Jurnal Elkolind Volume 10, Nomor 2, Juli 2023
DOI: http://dx.doi.org/10.33795/elkolind.v10i2.3720

Peningkatan Kestabilan Sistem Kontrol UGV melalui Optimalisasi Manajemen Core dan Free-RTOS pada ESP32

Gillang Al Azhar[1], Sungkono[2], Mas Nurul Achmadiyah[3], Syarifatul Izza[4]
e-mail: gillang_al_azhar@polinema.ac.id, sungkono@polinema.ac.id, masnurul@polinema.ac.id, syarifatulizza95@gmail.com
[1,2,3]Jurusan Teknik Elektro, Politeknik Negeri Malang, Jalan Soekarno Hatta No.9 Malang, Indonesia
[4]Jurusan Teknik Listrik, Politeknik Unisma Malang, Jalan Mayjen Haryono No.193, Malang, Indonesia

Informasi Artikel

Riwayat Artikel

Diterima 5 Juni 2023
Direvisi 4 Juli 2023
Diterbitkan 31 Juli 2023

Kata kunci:

UGV
DualCore
Free-RTOS
ESP32

ABSTRAK

Penggunaan Mikrokontroler yang memiliki kemampuan tinggi menjadi kebutuhan utama dalam implementasinya kedalam UGV(*Unmanned Grounded Vehicle*), dengan kelangkaan jumlah mikrokontroler dengan kemampuan tinggi, menjadikan ESP32 sebagai alternatif pengganti yang cocok untuk diimplementasikan bila mampu digunakan secara optimal. Optimalisasi penggunaan ESP32 dilakukan dengan memanfaatkan *DualCore* yang dimiliki ESP32 untuk melakukan manajemen aliran program pada ESP32. Hal ini bertujuan agar meminimalisir terjadinya delay dan latency data pada pemrosesan data. Pengujian untuk penggunaan fitur *DualCore* dan *Free-RTOS* ESP32 dilakukan dengan cara menentukan waktu sampling setiap 100 ms, dengan waktu looping maksimal adalah 80 ms. Hasil waktu aktual yang didapatkan memiliki catatan waktu yang stabil dengan rata – rata maksimal waktu sampling adalah 75,3 ms, sedangkan untuk catatan waktu tanpa penggunaan *DualCore* dan *Free-RTOS* memiliki waktu sampling maksimal sebesar 88.9 ms. Serta didapatkan rata - rata suhu dari chip ESP32 saat menggunakan *DualCore* dan *Free-RTOS* sebesar 58.8°C. Hal tersebut menunjukkan penggunaan *DualCore* ESP32 dan *Free-RTOS* dapat memberikan sampling waktu yang lebih stabil dengan rentang waktu yang sesuai dengan desain serta tidak membebani kerja chip ESP32 yang ditunjukkan suhu kerja yang dimiliki masih mendekati suhu kerja normalnya yaitu 53.3°C.

Keywords:

UGV
DualCore
Free-RTOS
ESP32

ABSTRACT

The use of high-capability microcontrollers is a primary requirement in their implementation into UGV (Unmanned Grounded Vehicle). Due to the scarcity of high-capability microcontrollers, ESP32 has emerged as a suitable alternative for implementation if used optimally. The optimization of ESP32 usage is achieved by leveraging its DualCore to manage program flow effectively. This optimization aims to minimize data delay and latency during data processing. Trials of the DualCore and Free-RTOS features of ESP32 was conducted by setting a sampling time of 100 ms, with a maximum loop time of 80 ms. The actual results showed a stable time record with a maximum sampling time of 77 ms when utilizing the DualCore and Free-RTOS features. In contrast, the time record without using DualCore and Free-RTOS had a maximum sampling time of 88.9 ms. Additionally, the average chip temperature of the ESP32 was measured at 58.8°C when using DualCore and Free-RTOS. These findings indicate that employing DualCore ESP32 and Free-RTOS can provide more stable sampling times within the desired time range as per the design. Furthermore, it does not overburden the ESP32 chip, as evidenced by its operating temperature remaining close to the normal working temperature of 53.3°C.

p-ISSN: 2356-0533; e-ISSN: 2355-9195

Penulis Korespondensi:
Gillang Al Azhar
Jurusan Teknik Elektro
Politeknik Negeri Malang,
Jalan Soekarno Hatta No.9 Malang, Indonesia, 65144
Email: gillang_al_azhar@polinema.ac.id
Nomor HP/WA aktif: +62 812-176-028-89

1. PENDAHULUAN

Implementasi teknologi robotika semakin beragam seiring dengan kemajuan teknologi otomasi yang pesat. Berbagai macam implementasi teknologi robotika diaplikasikan kedalam berbagai bidang otomasi [1], baik didunia industri [2], militer, maupun masyarakat [3]. Salah satu adopsi teknologi otomasi robotika dibidang industri adalah dikembangkannya sistem robot beroda [4], [5]. Berbagai macam jenis robot beroda diadopsi untuk dijadikan UGV (*Unmanned Grounded Vehicle*) seperti robot beroda dengan sistem penggerak *holonomic* [6], [7], [8] dan robot beroda dengan sistem penggerak *non-holonomic* [9], [10]. Dengan terus majunya teknologi robotika dan kemajuan dalam kecerdasan buatan, penelitian di bidang UGV menjadi semakin menarik dan relevan untuk menghadapi tantangan masa depan dan menciptakan solusi yang inovatif dan efisien untuk berbagai aplikasi di dunia nyata. Namun, dengan semakin kompleksnya implementasi robotika dibidang UGV, dibutuhkan performa kontroler yang lebih tinggi, dimana tingkat akurasi sampling waktu yang dibutuhkan untuk menjalankan program semakin menjadi kebutuhan utama [11] . Mikrokontroler ESP32 merupakan salah satu mikrokontroler yang memiliki kemampuan untuk memenuhi kebutuhan tersebut. Kemampuan ESP32 untuk menjalankan tugas berat [12] dapat dijadikan sebagai alternatif penggunaan mikrokontroler berkemampuan tinggi lainnya yang semakin langkahnya ketersediaannya seperti STM32, ataupun ARM. Sehingga penggunaan ESP32 sebagai kontroler UGV dapat dijadikan pertimbangan untuk direalisasikan, dimana hal tersebut bergantung bagaimana fitur yang dimiliki chip ESP32 digunakan secara optimal.

Dengan dilatar belakangi hal tersebut maka pada penelitian yang dilakukan digunakan ESP32 sebagai mikrokontroler utama dari purwarupa UGV. Untuk penggunaan ESP32 secara optimal, dilakukan dengan memanfaatkan 2 buah core pada ESP32 (*DualCore*) yang digunakan untuk menjalankan sistem purwarupa UGV. Selain itu dengan fitur *Free-RTOS*, memungkinkan ESP32 untuk dapat membagi beban kerja yang sebelumnya dibebankan pada 1 core saja, menjadi kepada 2 core yang dimiliki ESP32 [13], [14]. Untuk pengujian peforma dari manajemen *core* dan penggunaan fitur yang ada pada ESP32, dilakukan pengujian akurasi waktu sampling yang dijalankan secara aktual oleh ESP32 dengan dan tanpa mengaktifkan fitur *DualCore* untuk menyelesaikan seluruh beban kerja program yang ada. Selain itu juga dicek suhu dari chip ESP32 yang digunakan ketika menjalankan program dengan mengaktifkan fitur *DualCore* dan *Free-RTOS* serta digunakan untuk mengendalikan putaran motor melalui sensor suhu internal yang telah tersedia didalam board ESP32.

2. METODE PENELITIAN

Penelitian yang dilakukan mengacu pada purwarupa UGV yang telah dibuat pada penelitian sebelumnya [15], dimana purwarupa UGV memiliki 2 buah motor DC yang dilengkapi dengan internal rotary encoder, selain itu purwarupa UGV juga dilengkapi dengan sensor Kompas BNO055 yang pembacaannya dilakukan dengan menggunakan komunikasi data I2C, dan juga purwarupa UGV dilengkapi dengan sensor lidar yang dipasangkan pada kontroler lain yang berada pada purwarupa UGV berupa Mini Komputer (Mini PC) yang dihubungkan dengan ESP32 melalui komunikasi Serial. User Interface (UI) yang digunakan pada purwarupa robot menggunakan LCD OLED 128x64, yang mana tersambung dengan komunikasi I2C pada ESP32. Berdasarkan device yang telah terpasang pada purwarupa UGV dibutuhkan perancangan untuk penggunaan fitur *DualCore* dan *Free-RTOS* pada ESP32 agar semua device dapat beroperasi secara optimal.

2.1 Arah dan Tujuan Penelitian

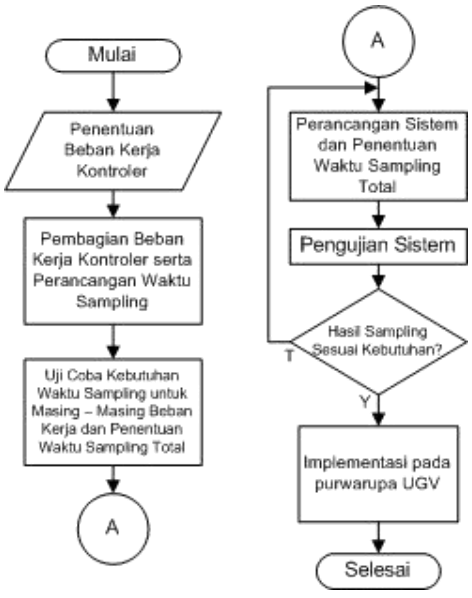
Arah dari penelitian yang dilakukan adalah untuk dapat membuat sistem kontrol dari purwarupa UGV dapat berjalan secara optimal, dimana sistem kontrol didesain untuk dapat menjalankan beberapa *task* program yang

harus dilakukan secara periodik dan secara bersamaan. Untuk daftar *task* program yang harus dijalankan dapat dilihat pada TABEL 1 berikut

TABEL 1 : DAFTAR TASK PROGRAM YANG HARUS DIJALANKAN KONTROLER

No	Jenis *Task* Program	Penjelasan *Task* Program
1.	User Interface	Interface menggunakan OLED 128x64
2.	Pembacaan Sensor Kompas	Pembacaan Sensor Kompas dilakukan melalui Komunikasi I2C
3.	Pembacaan 2 Sensor Encoder	Sensor Encoder terpasang pada motor kanan dan kiri penggerak purwarupa UGV, dibutuhkan fitur interupsi sebanyak 4x untuk masing - masing pembacaan sensor
4.	Pengolahan Data Encoder menjadi Kecepatan Motor	Konversi data encoder menjadi nilai RPM dilakukan secara periodic dengan sampling waktu tertentu
5.	Komunikasi Data Serial	Komunikasi Serial dilakukan untuk melakukan pertukaran data dengan device kontroler lain yang digunakan didalam purwarupa UGV
6.	Output PWM 4 channel	PWM digunakan untuk mengendalikan kecepatan dan arah putaran masing – masing motor
7.	Low Level Control untuk Motor DC	Pengendalian Putaran Motor dalam RPM berdasarkan nilai set Point yang didapatkan dari output kontrol invers-kinematik UGV

Berdasarkan daftar *task* program yang harus dijalankan tersebut, dapat dilihat bahwa sistem yang dijalankan memiliki sistem pengolahan data dengan kebutuhan tingkat kepresisian yang sangat tinggi. Sehingga tujuan dari penelitian yang dilakukan adalah agar mampu memanfaatkan ESP32 sebagai kontroler yang terpasang pada purwarupa UGV dapat dimanfaatkan untuk menjalankan *sistem* secara optimal dan sesuai dengan yang dirancang. Karena dengan semakin stabilnya sampling waktu untuk mengerjakan sistem looping didalam program, maka akan menghasilkan sistem kontrol yang semakin stabil, dengan begitu akurasi pergerakan dan manuver yang dijalankan purwarupa UGV akan semakin tepat dan akurat [16].



Gambar 1: Flowchart Alur Penyelesaian Penelitian

Untuk dapat tujuan tersebut maka penelitian ini dilaksanakan secara sistematis melalui beberapa tahapan seperti yang ditunjukkan pada Gambar 1.

DOI: http://dx.doi.org/10.33795/elkolind.v10i2.3720

2.2 Waktu Pengerjaan *Task* Program yang Harus dijalankan

Dalam menjalankan sistem control UGV, ESP32 menjalankan beberapa task program secara sekaligus, baik dari pembacaan sensor, komunikasi data, hingga pengendalian motor penggerak UGV tersebut. Untuk masing – masing task program yang dijalankan memiliki waktu yang dibutuhkan untuk melakukan satu kali proses looping program. Agar sistem yang dijalankan dapat berjalan secara optimal, yaitu masing – masing task program tidak akan mengganggu proses jalannya program antara satu dengan yang lainnya, maka perlu diperhitungkan kebutuhan waktu untuk masing – masing task program yang dijalankan. Kemudian dari catatan waktu yang dibutuhkan untuk setiap task program tersebut, dapat dilakukan pengelompokan task program mana saja yang dijalankan di masing – masing Core yang ada pada ESP32, dengan catatan total waktu task program yang dijalankan di setiap Core tidak melebihi sampling waktu yang telah ditentukan. Untuk sampling waktu ditentukan berdasarkan spesifikasi waktu sampling paling lambat untuk sistem UGV yaitu sebesar 100 ms. Namun untuk menghindari terjadinya crash program atau *hank* pada sistem, seluruh task program harus diselesaikan dalam waktu maksimal 80% dari waktu sampling yang dimiliki, sehingga sistem kontroler akan memiliki *rest time* sebesar 20% dari waktu sampling, sehingga prosentase kemungkinan terjadinya sistem error ketika dijalankan akan semakin kecil [17].

a. Waktu yang dibutuhkan komunikasi data serial

Komunikasi data serial digunakan oleh ESP32 untuk mengirimkan dan menerima data dengan mini komputer yang terpasang pada purwarupa UGV, dimana data yang dikirimkan ESP32 ke mini komputer adalah berupa data sensor – sensor yang terpasang pada purwarupa UGV, sedangkan untuk data yang diterima adalah nilai output dari perhitungan kontrol kinematik yang dijalankan didalam mini komputer. Beberapa parameter komunikasi serial yang digunakan adalah baudrate sebesar 115.200, hal ini bertujuan agar kecepatan komunikasi data serial antara ESP32 dan mini komputer dapat berjalan dengan cepat, sehingga waktu yang dibutuhkan untuk pertukaran data akan semakin cepat. Berdasarkan nilai baudrate tersebut, dapat dihitung kecepatan transfer data per bit seperti yang ditunjukkan pada (1).

$$bps = \frac{1}{baudrate} \tag{1}$$

$$bps = \frac{1}{115.200} = 8{,}6\ x\ 10^{-6}$$

Sehingga didapatkan kecepatan transfer data per bit sebesar $8{,}6\ x\ 10^{-6}$ detik atau sebesar 8,6 bit/mikrosekond. Dalam komunikasi data serial, satu paket data berisikan 10 bit data, dimana bit ke-1 merupakan sinyal start, kemudian bit ke 2 hingga 9 adalah data yang dikirimkan, dan bit ke-10 merupakan sinyal stop atau data karier. Berdasarkan protocol pengiriman data serial tersebut, dapat dihitung waktu yang dibutuhkan untuk mengirim 1 paket data dengan (2) sebagai berikut

$$\begin{aligned} t_{serial} &= bps\ x\ bit_{data} \\ &= 8{,}6\ x\ 10^{-6}\ x\ 10 \\ &= 8.6\ x\ 10^{-5}\ s = 86\ us \end{aligned} \tag{2}$$

Berdasarkan perhitungan yang telah dilakukan didapatkan waktu pengiriman data untuk 1 paket data atau 1 Byte serial adalah $86\ us$. Pada sistem yang dijalankan total paket data yang dikirim dan diterima oleh ESP32 adalah sebesar 600 Byte sehingga dapat dihitung waktu total dari proses komunikasi serial dengan perhitungan berikut

$$\begin{aligned} t_{total} &= t_{byte}\ x\ \Sigma\text{byte} \\ &= 86\ us\ x\ 600 \\ &= 51.600\ us = \ 51{,}6\ ms \end{aligned} \tag{3}$$

Dengan menggunakan (3) didapatkan hasil waktu total yang dibutuhkan untuk satu kali looping task program komunikasi serial adalah sebesar $51{,}6\ ms$.

b. Waktu yang dibutuhkan komunikasi data I2C

Konfigurasi kecepatan komunikasi data I2C dapat diatur berdasarkan frekuensinya. ESP32 memiliki frekuensi *default* untuk komunikasi melalui I2C bus adalah sebesar 100 KHz. Sehingga berdasarkan spesifikasi tersebut dapat dihitung waktu yang dibutuhkan untuk melakukan pengiriman 1 Byte data yaitu dengan cara berikut

$$T_{I2C} = \frac{1}{f_{I2C}} = \frac{1}{100.000} = 10\ us \quad (4)$$

$$T_{1b_{i2c}} = 10\ us\ x\ 8 = 80\ us$$

Berdasarkan (4) didapatkan waktu untuk mengirim 1 bit data dibutuhkan waktu sebesar $10\ us$, sedangkan untuk komunikasi I2C memiliki format pengiriman paket data sebesar 1 Byte, sehingga waktu yang dibutuhkan untuk melakukan pengiriman 1 Byte data adalah sebesar $80\ us$. Dalam penggunaannya, komunikasi I2C diaplikasikan untuk membaca data dari sensor Kompas Adafruit BNO055 dan mengirimkan data interface yang ditampilkan didalam display OLED 128x64. Untuk komunikasi data antara ESP32 dan sensor Kompas memiliki ukuran data sebesar 26 Byte, dan untuk pengiriman data UI pada OLED memiliki ukuran data sebesar 900 Byte. Sehingga dapat dihitung kebutuhan waktu untuk masing – masing task program tersebut sebagai berikut

$$T_{BNO055} = T_{1B_{i2c}}\ x\ B_{BNO055} \quad (5)$$

$$= 80\ us\ x\ 26 = 2080\ us = 2{,}08\ ms$$

$$T_{OLED} = T_{1B_{i2c}}\ x\ B_{OLED} \quad (6)$$

$$= 80\ us\ x\ 900 = 72.000\ us = 72\ ms$$

Diperlihatkan pada (5) waktu yang dibutuhkan untuk transfer data sensor Kompas sebesar $2{,}08\ ms$, dan untuk transfer data pada display OLED berdasarkan (6) didapatkan waktu sebesar $72\ ms$.

c. Manajemen Pembagian Beban Kerja pada ESP32

Selain *task* komunikasi Serial dan I2C, terdapat task lain seperti yang ditunjukkan pada TABEL 1. Sama seperti halnya komunikasi data, untuk masing – masing task program lain juga dihitung kebutuhan waktunya. Hal ini dilakukan dengan cara memanfaatkan fitur fungsi internal timer millis() untuk mengetahui berapa waktu yang dibutuhkan untuk melakukan 1 kali looping masing – masing task program tersebut. Untuk hasil pengujian waktu yang dibutuhkan dapat dilihat pada TABEL 2.

TABEL 2 : CATATAN WAKTU UNTUK PENGERJAAN MASING - MASING *TASK* PROGRAM

No	Jenis *Task* Program	Waktu
1.	UI dengan OLED melalui komunikasi I2C	72 ms
2.	Pembacaan Sensor Kompas melalui komunikasi I2C	2 ms
3.	Pembacaan 2 Sensor Encoder	3 ms
4.	Pengolahan Data Encoder menjadi Kecepatan Motor dan Low Level Control untuk Motor DC	25 ms
5.	Komunikasi Data Serial	51,6 ms
6.	Output PWM 4 channel	1 ms
7.	Scanning GPIO (Fungsi Pembacaan Tombol)	2 ms
Total Waktu		**156,6 ms**

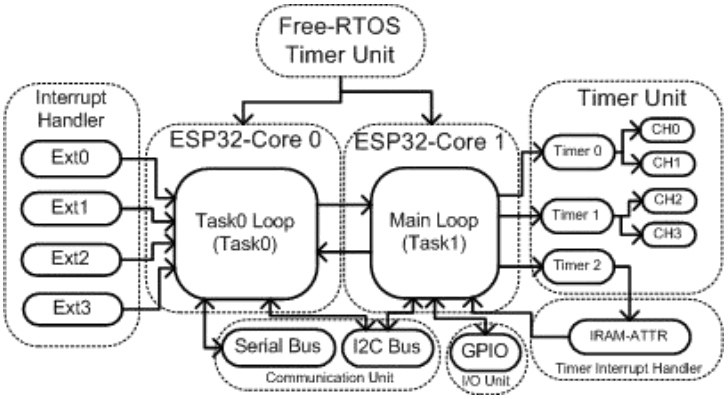
Berdasarkan TABEL 2. Didapatkan waktu total keseluruhan yang dibutuhkan untuk melakukan satu kali looping sebesar $156{,}6\ ms$, dimana hal tersebut menunjukkan kebutuhan waktu melebihi batas optimal dari waktu looping yaitu sebesar $100\ ms$. Sehingga dibutuhkan manajemen *task* program yang dijalankan didalam ESP32 agar masing – masing looping *task* program dapat berjalan maksimal 80% dari sampling maksimal ($100\ ms$), sehingga didapatkan waktu sebesar $80\ ms$. Maka pembagian task berdasarkan Core dan Free-RTOS ESP32 ada pada TABEL 3.

TABEL 3 : CATATAN WAKTU UNTUK PENGERJAAN MASING - MASING *TASK* PROGRAM BERDASARKAN PEMBAGIAN CORE

No	Pembagian Core					
	CORE0	**Waktu**	**CORE1**	**Waktu**	**IRAM_ATTR**	**Waktu**
1.	Sensor Kompas	2 ms	UI display OLED	72 ms	Pembacaan RPM dan Low Level Control	25 ms
2.	Pembacaan 2 Encoder	3 ms	Output PWM 4 Channel	1 ms		
3.	Komunikasi Data Serial	51,6 ms	Scanning GPIO	2 ms		
	Total Waktu	**56,6 ms**		**75 ms**		**25 ms**

Task program dijalankan didalam 3 proses yang mampu berjalan secara parallel, yaitu sistem *DualCore* yang dijalankan dengan memanfaatkan fitur *Free-RTOS* didalam ESP32 yang berisikan 2 proses, dan 1 proses lainnya menggunakan IRAM_ATTR sebagai unit yang berguna untuk menjalankan *ISR (Interrupt Service Routine)* didalam ESP32. Dengan pembagian *task* program berdasarkan Core yang digunakan didapatkan untuk masing – masing waktu looping adalah 56,6 ms untuk Core1, 75 ms untuk Core2, dan 25 ms untuk IRAM_ATTR, sehingga berdasarkan pembagian tersebut didapatkan masing – masing core unit ESP32 membutuhkan waktu yang kurang dari 80 ms untuk menjalankan looping sistemnya.

2.3 Perancangan Sistem *DualCore* dan *Free-RTOS* ESP32



Gambar 2: Diagram blok manajemen fitur ESP32

Untuk manajemen *task* program yang dibebankan pada masing – masing Core dirancang agar tidak menjalankan looping program selama lebih dari 80 ms. Pada Gambar 2 dapat dilihat diagram block sistem *DualCore* yang dijalankan didalam ESP32 berdasarkan task program yang dijalankan mengacu pada TABEL 3. Diasumsikan nilai waktu sampling 80 ms merupakan 80% dari sampling waktu maksimal yang didesain, sehingga untuk waktu sampling keseluruhan sistem dapat dihitung menjadi

$$\frac{80ms}{ts} = \frac{80\%}{100\%} \tag{7}$$

$$ts = \frac{80\ x\ 100\ x\ 10^{-2}}{80\ x\ 10^{-2}} = 100\ ms$$

Sehingga dengan perhitungan (7), didapatkan nilai time sampling untuk sistem adalah 100 ms, dengan rincian 80 ms merupakan waktu maksimal pengerjaan *task* program sistem keseluruhan, dan 20 ms adalah *resting time* untuk chip ESP32 agar mengurangi kemungkinan terjadinya error sistem atau *hank* pada sistem. Sehingga untuk membuat waktu sampling pada ESP dilakukan setiap 100 ms dengan asumsi batas maksimal pengerjaan *task* program dapat diselesaikan dalam waktu 80 ms, Langkah pertama adalah dengan menghitung clock yang diberikan pada unit timer apda ESP32 dengan melalui berikut

$$f_{timer2} = \frac{f_{clk}}{Prescaler} \tag{8}$$

$$= \frac{80.000.000}{80}$$
$$= 1.000.000 = 10^6$$

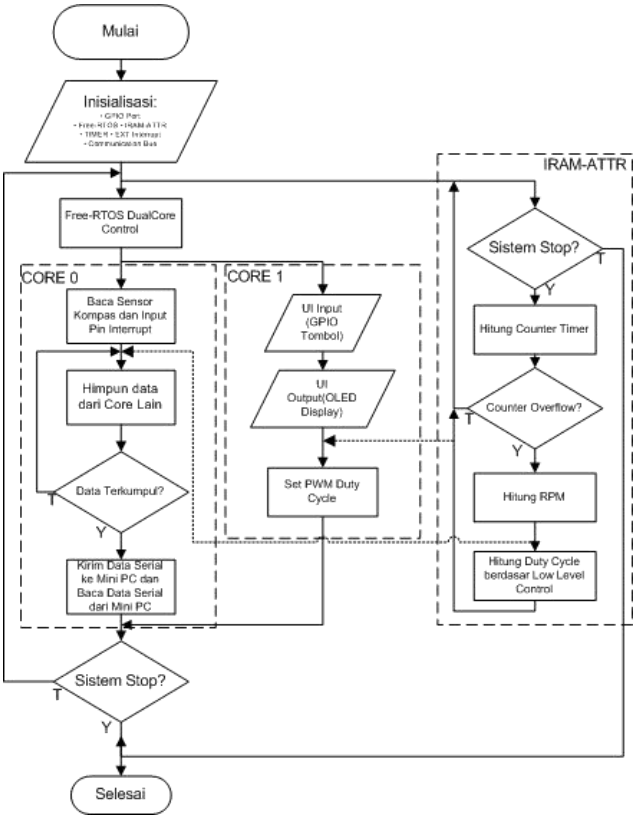
Berdasarkan (8), dengan menggunakan prescaler 80, didapatkan hasil frekuensi clock untuk timer sebesar $1\,MHz$, dimana diselanjutnya dapat diketahui periode clock sebagai trigger counter timer dengan cara berikut

$$T_{clock} = \frac{1}{f_{timer2}} \quad (9)$$
$$= \frac{1}{10^6} = 1uS$$

Dari hasil yang didapatkan (9), diketahui dengan menggunakan prescaler 80, periode 1 clock untuk melakukan hitungan timer adalah $1\,us$. Berdasarkan nilai periode tersebut dapat dihitung berapa banyak counter dari timer agar dapat menghasilkan interupsi setiap $100\,ms$ sebagai waktu sampling sistem yang dijalankan didalam ESP32. Berikut merupakan perhitungan nilai counter yang digunakan untuk mengkonfigurasi interrupt timer

$$T_{OVF} = T_{clock}\ x\ Count \quad (10)$$
$$0{,}1 = 0{,}000001\ x\ Count$$
$$Count = \frac{0{,}1}{0{,}000001} = 100.000$$

Dari hasil (10), dapat dilihat bahwa untuk dapat melakukan time sampling sebesar $100\,ms$, counter dari timer harus melakukan mencapai 100.000 kali.



Gambar 3: Flowchart jalannya sistem didalam ESP32

Sehingga seluruh konfigurasi ESP32 untuk menjalankan *task* program agar sesuai dengan kebutuhan yang telah dirancang untuk diselesaikan dalam waktu kurang dari $80\,ms$ pada setiap core yang digunakan. Sehingga untuk jalannya sistem yang ada didalam ESP32 dapat dilihat pada Flowchart jalannya program pada Gambar 3.

2.4 Rancangan Pengujian

Untuk mengetahui peforma hasil perancangan sistem *DualCore* yang dijalankan dalam ESP maka dilakukan pengujian sebagai berikut

a. Pengujian Waktu Sampling Tanpa Menggunakan *DualCore* pada ESP32

Pengujian dilakukan dengan menjalankan seluruh sistem dengan menggunakan Core *default* dari ESP32 dengan bantuan IRAM-ATTR saja, dimana data yang diambil adalah waktu yang dibutuhkan untuk menyelesaikan seluruh program dalam satu siklus loop didalam ESP32. Penghitungan waktu sampling aktual dilakukan dengan memanfaatkan fitur internal timer yaitu fungsi millis(), dimana waktu looping adalah selisih antara nilai millis pada saat memulai mengerjakan satu siklus *looping* program hingga memulai siklus pengerjaan siklus berikutnya. Pengambilan data akan dilakukan dengan cara program dijalankan selama 1 menit dengan selang waktu pengambilan data setiap 1 detik sekali.

b. Pengujian Waktu Sampling Dengan Menggunakan *DualCore* pada ESP32

Pengujian dilakukan dengan menjalankan seluruh sistem dengan menggunakan *DualCore* dari ESP32 dengan bantuan IRAM-ATTR sesuai dengan pembagian sistem yang telah dibuat, dimana data yang diambil adalah waktu dari masing – masing Core dalam menyelesaikan seluruh program yang telah dibebankan dalam satu siklus loop didalam ESP32. Penghitungan waktu sampling aktual dilakukan dengan memanfaatkan fitur internal timer yaitu fungsi millis(), dimana waktu looping adalah selisih antara nilai millis pada saat memulai mengerjakan satu siklus *looping* program hingga memulai siklus pengerjaan siklus berikutnya. Pengambilan data akan dilakukan dengan cara program dijalankan selama 1 menit dengan selang waktu pengambilan data setiap 1 detik sekali.

c. Pengujian Suhu Chip ESP32

Pengujian dilakukan dengan menjalankan seluruh sistem dengan menggunakan *DualCore* dari ESP32 dengan bantuan IRAM-ATTR sesuai dengan pembagian sistem yang telah dibuat, dimana data yang diambil adalah suhu dari chip ESP32 ketika menjalankan program dalam kondisi full load, untuk mengetahui seberapa berat beban yang diberikan pada ESP32 ketika mengaktifkan fungsi *DualCore* dan dibantu dengan IRAM-ATTR untuk menjalankan seluruh sistem pada UGV. Hasil yang didapat akan dirata – rata dan dibandingkan dengan suhu normal chip yaitu $53{,}3^{o}C$. Untuk pengukuran suhu dilakukan dengan memanfaatkan sensor suhu internal yang ada didalam chip ESP32. Pengambilan data akan dilakukan dengan cara program dijalankan selama 1 menit dengan selang waktu pengambilan data setiap 1 detik sekali.

3. HASIL DAN PEMBAHASAN

Berikut ini merupakan hasil yang didapatkan dari pengujian yang telah dirancang sebelumnya, dimana terdapat 3 jenis pengujian yang dilakukan.

3.1 Hasil Pengujian Waktu Sampling Tanpa Menggunakan *DualCore* pada ESP32

Berdasarkan uji coba yang dilakukan untuk menjalankan seluruh *task* program tanpa menggunakan DualCore dari ESP32, didapatkan catatan waktu sebagai berikut

TABEL 4: CATATAN WAKTU RATA – RATA SETIAP DETIK UNTUK LOOPING PADA CORE0 DAN IRAM ESP32

Rata – rata Waktu Looping								
detik	**Core1**	**IRAM**	**detik**	**Core1**	**IRAM**	**detik**	**Core1**	**IRAM**
1.	79,1 ms	70,2 ms	21.	87,9 ms	71,1 ms	41.	82,3 ms	70,6 ms
2.	82,3 ms	71,4 ms	22.	94,1 ms	70,9 ms	42.	86,2 ms	71,1 ms
3.	86,2 ms	69,9 ms	23.	97,3 ms	69,6 ms	43.	83,4 ms	69,9 ms
4.	83,4 ms	70,2 ms	24.	98,4 ms	69,9 ms	44.	94,1 ms	70,2 ms
5.	89,8 ms	70,4 ms	25.	91,2 ms	70,2 ms	45.	97,3 ms	70,4 ms
6.	91,4 ms	71,2 ms	26.	88,6 ms	70,4 ms	46.	98,4 ms	70,9 ms

DOI: http://dx.doi.org/10.33795/elkolind.v10i2.3720

7.	87,5 ms	70,6 ms	27.	79,1 ms	69,7 ms	47.	91,2 ms	69,6 ms
8.	84,6 ms	71,1 ms	28.	82,3 ms	70,1 ms	48..	88,6 ms	69,9 ms
9	84,6 ms	70,9 ms	29.	86,2 ms	70,3 ms	49.	87,9 ms	70,2 ms
10.	87,9 ms	69,6 ms	30.	83,4 ms	70,2 ms	50.	94,1 ms	70,4 ms
11.	94,1 ms	69,9 ms	31.	89,8 ms	69,9 ms	51.	97,3 ms	69,7 ms
12.	97,3 ms	70.8 ms	32.	87,5 ms	69,7 ms	52.	98,4 ms	70,1 ms
13.	98,4 ms	70,3 ms	33.	84,6 ms	70,2 ms	53.	82,3 ms	70.8 ms
14.	91,2 ms	70,2 ms	34.	84,6 ms	70,4 ms	54.	86,2 ms	70,3 ms
15.	88,6 ms	69,9 ms	35.	87,9 ms	71,2 ms	55.	83,4 ms	70,2 ms
16.	94,4 ms	69,7 ms	36.	91,2 ms	69,9 ms	56.	91,2 ms	69,9 ms
17.	98,4 ms	70,1 ms	37.	86,2 ms	70.8 ms	57.	88,6 ms	71,1 ms
18.	93,2 ms	70,3 ms	38.	83,4 ms	70,3 ms	58.	82,3 ms	69,9 ms
19.	86,2 ms	71,1 ms	39.	89,8 ms	70,6 ms	59.	86,2 ms	70,2 ms
20.	88,9 ms	70,9 ms	40.	87,9 ms	71,1 ms	60.	83,4 ms	70,4 ms
Rata – rata waktu Core1 : $88,9\ ms$								
Rata – rata waktu IRAM : $70,6\ ms$								

Berdasarkan TABEL 4, dapat dilihat bahwa waktu yang dibutuhkan untuk melakukan satu kali looping program membutuhkan rata – rata waktu sebesar 88,9 ms untuk penggunaan *SingleCore*, dimana kondisi ini hanya Core1 sebagai *core default* yang menjalankan program utama, dan untuk *IRAM-ATTR* yang membantu kinerja Core1 memiliki catatan waktu rata – rata sebesar 70,6 ms. Memang untuk catatan waktu masih ada dibawah 100 ms, namun berdasarkan desain yang telah dirancang, maksimal waktu looping adalah 80% dari waktu sampling, yaitu 80 ms, sehingga catatan waktu yang diperoleh pada Core1 melebihi batas tersebut, sehingga dapat disimpulkan bahwa penggunaan *SingleCore* tidak mampu memenuhi kebutuhan waktu yang ditentukan.

3.2 Hasil Pengujian Waktu Sampling Dengan Menggunakan *DualCore* pada ESP32

Berdasarkan uji coba yang dilakukan untuk menjalankan seluruh *task* program dengan menggunakan *DualCore* dari ESP32, didapatkan catatan waktu sebagai berikut

TABEL 5: CATATAN WAKTU RATA – RATA SETIAP DETIK UNTUK LOOPING PADA CORE0, CORE1 DAN IRAM ESP32

Rata – rata Waktu Looping											
detik	**Core0**	**Core1**	**IRAM**	**detik**	**Core 0**	**Core1**	**IRAM**	**detik**	**Core0**	**Core1**	**IRAM**
1.	57,1 ms	75,3 ms	25,1 ms	21.	57,0 ms	75,1 ms	25,0 ms	41.	57,0 ms	75,2 ms	25,0 ms
2.	56,9 ms	75,6 ms	25,0 ms	22.	56,7 ms	75,4 ms	25,0 ms	42.	56,7 ms	75,3 ms	25,0 ms
3.	56,7 ms	75,8 ms	25,1 ms	23.	56,8 ms	75,2 ms	25,0 ms	43.	56,8 ms	75,9 ms	25,0 ms
4.	57,1 ms	75,1 ms	25,0 ms	24.	56,8 ms	75,6 ms	25,0 ms	44.	56,8 ms	75,4 ms	25,0 ms
5.	57,0 ms	76,1 ms	25,0 ms	25.	56,6 ms	75,8 ms	25,0 ms	45.	56,6 ms	75,1 ms	25,1 ms
6.	56,7 ms	75,2 ms	25,0 ms	26.	56,8 ms	75,1 ms	25,0 ms	46.	56,8 ms	75,4 ms	25,0 ms
7.	56,8 ms	75,3 ms	25,0 ms	27.	57,0 ms	75,1 ms	25,0 ms	47.	56,7 ms	75,2 ms	25,0 ms
8.	56,8 ms	75,9 ms	25,1 ms	28.	57,1 ms	75,4 ms	25,1 ms	48..	56,8 ms	75,3 ms	25,0 ms
9	56,6 ms	75,4 ms	25,0 ms	29.	56,9 ms	75,2 ms	25,0 ms	49.	56,8 ms	75,2 ms	25,0 ms
10.	56,8 ms	75,2 ms	25,0 ms	30.	56,7 ms	75,6 ms	25,0 ms	50.	56,6 ms	75,3 ms	25,0 ms
11.	57,0 ms	75,8 ms	25,0 ms	31.	57,1 ms	75,2 ms	25,0 ms	51.	56,7 ms	75,1 ms	25,0 ms
12.	56,7 ms	75,1 ms	25,0 ms	32.	57,0 ms	75,3 ms	25,0 ms	52.	56,7 ms	75,4 ms	25,0 ms
13.	56,8 ms	75,4 ms	25,0 ms	33.	56,9 ms	75,9 ms	25,1 ms	53.	56,8 ms	75,2 ms	25,0 ms
14.	57,1 ms	75,2 ms	25,0 ms	34.	56,7 ms	75,4 ms	25,0 ms	54.	56,9 ms	75,6 ms	25,0 ms
15.	56,9 ms	75,3 ms	25,0 ms	35.	56,7 ms	75,1 ms	25,0 ms	55.	56,9 ms	75,8 ms	25,0 ms
16.	56,7 ms	75,4 ms	25,1 ms	36.	56,8 ms	75,4 ms	25,0 ms	56.	56,7 ms	75,1 ms	25,1 ms
17.	56,7 ms	75,2 ms	25,0 ms	37.	56,9 ms	75,2 ms	25,0 ms	57.	56,7 ms	75,1 ms	25,0 ms
18.	56,8 ms	75,3 ms	25,0 ms	38.	56,7 ms	75,3 ms	25,1 ms	58.	56,9 ms	75,4 ms	25,0 ms
19.	56,9 ms	75,1 ms	25,0 ms	39.	56,7 ms	75,2 ms	25,0 ms	59.	56,7 ms	75,2 ms	25,0 ms
20.	56,7 ms	75,4 ms	25,1 ms	40.	56,8 ms	75,3 ms	25,0 ms	60.	56,7 ms	75,3 ms	25,0 ms

p-ISSN: 2356-0533; e-ISSN: 2355-9195

Rata – rata waktu Core0 : $56,8\ ms$
Rata – rata waktu Core1 : $75,3\ ms$
Rata – rata waktu IRAM : $25\ ms$

Berdasarkan TABEL 5, dapat dilihat bahwa waktu yang dibutuhkan untuk melakukan satu kali looping program membutuhkan rata – rata waktu maksimal sebesar 75,3 ms, dimana kondisi ini membebankan beberapa task program secara terpisah pada Core1 dan Core0, dan *IRAM-ATTR*. Dari hasil yang ditunjukkan rata – rata waktu untuk Core0 adalah 56,8 ms, untuk Core1 sebesar 75,3 ms dan IRAM-ATTR sebesar 25 ms. Sehingga berdasarkan pengujian tersebut dapat dilihat bahwa penggunaan *DualCore* ESP 32 mampu memberikan waktu sampling untuk masing – masing corenya masih kurang dari 80 ms, hal ini dapat terjadi karena Core0, Core1, dan IRAM-ATTR mampu bekerja secara parallel, sehingga seluruh proses dapat selesai dengan catatan waktu sampling yang kurang datri 80 ms.

3.3 Hasil Pengujian Suhu Chip ESP32

Berdasarkan uji coba yang dilakukan untuk dengan mengetahui berapa kenaikan suhu chip ESP32 saat menggunakan mode *DualCore*, didapatkan data sebagai berikut

TABEL 6: SUHU CHIP ESP32 SAAT DALAM MODE *DUALCORE*

Detik	Suhu	Detik	Suhu	Detik	Suhu
1.	58,7 ^oC	21.	58,7 ^oC	41.	58,7 ^oC
2.	58.5 ^oC	22.	58,8 ^oC	42.	58,8 ^oC
3.	58,6 ^oC	23.	58,8 ^oC	43.	58,8 ^oC
4.	58,9 ^oC	24.	58,7 ^oC	44.	58,7 ^oC
5.	58,5 ^oC	25.	58,7 ^oC	45.	58,7 ^oC
6.	58,6 ^oC	26.	58,8 ^oC	46.	58,8 ^oC
7.	58,6 ^oC	27.	58,8 ^oC	47.	58,8 ^oC
8.	58,9 ^oC	28.	58,7 ^oC	48.	58,9 ^oC
9.	59,1 ^oC	29.	58,8 ^oC	49.	58,9 ^oC
10.	59,1 ^oC	30.	58,8 ^oC	50.	59,0 ^oC
11.	58,9 ^oC	31.	58,9 ^oC	51.	59,0 ^oC
12.	58,9 ^oC	32.	58,8 ^oC	52.	59,1 ^oC
13.	58,8 ^oC	33.	58,7 ^oC	53.	59,0 ^oC
14.	58,9 ^oC	34.	58,8 ^oC	54.	59,0 ^oC
15.	58,9 ^oC	35.	58,8 ^oC	55.	58,9 ^oC
16.	58,8 ^oC	36.	58,9 ^oC	56.	58,8 ^oC
17.	58,7 ^oC	37.	58,8 ^oC	57.	58,7 ^oC
18.	58,8 ^oC	38.	58,7 ^oC	58.	58,8 ^oC
19.	58,8 ^oC	39.	58,7 ^oC	59.	58,8 ^oC
20.	58,7 ^oC	40.	58,7 ^oC	60.	58,7 ^oC
Rata – rata suhu chip ESP32 : 58,8 ^oC					

Berdasarkan hasil yang ditunjukkan pada TABEL 6. dapat dilihat bahwa Ketika menjalankan sistem *DualCore*, Suhu kerja dari chip ESP32 meningkat dari suhu normalnya yaitu $53,33^oC$, menjadi $58,8^oC$. Hal ini menunjukkan bahwa kenaikan suhu kerja chip ESP32 walaupun digunakan untuk menggunakan sistem *DualCore* tidak membebani kinerja dari chip tersebut, karena perubahan suhu kerja pada chip tidak berbeda jauh, hanya sebesar $5,5^oC$. Hal ini menunjukkan bahwa penggunaan sistem *DualCore* dapat digunakan sebagai solusi yang optimal.

4. KESIMPULAN

Berdasarkan hasil penelitian yang dilakukan dapat disimpulkan bahwa ESP32 dapat dirancang untuk menjadi kontroler yang optimal bagi sistem UGV, hal ini didasari dari pengujian yang telah dilakukan dengan mencoba kemampuan waktu sampling mikrokontroler ESP32 ketika menggunakan sistem *SingleCore* dan *DualCore*. Terbukti dari hasil percobaan, ESP32 yang beroperasi didalam mode DualCore mampu bekerja lebih optimal dibandingkan dengan operasi mode SingleCore, dimana catatan rata - rata waktu maksimal yang dibutuhkan untuk melakukan

satu kali looping sistem adalah 75,3 ms, sedangkan untuk mode *SingleCore* memiliki rata – rata waktu maksimal adalah 88,9 ms. Hal ini menunjukkan bahwa dalam mode *DualCore,* ESP32 mampu memenuhi kriteria penyelesaian looping program yang tidak boleh lebih dari 80% dari time sampling yang dibutuhkan sistem, dimana time sampling yang digunakan adalah 100 ms, sehingga batas maksimal waktu looping program adalah 80 ms. Serta berdasarkan pengujian dapat disimpulkan juga bahwa mode *DualCore* tidak membebani Chip ESP32 secara signifikan yang ditunjukkan dari suhu kerja saat mode DualCore adalah $58{,}8^oC$ meningkat $5{,}5^oC$ dibandingkan saat beroperasi normal yaitu $53{,}3^oC$. Sehingga ESP32 yang beroperasi dalam mode *DualCore,* dapat digunakan sebagai kontroler yang memiliki tingkat presisi pengaturan waktu yang baik, dimana semakin akurat waktu sampling yang dapat dijalankan akan semakin stabil peforma sistem control yang dimiliki UGV.

5. UCAPAN TERIMAKASIH

Ucapan terima kasih terutama ditujukan kepada Politeknik Negeri Malang telah memberikan sarana dan prasarana laboratorium sehingga penelitian dapat terlaksana dengan baik, serta telah memberikan bantuan dana penelitian melalui Hibah P2M Polinema. Ucapan terima kasih dapat juga disampaikan kepada pihak-pihak yang membantu pelaksanaan penelitian sehingga penelitian dapat selesai dengan tepat waktu.

DAFTAR PUSTAKA

[1] A. Elmquist and D. Negrut, "Methods and Models for Simulating Autonomous Vehicle Sensors," *IEEE Transactions on Intelligent Vehicles,* vol. 5, pp. 684-692, 2020.

[2] T. Shu, S. Gharaaty, W. Xie, A. Joubair and I. A. Bonev, "Dynamic Path Tracking of Industrial Robots With High Accuracy Using Photogrammetry Sensor," *IEEE/ASME Transactions on Mechatronics,* vol. 23, pp. 1159-1170, 2018.

[3] G. A. Azhar, T. Winarno and A. Komarudin, "Kontrol Sudut Elevasi Robot Pelontar Softsaucer dengan Metode PID," *Jurnal Elektronika Otomasi Industri,* vol. 4, pp. 9-14, 2020.

[4] I. Siradjuddin, G. Azhar, S. Wibowo, F. Ronilaya, C. Rahmad and R. O. H. A. D. I. Erfan, "A General Inverse Kinematic Formulation and Control Schemes for Omnidirectional Robots," *Engineering Letters,* vol. 29, 1 2022.

[5] M. Sorour, A. Cherubini, P. Fraisse and R. Passama, "Motion Discontinuity-Robust Controller for Steerable Mobile Robots," *IEEE Robotics and Automation Letters,* vol. 2, pp. 452-459, 2017.

[6] I. Siradjuddin, G. A. Azhar, A. Murdani and M. L. M. Faizin, "Desain dan pemodelan kontrol kinematik pergerakan robot beroda dengan menggunakan 6 roda omni-wheels," *Jurnal Eltek,* vol. 18, pp. 116-127, 2020.

[7] I. Siradjuddin, L. Kamajaya, S. Wibowo, A. A. Rofiq, G. A. Azhar and M. Khairuddin, "Near Real Time Simulation of an Independent Steering Independent Driving Mobile Robot," in *2021 3rd International Conference on Control Systems, Mathematical Modeling, Automation and Energy Efficiency (SUMMA)*, 2021.

[8] J. Li, M. Ran and L. Xie, "Efficient Trajectory Planning for Multiple Non-Holonomic Mobile Robots via Prioritized Trajectory Optimization," *IEEE Robotics and Automation Letters,* vol. 6, pp. 405-412, 2021.

[9] Y. Maddahi and K. Zareinia, "Nonparametric Bootstrap Technique to Improve Positional Accuracy in Mobile Robots With Differential Drive Mechanism," *IEEE Access,* vol. 8, pp. 158502-158511, 2020.

[10] G. A. Azhar, T. Winarno and S. Izza, "Implementasi g-h Filter Pada Sensor Kompas Sebagai Peningkatan Akurasi Trajectory Tracking Robot Differential Drive," *Journal of Mechanical and Electrical Technology,* vol. 1, 2022.

[11] Y. Shi, M. R. Elara, A. V. Le, V. Prabakaran and K. L. Wood, "Path Tracking Control of Self-Reconfigurable Robot hTetro With Four Differential Drive Units," *IEEE Robotics and Automation Letters,* vol. 5, pp. 3998-4005, 2020.

[12] R. Kumar, S. Singh and V. K. Chaurasiya, "A Low-Cost and Efficient Spatial–Temporal Model for Indoor Localization "H-LSTMF"," *IEEE Sensors Journal,* vol. 23, pp. 6117-6128, 2023.

[13] G. Fabregat, J. A. Belloch, J. M. Badía and M. Cobos, "Design and Implementation of Acoustic Source Localization on a Low-Cost IoT Edge Platform," *IEEE Transactions on Circuits and Systems II: Express Briefs,* vol. 67, pp. 3547-3551, 2020.

[14] V. Barral Vales, O. C. Fernández, T. Domínguez-Bolaño, C. J. Escudero and J. A. García-Naya, "Fine Time Measurement for the Internet of Things: A Practical Approach Using ESP32," *IEEE Internet of Things Journal,* vol. 9, pp. 18305-18318, 2022.

[15] S. Sungkono, G. A. Azhar, A. C. Kusuma and S. Izza, "Differential Drive Mobile Robot Motion Accuracy Improvement with Odometry-Compass Sensor Fusion Implementation," *ELKHA:Jurnal Teknik Elektro,* vol. 15, 2023.

[16] G. A. Azhar, T. Winarno and S. Izza, "Sistem Distribusi Data Kontrol Pada Differential Drive Mobile Robot Menggunakan Robot Operating System," *Journal of Mechanical and Electrical Technology,* vol. 1, 2022.

[17] B. Kim and K. Yi, "Probabilistic and Holistic Prediction of Vehicle States Using Sensor Fusion for Application to Integrated Vehicle Safety Systems," *IEEE Transactions on Intelligent Transportation Systems,* vol. 15, pp. 2178-2190, 2014.

DOI: http://dx.doi.org/10.33795/elkolind.v10i2.3720