LAPORAN UJIAN TENGAH SEMESTER ROBOTIKA

“Obstacle Avoiding Robot”

Ditujukan Untuk Memenuhi Ujian Tengah Semester Robotika

Disusun Oleh:

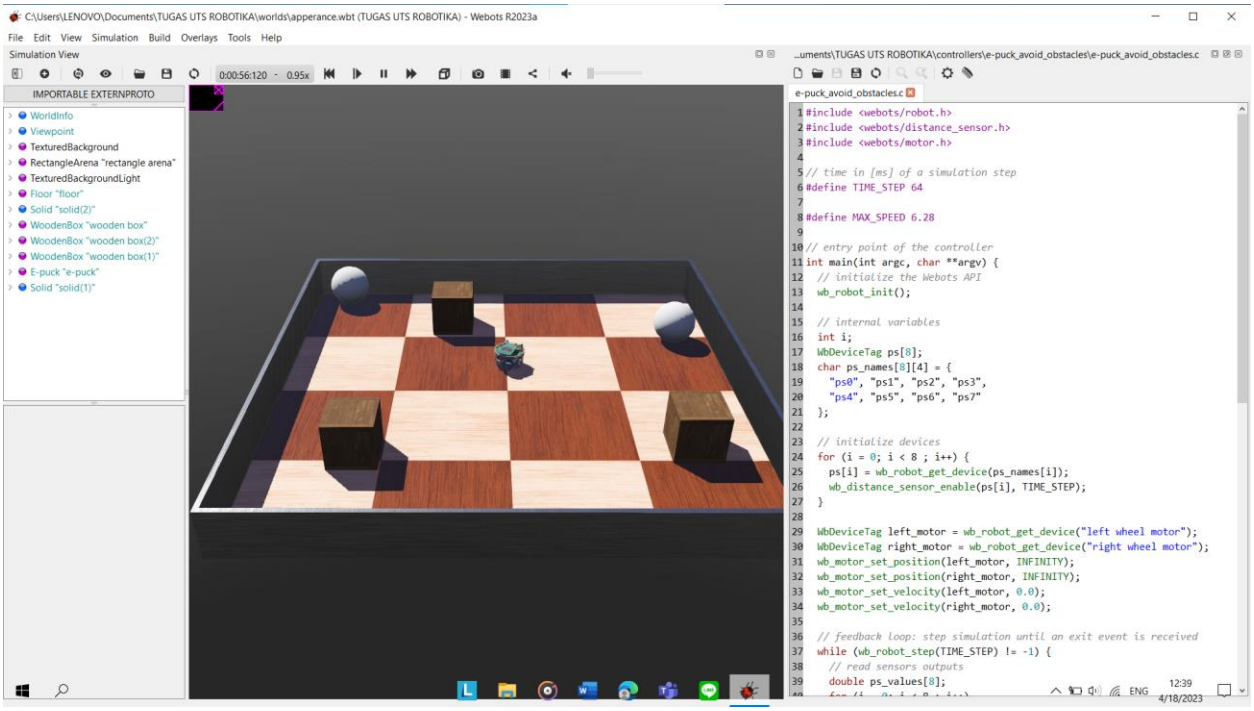
April Hamonangan Marbun (1103202039)

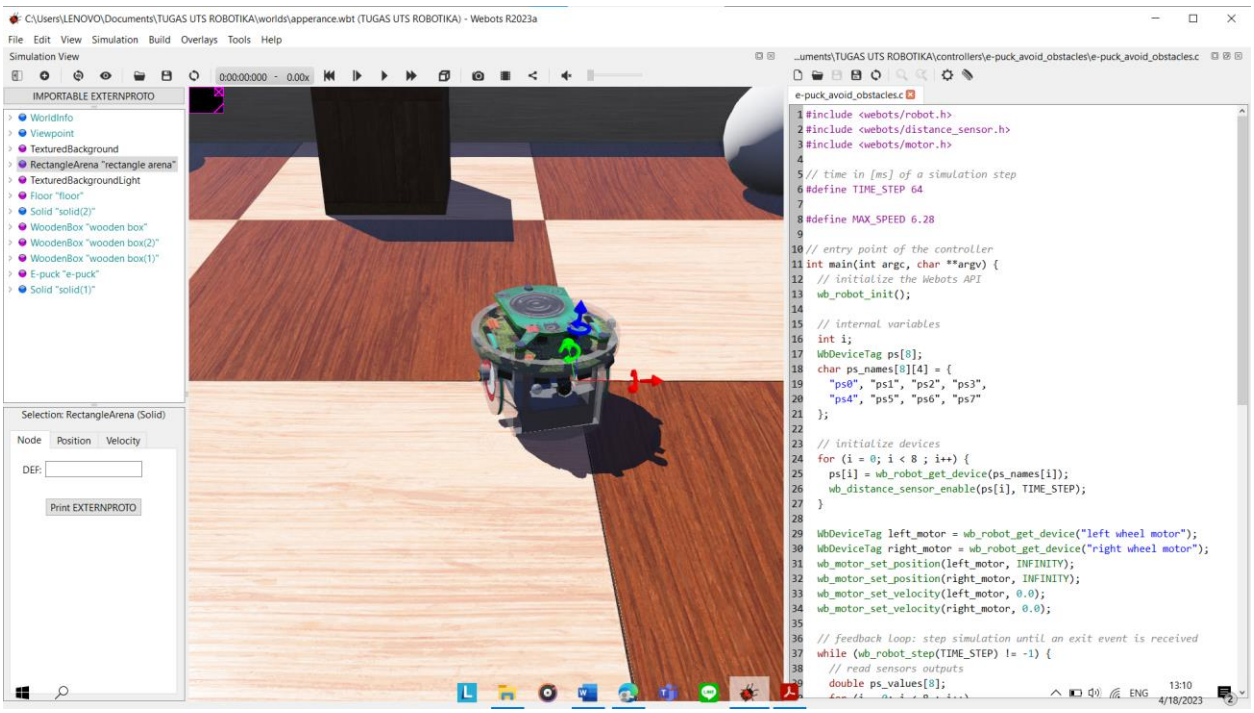
Universitas Telkom

Bandung

2023

BAB I


C:\Users\LENOVO\Documents\TUGAS UTS ROBOTIKA\worlds\apperance.wbt (TUGAS UTS ROBOTIKA) - Webots R2023a
File Edit View Simulation Build Overlays Tools Help
Simulation View
IMPORTABLE EXTERNPROTO
WorldInfo
Viewpoint
TexturedBackground
RectangleArena "rectangle arena"
TexturedBackgroundLight
Floor "floor"
Solid "solid(2)"
WoodenBox "wooden box"
WoodenBox "wooden box(2)"
WoodenBox "wooden box(1)"
E-puck "e-puck"
Solid "solid(1)"
...uments\TUGAS UTS ROBOTIKA\controllers\e-puck_avoid_obstacles\e-puck_avoid_obstacles.c
e-puck_avoid_obstacles.c
#include <webots/robot.h>
#include <webots/distance_sensor.h>
#include <webots/motor.h>
// time in [ms] of a simulation step
#define TIME_STEP 64
#define MAX_SPEED 6.28
// entry point of the controller
int main(int argc, char **argv) {
// initialize the Webots API
wb_robot_init();
// internal variables
int i;
WbDeviceTag ps[8];
char ps_names[8][4] = {
"ps0", "ps1", "ps2", "ps3",
"ps4", "ps5", "ps6", "ps7"
};
// initialize devices
for (i = 0; i < 8 ; i++) {
ps[i] = wb_robot_get_device(ps_names[i]);
wb_distance_sensor_enable(ps[i], TIME_STEP);
}
WbDeviceTag left_motor = wb_robot_get_device("left wheel motor");
WbDeviceTag right_motor = wb_robot_get_device("right wheel motor");
wb_motor_set_position(left_motor, INFINITY);
wb_motor_set_position(right_motor, INFINITY);
wb_motor_set_velocity(left_motor, 0.0);
wb_motor_set_velocity(right_motor, 0.0);
// feedback loop: step simulation until an exit event is received
while (wb_robot_step(TIME_STEP) != -1) {
// read sensors outputs
double ps_values[8];
ENG
12:39
4/18/2023


C:\Users\LENOVO\Documents\TUGAS UTS ROBOTIKA\worlds\apperance.wbt (TUGAS UTS ROBOTIKA) - Webots R2023a
File Edit View Simulation Build Overlays Tools Help
Simulation View
IMPORTABLE EXTERNPROTO
WorldInfo
Viewpoint
TexturedBackground
RectangleArena "rectangle arena"
TexturedBackgroundLight
Floor "floor"
Solid "solid(2)"
WoodenBox "wooden box"
WoodenBox "wooden box(2)"
WoodenBox "wooden box(1)"
E-puck "e-puck"
Solid "solid(1)"
Selection: RectangleArena (Solid)
Node
Position
Velocity
DEF:
Print EXTERNPROTO
...uments\TUGAS UTS ROBOTIKA\controllers\e-puck_avoid_obstacles\e-puck_avoid_obstacles.c
e-puck_avoid_obstacles.c
#include <webots/robot.h>
#include <webots/distance_sensor.h>
#include <webots/motor.h>
// time in [ms] of a simulation step
#define TIME_STEP 64
#define MAX_SPEED 6.28
// entry point of the controller
int main(int argc, char **argv) {
// initialize the Webots API
wb_robot_init();
// internal variables
int i;
WbDeviceTag ps[8];
char ps_names[8][4] = {
"ps0", "ps1", "ps2", "ps3",
"ps4", "ps5", "ps6", "ps7"
};
// initialize devices
for (i = 0; i < 8 ; i++) {
ps[i] = wb_robot_get_device(ps_names[i]);
wb_distance_sensor_enable(ps[i], TIME_STEP);
}
WbDeviceTag left_motor = wb_robot_get_device("left wheel motor");
WbDeviceTag right_motor = wb_robot_get_device("right wheel motor");
wb_motor_set_position(left_motor, INFINITY);
wb_motor_set_position(right_motor, INFINITY);
wb_motor_set_velocity(left_motor, 0.0);
wb_motor_set_velocity(right_motor, 0.0);
// feedback loop: step simulation until an exit event is received
while (wb_robot_step(TIME_STEP) != -1) {
// read sensors outputs
double ps_values[8];
ENG
13:10
4/18/2023

Robot penghindar rintangan adalah sebuah robot yang dirancang untuk menghindari rintangan saat bergerak. Tujuan dari pembuatan robot ini adalah untuk memberikan kemampuan kepada robot untuk menghindari rintangan yang ada di sekitarnya secara otomatis. Robot penghindar rintangan dapat digunakan untuk berbagai aplikasi, seperti pada industri otomotif, transportasi, dan lain sebagainya. Desain robot penghindar rintangan dapat bervariasi tergantung pada aplikasi yang diinginkan. Namun, pada umumnya, robot penghindar rintangan dilengkapi dengan sensor jarak, motor penggerak, dan kontroler yang digunakan untuk mengendalikan pergerakan robot. Sensor jarak digunakan untuk mendeteksi rintangan yang ada di sekitar robot, motor penggerak digunakan untuk menggerakkan robot, sedangkan kontroler digunakan untuk mengatur dan mengendalikan pergerakan robot.

Dalam pembuatan robot penghindar rintangan, terdapat beberapa teknologi yang dapat digunakan, seperti teknologi machine learning dan computer vision untuk meningkatkan kinerja robot dalam mengenali rintangan dan menghindarinya. Selain itu, teknologi IoT (Internet of Things) juga dapat digunakan untuk menghubungkan robot dengan perangkat lain dan mengirimkan data ke cloud untuk analisis lebih lanjut. Dengan adanya robot penghindar rintangan, diharapkan dapat membantu dalam meningkatkan efisiensi dan produktivitas pada berbagai aplikasi. Oleh karena itu, pembuatan robot penghindar rintangan menjadi suatu hal yang sangat penting untuk dilakukan.

BAB II

```c
#include <webots/robot.h>
#include <webots/distance_sensor.h>
#include <webots/motor.h>

// time in [ms] of a simulation step
#define TIME_STEP 64

#define MAX_SPEED 6.28

// entry point of the controller
int main(int argc, char **argv) {
  // initialize the Webots API
  wb_robot_init();

  // internal variables
  int i;
  WbDeviceTag ps[8];
  char ps_names[8][4] = {
    "ps0", "ps1", "ps2", "ps3",
    "ps4", "ps5", "ps6", "ps7"
  };
```

```
// initialize devices
for (i = 0; i < 8 ; i++) {
  ps[i] = wb_robot_get_device(ps_names[i]);
  wb_distance_sensor_enable(ps[i], TIME_STEP);
}

WbDeviceTag left_motor = wb_robot_get_device("left wheel motor");
WbDeviceTag right_motor = wb_robot_get_device("right wheel motor");
wb_motor_set_position(left_motor, INFINITY);
wb_motor_set_position(right_motor, INFINITY);
wb_motor_set_velocity(left_motor, 0.0);
wb_motor_set_velocity(right_motor, 0.0);

// feedback loop: step simulation until an exit event is received
while (wb_robot_step(TIME_STEP) != -1) {
  // read sensors outputs
  double ps_values[8];
  for (i = 0; i < 8 ; i++)
    ps_values[i] = wb_distance_sensor_get_value(ps[i]);

  // detect obstacles
  bool right_obstacle =
    ps_values[0] > 80.0 ||
    ps_values[1] > 80.0 ||
    ps_values[2] > 80.0;
  bool left_obstacle =
    ps_values[5] > 80.0 ||
    ps_values[6] > 80.0 ||
    ps_values[7] > 80.0;

  // initialize motor speeds at 50% of MAX_SPEED.
  double left_speed  = 0.5 * MAX_SPEED;
  double right_speed = 0.5 * MAX_SPEED;

  // modify speeds according to obstacles
  if (left_obstacle) {
    // turn right
    left_speed  = 0.5 * MAX_SPEED;
    right_speed = -0.5 * MAX_SPEED;
  }
  else if (right_obstacle) {
    // turn left
    left_speed  = -0.5 * MAX_SPEED;
    right_speed = 0.5 * MAX_SPEED;
  }
```

```
    // write actuators inputs
    wb_motor_set_velocity(left_motor, left_speed);
    wb_motor_set_velocity(right_motor, right_speed);
  }

  // cleanup the Webots API
  wb_robot_cleanup();
  return 0; //EXIT_SUCCESS
}
```

Program ini terdiri dari 3 bagian utama:

1. Inisialisasi

Pada bagian inisialisasi, program memanggil fungsi wb_robot_init() untuk menginisialisasi Webots API. Selanjutnya, program mendefinisikan variabel-variabel yang dibutuhkan untuk menyimpan objek-objek sensor dan motor pada robot. Variabel ps adalah array yang menyimpan objek-objek sensor jarak, sedangkan left_motor dan right_motor menyimpan objek motor kiri dan kanan pada robot. Kemudian, program mengaktifkan sensor jarak pada robot dengan memanggil fungsi wb_distance_sensor_enable() untuk setiap objek sensor jarak yang telah didaftarkan. Program juga menyetel posisi motor menjadi INFINITY dan kecepatan awal menjadi 0.0. Loop Utama

2. Loop Utama

Setelah selesai melakukan inisialisasi, program akan memasuki loop utama. Loop utama berjalan terus-menerus sampai program dihentikan oleh pengguna. Pada setiap iterasi, program akan membaca nilai sensor jarak yang telah diaktifkan pada bagian inisialisasi dan menyimpannya ke dalam array ps_values. Kemudian, program melakukan deteksi rintangan dengan memeriksa apakah nilai sensor jarak pada sensor-sensor di sisi kiri atau kanan robot melebihi ambang batas tertentu (80.0 dalam kasus ini). Jika sensor di sebelah kiri yang mendeteksi rintangan, maka robot akan berbelok ke kanan dengan mengurangi kecepatan motor kiri dan menambah kecepatan motor kanan. Jika sensor di sebelah kanan yang mendeteksi rintangan, maka robot akan berbelok ke kiri dengan mengurangi kecepatan motor kanan dan menambah kecepatan motor kiri. Jika tidak ada rintangan yang terdeteksi, maka kecepatan motor kiri dan kanan tetap sama dengan kecepatan maksimum (MAX_SPEED) yang telah ditetapkan sebelumnya. Setelah kecepatan motor kiri dan kanan diatur sesuai dengan deteksi rintangan, program menulis nilai kecepatan motor ke objek motor kiri dan kanan dengan memanggil fungsi wb_motor_set_velocity().

3. Cleanup

Setelah loop utama selesai, program membersihkan Webots API dengan memanggil fungsi wb_robot_cleanup(). Kesimpulan Program pengendali robot penghindar rintangan ini merupakan contoh sederhana dari bagaimana menggunakan sensor dan motor pada robot untuk

mengimplementasikan sebuah algoritma yang memungkinkan robot untuk menghindari rintangan secara otomatis. Program ini dapat dikembangkan lebih jauh dengan menambahkan sensor lain, seperti sensor garis atau sensor kamera, dan memodifikasi algoritma penghindar rintangan agar lebih efisien dan akurat dalam menghindari rintangan.