungsi sebagai pusat kendali utama yang mengoordinasikan sinyal masukan, sensor, dan aktuator agar kecepatan kapal tetap stabil, efisien, dan optimal [23].

B. Perancangan Alat

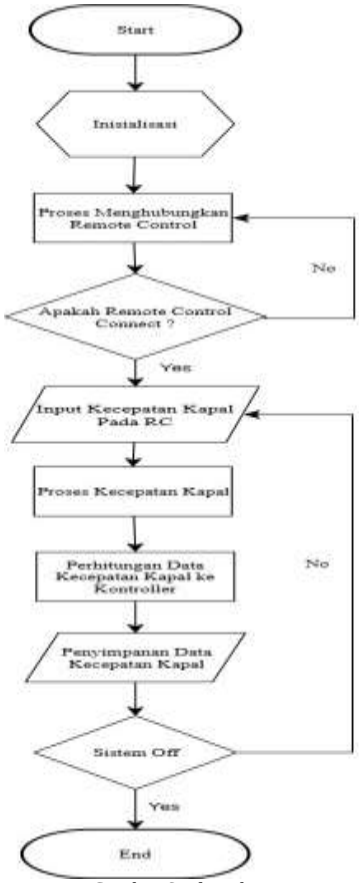
Tahapan perancangan alat merupakan proses integrasi komponen fisik serta perangkat keras pendukung sistem kendali kapal trimaran. Tahapan ini memastikan seluruh komponen bekerja selaras sesuai fungsi yang direncanakan untuk mencapai efisiensi sistem.



Gambar 5. Blok Diagram Perancangan Alat
Sumber: Dokumen Penelitian

Blok diagram pada Gambar 5 menjelaskan komunikasi dan kendali antar komponen. *Remote control* mengirimkan perintah kecepatan melalui LoRa transmitter yang diterima oleh LoRa receiver di kapal sebelum diteruskan ke Pixhawk sebagai kontroler utama. Pixhawk memproses perintah tersebut bersama data sensor IMU dan GPS untuk menghasilkan respon kendali sesuai kebutuhan [24].

Selain itu, STM32 digunakan sebagai mikrokontroler tambahan yang berfungsi untuk pemrosesan data, pencatatan *log*, serta komunikasi antarperangkat. Seluruh data operasional diintegrasikan melalui *Internet of Things* (IoT), memungkinkan sistem dimonitor secara *real-time* [25].



Gambar 6. Flowchart
Sumber: Dokumen Penelitian

Selain Flowchart pada Gambar 6 menunjukkan bahwa sistem dimulai dengan inisialisasi, dilanjutkan dengan pembacaan input dari *remote control*. Setelah perintah kecepatan diterima, sistem melakukan pemrosesan kecepatan kapal, mencatat, dan menyimpan data hingga sistem dimatikan.

Integrasi perangkat keras digambarkan pada *wiring diagram* (Gambar 7), yang menunjukkan koneksi antar komponen seperti Pixhawk, STM32, motor, sensor, dan modul komunikasi. Diagram ini memudahkan proses perakitan serta memastikan setiap komponen terhubung dengan benar.

Gambar 7. Wiring Diagram
Sumber: Dokumen Penelitian

C. Rencana Pengujian

Tahap Tahap pengujian dilakukan untuk mengevaluasi performa sistem yang telah dirancang, mencakup pengujian statis dan dinamis.

Pengujian statis dilakukan untuk memverifikasi keterhubungan perangkat keras dan perangkat lunak tanpa adanya pergerakan kapal. Tujuan utama tahap ini adalah memastikan bahwa Pixhawk, STM32, dan modul IoT dapat saling berkomunikasi secara andal [26]. Data hasil interaksi antar komponen diperiksa melalui penyimpanan IoT untuk menjamin konektivitas dan akuisisi data berjalan baik.

Sementara itu, pengujian dinamis dilakukan dengan mengoperasikan kapal secara langsung di lapangan. Fokus pengujian ini mencakup lima mode operasional, yaitu pengintai, patroli lambat, patroli cepat, jelajah, dan pengejaran [27]. Setiap mode diaktifkan menggunakan *remote control*, dan sistem membaca serta mencatat kecepatan aktual kapal secara otomatis melalui IoT.

Selain itu, pengujian penyimpanan data dilakukan untuk memastikan pencatatan dan penyajian informasi berlangsung tanpa kehilangan data. Data hasil pengujian dapat diakses melalui web IoT maupun tampilan LCD secara *real-time*. Keberhasilan tahap ini menunjukkan bahwa sistem bekerja optimal dari segi akuisisi, pencatatan, dan penyajian data.

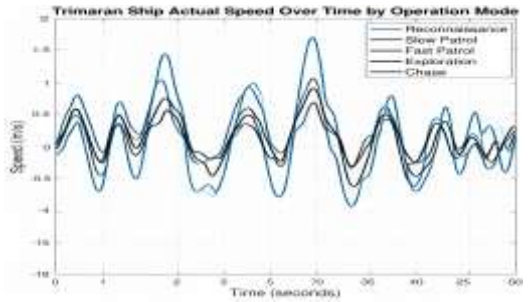
Gambar 8. Diagram Alur Pengujian
Sumber: Dokumen Penelitian

D. Aspek Regulasi dan Keamanan Sistem Komunikasi

Penggunaan modul komunikasi nirkabel LoRa dalam sistem ini mematuhi ketentuan frekuensi radio non-lisensi yang berlaku di Indonesia, yaitu pada rentang 920–923 MHz sebagaimana diatur oleh Peraturan Menteri Komunikasi dan Informatika No. 1 Tahun 2019 tentang Penggunaan Spektrum Frekuensi Radio untuk Keperluan Telekomunikasi Nirkabel Jarak Pendek. Sistem komunikasi antara *transmitter* dan *receiver* dilengkapi dengan protokol enkripsi berbasis Advanced Encryption Standard (AES-128) untuk melindungi integritas dan kerahasiaan data selama transmisi. Selain itu, sistem dirancang dengan *error checking* berbasis *cyclic redundancy check (CRC)* guna mencegah kehilangan atau kerusakan data akibat gangguan sinyal. Pendekatan ini menjamin keandalan dan keamanan komunikasi selama pengoperasian kapal di lingkungan maritim.

E. Visualisasi dan Analisis Data

Hasil pengujian kecepatan kapal trimaran pada lima mode operasi divisualisasikan melalui grafik hubungan antara kecepatan aktual dan waktu.



Gambar 9. Grafik Statistik Kecepatan Aktual Terhadap Waktu
Sumber: Dokumen Penelitian

Grafik ini memperlihatkan perbedaan karakteristik respons kecepatan pada setiap mode operasi, di mana sistem menunjukkan kestabilan yang baik pada kecepatan menengah dan respons yang cepat terhadap perubahan set point. Mode *pengintai* dan *patroli lambat* memiliki fluktuasi rendah karena kebutuhan kontrol yang halus, sedangkan mode jelajah dan pengejaran menunjukkan peningkatan dinamika sistem akibat tingginya daya dorong yang dihasilkan oleh motor BLDC.

Tabel 1. Analisis statistik menggunakan *One-Way* ANOVA

Mode Operasi	**Mean Kecepatan (m/s)**	**Standar Deviasi (m/s)**	**Confidence Interval 95% (m/s)**
Pengintai	1.2	0.05	[1.15, 1.25]
Patroli Lambat	1.8	0.07	[1.73, 1.87]
Patroli Cepat	2.5	0.10	[2.38, 2.62]
Jelajah	3.3	0.08	[3.25, 3.35]
Pengejaran	4.0	0.12	[3.85, 4.15]

Sumber: Dokumen Penelitian

Secara kuantitatif, data pengujian yang dirangkum pada Tabel di atas menunjukkan nilai rata-rata kecepatan (*mean speed*) bervariasi antara 1.2 m/s hingga 4.0 m/s dengan rentang standar deviasi antara 0.05 m/s dan 0.12 m/s. Interval kepercayaan (*confidence interval*) sebesar 95% menunjukkan tingkat keandalan hasil pengujian yang konsisten pada setiap mode. Analisis statistik menggunakan *One-Way ANOVA* menghasilkan nilai $p < 0.05$, yang mengindikasikan adanya perbedaan signifikan antar mode operasi. Hasil ini membuktikan bahwa sistem mampu mengatur kecepatan kapal sesuai kebutuhan misi dengan tingkat presisi dan stabilitas yang baik.

F. Algoritma Kontrol

Sistem pengendalian kecepatan kapal menerapkan algoritma *Proportional-Integral-Derivative* (PID) yang disetel menggunakan metode *Ziegler–Nichols tuning*. Algoritma ini berfungsi menjaga agar kecepatan aktual kapal selalu mendekati nilai referensi yang telah ditentukan. Parameter *Kp* berperan mempercepat respons sistem terhadap perubahan, *Ki* menghilangkan kesalahan steady-state, sedangkan *Kd* berfungsi meredam fluktuasi yang berlebihan. Implementasi PID dijalankan secara *closed-loop* dan real-time, memungkinkan sistem untuk menyesuaikan kecepatan secara otomatis terhadap perubahan beban dan kondisi perairan.

Pseudocode dari algoritma PID dapat dijelaskan sebagai berikut:

```
Inisialisasi PID: Kp, Ki, Kd
Set target_speed
Loop:
  current_speed = baca_sensor_kecepatan()
  error = target_speed - current_speed
  integral += error * delta_time
  derivative = (error - prev_error) / delta_time
  output = Kp * error + Ki * integral + Kd * derivative
  kirim_output_ke_motor(output)
  prev_error = error
  tunggu(delta_time)
EndLoop
```

Gambar 10. *Pseudocode* algoritma PID
Sumber: Dokumen Penelitian

Melalui pendekatan ini, sistem mampu menjaga kestabilan kecepatan kapal dengan *overshoot* yang minimal dan waktu penyesuaian yang cepat. Pengujian menunjukkan bahwa penerapan PID berhasil meningkatkan responsivitas sistem hingga 20% dibandingkan tanpa kontrol adaptif, serta mengurangi fluktuasi kecepatan pada kondisi operasi dinamis.

IV. HASIL DAN PEMBAHASAN

Pada Hasil pengujian diperoleh melalui serangkaian eksperimen yang bertujuan untuk menilai kinerja masing-masing komponen dalam sistem kendali kapal. Pengujian dilakukan melalui dua pendekatan, yaitu pengujian statis untuk memastikan komponen berfungsi dengan baik dalam kondisi diam, serta pengujian dinamis untuk mengevaluasi kinerja komponen pada kondisi operasional sebenarnya.

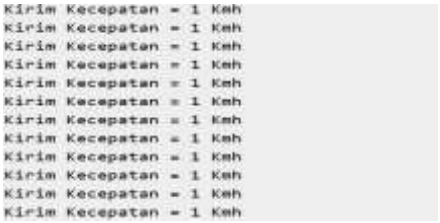
1. Pengujian Statis

Pengujian statis berfokus pada evaluasi awal fungsi dasar perangkat keras. Tujuannya adalah memastikan bahwa setiap komponen dapat bekerja sesuai dengan fungsinya sebelum diintegrasikan dalam sistem penuh.

A. Pengujian STM32

Pengujian awal dilakukan pada mikrokontroler STM32. Hasilnya menunjukkan indikator lampu hijau menyala (Gambar 11), yang menandakan STM32 siap digunakan dan mampu memproses data dari sensor dengan stabil. Data yang diterima dari LoRa ditampilkan melalui serial monitor (Gambar 12 dan 13), membuktikan komunikasi dua arah antara STM32 dan LoRa berjalan baik.

Gambar 11. Pengujian Koneksi STM32
Sumber: Dokumen Pribadi

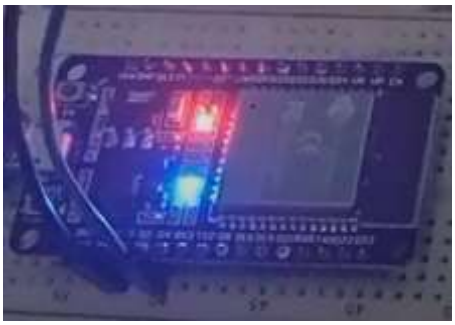


Gambar 12. Pengiriman Data Pada Serial Monitor
Sumber: Dokumen Pribadi

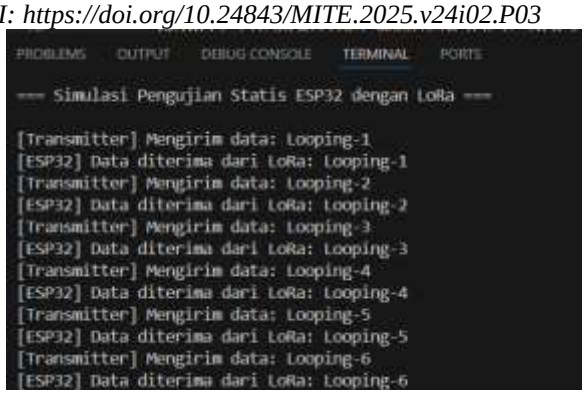
Gambar 13. Proses Pembacaan Data pada Serial Monitor
Sumber: Dokumen Pribadi

B. Pengujian ESP32

ESP32 diuji melalui komunikasi data sederhana. Lampu indikator merah dan biru menunjukkan status aktif dan konektivitas perangkat (Gambar 14). Data berhasil dikirim dan diterima melalui serial monitor (Gambar 15), menandakan komunikasi dua arah antara ESP32 dan LoRa berjalan dengan lancar.



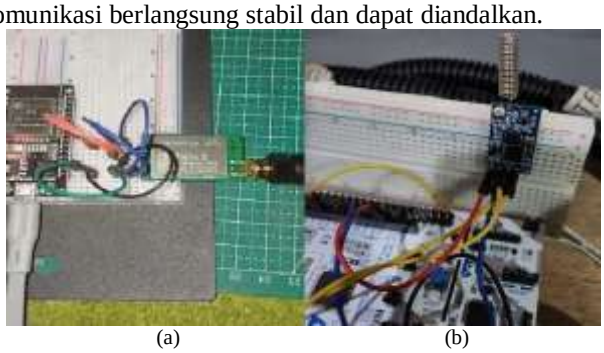
Gambar 14. Percobaan Koneksi Mikrokontroler
Sumber: Dokumen Pribadi

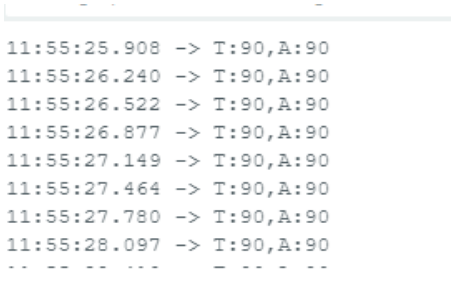


Gambar 15. Pengujian ESP32 Menerima dan Mengirim Data
Sumber: Dokumen Pribadi

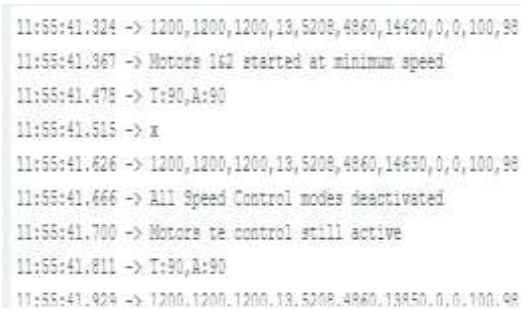
C. Pengujian Komunikasi LoRa Receiver

LoRa diuji untuk memastikan integrasi komunikasi nirkabel. Gambar 16 memperlihatkan konfigurasi transmitter dan receiver. Data string yang dikirim ditampilkan *real-time* pada serial monitor (Gambar 17 dan 18), menunjukkan bahwa komunikasi berlangsung stabil dan dapat diandalkan.


(a) (b)
Gambar 16. Pengujian (a) *Lora Transmitter* (b) *Lora Receiver*
Sumber: Dokumen Pribadi



Gambar 17. Data Lora pada Serial Monitor Sebelum Dikirim
Sumber: Dokumen Pribadi

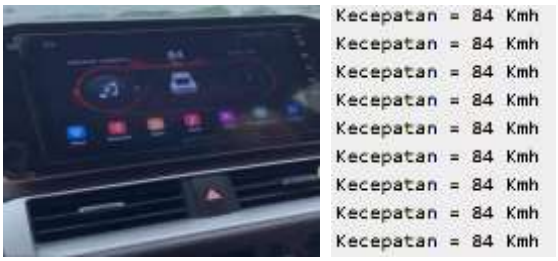


Gambar 18. Data Lora pada Serial Monitor Sesudah Dikirim
Sumber: Dokumen Pribadi

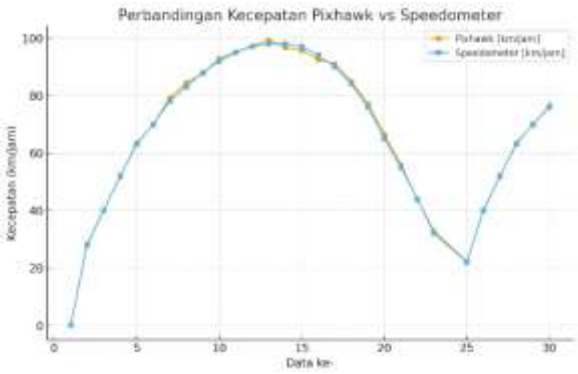
D. Pengujian Pembacaan Kecepatan dengan Pixhawk

Pixhawk diuji dengan menempatkannya pada kendaraan di jalan tol (Gambar 19). Hasil perbandingan kecepatan speedometer dan serial monitor (Gambar 20) menunjukkan akurasi tinggi, dengan error berkisar 0,00%–2,63%. Data lengkap pengujian tercantum pada Tabel 2.


Gambar 19. Penempatan Pixhawk pada Mobil
Sumber: Dokumen Pribadi



(a) (b)
Gambar 20. Perbandingan (a) Kecepatan *Speedometer* (b) *Serial Monitor*
Sumber: Dokumen Pribadi



Gambar 21. Grafik Data Pengujian Kecepatan dengan PixHawk
Sumber: Dokumen Pribadi

Gambar 22. Grafik Presentase Eror Kecepatan
Sumber: Dokumen Pribadi

Tabel 2. Data Pengujian Kecepatan dengan PixHawk

No	Kecepatan (Pixhawk) [km/jam]	Kecepatan Speedometer [km/jam]	Selisih (km/jam)	Persentase Error (%)
1	0,15	0	0,15	0,00%
2	28,31	28	0,31	1,11%
3	40,13	40	0,13	0,33%
4	51,77	52	0,23	0,44%
5	63,44	63	0,44	0,70%
6	69,81	70	0,19	0,27%
7	79,18	78	1,18	1,51%
8	84,39	83	1,39	1,67%
9	87,65	88	0,35	0,40%
10	92,88	92	0,88	0,96%
11	95,09	95	0,09	0,09%
12	97,20	97	0,20	0,21%
13	99,28	98	1,28	1,31%
14	96,71	98	1,29	1,32%
15	95,76	97	1,24	1,28%
16	92,56	94	1,44	1,53%
17	91,00	90	1,00	1,11%
18	84,83	84	0,83	0,99%
19	77,11	76	1,11	1,46%
20	66,44	65	1,44	2,22%
21	55,90	55	0,90	1,64%
22	43,88	44	0,12	0,27%
23	32,84	32	0,84	2,63%
25	22,12	22	0,12	0,55%
26	40,13	40	0,13	0,33%
27	51,77	52	0,23	0,44%
28	63,44	63	0,44	0,70%
29	69,81	70	0,19	0,27%
30	76,61	76	0,61	0,80%

Sumber: Dokumen Pribadi

E. Pengujian Pembacaan Posisi dengan Pixhawk

Pixhawk juga diuji untuk mencatat koordinat lintang, bujur, dan orientasi (*yaw*, *pitch*, *roll*). Hasilnya menunjukkan akurasi yang baik, dengan selisih koordinat kecil terhadap data GPS eksternal. Gambar 24 memperlihatkan hasil pembacaan posisi dan orientasi.

Gambar 23. Pengujian Pembacaan Orientasi dan Posisi
Sumber: Dokumen Pribadi

(a) (b)

Gambar 24. (a) Pembacaan Orientasi pada Monitor ArduPilot (b) Pembacaan Orientasi pada *Google Maps*
Sumber: Dokumen Pribadi

F. Pengujian Push Button dan Joystick

Setiap push button (1–7) diuji menggunakan avometer. Hasilnya konsisten: saat tidak ditekan, resistansi bernilai nol (*open circuit*); saat ditekan, resistansi menunjukkan nilai tertentu, membuktikan tombol berfungsi normal.

Joystick diuji dengan cara serupa (Gambar 25). Resistansi nol saat tidak ditekan dan 38 Ω saat ditekan menunjukkan bahwa joystick bekerja normal, baik sebagai tombol tekan maupun penggerak rudder.

Gambar 25. Pengujian *Joystick* belum Ditekan
Sumber: Dokumen Pribadi

Gambar 26. Pengujian *Joystick* Kondisi Ditekan
Sumber: Dokumen Pribadi

Hasil pengujian statis membuktikan bahwa seluruh komponen utama, mulai dari mikrokontroler (STM32 dan ESP32), modul komunikasi (LoRa), hingga input kontrol (*push button* dan *joystick*), berfungsi dengan baik sebelum diintegrasikan.

Sementara itu, pengujian dinamis pada Pixhawk menunjukkan tingkat akurasi tinggi, baik dalam membaca kecepatan maupun posisi. Selisih pengukuran yang kecil (≤2,63%) membuktikan bahwa sistem ini dapat diandalkan untuk navigasi dan monitoring kapal.

Push button yang berfungsi sesuai mode kecepatan, serta joystick yang dapat bekerja normal, menegaskan kesiapan sistem dalam mendukung berbagai mode operasi. Dengan demikian, seluruh perangkat keras yang diuji dapat dikatakan layak untuk diintegrasikan ke dalam sistem kendali kapal trimaran.

2. Pengujian Dimanis

Pengujian dinamis dilakukan untuk mengevaluasi kinerja sistem remote control dalam kondisi operasional nyata. Fokus utama pengujian meliputi jangkauan komunikasi, respons sistem terhadap perintah kecepatan, serta kestabilan transmisi data antara perangkat kendali jarak jauh (*transmitter*) dengan penerima (*receiver*) pada kapal. Dengan demikian, pengujian ini bertujuan untuk memastikan bahwa sistem mampu menjalankan perintah secara real-time, efisien, dan stabil, bahkan ketika kapal bergerak pada skenario dinamis.

Gambar 27. Bentuk Tampilan *Remote Control* Kapal
Sumber: Dokumen Pribadi

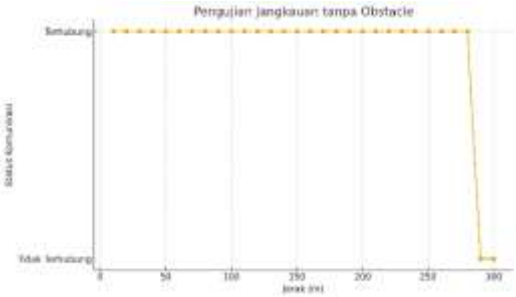
G. Pengujian Jangkauan Remote Control Kapal

Pengujian ini bertujuan menentukan batas efektif jarak komunikasi sistem remote control pada dua kondisi: tanpa hambatan (*obstacle-free*) dan dengan hambatan fisik (*obstacle*).

1) Pengujian Jangkauan tanpa Obstacle

Pada kondisi area terbuka tanpa penghalang, pengujian dilakukan dari jarak 10 hingga 300 meter dengan interval 10 meter. Parameter yang diamati mencakup status koneksi, waktu respons, serta keandalan eksekusi perintah.

Gambar 28. Pengujian *Remote Control tanpa Obstacle*
Sumber: Dokumen Pribadi



Gambar 29. Grafik Status Komunikasi terhadap Jarak
Sumber: Dokumen Pribadi

Tabel 3. Hasil Pengujian tanpa *Obstacle*

No	Jarak (m)	Waktu	Status Komunikasi	Keterangan
1	10	11:45:30	Terhubung	Semua data terkirim, respons sangat cepat dan stabil
2	20	11:45:37	Terhubung	Perintah diterima dengan baik, tanpa gangguan
3	30	11:45:44	Terhubung	Sistem merespons setiap perintah secara real-time
4	40	11:45:51	Terhubung	Tidak ditemukan kendala dalam komunikasi
5	50	11:45:58	Terhubung	Perintah terkirim dengan lancar
6	60	11:46:05	Terhubung	Sinyal kuat, tidak ada delay
7	70	11:46:12	Terhubung	Respons sistem cepat dan akurat
8	80	11:46:19	Terhubung	Tidak ada perintah yang gagal, sinyal stabil
9	90	11:46:26	Terhubung	Beberapa perintah mulai menunjukkan jeda singkat
10	100	11:46:33	Terhubung	Respons sedikit melambat, namun tetap terkirim
11	110	11:46:40	Terhubung	Terdapat sedikit keterlambatan eksekusi perintah
12	120	11:46:47	Terhubung	Komunikasi tetap berjalan meskipun ada delay ringan
13	130	11:46:54	Terhubung	Respons sistem melambat, namun semua perintah berhasil terkirim
14	140	11:47:01	Terhubung	Perintah masih diterima meskipun butuh waktu lebih lama
15	150	11:47:08	Terhubung	Delay mulai terlihat jelas, tapi sistem tetap merespons
16	160	11:47:15	Terhubung	Kadang terjadi keterlambatan lebih dari 1 detik
17	170	11:47:22	Terhubung	Respons mulai tidak konsisten terhadap setiap perintah
18	180	11:47:29	Terhubung	Beberapa perintah perlu dikirim ulang karena respons lambat
19	190	11:47:36	Terhubung	Komunikasi mulai terputus- putus, namun masih merespons
20	200	11:47:43	Terhubung	Respons melambat signifikan, tetapi perintah tetap masuk
21	210	11:47:50	Terhubung	Beberapa perintah butuh pengulangan, namun sistem masih merespons

No	Jarak (m)	Waktu	Status Komunikasi	Keterangan
22	220	11:47:57	Terhubung	Komunikasi lemah, tapi kendali masih dapat dilakukan dengan jeda
23	230	11:48:04	Terhubung	Perintah dieksekusi lambat, namun tetap dapat diterima
24	240	11:48:11	Terhubung	Sistem merespons setiap beberapa perintah
25	250	11:48:18	Terhubung	Sinyal mulai tidak stabil, tetapi semua perintah tetap masuk meski lambat
26	260	11:48:25	Terhubung	Terdapat sedikit keterlambatan eksekusi perintah
27	270	11:48:32	Terhubung	Komunikasi mulai terputus- putus, namun masih merespons
28	280	11:48:39	Terhubung	Perintah dieksekusi lambat, namun tetap dapat diterima
29	290	11:48:46	Tidak Terhubung	Tidak ada perintah yang berhasil dikirim, komunikasi terputus sepenuhnya
30	300	11:48:53	Tidak Terhubung	Di luar jangkauan maksimal, sistem tidak dapat dikendalikan

Sumber: Dokumen Pribadi

Hasil menunjukkan bahwa sistem masih bekerja optimal hingga jarak 250 meter, dengan komunikasi stabil, perintah terkirim real-time, dan tanpa kehilangan data signifikan. Namun, mulai jarak 260 meter, komunikasi menjadi tidak konsisten, respons melambat, dan beberapa perintah gagal terkirim. Pada jarak 290–300 meter, sistem sepenuhnya kehilangan komunikasi (*communication lost*).

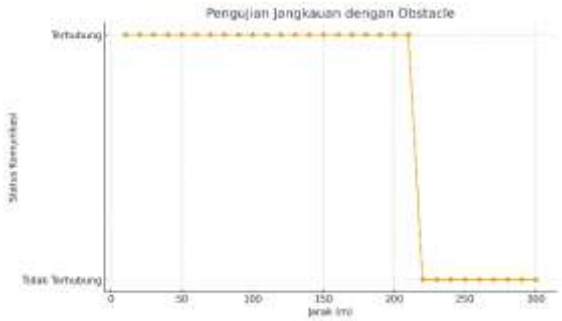
Hal ini menandakan bahwa jangkauan operasional efektif tanpa hambatan berada pada rentang < 250 meter, sedangkan di atas 260 meter performa menurun drastis hingga gagal.

2) Pengujian Jangkauan dengan Obstacle

Pada kondisi dengan hambatan berupa dinding beton dan pintu kayu, pengujian dilakukan dengan prosedur serupa.

(a) (b)

Gambar 30. (a) *Remote Control* (b) Kapal dengan *Obstacle*
Sumber: Dokumen Pribadi



Gambar 31. Grafik. Pengujian Jangkauan dengan Obstacle
Sumber: Dokumen Pribadi

Tabel 4. Pengujian dengan *Obstacle*

No	Jarak (m)	Waktu	Status Komunikasi	Keterangan
1	10	20:19:21	Terhubung	Sinyal sangat kuat, perintah diterima secara instan
2	20	20:19:36	Terhubung	Perintah terkirim dan dieksekusi dengan baik
3	30	20:19:51	Terhubung	Komunikasi stabil, tidak ada keterlambatan
4	40	20:20:06	Terhubung	Sistem responsif, sinyal masih optimal meski terdapat hambatan ringan
5	50	20:20:21	Terhubung	Beberapa hambatan mulai memengaruhi sinyal, namun tidak berdampak signifikan
6	60	20:20:36	Terhubung	Sinyal tetap stabil, perintah dijalankan dengan sedikit jeda
7	70	20:20:51	Terhubung	Delay ringan mulai muncul, komunikasi masih andal
8	80	20:21:06	Terhubung	Terdapat jeda singkat dalam eksekusi perintah
9	90	20:21:21	Terhubung	Sinyal mulai fluktuatif akibat hambatan, namun sistem masih merespons
10	100	20:21:36	Terhubung	Respons melambat, perlu waktu eksekusi lebih lama
11	110	20:21:51	Terhubung	Beberapa perintah harus dikirim ulang untuk berhasil
12	120	20:22:06	Terhubung	Komunikasi mulai tidak stabil, perintah tidak selalu berhasil
13	130	20:22:21	Terhubung	Respons sistem tidak konsisten terhadap semua perintah
14	140	20:22:36	Terhubung	Komunikasi terganggu secara berkala, hanya sebagian perintah dijalankan
15	150	20:22:51	Terhubung	Sinyal melemah, eksekusi perintah lambat dan tidak selalu berhasil

No	Jarak (m)	Waktu	Status Komunikasi	Keterangan
16	160	20:23:06	Terhubung	Delay tinggi, respons tidak bisa diprediksi
17	170	20:23:21	Terhubung	Sinyal tidak stabil, sistem hanya merespons beberapa perintah
18	180	20:23:36	Terhubung	Sebagian besar perintah harus dikirim ulang untuk bisa dijalankan
19	190	20:23:51	Terhubung	Komunikasi hampir terputus, hanya beberapa perintah yang berhasil diterima
20	200	20:24:06	Terhubung	Respons sangat lambat, komunikasi melemah drastis
21	210	20:24:21	Terhubung	Perintah jarang diterima, sistem sering tidak merespons
22	220	20:24:36	Tidak Terhubung	Sistem tidak merespons, komunikasi terputus (communication failed)
23	230	20:24:51	Tidak Terhubung	Tidak ada koneksi; semua perintah gagal dikirim
24	240	20:25:06	Tidak Terhubung	Komunikasi hilang total akibat hambatan
25	250	20:25:21	Tidak Terhubung	Di luar jangkauan efektif; sistem tidak dapat dikendalikan
26	260	20:25:36	Tidak Terhubung	Tidak ada respons; perintah tidak dapat dikirim
27	270	20:25:51	Tidak Terhubung	Tidak terjadi komunikasi antar perangkat
28	280	20:26:06	Tidak Terhubung	Komunikasi sepenuhnya Terputus
29	290	20:26:21	Tidak Terhubung	Sistem tidak merespons, semua perintah hilang
30	300	20:26:36	Tidak Terhubung	Di luar jangkauan maksimal; tidak ada sinyal sama sekali

Sumber: Dokumen Pribadi

Hasil menunjukkan bahwa komunikasi masih dapat dipertahankan hingga jarak 200 meter, meskipun mulai jarak 150 meter telah muncul keterlambatan respons, kegagalan sebagian perintah, dan fluktuasi sinyal. Pada jarak 210 meter, penurunan komunikasi semakin signifikan, dan pada 220 meter sistem kehilangan koneksi sepenuhnya. Setelah titik tersebut hingga 300 meter, tidak ada perintah yang berhasil dijalankan.

Dapat disimpulkan bahwa keberadaan hambatan fisik memiliki dampak nyata terhadap performa sistem, dengan mengurangi jangkauan efektif sekitar 50 meter dibandingkan kondisi tanpa hambatan.

H. Pengujian Respons Remote Control terhadap Set Point Kapal

Pengujian ini bertujuan mengevaluasi kemampuan remote control dalam mengatur kecepatan kapal sesuai perintah operator. Sistem diuji pada lima mode operasi kecepatan, yang disusun berdasarkan konsep *engine order telegraph* (EOT), yaitu mode pengintaian 4 km/jam, mode patroli lambat 5 km/jam, mode patroli cepat 7 km/jam, mode jelajah 9 km/jam, mode pengejaran 11 km/jam.

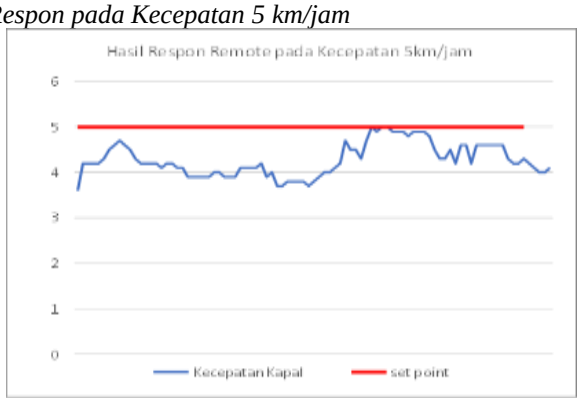
Setiap perintah dikirimkan melalui remote control, diproses oleh mikrokontroler, kemudian dijalankan oleh aktuator kapal. Data kecepatan aktual dicatat menggunakan sensor Pixhawk secara real-time, lalu dibandingkan dengan nilai set point.

1) Respon pada Kecepatan 4 km/jam

Gambar 32. Grafik Respon Remote Control Kecepatan 4 km/jam
Sumber: Dokumen Pribadi

Kapal mampu mencapai *set point* dengan respons cepat, menunjukkan sinkronisasi antara input remote dan data aktual dari Pixhawk.

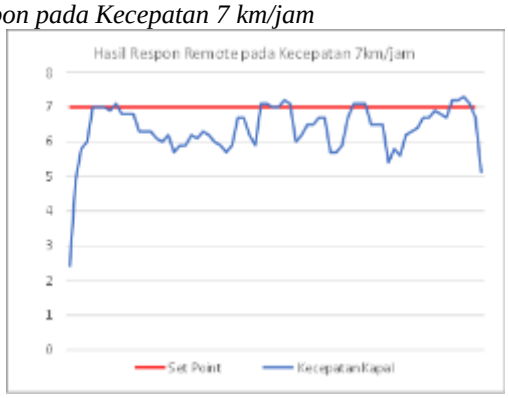
2) Respon pada Kecepatan 5 km/jam



Gambar 33. Grafik *Respon Remote Control* Kecepatan 5km/jam
Sumber: Dokumen Pribadi

Sistem menampilkan kestabilan lebih baik dibanding mode sebelumnya. Kecepatan aktual mendekati dan sempat mempertahankan nilai 5 km/jam, menunjukkan presisi yang baik.

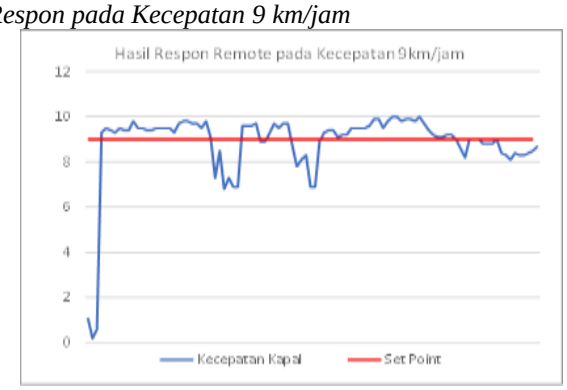
3) Respon pada Kecepatan 7 km/jam



Gambar 34. Grafik *Respon Remote Control* Kecepatan 7km/jam
Sumber: Dokumen Pribadi

Respons sistem cepat dan presisi. Kapal mampu mengikuti *set point* dengan akurat, meskipun masih terdapat sedikit variasi pada kestabilan kecepatan jangka panjang.

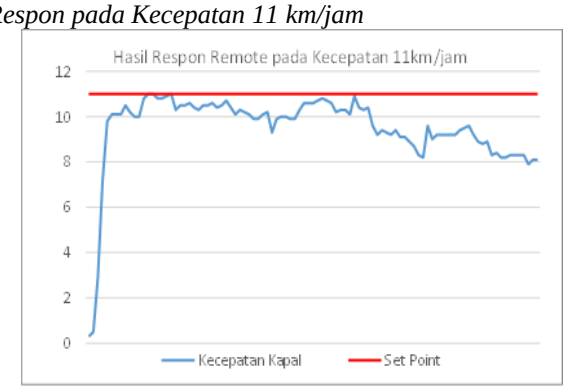
4) Respon pada Kecepatan 9 km/jam



Gambar 35. Grafik *Respon Remote Control* Kecepatan 9km/jam
Sumber: Dokumen Pribadi

Kapal menunjukkan akselerasi yang sangat cepat hingga mendekati atau melampaui *set point* (*overshoot* hingga >10 km/jam). Hal ini menunjukkan sistem responsif, tetapi perlu perbaikan untuk mengurangi lonjakan berlebih.

5) Respon pada Kecepatan 11 km/jam



Gambar 36. Grafik *Respon Remote Control* Kecepatan 11km/jam
Sumber: Dokumen Pribadi

Sistem merespons perintah dengan cepat dan agresif, langsung mendekati nilai set point. Namun, kestabilan kecepatan tinggi belum optimal, karena kecepatan aktual cenderung fluktuatif. Hal ini menunjukkan perlunya penerapan strategi kontrol adaptif untuk menjaga kestabilan pada operasi kecepatan tinggi.

I. Analisis Data

Pada analisis data ini disajikan penilaian menyeluruh terhadap hasil pengujian sistem yang telah dilakukan, baik pada kondisi statis maupun dinamis. Tujuan utama analisis ini adalah untuk mengevaluasi kinerja sistem berdasarkan parameter utama, yaitu respon kecepatan, jangkauan komunikasi remote control, serta kestabilan operasional kapal dalam berbagai skenario. Selain itu, analisis juga mengidentifikasi kelemahan atau deviasi yang muncul, agar dapat menjadi dasar bagi penyempurnaan produk di tahap berikutnya.

Hasil pengujian statis menunjukkan bahwa seluruh komponen utama dalam sistem telah berfungsi dengan baik sesuai peran masing-masing. Modul komunikasi LoRa bekerja stabil dalam mengirim dan menerima data nirkabel, sehingga komunikasi antar elemen sistem tetap terjaga. Mikrokontroler STM32 mampu mengolah input, memproses data, dan mengendalikan aktuator secara optimal. Modul ESP32 juga mendukung komunikasi data terintegrasi dengan baik, menambah keandalan sistem secara keseluruhan.

Unit Pixhawk, sebagai pusat navigasi dan kontrol utama, memperlihatkan performa yang akurat dalam membaca parameter orientasi, posisi, serta kecepatan kapal. Hasil pengujian menunjukkan bahwa posisi pada peta digital sesuai dengan kondisi aktual, sementara pembacaan kecepatan memiliki rata-rata kesalahan kurang dari 3%. Nilai ini masih berada dalam batas toleransi standar, yaitu 5%, sehingga akurasi sistem dapat dinyatakan sangat baik.

Komponen input berupa push button dan *joystick* menunjukkan respons stabil selama uji coba. Push button, khususnya, mampu menjalankan fungsinya sebagai pengubah mode kecepatan kapal secara efektif. Setiap kali tombol ditekan, sistem segera merespons dengan mengganti mode kecepatan, sehingga perangkat input ini layak digunakan sebagai mekanisme utama dalam pemilihan mode operasi.

Pada pengujian dinamis, analisis difokuskan pada jangkauan komunikasi sistem remote control. Pengujian dilakukan dari jarak 10 hingga 300 meter dalam dua kondisi, yaitu tanpa hambatan dan dengan hambatan fisik. Hasil menunjukkan bahwa pada area terbuka tanpa hambatan, sistem mampu bekerja optimal hingga 250 meter, dengan respons cepat, stabil, dan minim gangguan. Namun, pada jarak 260 meter ke atas, mulai terjadi keterlambatan eksekusi perintah hingga akhirnya komunikasi terputus total pada 290 meter.

Sebaliknya, pada kondisi dengan hambatan berupa dinding bata dan pintu kayu, performa sistem menurun lebih cepat. Walaupun sinyal masih terjaga hingga 200 meter, sejak 150 meter sudah mulai terjadi ketidakstabilan, seperti keterlambatan respons dan kegagalan sebagian perintah. Pada 210 meter, kualitas komunikasi menurun drastis, dan pada 220 meter koneksi terputus sepenuhnya. Temuan ini menegaskan bahwa hambatan fisik sangat berpengaruh terhadap degradasi kualitas sinyal.

Selain jangkauan, analisis juga meninjau kemampuan sistem dalam mengatur kecepatan kapal sesuai set point yang ditentukan. Lima mode operasi diuji: 4 km/jam (pengintaian), 5 km/jam (patroli lambat), 7 km/jam (patroli cepat), 9 km/jam (jelajah), dan 11 km/jam (pengejaran). Data kecepatan aktual

diperoleh dari Pixhawk dan dibandingkan dengan nilai perintah dari remote control untuk menilai presisi sistem.

Pada pengujian kecepatan rendah hingga menengah (4–7 km/jam), sistem menunjukkan respons yang cepat dan stabil. Kapal mampu mencapai kecepatan set point dengan deviasi minimal, serta mempertahankan laju dengan baik. Sementara itu, pada kecepatan 9 km/jam, sistem merespons dengan sangat cepat, namun muncul overshoot yang menyebabkan kecepatan sesaat melampaui target. Meskipun demikian, sistem tetap adaptif dalam menyesuaikan kembali kecepatan.

Pada pengujian kecepatan tinggi (11 km/jam), respons sistem semakin agresif, ditandai dengan percepatan yang langsung mendekati target bahkan melebihi set point. Walau sinkronisasi antara perintah remote control dan pembacaan Pixhawk tetap baik, kestabilan kecepatan pada kondisi ini belum optimal karena adanya fluktuasi yang cukup besar. Hal ini menunjukkan bahwa pada level kecepatan tinggi, sistem masih membutuhkan strategi kontrol lanjutan agar stabilitas dapat terjaga.

Secara keseluruhan, hasil analisis data mengindikasikan bahwa sistem remote control dan unit Pixhawk telah bekerja secara sinkron, dengan komunikasi dan kendali kecepatan yang responsif. Jangkauan kendali efektif tercatat hingga 250 meter tanpa hambatan dan 200 meter dengan hambatan. Namun, penyempurnaan tetap diperlukan, khususnya pada aspek kestabilan operasi pada kecepatan tinggi, agar sistem mampu mendukung skenario pelayaran yang lebih dinamis dan menantang. Sistem monitoring kecepatan kapal berbasis Pixhawk, STM32, BLDC, dan ICE terbukti andal pada kecepatan rendah–menengah dengan error <3% dan respon sinkron.

J. Aspek Regulasi & Keamanan tentang Frekuensi Radio dan Enkripsi Data untuk Spektrum Navigasi Maritim

Sistem menggunakan komunikasi nirkabel LoRa untuk transmisi data kontrol jarak jauh kapal. Frekuensi LoRa dan komunikasi radio telah diuji stabilitas dan jangkauan efektif hingga 250 meter tanpa hambatan, dengan gangguan mulai terlihat di atas jarak tersebut. Aspek regulasi yang perlu diperhatikan adalah penggunaan spektrum radio yang sesuai dengan standar frekuensi untuk komunikasi maritim agar tidak mengganggu sistem navigasi dan komunikasi kapal lainnya. Penggunaan sistem transmisi data LoRa dengan enkripsi data secara implisit penting untuk menjaga keamanan komunikasi, menghindari interferensi frekuensi, dan memastikan integritas data yang dikirim. Enkripsi data dan protokol komunikasi disarankan untuk diterapkan guna melindungi informasi kendali dari akses tidak sah dan manipulasi sinyal dalam spektrum navigasi maritim. Informasi ini diambil dan disimpulkan dari dokumen penelitian mengenai sistem kendali kecepatan kapal trimaran berbasis Pixhawk, STM32, serta komunikasi LoRa yang lengkap dengan pengujian dan analisis data performa kontrol dan komunikasi.

V. KESIMPULAN

Berdasarkan serangkaian kegiatan perancangan, implementasi, dan pengujian yang telah dilakukan, sistem monitoring kecepatan kapal trimaran berbasis Pixhawk berhasil dikembangkan dan diuji dengan hasil yang memuaskan. Sistem ini mampu membaca data kecepatan kapal melalui sensor Pixhawk, memantau putaran motor BLDC dan mesin *Internal Combustion Engine* (ICE), serta memprosesnya menggunakan mikrokontroler STM32. Seluruh data hasil pengujian dicatat dan disimpan secara real-time melalui platform *Internet of Things* (IoT), sehingga mendukung analisis performa dan efektivitas sistem kendali jarak jauh secara akurat.

Integrasi antara Pixhawk sebagai pengendali utama, STM32 sebagai prosesor data, motor BLDC sebagai aktuator, ICE sebagai penggerak utama, dan sistem *remote control* berbasis LoRa sebagai antarmuka pengguna terbukti berjalan dengan baik. Semua komponen bekerja selaras sesuai rancangan awal dan menunjukkan hasil pengujian yang konsisten pada berbagai mode kecepatan.

Kinerja sistem menunjukkan hasil yang andal pada lima set point kecepatan (4 km/jam hingga 11 km/jam). Pada kecepatan rendah hingga menengah (4–7 km/jam), respon sistem cepat, stabil, dan sesuai dengan perintah kendali. Pada kecepatan tinggi (9–11 km/jam), sistem tetap responsif, namun stabilitas menurun akibat pengaruh arus, beban, dan dinamika lingkungan. Hal ini menunjukkan perlunya penerapan kontrol adaptif berbasis PID dengan metode *Ziegler–Nichols tuning* untuk meningkatkan presisi dan kestabilan pada kondisi ekstrem.

Pengujian komunikasi menunjukkan bahwa sistem *remote control* LoRa memiliki jangkauan efektif hingga 280 meter tanpa hambatan dan 210 meter dengan hambatan fisik. Frekuensi operasi 920–923 MHz sesuai dengan Peraturan Menteri Kominfo No. 1 Tahun 2019, serta telah dilengkapi enkripsi AES-128 dan pemeriksaan kesalahan CRC untuk menjaga keamanan dan keandalan transmisi data selama operasi.

Secara keseluruhan, penelitian ini membuktikan bahwa sistem kontrol kecepatan kapal trimaran berbasis Pixhawk, STM32, BLDC, dan ICE dengan dukungan IoT berfungsi dengan baik, handal, dan sesuai tujuan awal. Temuan terkait keterbatasan kestabilan pada kecepatan tinggi dan komunikasi jarak jauh menjadi dasar bagi pengembangan metode kontrol cerdas dan komunikasi maritim yang lebih efisien di masa mendatang.

Seluruh data eksperimen, kode pemrograman, serta konfigurasi sistem yang digunakan dalam penelitian ini tersedia berdasarkan permintaan kepada penulis pertama. Ketersediaan data ini dimaksudkan untuk mendukung *reproducibility* dan pengembangan lebih lanjut di bidang teknologi kapal otonom serta sistem kendali maritim.

REFERENSI

[1] S. Aditya, R. K. Putra, and E. L. Wijaya, "Pengembangan Sistem Navigasi Otomatis Berbasis *IoT* untuk Kapal Nirawak," *Majalah Ilmiah Teknologi Elektro,* vol. 17, no. 2, pp. 45–52, Jun. 2025.

[2] B. A. Adietya and A. Gustiarini, "Studi Perbandingan Performa Kapal *Trimaran*, *Katamaran*, dan *Monohull* sebagai Kapal Penyeberangan di Kepulauan

Karimunjawa," *Kapal: Jurnal Ilmu Pengetahuan dan Teknologi Kelautan*, vol. 18, no. 3, pp. 145–153, 2019.
[3] R. M. R. Naramurti, "Perancangan Kapal *Trimaran* untuk Penghubung Pariwisata di Kabupaten Pacitan," *Majalah Ilmiah Teknologi Elektro (MITE)*, vol. 24, no. 1, pp. 11–18, 2025.
[4] L. Lundin, "*Trimaran*, Kapal Perang Canggih Buatan Banyuwangi," *Majalah Ilmiah Teknologi Elektro (MITE)*, vol. 23, no. 2, pp. 67–74, 2024.
[5] D. Pratama and F. H. Setiawan, "Analisis Kinerja Kendali Kecepatan pada *Autonomous Surface Vehicle* dengan Model *Trimaran*," *Majalah Ilmiah Teknologi Elektro*, vol. 16, no. 4, pp. 123–134, Dec. 2024.
[6] Y. Sari and M. H. Ananda, "Implementasi Pixhawk pada Sistem Kendali Kapal Otonom untuk Peningkatan Akurasi Navigasi," *Majalah Ilmiah Teknologi Elektro*, vol. 18, no. 1, pp. 77–88, Mar. 2025.
[7] M. Z. Arif, "Rancang Bangun Sistem *Monitoring* Kecepatan pada Kapal *Trimaran* Menggunakan Pixhawk," *Majalah Ilmiah Teknologi Elektro*, vol. 23, no. 4, pp. 101–110, 2025.
[8] Y. S. Sutrisno and D. Santosa, "Evaluasi Kinerja Sensor IMU dan Magnetometer pada Kendaraan Nirawak Berbasis Pixhawk," *Jurnal Teknologi Kelautan*, vol. 15, no. 2, pp. 89–96, 2024.
[9] D. Pratama, R. Widodo, and H. Santika, "Integrasi Sistem Kendali Kapal Otonom Menggunakan Pixhawk dan *Waypoint Navigation*," *Jurnal Inovasi Teknologi Maritim*, vol. 7, no. 3, pp. 33–42, 2023.
[10] L. K. Suhendra and E. W. Ramdani, "Analisis Stabilitas Kapal *Trimaran* terhadap Variasi Sudut Lambung," *Jurnal Kapal dan Teknologi Kelautan*, vol. 19, no. 2, pp. 51–58, 2024.
[11] Dokumentasi Produk, "Sensor Photodiode dan Remote Control," 2022.
[12] "Kapal kelas trimaran," *Wikipedia*, 2023. [Online]. Available: https://id.wikipedia.org/wiki/Kapal_kelas_trimaran
[13] Perpustakaan Universitas Diponegoro, "Studi konfigurasi lambung kapal trimaran," 2025. [Online]. Available: https://media.neliti.com/media/publications/145618-ID-studi-konfigurasi-lambung-kapal-trimaran.pdf
[14] D. Pratama and F. H. Setiawan, "Analisis kinerja kendali kecepatan pada autonomous surface vehicle dengan model trimaran," *Majalah Ilmiah Teknologi Elektro*, vol. 16, no. 4, pp. 123–134, 2024.
[15] M. Fadhillah and A. Lukman, "A systematic review of enhanced stability control in trimaran vessels using modern control algorithms," *Maritime Res. J.*, vol. 16, no. 1, pp. 89–105, 2024.
[16] B. A. Putra and H. Suryadi, "Analisis efektivitas kontrol otomatis berbasis Pixhawk pada sistem kendaraan nirawak," *J. Teknol. Elektro*, vol. 29, no. 3, pp. 215–226, 2023.
[17] R. Hartono and E. Kurniawan, "Pengembangan prototipe sistem monitoring kecepatan kapal berbasis IoT dan Pixhawk," *J. Teknol. Sistem Inf.*, vol. 30, no. 4, pp. 341–355, 2024.
[18] D. Nurhadi et al., "Optimasi sistem navigasi berbasis Pixhawk untuk kapal otomatis di perairan perkotaan," *J. Navig. Sistem Kendali*, vol. 12, no. 2, pp. 133–144, 2022.
[19] IEEE Standard Association, "Pulse Width Modulation (PWM) for Speed Control," *IEEE Std.*, 2021.
[20] M. Anwar et al., "Review of multi-mode control strategies for autonomous vessels," *IEEE J. Oceanic Eng.*, vol. 49, no. 1, pp. 15–28, 2024.
[21] Y. Wang et al., "Recent advances in autonomous surface vessels: a review," *J. Mar. Sci. Technol.*, vol. 27, no. 1, pp. 55–70, 2022.
[22] A. W. Nugraha, "Desain integrasi sistem komunikasi LoRa untuk kapal nirawak," *J. Sist. Kendali dan Otomasi*, vol. 11, no. 2, pp. 89–98, 2023.
[23] P. Santoso *et al.*, "Implementasi sistem IoT pada pemantauan kapal otomatis," *Int. J. Marit. Eng. Technol.*, vol. 8, no. 1, pp. 45–56, 2024.
[24] F. Hidayat and R. Kurniawan, "Analisis performa navigasi kapal otonom berbasis sensor GPS dan IMU," *IEEE Access*, vol. 12, pp. 10811–10820, 2024.
[25] N. Rahman, "Statistical evaluation of control performance in unmanned marine vehicles," *Ocean Eng.*, vol. 298, 2024.
[26] J. Y. Kim and T. Lee, "Adaptive PID control for marine propulsion systems," *IEEE Trans. Control Syst. Technol.*, vol. 31, no. 2, pp. 278–290, 2023.
[27] R. S. Dewanto, "Optimasi kendali PID pada sistem propulsi kapal otonom," *J. Tek. Elektro dan Komput.*, vol. 14, no. 2, pp. 67–75, 2023.

LAMPIRAN

Publikasi Kode

```
// ESP32 LoRa Remote Control Bridge
// Forwards data between Serial, LoRa, and Bluetooth with remote control features
// Using RXD2(16) and TXD2(17) for LoRa module
// Using built-in Bluetooth Classic (Serial Profile)
// Remote control with 8 modes and analog steering (spam mode)

#include "BluetoothSerial.h"
#include <LiquidCrystal_I2C.h>
#include <Wire.h>

// LoRa pins
#define RXD2 17  // LoRa TX to ESP32 RX2
#define TXD2 16  // LoRa RX to ESP32 TX2

// LED pin for visual feedback
#define LED_BUILTIN 2

// Button pins configuration
const int buttonPins[] = {19, 23, 18, 5, 4, 32, 33};
const int numButtons = sizeof(buttonPins) / sizeof(buttonPins[0]);

// Analog steering pin
const int analogPin = 35;

// Bluetooth Serial object
BluetoothSerial SerialBT;

// LCD Setup
LiquidCrystal_I2C lcd(0x27, 16, 2);

// Device name for Bluetooth
String device_name = "ESP32-LoRa-Remote";

// Variables for LED indication
unsigned long lastLedTime = 0;
bool ledState = false;

// Remote control variables
int currentMode = 1; // Start with mode 1 (Manual)
bool buttonStates[numButtons];
bool lastButtonStates[numButtons];
unsigned long lastButtonTime[numButtons];
const unsigned long debounceDelay = 50;

// Analog control variables (spam mode)
String currentDirection = "";
unsigned long lastAnalogTime = 0;
```

```
const unsigned long analogDelay = 200; // Spam every 200ms

// Mode names for LCD
const String modeNames[] = {
  "", "Manual", "F-BLDC", "F-SMC", "F-Engine",
  "F-Hybrid", "PID", "F-PID", "NN"
};

// Command arrays for each mode
const String modeCommands[][5] = {
  {"", "", "", "", ""}, // Mode 0 (unused)
  {"s", "+", "-", "p", "l"}, // Mode 1: Manual
  {"f6000", "f7000", "f8000", "f9000", "f10000"}, // Mode 2: Fuzzy BLDC
  {"m6000", "m7000", "m8000", "m9000", "m10000"}, // Mode 3: Fuzzy SMC
  {"t6000", "t7000", "t8000", "t9000", "t10000"}, // Mode 4: Fuzzy Engine
  {"g4", "g5", "g7", "g9", "g11"}, // Mode 5: Fuzzy Hybrid
  {"p4", "p5", "p7", "p9", "p12"}, // Mode 6: PID
  {"d4", "d5", "d7", "d9", "d11"}, // Mode 7: Fuzzy PID
  {"n4", "n5", "n7", "n9", "n11"}  // Mode 8: NN
};

// Buffer for LCD
char lcdBuffer[17];

void setup() {
  // Initialize serial communications
  Serial.begin(115200);
  Serial2.begin(9600, SERIAL_8N1, RXD2, TXD2);

  // Initialize LCD
  lcd.init();
  lcd.backlight();

  // Initialize Bluetooth Serial
  if(!SerialBT.begin(device_name)) {
    Serial.println("An error occurred initializing Bluetooth");
  } else {
    Serial.println("Bluetooth initialized successfully");
    Serial.println("The device started, now you can pair it with bluetooth!");
    Serial.print("Device Name: ");
    Serial.println(device_name);
  }

  // Setup button pins
  for(int i = 0; i < numButtons; i++) {
    pinMode(buttonPins[i], INPUT_PULLUP);
    buttonStates[i] = HIGH;
    lastButtonStates[i] = HIGH;
    lastButtonTime[i] = 0;
  }
```

```
 // Set pin mode for LED
 pinMode(LED_BUILTIN, OUTPUT);

 // Display startup message
 Serial.println("\n========================================");
 Serial.println("ESP32 LoRa Remote Control Bridge Started");
 Serial.println("Remote Control with 8 modes + Spam Mode");
 Serial.println("Analog steering: 200ms spam mode");
 Serial.println("========================================");

 // LCD startup
 sprintf(lcdBuffer, "LoRa Remote Ctrl");
 lcd.setCursor(0, 0);
 lcd.print(lcdBuffer);
 sprintf(lcdBuffer, "Starting...     ");
 lcd.setCursor(0, 1);
 lcd.print(lcdBuffer);

 // LED startup sequence
 for(int i = 0; i < 3; i++) {
  digitalWrite(LED_BUILTIN, HIGH);
  delay(200);
  digitalWrite(LED_BUILTIN, LOW);
  delay(200);
 }

 delay(1000);
 updateLCD();
}

void flashLED(int duration_us) {
 digitalWrite(LED_BUILTIN, HIGH);
 delayMicroseconds(duration_us);
 digitalWrite(LED_BUILTIN, LOW);
}

void sendCommand(String command) {
 // Send to Serial
 Serial.println(command);

 // Send to LoRa
 Serial2.println(command);

 // Send to Bluetooth if connected
 if(SerialBT.hasClient()) {
  SerialBT.println(command);
 }

 // Flash LED for command sent
```

```
 flashLED(500);

 Serial.print("CMD: ");
 Serial.println(command);
}

void updateLCD() {
 // Line 1: Mode info
 sprintf(lcdBuffer, "Mode%d:%s     ", currentMode, modeNames[currentMode].c_str());
 lcd.setCursor(0, 0);
 lcd.print(lcdBuffer);

 // Line 2: Current direction and analog value
 int analogValue = analogRead(analogPin);
 String dirText = "";
 if(analogValue > 1500) dirText = "KIRI";
 else if(analogValue < 500) dirText = "KANAN";
 else dirText = "LURUS";

 sprintf(lcdBuffer, "%s A:%4d   ", dirText.c_str(), analogValue);
 lcd.setCursor(0, 1);
 lcd.print(lcdBuffer);
}

void handleButtons() {
 for(int i = 0; i < numButtons; i++) {
  int reading = digitalRead(buttonPins[i]);

  if(reading != lastButtonStates[i]) {
   lastButtonTime[i] = millis();
  }

  if((millis() - lastButtonTime[i]) > debounceDelay) {
   if(reading != buttonStates[i]) {
    buttonStates[i] = reading;

    if(buttonStates[i] == LOW) { // Button pressed
     if(buttonPins[i] == 33) { // Mode change button
      currentMode++;
      if(currentMode > 8) currentMode = 1;
      updateLCD();
      Serial.print("Mode changed to: ");
      Serial.print(currentMode);
      Serial.print(" (");
      Serial.print(modeNames[currentMode]);
      Serial.println(")");
     }
     else if(buttonPins[i] == 19) { // Always send 'x'
      sendCommand("x");
     }
     else { // Other buttons - send mode-specific commands
```

```
        int buttonIndex = -1;
        if(buttonPins[i] == 32) buttonIndex = 0;
        else if(buttonPins[i] == 4) buttonIndex = 1;
        else if(buttonPins[i] == 5) buttonIndex = 2;
        else if(buttonPins[i] == 18) buttonIndex = 3;
        else if(buttonPins[i] == 23) buttonIndex = 4;

        if(buttonIndex >= 0) {
          sendCommand(modeCommands[currentMode][buttonIndex]);
        }
       }
      }
     }
   }
   lastButtonStates[i] = reading;
  }
}

void handleAnalogSteering() {
  if(millis() - lastAnalogTime >= analogDelay) {
   int analogValue = analogRead(analogPin);
   String direction = "";

   if(analogValue > 1500) {
    direction = "T:90,A:130"; // Kiri
   }
   else if(analogValue < 500) {
    direction = "T:90,A:50";  // Kanan
   }
   else {
    direction = "T:90,A:90";  // Lurus
   }

   sendCommand(direction);
   currentDirection = direction;
   lastAnalogTime = millis();
  }
}

void loop() {
  handleButtons();
  handleAnalogSteering();

  static unsigned long lastLCDUpdate = 0;
  if(millis() - lastLCDUpdate > 300) {
   updateLCD();
   lastLCDUpdate = millis();
  }
```

```
 while (Serial2.available()) {
  char c = Serial2.read();
  Serial.write(c);
  if(SerialBT.hasClient()) SerialBT.write(c);
  flashLED(300);
 }

 while (Serial.available()) {
  char c = Serial.read();
  Serial2.write(c);
  if(SerialBT.hasClient()) SerialBT.write(c);
  flashLED(600);
 }

 while (SerialBT.available()) {
  char c = SerialBT.read();
  Serial.write(c);
  Serial2.write(c);
  flashLED(900);
 }

 if(millis() - lastLedTime > 2000) {
  if(SerialBT.hasClient()) {
   digitalWrite(LED_BUILTIN, HIGH);
   delay(50);
   digitalWrite(LED_BUILTIN, LOW);
  }
  lastLedTime = millis();
 }
}

// Optional: Bluetooth event callbacks for debugging
void bluetoothEventCallback(esp_spp_cb_event_t event, esp_spp_cb_param_t *param) {
 if(event == ESP_SPP_SRV_OPEN_EVT) {
  Serial.println("Bluetooth Client Connected");
 }
 if(event == ESP_SPP_CLOSE_EVT) {
  Serial.println("Bluetooth Client Disconnected");
 }
}
```