DAFTAR PUSTAKA

Anggoro, B. (2013). Desain Pemodelan Kinematik dan Dinamik Humanoid Robot. Skripsi. Univesitas Diponegoro, Bandung.

Arifin, H. Z. D., Maulana, R., & Fitriyah, H. (2022). Implementasi Sistem Robot Otonom dengan Sensor Kinect menggunakan Algoritme Gmapping dan Timed Elastic Band. Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer, 6(8), 3570–3577. Diambil dari https://j-ptiik.ub.ac.id/index.php/j-ptiik/article/view/11393

America, N. (2019). Executive Summary World Robotics 2019 Industrial Robots," pp. 13–16.

Brock, O and O. Khatib. (1999). High-speed navigation using the global dynamic window approach," Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on, 1999, pp. 341-346 vol.1.

Borenstein, J and Y. Koren, "Real-time obstacle avoidance for fast mobile robots in cluttered environments," vol. 19, no. 5, pp. 572–577, 1990.

Corke., P. (2017). Robotics, Vision and Control: Fundamental Algorithms In MATLAB® Second, Completely Revised. Springer.

Dieter, F., Wolfram, F., and Sebastian, T. (1997). The Dynamic WindowApproachto Collision Avoidance. pp. 137–146.

Dudek, G. and Jenkin, M. (2010). Computational Principles of Mobile Robotics, 2nd ed. New York: Cambridge University Press.

E. Dwiky, E., Endang, D. Rosalia, H., S. Sunarto, T. S, Kuat, G. Ferrianto. (2019). Sistem Navigasi Mobile Robot Dalam Ruangan Berbasis Autonomous Navigation. Journal of Mechanical Engineering and Mechatronics, pp 78-86.

Gatesichapakorn,S., Takamatsu, J., Ruchanuruck, S. (2019). ROS based Autonomous Mobile Robot Navigation using 2D LiDAR and RGB-D Camera. Conference: 2019 First International Symposium on Instrumentation, Control, Artificial Intelligence, and Robotics (ICA-SYMP). Diambil dari https://ieeexplore.ieee.org/document/8645984/.

.V.N., Prasthech, K., Laksmhi, K.R.V., Kadam, S.S., Bailey, K. (2021). tonomous Navigation in Dynamic Environment. International Research

Journal of Engineering and Technology (IRJET). Vol. 08 (6). BMS College of Engineering, Karnataka, India.

Ilham (2019). Navigasi Mobile Robot Darat Menggunakan Odometri Visual Berbasis Citra Stereo. Tesis. Institut Teknologi Sepuluh Nopember Surabaya

Jianwei, Z., Shengyi, Jinyu., L. (2022). Research and Implementation of Autonomous Navigation for Mobile Robots Based on SLAM Algorithm under ROS.

Joseph, Lentin. (2018). Robot Operating System for Absolute Beginners. Apress. India.

Litman., T. (2019). Autonomous Vehicle Implementation Predictions: Implications for Transport Planning,Transp. Res. Board Annu. Meet., Vol. 42, no. January 2014, pp. 36–42, 2019.

Minguez, J and L. Montano. (2000). Nearness Diagram Navigation (ND): A New Real Time Collision Avoidance Approach. In Proc. of the ieee/rsj international conference on intelligent robots and systems (iros'00, vol. 60, 2000, pp. 2094-- 2100.

Muhammad, F., Wicaksana, C. A. (2021). Literature Review Sistem Navigasi Autonomous Mobile Robot Berbasis ROS (Robot Operating System). Jurnal Ilmiah Setrum. Vol.10 No.1 (2021) 103-112.

Meng, Z. Sun, H., Qin, Z., Chen, C. Zhou, and M. H. (2018). Intelligent Robotic System for Autonomous Exploration and Active SLAM in Unknown Environments. SII 2017 - 2017 IEEE/SICE Int. Symp. Syst. Integr., Vol. 2018-Janua, pp. 651–656.

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

Morales, L., Herrera, M., Camacho, O., Leica, P. and J. Aguilar. (2021). LAMDA Control Approaches Applied to Trajectory Tracking for Mobile Robots," IEEE Access, vol. 9, pp. 37179-37195.

, C., Feiten, W. Wösch., Hoffmann, F., and T. Bertram. (2012). ajectory Modification Considering Dynamic Constraints of Autonomous ɔots," 7th Ger. Conf. Robot. Robot. 2012, pp. 74–79.

Siegwart,R and Nourbakhsh., I., R. (2004). Introduction to Autonomous Mobile Robots. USA: Bradford Company.

Sun, Y, Guan, L., Chang, Z. et al., (2019). Design of a low-cost indoor navigation system for food delivery robot based on multi-sensor information fusion," Sensors, vol. 19, no. 22, 4980 pages.

Ulrich, I and J. Borenstein. (2000). VFH: local obstacle avoidance with look aheadverification, Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on, 2000.

Takbir, M., Anshar. M., Dewiani. (2019). Sistem Navigasi pada Prototipe Robot Kursi Beroda untuk Penyandang Disabilitas. Skripsi. Departemen Elektro Universitas Hasanuddin, Makassar.

Wang, C. and Du, D. (2016). Research on Logistics Autonomous Mobile Robot System. IEEE Int. Conf. Mechatronics Autom. IEEE ICMA 2016, pp. 275–280.

Wannacot, D., dkk.. (2012). Autonomous Navigation Planning with Ros. Journal.Michigan Technological University, USA.

Y. Pyo, H. Cho, L. J. Jung, and D. Lim. (2017). ROS Robot Programming (English). ROBOTIS. [Online]. Available: http:// community.robotsource.org/t/download-the-ros-robot-programming-book-for-free/ 51.

LAMPIRAN

Lampiran 1 Data Sampling Perhitungan Kecepatan Robot

a. Data sampling kecepatan robot pada skenario 1 Lingkungan Dinamis

Time	*Linear Speed (m/s)*	*Angular Speed (rad/s)*
1	0,219999999	-0,057435896
2	0,219999999	-0,073846154
3	0,219999999	-0,073846154
4	0,219999999	-0,073846154
5	0,219999999	-0,073846154
6	0,219999999	-0,057435896
7	0,219999999	-0,057435896
8	0,219999999	-0,057435896
9	0,219999999	-0,024615385
10	0,219999999	-0,008205128
11	0,219999999	-0,008205128
12	0,196842104	0,008205128
13	0,219999999	0,254358977
14	0,219999999	0,254358977
15	0,219999999	0,270769238
16	0,196842104	0,303589731
17	0,196842104	0,28717947
18	0,219999999	0,237948716
19	0,219999999	0,172307685
20	0,219999999	0,123076923
21	0,196842104	0,106666662
22	0,196842104	0,057435896
23	0,196842104	0
24	0,17368421	-0,073846154
25	0,17368421	-0,073846154
26	0,196842104	-0,139487177
27	0,17368421	-0,155897439
28	0,17368421	-0,172307685
29	0,196842104	-0,188717946
30	0,17368421	-0,139487177
31	0,17368421	-0,139487177
32	0,17368421	-0,139487177
33	0,17368421	-0,123076923
34	0,17368421	-0,106666662
35	0,17368421	-0,090256408
36	0,17368421	-0,090256408
37	0,17368421	-0,041025639
38	0,17368421	-0,041025639

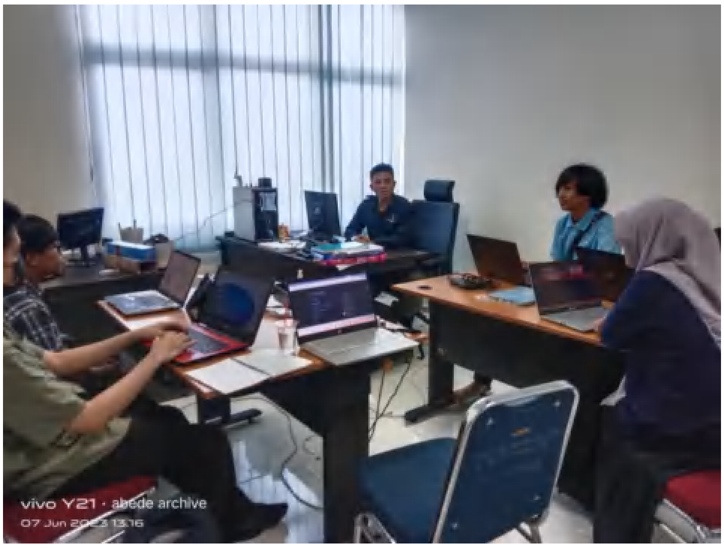
39	0,17368421	0,024615385
40	0,17368421	0,041025639
41	0,17368421	0,024615385
42	0,17368421	0,024615385
43	0,17368421	0,008205128
44	0,17368421	0,008205128
45	0,17368421	-0,057435896
46	0,17368421	-0,057435896
47	0,17368421	-0,057435896
48	0,150526315	-0,041025639
49	0,150526315	-0,057435896
50	0,150526315	0,024615385
51	0,150526315	0,008205128
52	0,150526315	0,008205128
53	0,150526315	-0,090256408
54	0,150526315	-0,106666662
55	0,150526315	-0,106666662
56	0,150526315	-0,106666662
57	0,150526315	0
58	0,150526315	0
59	0,150526315	0,008205128
60	0,150526315	0,024615385
61	0,150526315	0,024615385
62	0,12736842	0,041025639
63	0,150526315	-0,057435896
64	0,150526315	-0,008205128
65	0,150526315	-0,008205128
66	0,150526315	-0,008205128
67	0,150526315	-0,057435896
68	0,150526315	-0,008205128
69	0,150526315	-0,008205128
70	0,12736842	-0,057435896
71	0,12736842	0,024615385
72	0,12736842	0,024615385
73	0,12736842	0,024615385
74	0,12736842	0,024615385
75	0,12736842	0,024615385
76	0,12736842	-0,090256408
77	0,12736842	-0,106666662
78	0,12736842	-0,106666662
79	0,12736842	-0,106666662
80	0,12736842	-0,106666662
81	0,12736842	0
82	0,12736842	0,008205128

83	0,12736842	0,008205128
84	0,12736842	0,024615385
85	0,12736842	0,024615385
86	0,12736842	0
87	0,12736842	-0,106666662
88	0,12736842	-0,008205128
89	0,12736842	-0,008205128
90	0,12736842	-0,008205128
91	0,12736842	-0,008205128
92	0,12736842	-0,106666662
93	0,12736842	-0,106666662
94	0,12736842	-0,123076923
95	0,12736842	-0,008205128
96	0,12736842	-0,008205128
97	0,12736842	0
98	0,12736842	0,008205128
99	0,12736842	0
100	0,12736842	0

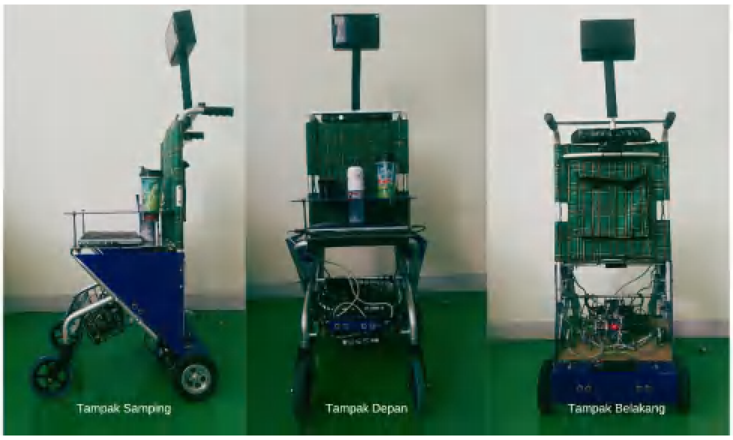
Lampiran 2 Dokumentasi Pelaksanaan Penelitian

a. Tahap Persiapan

Riset & studi literatur

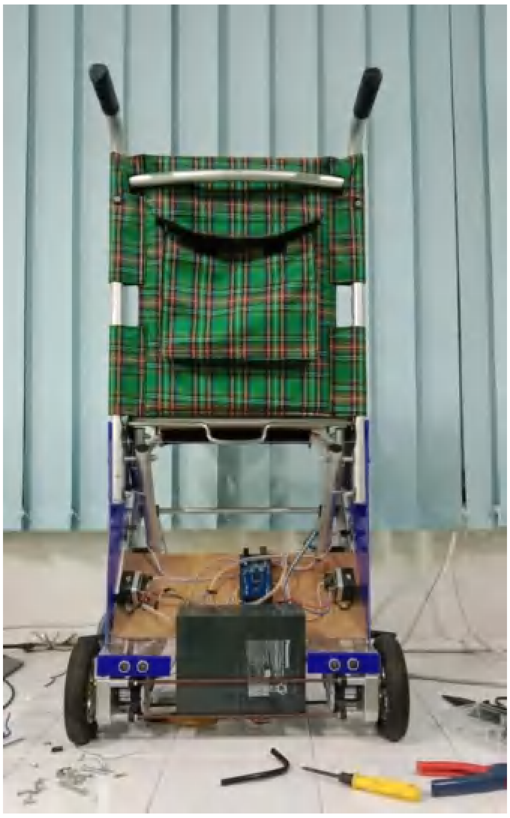


Diskusi dan konsultasi



Rancang Bangun Robot

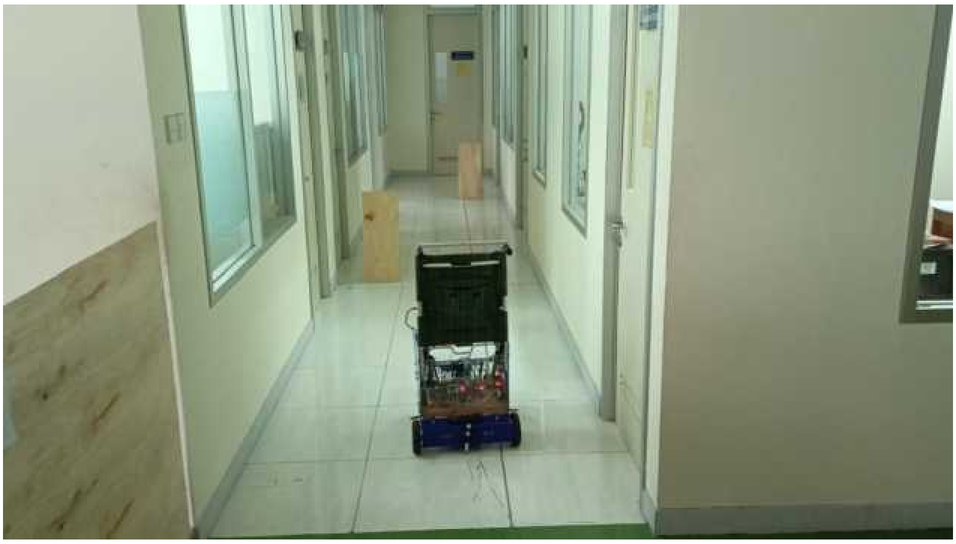
b. Tahap Perancangan



Perancangan Awal: *Redesign* Kursi Roda Menjadi *Waiter-Bot*

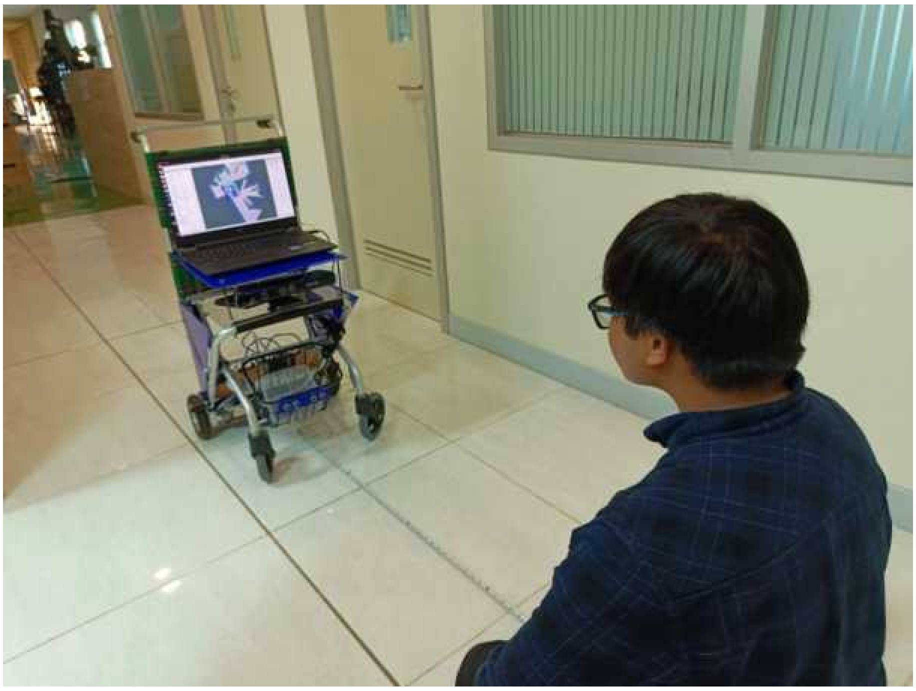
Instalasi perangkat keras *Waiter-Bot*

c. Tahap Pengujian

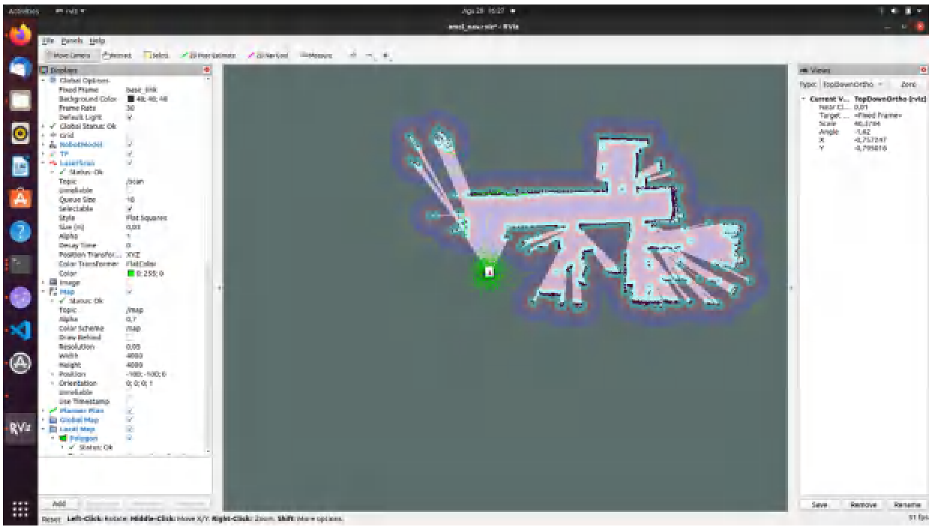


Pengujian Robot mengenali objek penghalang

Pengujian robot melakukan pivot dalam menghindari *obstacle*

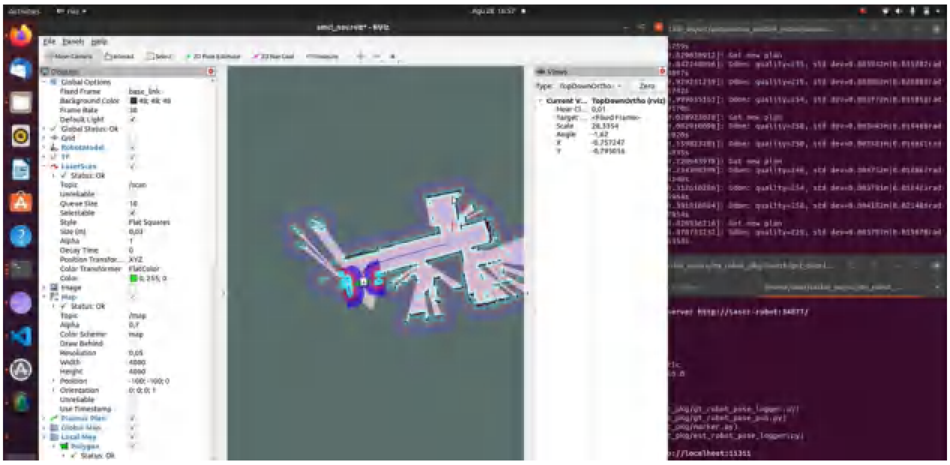


Pengukuran Jarak antara robot dan objek penghalang



Tampilan peta pengujian *Waiter-Bot* dalam RViz

Jejak *Waiter-Bot* dalam pengamatan trayektori perencanaan jalur



Waiter-Bot Sedang dalam melakukan navigasi menuju misi

Video Pengujian Sistem Navigasi *Waiter-Bot*

Lampiran 3 Kode program *Waiter-Bot*

Program Arduino *(Low Level)*

```
// Variabel untuk menyimpan data dari Python
int pwm_left = 0;
int pwm_right = 0;
int direction_left = 0;
int direction_right = 0;

// Pin untuk mengontrol driver motor
#include <Servo.h>  //  You need to include Servo.h as it is used
by the HB-25 Library
#include <HB25MotorControl.h>

const byte controlPinL = 6;
const byte controlPinR = 7;
HB25MotorControl motorControlL(controlPinL);
HB25MotorControl motorControlR(controlPinR);

void setup() {
  motorControlL.begin();
  motorControlR.begin();

  // Inisialisasi Serial
  Serial.begin(57600);
}

void loop() {
  // Jika ada data yang diterima dari Python
  if (Serial.available()) {

    // Membaca data dari Serial
    String input = Serial.readStringUntil('\n');
    input.trim();

      Memisahkan data menjadi PWM kiri, arah kiri, PWM kanan, dan
      an
      separator1 = input.indexOf(',');
      separator2 = input.indexOf(',', separator1 + 1);
```

```
    int separator3 = input.indexOf(',', separator2 + 1);

    pwm_left = input.substring(0, separator1).toInt();
    direction_left    =    input.substring(separator1    +    1,
separator2).toInt();
    pwm_right    =    input.substring(separator2    +    1,
separator3).toInt();
    direction_right = input.substring(separator3 + 1).toInt();


    // Mengontrol motor
    motorControlL.moveAtSpeed(pwm_left);
    motorControlR.moveAtSpeed(pwm_right);
  }
}
```

Ros_serial_encoder

```
/*
 * Author: Automatic Addison
 * Website: https://automaticaddison.com
 * Description: ROS node that publishes the accumulated ticks for
each wheel
 * (right_ticks and left_ticks topics) using the built-in encoder
 * (forward = positive; reverse = negative)
 */


#include <ros.h>
#include <std_msgs/Int16.h>

// Handles startup and shutdown of ROS
ros::NodeHandle nh;

// Encoder output to Arduino Interrupt pin. Tracks the tick count.
#define ENC_IN_LEFT_A 2
        ENC_IN_RIGHT_A 19

        : encoder output to Arduino to keep track of wheel direction
        ks the direction of rotation.
```

```
#define ENC_IN_LEFT_B 3
#define ENC_IN_RIGHT_B 18

// True = Forward; False = Reverse
boolean Direction_left = true;
boolean Direction_right = true;

// Minumum and maximum values for 16-bit integers
const int encoder_minimum = -32768;
const int encoder_maximum = 32767;

// Keep track of the number of wheel ticks
std_msgs::Int16 right_wheel_tick_count;
ros::Publisher rightPub("right_ticks", &right_wheel_tick_count);

std_msgs::Int16 left_wheel_tick_count;
ros::Publisher leftPub("left_ticks", &left_wheel_tick_count);

// 100ms interval for measurements
const int interval = 100;
long previousMillis = 0;
long currentMillis = 0;

// Increment the number of ticks
void right_wheel_tick() {

  // Read the value for the encoder for the right wheel
  int val = digitalRead(ENC_IN_RIGHT_B);

  if(val == LOW) {
    Direction_right = false; // Reverse
  }
  else {
    Direction_right = true; // Forward
  }

      irection_right) {

      (right_wheel_tick_count.data == encoder_maximum) {
```

```
      right_wheel_tick_count.data = encoder_minimum;
    }
    else {
      right_wheel_tick_count.data++;
    }
  }
  else {
    if (right_wheel_tick_count.data == encoder_minimum) {
      right_wheel_tick_count.data = encoder_maximum;
    }
    else {
      right_wheel_tick_count.data--;
    }
  }
}

// Increment the number of ticks
void left_wheel_tick() {

  // Read the value for the encoder for the left wheel
  int val = digitalRead(ENC_IN_LEFT_B);

  if(val == LOW) {
    Direction_left = true; // Reverse
  }
  else {
    Direction_left = false; // Forward
  }

  if (Direction_left) {
    if (left_wheel_tick_count.data == encoder_maximum) {
      left_wheel_tick_count.data = encoder_minimum;
    }
    else {
      left_wheel_tick_count.data++;


       {
       (left_wheel_tick_count.data == encoder_minimum) {
```

```
      left_wheel_tick_count.data = encoder_maximum;
    }
    else {
      left_wheel_tick_count.data--;
    }
  }
}

void setup() {

  // Set pin states of the encoder
  pinMode(ENC_IN_LEFT_A , INPUT_PULLUP);
  pinMode(ENC_IN_LEFT_B , INPUT);
  pinMode(ENC_IN_RIGHT_A , INPUT_PULLUP);
  pinMode(ENC_IN_RIGHT_B , INPUT);

  // Every time the pin goes high, this is a tick
  attachInterrupt(digitalPinToInterrupt(ENC_IN_LEFT_A),
left_wheel_tick, RISING);
  attachInterrupt(digitalPinToInterrupt(ENC_IN_RIGHT_A),
right_wheel_tick, RISING);

  // ROS Setup
  nh.getHardware()->setBaud(115200);
  nh.initNode();
  nh.advertise(rightPub);
  nh.advertise(leftPub);
}

void loop() {

  // Record the time
  currentMillis = millis();

  // If 100ms have passed, print the number of ticks
       urrentMillis - previousMillis > interval) {

       viousMillis = currentMillis;
```

```
    rightPub.publish( &right_wheel_tick_count );
    leftPub.publish( &left_wheel_tick_count );
    nh.spinOnce();
  }
}
```

ROS NODE: *serial_encoder_imu*

```
/*
 * Author: Automatic Addison
 * Website: https://automaticaddison.com
 * Description: ROS node that publishes the accumulated ticks for
each wheel
 * (right_ticks and left_ticks topics) using the built-in encoder
 * (forward = positive; reverse = negative)
 */

#include "MPU9250.h"
#include <ros.h>
#include <std_msgs/Int16.h>
#include <sensor_msgs/Imu.h>

// Handles startup and shutdown of ROS
ros::NodeHandle nh;

// Encoder output to Arduino Interrupt pin. Tracks the tick count.
#define ENC_IN_LEFT_A 2
#define ENC_IN_RIGHT_A 19

// Other encoder output to Arduino to keep track of wheel direction
// Tracks the direction of rotation.
#define ENC_IN_LEFT_B 3
#define ENC_IN_RIGHT_B 18

// True = Forward; False = Reverse
boolean Direction_left = true;
        Direction_right = true;

        num and maximum values for 16-bit integers
        nt encoder_minimum = -32768;
```

```
const int encoder_maximum = 32767;

// Keep track of the number of wheel ticks
std_msgs::Int16 right_wheel_tick_count;
ros::Publisher rightPub("right_ticks", &right_wheel_tick_count);

std_msgs::Int16 left_wheel_tick_count;
ros::Publisher leftPub("left_ticks", &left_wheel_tick_count);

// Keep track of mpu9250 imu sensor
sensor_msgs::Imu imu_msg;
ros::Publisher imu_pub("imu_data", &imu_msg);
MPU9250 imu;

// 100ms interval for measurements
const int interval = 100;
long previousMillis = 0;
long currentMillis = 0;

//imu sensor reading
void print_roll_pitch_yaw() {
  // Baca yaw, pitch, dan roll dari MPU9250
  float yaw = imu.getYaw();     // Mendapatkan nilai yaw menggunakan fungsi getYaw()
  float pitch = imu.getPitch();    // Mendapatkan nilai pitch menggunakan fungsi getPitch()
  float roll = imu.getRoll();       // Mendapatkan nilai roll menggunakan fungsi getRoll()

  // Baca kecepatan sudut (angular velocity) dari MPU9250
  float angular_velocity_x = imu.getGyroX();  // Mendapatkan nilai kecepatan sudut pada sumbu x menggunakan fungsi getGyroX()
  float angular_velocity_y = imu.getGyroY();  // Mendapatkan nilai kecepatan sudut pada sumbu y menggunakan fungsi getGyroY()
  float angular_velocity_z = imu.getGyroZ();  // Mendapatkan nilai
       an sudut pada sumbu z menggunakan fungsi getGyroZ()

       ca percepatan linier (linear acceleration) dari MPU9250
```

```
  float   linear_acceleration_x   =   imu.getLinearAccX();     //
Mendapatkan nilai percepatan linier pada sumbu x menggunakan fungsi
getLinearAccX()
  float   linear_acceleration_y   =   imu.getLinearAccY();     //
Mendapatkan nilai percepatan linier pada sumbu y menggunakan fungsi
getLinearAccY()
  float   linear_acceleration_z   =   imu.getLinearAccZ();     //
Mendapatkan nilai percepatan linier pada sumbu z menggunakan fungsi
getLinearAccZ()

  // Isi data IMU
  imu_msg.header.stamp = nh.now();
  imu_msg.header.frame_id = "imu_link";
  // Konversi Euler angles menjadi quaternion
  float cy = cos(yaw * 0.5);
  float sy = sin(yaw * 0.5);
  float cp = cos(pitch * 0.5);
  float sp = sin(pitch * 0.5);
  float cr = cos(roll * 0.5);
  float sr = sin(roll * 0.5);

  float qw = cy * cp * cr + sy * sp * sr;
  float qx = cy * cp * sr - sy * sp * cr;
  float qy = sy * cp * sr + cy * sp * cr;
  float qz = sy * cp * cr - cy * sp * sr;

  // Isi data quaternion ke pesan IMU
  imu_msg.orientation.x = qx;
  imu_msg.orientation.y = qy;
  imu_msg.orientation.z = qz;
  imu_msg.orientation.w = qw;

  imu_msg.angular_velocity.x = angular_velocity_x;
  imu_msg.angular_velocity.y = angular_velocity_y;
  imu_msg.angular_velocity.z = angular_velocity_z;

       sg.linear_acceleration.x = linear_acceleration_x;
       sg.linear_acceleration.y = linear_acceleration_y;
       sg.linear_acceleration.z = linear_acceleration_z;
```

```
  // Serial.print("Yaw, Pitch, Roll: ");
  // Serial.print(yaw, 2);
  // Serial.print(", ");
  // Serial.print(pitch, 2);
  // Serial.print(", ");
  // Serial.println(roll, 2);
}

// Increment the number of ticks
void right_wheel_tick() {

  // Read the value for the encoder for the right wheel
  int val = digitalRead(ENC_IN_RIGHT_B);

  if (val == LOW) {
    Direction_right = false;  // Reverse
  } else {
    Direction_right = true;  // Forward
  }

  if (Direction_right) {

    if (right_wheel_tick_count.data == encoder_maximum) {
      right_wheel_tick_count.data = encoder_minimum;
    } else {
      right_wheel_tick_count.data++;
    }
  } else {
    if (right_wheel_tick_count.data == encoder_minimum) {
      right_wheel_tick_count.data = encoder_maximum;
    } else {
      right_wheel_tick_count.data--;
    }
  }
}

       ement the number of ticks
       ft_wheel_tick() {
```

```
  // Read the value for the encoder for the left wheel
  int val = digitalRead(ENC_IN_LEFT_B);

  if (val == LOW) {
    Direction_left = true;  // Reverse
  } else {
    Direction_left = false;  // Forward
  }

  if (Direction_left) {
    if (left_wheel_tick_count.data == encoder_maximum) {
      left_wheel_tick_count.data = encoder_minimum;
    } else {
      left_wheel_tick_count.data++;
    }
  } else {
    if (left_wheel_tick_count.data == encoder_minimum) {
      left_wheel_tick_count.data = encoder_maximum;
    } else {
      left_wheel_tick_count.data--;
    }
  }
}

void setup() {
  //set i2c connection from mpu9250
  Wire.begin();
  delay(2000);
  if (!imu.setup(0x68)) {  // change to your own address
    while (1) {
      Serial.println("MPU connection failed. Please check your connection with `connection_check` example.");
      delay(5000);
    }
  }

      pin states of the encoder
      de(ENC_IN_LEFT_A, INPUT_PULLUP);
      de(ENC_IN_LEFT_B, INPUT);
```

```
  pinMode(ENC_IN_RIGHT_A, INPUT_PULLUP);
  pinMode(ENC_IN_RIGHT_B, INPUT);

  // Every time the pin goes high, this is a tick
  attachInterrupt(digitalPinToInterrupt(ENC_IN_LEFT_A),
left_wheel_tick, RISING);
  attachInterrupt(digitalPinToInterrupt(ENC_IN_RIGHT_A),
right_wheel_tick, RISING);

  // ROS Setup
  nh.getHardware()->setBaud(115200);
  nh.initNode();
  nh.advertise(rightPub);
  nh.advertise(leftPub);
  nh.advertise(imu_pub);
}
void loop() {
  //update mpu value if there any change
  if (imu.update()) {
    static uint32_t prev_ms = millis();
    if (millis() > prev_ms + 25) {
      print_roll_pitch_yaw();
      prev_ms = millis();
    }
  }
  // Record the time
  currentMillis = millis();

  // If 100ms have passed, print the number of ticks
  if (currentMillis - previousMillis > interval) {

    previousMillis = currentMillis;

    rightPub.publish(&right_wheel_tick_count);
    leftPub.publish(&left_wheel_tick_count);
       _pub.publish(&imu_msg);
       spinOnce();
```

ROS NODE: *serial_imu*

```
#include "MPU9250.h"
#include <ros.h>
#include <sensor_msgs/Imu.h>
ros::NodeHandle nh;
sensor_msgs::Imu imu_msg;
ros::Publisher imu_pub("imu_data", &imu_msg);

MPU9250 mpu;

void setup() {
  nh.initNode();
  nh.advertise(imu_pub);
  Serial.begin(115200);
  Wire.begin();
  delay(2000);

  if (!mpu.setup(0x68)) {  // change to your own address
    while (1) {
      Serial.println("MPU connection failed. Please check your
connection with `connection_check` example.");
      delay(5000);
    }
  }
}

void loop() {
  if (mpu.update()) {
    static uint32_t prev_ms = millis();
    if (millis() > prev_ms + 25) {
      print_roll_pitch_yaw();
      imu_pub.publish(&imu_msg);
      nh.spinOnce();
      prev_ms = millis();
    }



       int_roll_pitch_yaw() {
```

```
  float yaw = mpu.getYaw();
  float pitch = mpu.getPitch();
  float roll = mpu.getRoll();
  imu_msg.header.stamp = nh.now();
  imu_msg.orientation.x = roll;
  imu_msg.orientation.y = pitch;
  imu_msg.orientation.z = yaw;
  Serial.print("Yaw, Pitch, Roll: ");
  Serial.print(yaw, 2);
  Serial.print(", ");
  Serial.print(pitch, 2);
  Serial.print(", ");
  Serial.println(roll, 2);
}
```

Kode Program Navigasi Dalam ROS *(High Level)*

Konfigurasi Node Perangkat Keras Robot

```
#!/usr/bin/env python3

import rospy
from sensor_msgs.msg import JointState
from std_msgs.msg import Float64
import serial

class ROBOTHardwareInterface:
    def __init__(self):
        self.joint_names = ['left_wheel_joint',
'right_wheel_joint']
        self.joint_positions = [0.0, 0.0]
        self.joint_velocities = [0.0, 0.0]
        self.joint_efforts = [0.0, 0.0]
        self.joint_velocity_commands = [0.0, 0.0]
        self.left_motor_pos = 0
        self.right_motor_pos = 0
        self.left_prev_cmd = 0
        self.right_prev_cmd = 0
        self.serial = serial.Serial('/dev/ttyUSB0', 9600,
=0.1)
```

```
    def init(self):
        rospy.init_node('robot_hardware_interface')

        self.joint_state_pub  =  rospy.Publisher('joint_states',
JointState, queue_size=10)
        self.left_velocity_sub                                    =
rospy.Subscriber('left_wheel_joint_velocity_controller/command',
Float64, self.left_velocity_callback)
        self.right_velocity_sub                                   =
rospy.Subscriber('right_wheel_joint_velocity_controller/command',
Float64, self.right_velocity_callback)

    def left_velocity_callback(self, msg):
        self.joint_velocity_commands[0] = msg.data

    def right_velocity_callback(self, msg):
        self.joint_velocity_commands[1] = msg.data

    def read(self):
        data = self.serial.readline().decode('utf-8').strip()
        if data:
            left_encoder_value,  right_encoder_value  =  map(int,
data.split(','))
            self.joint_position[0] = left_encoder_value
            self.joint_position[1] = right_encoder_value
        else:
            print("Invalid data received from Arduino")



    def write(self):
        left_motor_velocity = int(self.joint_velocity_commands[0])
        right_motor_velocity                                      =
int(self.joint_velocity_commands[1])

        if   self.left_prev_cmd   !=   left_motor_velocity   or
ght_prev_cmd != right_motor_velocity:
            command                                               =
_motor_velocity},{right_motor_velocity}\n'
            self.serial.write(command.encode())
```

```
            self.left_prev_cmd = left_motor_velocity
            self.right_prev_cmd = right_motor_velocity

    def update(self):
        joint_state = JointState()
        joint_state.header.stamp = rospy.Time.now()
        joint_state.name = self.joint_names
        joint_state.position = self.joint_positions
        joint_state.velocity = self.joint_velocities
        joint_state.effort = self.joint_efforts

        self.joint_state_pub.publish(joint_state)

        self.read()
        self.write()

    def run(self):
        rate = rospy.Rate(10)  # 10 Hz

        while not rospy.is_shutdown():
            self.update()
            rate.sleep()

if __name__ == '__main__':
    robot_hw = ROBOTHardwareInterface()
    robot_hw.init()
    robot_hw.run()
```

Move Base Package

```
<launch>
  <!-- Arguments -->
  <arg name="cmd_vel_topic" default="/cmd_vel" />
  <arg name="odom_topic" default="odom" />
  <arg name="move_forward_only" default="false"/>

      nove_base -->
          pkg="move_base"    type="move_base"    respawn="false"
      ove_base" output="screen">
```

```
    <param name="base_local_planner"
value="dwa_local_planner/DWAPlannerROS" />
    <rosparam file="$(find
autonomus_mobile_robot)/param/costmap_common_params.yaml"
command="load" ns="global_costmap" />
    <rosparam file="$(find
autonomus_mobile_robot)/param/costmap_common_params.yaml"
command="load" ns="local_costmap" />
    <rosparam file="$(find
autonomus_mobile_robot)/param/local_costmap_params.yaml"
command="load" />
    <rosparam file="$(find
autonomus_mobile_robot)/param/global_costmap_params.yaml"
command="load" />
    <rosparam file="$(find
autonomus_mobile_robot)/param/move_base_params.yaml"
command="load" />
    <rosparam file="$(find
autonomus_mobile_robot)/param/dwa_local_planner_params.yaml"
command="load" />
    <remap from="cmd_vel" to="$(arg cmd_vel_topic)"/>
    <remap from="odom" to="$(arg odom_topic)"/>
    <param name="DWAPlannerROS/min_vel_x" value="0.0" if="$(arg
move_forward_only)" />
  </node>
</launch>
```

Global Costmap

```
global_frame: map
rolling_window: false
track_unknown_space: true

plugins:
  - {name: static,          type: "costmap_2d::StaticLayer"}
  - {name: inflation_g,     type: "costmap_2d::InflationLayer"}
```

Local Costmap

```
global_frame: odom
rolling_window: true
width: 1.5
height: 1.5

plugins:
  - {name: obstacles_laser, type:
"costmap_2d::ObstacleLayer"}
  - {name: inflation_l, type:
"costmap_2d::InflationLayer"}
```

Algoritma DWA *Local Planner*

```
# Robot Configuration Parameters
  max_vel_x: 0.4
  min_vel_x: -0.4

  max_vel_y: 0.0
  min_vel_y: 0.0

# The velocity when robot is moving in a straight line
  max_vel_trans:  0.4
  min_vel_trans:  0.3

  max_vel_theta: 0.4
  min_vel_theta: 0.3

  acc_lim_x: 2.5
  acc_lim_y: 0.0
  acc_lim_theta: 3.2

# Goal Tolerance Parameters
  xy_goal_tolerance: 0.05
  yaw_goal_tolerance: 0.17
  latch_xy_goal_tolerance: false

        rd Simulation Parameters
        ime: 2.0
        nples: 20
```

```
  vy_samples: 0
  vth_samples: 40
  controller_frequency: 10.0

# Trajectory Scoring Parameters
  path_distance_bias: 32.0
  goal_distance_bias: 20.0
  occdist_scale: 0.02
  forward_point_distance: 0.325
  stop_time_buffer: 0.2
  scaling_speed: 0.25
  max_scaling_factor: 0.2

# Oscillation Prevention Parameters
  oscillation_reset_dist: 0.05

# Debugging
  publish_traj_pc : true
  publish_cost_grid_pc: true
```

Kode Program ROS: *base_local_planner_params.yaml*

```
TrajectoryPlannerROS:

# Robot Configuration Parameters
  max_vel_x: 0.18
  min_vel_x: 0.08

  max_vel_theta:  1.0
  min_vel_theta: -1.0
  min_in_place_vel_theta: 1.0

  acc_lim_x: 1.0
  acc_lim_y: 0.0
  acc_lim_theta: 0.6

# G     Tolerance Parameters
        al_tolerance: 0.10
        oal_tolerance: 0.05
```

```
# Differential-drive robot configuration
  holonomic_robot: false

# Forward Simulation Parameters
  sim_time: 0.8
  vx_samples: 18
  vtheta_samples: 20
  sim_granularity: 0.05
```