Nama : Yusran Yasir

NIM : 1103213166

1. Camera robot untuk Mendeteksi blob warna (merah, hijau, dan biru)

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <webots/camera.h>
#include <webots/motor.h>
#include <webots/robot.h>
#include <webots/utils/system.h>
```

Header: Menyediakan fungsi untuk operasi dasar seperti input/output, manajemen memori, serta modul Webots seperti kamera, motor, dan robot.
Konstanta ANSI Color: Digunakan untuk menampilkan teks berwarna pada terminal (misalnya untuk menunjukkan warna blob yang terdeteksi).
Konstanta SPEED: Menentukan kecepatan dasar motor robot.

```
enum BLOB_TYPE { RED, GREEN, BLUE, NONE };
```

BLOB_TYPE: Enum yang digunakan untuk mendefinisikan tipe warna blob yang dapat dideteksi (merah, hijau, biru, atau tidak ada).

```
wb_robot_init();
const int time_step = wb_robot_get_basic_time_step();
```

wb_robot_init: Inisialisasi robot dan mempersiapkan simulasi.
time_step: Mengambil interval waktu simulasi (biasanya dalam milidetik), yang akan digunakan dalam loop utama.

```
camera = wb_robot_get_device("camera");
wb_camera_enable(camera, time_step);
width = wb_camera_get_width(camera);
height = wb_camera_get_height(camera);
```

Menginisialisasi kamera, mengaktifkannya, dan mendapatkan dimensi gambar yang dihasilkan.

```
left_motor = wb_robot_get_device("left wheel motor");
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);
```

Motor kiri dan kanan diatur pada mode kecepatan dengan posisi target tak hingga (INFINITY).

```
while (wb_robot_step(time_step) != -1) {
   const unsigned char *image = wb_camera_get_image(camera);
```

wb_robot_step: Menjalankan simulasi satu langkah, sinkron dengan interval waktu.
wb_camera_get_image: Mendapatkan gambar terbaru dari kamera.

```
if (pause_counter > 640 / time_step) {
   left_speed = 0;
   right_speed = 0;
}
```
Jika masih dalam durasi "menunggu", robot berhenti.
```
else if (pause_counter > 0) {
   left_speed = -SPEED;
   right_speed = SPEED;
}
```
Robot berputar untuk menghindari deteksi ulang blob yang sama.
```
else {
   // Analisis gambar untuk mencari blob
}
```
Robot menganalisis gambar dari kamera untuk mendeteksi warna tertentu.
```
if ((red > 3 * green) && (red > 3 * blue))
   current_blob = RED;
else if ((green > 3 * red) && (green > 3 * blue))
   current_blob = GREEN;
else if ((blue > 3 * red) && (blue > 3 * green))
   current_blob = BLUE;
else
   current_blob = NONE;
```
Pixel di area tengah gambar dihitung untuk intensitas merah, hijau, dan biru.
Jika salah satu warna dominan (3 kali lebih banyak dari yang lain), warna tersebut dianggap sebagai blob yang terdeteksi.
```
wb_camera_save_image(camera, filepath, 100);
```
pause_counter direset untuk memberikan waktu jeda.
```
wb_motor_set_velocity(left_motor, left_speed);
wb_motor_set_velocity(right_motor, right_speed);
```
Kecepatan motor kiri dan kanan diatur berdasarkan logika di atas.
```
wb_robot_cleanup();
```
Membersihkan sumber daya yang digunakan sebelum keluar dari program.

Program ini mengilustrasikan logika sederhana untuk robot berbasis kamera yang:
Mendeteksi warna (merah, hijau, biru) pada gambar kamera.
Mengontrol motor untuk mencari dan berhenti di depan blob warna.
Menyimpan gambar blob yang terdeteksi.
Kode ini dapat diperluas dengan menambahkan logika lebih kompleks seperti navigasi berbasis warna atau integrasi sensor tambahan.

2. Camera robot dengan fokus kamera berdasarkan objek yang ada di depannya
```
#include <webots/camera.h>
#include <webots/distance_sensor.h>
#include <webots/motor.h>
```

#include <webots/robot.h>
Mengimpor modul API Webots:
camera.h: Untuk kontrol kamera.
distance_sensor.h: Untuk membaca nilai dari sensor jarak.
motor.h: Untuk mengontrol motor robot.
robot.h: Untuk inisialisasi dan kontrol robot.
#define SPEED 1
#define TIME_STEP 32
SPEED: Kecepatan rotasi motor robot.
TIME_STEP: Interval waktu dalam milidetik antara setiap langkah simulasi.
wb_robot_init();

camera = wb_robot_get_device("camera");
wb_camera_enable(camera, TIME_STEP);

distance_sensor = wb_robot_get_device("distance sensor");
wb_distance_sensor_enable(distance_sensor, TIME_STEP);

left_motor = wb_robot_get_device("left wheel motor");
right_motor = wb_robot_get_device("right wheel motor");
wb_motor_set_position(left_motor, INFINITY);
wb_motor_set_position(right_motor, INFINITY);
wb_robot_init(): Menginisialisasi robot dan perangkat.
Kamera diaktifkan dengan wb_camera_enable().
Sensor jarak diaktifkan dengan wb_distance_sensor_enable().
Motor kiri dan kanan diatur ke mode kecepatan bebas dengan posisi tak terbatas.
wb_motor_set_velocity(left_motor, -SPEED);
wb_motor_set_velocity(right_motor, SPEED);
Motor kiri bergerak mundur dengan kecepatan -1.
Motor kanan bergerak maju dengan kecepatan 1.
while (wb_robot_step(TIME_STEP) != -1) {
  const double object_distance = wb_distance_sensor_get_value(distance_sensor) / 1000;
  wb_camera_set_focal_distance(camera, object_distance);
}
wb_robot_step(TIME_STEP): Memastikan simulasi berjalan pada setiap langkah waktu.
wb_distance_sensor_get_value(): Membaca jarak objek dari sensor dalam satuan milimeter.
Nilai dikonversi ke meter dengan membaginya dengan 1000.
wb_camera_set_focal_distance(): Menyesuaikan jarak fokus kamera berdasarkan jarak objek yang terdeteksi oleh sensor.
wb_robot_cleanup();

Mengakhiri simulasi dengan membersihkan sumber daya yang digunakan.

3. Camera robot Deteksi Blob Berwarna pada Robot dengan efek Motion Blur kamera

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <webots/camera.h>
#include <webots/motor.h>
#include <webots/robot.h>
#include <webots/utils/system.h>
```

Header file menyediakan fungsi untuk:
I/O dasar (stdio.h).
Alokasi memori (stdlib.h).
String manipulasi (string.h).
API Webots untuk kamera, motor, dan kontrol robot.

```
#define SPEED 4
enum BLOB_TYPE { RED, GREEN, BLUE, NONE };
```

SPEED: Kecepatan dasar untuk motor.
enum BLOB_TYPE: Enumerator untuk mendefinisikan tipe warna blob (RED, GREEN, BLUE, atau NONE jika tidak ada blob terdeteksi).

```
wb_robot_init();

const int time_step = wb_robot_get_basic_time_step();
```

wb_robot_init(): Menginisialisasi robot dan perangkat.
wb_robot_get_basic_time_step(): Mengambil langkah waktu (time step) default dari simulasi.

```
camera = wb_robot_get_device("camera");
wb_camera_enable(camera, time_step);
width = wb_camera_get_width(camera);
height = wb_camera_get_height(camera);
```

Kamera diaktifkan, dan dimensinya diambil untuk menganalisis gambar.

```
left_motor = wb_robot_get_device("left wheel motor");
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);
```

Motor kiri dan kanan diatur untuk mode kecepatan bebas (position **INFINITY**).

```
while (wb_robot_step(time_step) != -1) {
```

Loop utama berjalan selama simulasi aktif.

```
const unsigned char *image = wb_camera_get_image(camera);
```

Mengambil Gambar dari Kamera

```
red = 0; green = 0; blue = 0;
```

Deklarasi Variabel Warna

```
for (i = width / 3; i < 2 * width / 3; i++) {
  for (j = height / 2; j < 3 * height / 4; j++) {
    red += wb_camera_image_get_red(image, width, i, j);
    blue += wb_camera_image_get_blue(image, width, i, j);
    green += wb_camera_image_get_green(image, width, i, j);
  }
}
```

Program memindai piksel di bagian tengah gambar.
Setiap piksel ditambahkan ke total komponen warna masing-masing (merah, hijau, biru).

```
if ((red > 3 * green) && (red > 3 * blue))
  current_blob = RED;
else if ((green > 3 * red) && (green > 3 * blue))
  current_blob = GREEN;
else if ((blue > 3 * red) && (blue > 3 * green))
  current_blob = BLUE;
else
  current_blob = NONE;
```

Warna dianggap blob jika satu komponen warna 3 kali lebih dominan daripada yang lain.

```
if (current_blob == NONE) {
  left_speed = -SPEED;
  right_speed = SPEED;
}
```

Robot terus berputar untuk mencari blob baru.

```
left_speed = 0;
right_speed = 0;
printf("Looks like I found a %s%s%s blob.\n", ansi_colors[current_blob],
color_names[current_blob], ANSI_COLOR_RESET);
```

Robot berhenti dan mencetak warna blob yang ditemukan.

```
wb_camera_save_image(camera, filepath, 100);
```

Gambar blob disimpan ke file dalam direktori pengguna.

```
pause_counter = 1280 / time_step;
```

Robot berhenti sementara untuk menghindari mendeteksi blob yang sama.

```
wb_motor_set_velocity(left_motor, left_speed);
wb_motor_set_velocity(right_motor, right_speed);
```

Kecepatan motor kiri dan kanan diperbarui setiap langkah waktu simulasi.

```
wb_robot_cleanup();
```

Membersihkan sumber daya sebelum keluar dari program.

4. Robot dengan kamera Deteksi Blob berwarna dengan noise mask
   Inisialisasi dan Konstanta
   Header Files: Berisi pustaka yang diperlukan, seperti pengontrol robot, kamera, motor, dan fungsi sistem.

ANSI Color: Digunakan untuk mencetak teks berwarna di terminal.
enum BLOB_TYPE: Digunakan untuk mendefinisikan jenis blob yang akan dideteksi (RED, GREEN, BLUE, atau NONE jika tidak ada).
wb_robot_init();
const int time_step = wb_robot_get_basic_time_step();
wb_robot_init(): Menginisialisasi robot.
time_step: Mendapatkan langkah waktu (time step) dari simulasi.
camera = wb_robot_get_device("camera");
wb_camera_enable(camera, time_step);
width = wb_camera_get_width(camera);
height = wb_camera_get_height(camera);
Kamera diaktifkan dan dimensinya diperoleh untuk analisis gambar.
left_motor = wb_robot_get_device("left wheel motor");
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);
Motor kiri dan kanan diatur untuk mode kontrol kecepatan (target posisi INFINITY).
while (wb_robot_step(time_step) != -1) {
Loop utama berjalan selama simulasi aktif.
const unsigned char *image = wb_camera_get_image(camera);
Pengambilan Gambar
if (pause_counter > 0)
  pause_counter--;
pause_counter digunakan untuk menghentikan robot sementara setelah mendeteksi blob.
else if (!image) {
  left_speed = 0;
  right_speed = 0;
}
Jika kamera tidak mengembalikan gambar (sinkronisasi dinonaktifkan), robot berhenti.
red = 0;
green = 0;
blue = 0;
for (i = width / 3; i < 2 * width / 3; i++) {
  for (j = height / 2; j < 3 * height / 4; j++) {
    red += wb_camera_image_get_red(image, width, i, j);
    blue += wb_camera_image_get_blue(image, width, i, j);
    green += wb_camera_image_get_green(image, width, i, j);
  }
}

Program memindai piksel di tengah layar.
Komponen warna (merah, hijau, biru) dihitung untuk mendeteksi warna dominan.

```
if ((red > 3 * green) && (red > 3 * blue))
  current_blob = RED;
else if ((green > 3 * red) && (green > 3 * blue))
  current_blob = GREEN;
else if ((blue > 3 * red) && (blue > 3 * green))
  current_blob = BLUE;
else
  current_blob = NONE;
```

Blob warna terdeteksi jika salah satu komponen warna dominan 3 kali lebih besar dari komponen lainnya.

```
if (current_blob == NONE) {
  left_speed = -SPEED;
  right_speed = SPEED;
}
```

Robot berputar untuk mencari blob baru.

```
else {
  left_speed = 0;
  right_speed = 0;
  printf("Looks like I found a %s%s%s blob.\n", ansi_colors[current_blob],
color_names[current_blob], ANSI_COLOR_RESET);
}
```

Jika blob ditemukan, robot berhenti dan mencetak warna blob yang terdeteksi.

```
wb_camera_save_image(camera, filepath, 100);
pause_counter = 1280 / time_step;
```

Gambar disimpan ke file di direktori pengguna.
pause_counter diatur untuk memberi waktu jeda sebelum melanjutkan pencarian.

```
wb_motor_set_velocity(left_motor, left_speed);
wb_motor_set_velocity(right_motor, right_speed);
```

Kecepatan motor diperbarui berdasarkan kondisi saat ini (mencari blob, berhenti, atau berputar).

```
wb_robot_cleanup();
```

Membersihkan semua sumber daya sebelum keluar dari program.

5. Deteksi Objek dengan kamera dan pengenalan Objek pada robot
   Header Files
   <webots/camera.h>: Untuk mengakses kamera dan fitur-fitur terkait.
   <webots/camera_recognition_object.h>: Untuk pengenalan objek dengan kamera.
   <webots/motor.h>: Untuk kontrol motor robot.
   <webots/robot.h>: Untuk fungsi utama robot.

```
#define SPEED 1.5
#define TIME_STEP 64
```

SPEED: Kecepatan motor.

TIME_STEP: Interval waktu antar langkah simulasi dalam milidetik.
wb_robot_init();
Menginisialisasi perangkat robot dan memulai simulasi.
camera = wb_robot_get_device("camera");
wb_camera_enable(camera, TIME_STEP);
wb_camera_recognition_enable(camera, TIME_STEP);
wb_camera_enable: Mengaktifkan kamera robot.
wb_camera_recognition_enable: Mengaktifkan fitur pengenalan objek kamera, sehingga robot dapat mendeteksi dan mengenali objek di lingkungan simulasi.
left_motor = wb_robot_get_device("left wheel motor");
right_motor = wb_robot_get_device("right wheel motor");
wb_motor_set_position(left_motor, INFINITY);
wb_motor_set_position(right_motor, INFINITY);
Motor kiri dan kanan diatur ke mode kontrol kecepatan dengan target posisi INFINITY.
wb_motor_set_velocity(left_motor, -SPEED);
wb_motor_set_velocity(right_motor, SPEED);
Robot diberi kecepatan diferensial untuk berputar di tempat.
while (wb_robot_step(TIME_STEP) != -1) {
Loop utama berjalan selama simulasi aktif, dengan langkah waktu sesuai dengan TIME_STEP.
int number_of_objects = wb_camera_recognition_get_number_of_objects(camera);
printf("\nRecognized %d objects.\n", number_of_objects);
wb_camera_recognition_get_number_of_objects: Mengembalikan jumlah objek yang dikenali oleh kamera.
Dicetak jumlah objek yang terdeteksi.
const WbCameraRecognitionObject *objects =
wb_camera_recognition_get_objects(camera);
for (i = 0; i < number_of_objects; ++i) {
wb_camera_recognition_get_objects: Mengembalikan array berisi detail setiap objek yang dikenali.
printf("Model of object %d: %s\n", i, objects[i].model);
printf("Id of object %d: %d\n", i, objects[i].id);
Model: Nama model objek.
ID: ID unik dari objek.
printf("Relative position of object %d: %lf %lf %lf\n", i, objects[i].position[0], objects[i].position[1], objects[i].position[2]);
Posisi objek dalam koordinat relatif terhadap robot.
printf("Relative orientation of object %d: %lf %lf %lf %lf\n", i, objects[i].orientation[0], objects[i].orientation[1], objects[i].orientation[2], objects[i].orientation[3]);
Orientasi objek dalam quaternion relatif terhadap robot.
printf("Size of object %d: %lf %lf\n", i, objects[i].size[0], objects[i].size[1]);

Ukuran objek dalam dimensi dunia simulasi (lebar dan tinggi).
printf("Position of the object %d on the camera image: %d %d\n", i,
objects[i].position_on_image[0], objects[i].position_on_image[1]);
printf("Size of the object %d on the camera image: %d %d\n", i,
objects[i].size_on_image[0], objects[i].size_on_image[1]);
Posisi dan ukuran objek pada gambar yang diambil oleh kamera.
for (j = 0; j < objects[i].number_of_colors; ++j)
  printf("- Color %d/%d: %lf %lf %lf\n", j + 1, objects[i].number_of_colors,
objects[i].colors[3 * j], objects[i].colors[3 * j + 1], objects[i].colors[3 * j + 2]);
Daftar warna objek (dalam format RGB).
wb_robot_cleanup();
wb_robot_cleanup membersihkan semua sumber daya yang digunakan oleh simulasi sebelum program selesai.

6. Implementasi Segmentasi kamera pada Robot
<webots/camera.h>: Untuk mengakses perangkat kamera dan fungsinya, termasuk pengenalan objek dan segmentasi.
<webots/display.h>: Untuk mengakses perangkat display, yang digunakan untuk menampilkan hasil segmentasi.
<webots/motor.h>: Untuk mengontrol pergerakan motor robot.
<webots/robot.h>: Untuk fungsi utama pengendalian robot.
#define SPEED 1.5
#define TIME_STEP 64
SPEED: Kecepatan motor robot.
TIME_STEP: Interval waktu langkah simulasi dalam milidetik.
wb_robot_init();
Fungsi ini memulai simulasi robot dan menginisialisasi semua perangkat yang digunakan.
camera = wb_robot_get_device("camera");
wb_camera_enable(camera, TIME_STEP);
wb_camera_recognition_enable(camera, TIME_STEP);
wb_camera_recognition_enable_segmentation(camera);
wb_robot_get_device: Mengambil referensi ke perangkat kamera.
wb_camera_enable: Mengaktifkan kamera untuk mengambil gambar.
wb_camera_recognition_enable: Mengaktifkan kemampuan pengenalan objek pada kamera.
wb_camera_recognition_enable_segmentation: Mengaktifkan fitur segmentasi kamera, yang menghasilkan gambar tersegmentasi.
const int width = wb_camera_get_width(camera);
const int height = wb_camera_get_height(camera);
Mendapatkan lebar dan tinggi gambar dari kamera.
display = wb_robot_get_device("segmented image display");
Mengambil referensi ke perangkat display, yang digunakan untuk menampilkan hasil gambar tersegmentasi.

left_motor = wb_robot_get_device("left wheel motor");
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, -SPEED);
wb_motor_set_velocity(right_motor, SPEED);
Mengambil referensi ke motor kiri dan kanan.
Mengatur mode motor ke kontrol kecepatan (dengan target posisi tak terbatas).
Memberikan kecepatan diferensial pada motor untuk membuat robot bergerak berputar di tempat.
while (wb_robot_step(TIME_STEP) != -1) {
Loop utama berjalan selama simulasi aktif, dengan langkah waktu sesuai TIME_STEP.
if (wb_camera_recognition_is_segmentation_enabled(camera) &&
wb_camera_recognition_get_sampling_period(camera) > 0) {
Mengecek apakah fitur segmentasi kamera aktif dan pengambilan gambar kamera berjalan.
const unsigned char *data =
wb_camera_recognition_get_segmentation_image(camera);
wb_camera_recognition_get_segmentation_image: Mengembalikan pointer ke data gambar tersegmentasi (format BGRA, yaitu Blue-Green-Red-Alpha).
segmented_image = wb_display_image_new(display, width, height, data, WB_IMAGE_BGRA);
wb_display_image_paste(display, segmented_image, 0, 0, false);
wb_display_image_delete(display, segmented_image);
wb_display_image_new: Membuat gambar baru di perangkat display berdasarkan data gambar tersegmentasi.
wb_display_image_paste: Menempelkan gambar di posisi (0,0) pada display.
wb_display_image_delete: Menghapus gambar setelah selesai ditampilkan untuk menghindari kebocoran memori.
wb_robot_cleanup
ungsi ini digunakan untuk membersihkan semua sumber daya yang digunakan oleh simulasi sebelum program selesai.

7. Implementasi Penggunaan Kamera Bola pada robot
<webots/camera.h>: Mengontrol kamera untuk menangkap gambar.
<webots/distance_sensor.h>: Mengontrol sensor jarak untuk mendeteksi objek.
<webots/motor.h>: Mengontrol motor robot.
<webots/robot.h>: Fungsi utama kontrol robot.
<webots/utils/ansi_codes.h>: Memungkinkan manipulasi tampilan konsol (seperti membersihkan layar).
<math.h>: Untuk fungsi matematika (seperti atan untuk perhitungan sudut).
<stdio.h>: Untuk mencetak informasi ke terminal.
#define TIME_STEP 64

#define THRESHOLD 200
TIME_STEP: Interval waktu antar langkah simulasi dalam milidetik.
THRESHOLD: Batas nilai intensitas untuk menentukan warna dominan (merah, hijau, atau biru).
#define RED 0
#define GREEN 1
#define BLUE 2
#define LEFT 0
#define RIGHT 1
#define X 0
#define Y 1
Digunakan untuk indeks warna, motor (kiri/kanan), dan koordinat (X, Y).
double coord2D_to_angle(double x, double y) { ... }
Tujuan: Menghitung sudut arah blob berdasarkan koordinat 2D relatif terhadap kamera.
Logika:
Jika koordinat berada di kuadran tertentu, sudut dihitung dengan fungsi atan.
Jika koordinat berada di sumbu X atau Y, sudutnya diatur ke nilai tertentu (misalnya, 90° untuk sumbu Y).
wb_robot_init();
Fungsi ini memulai simulasi robot dan menginisialisasi semua perangkat.
WbDeviceTag camera = wb_robot_get_device("camera");
wb_camera_enable(camera, 2 * TIME_STEP);
int width = wb_camera_get_width(camera);
int height = wb_camera_get_height(camera);
Kamera diaktifkan, dan dimensinya (lebar dan tinggi gambar) diambil untuk analisis gambar.
WbDeviceTag us[2];
double us_values[2];
double coefficients[2][2] = {{6.0, -3.0}, {-5.0, 4.0}};
us[LEFT] = wb_robot_get_device("us0");
us[RIGHT] = wb_robot_get_device("us1");
for (i = 0; i < 2; i++)
  wb_distance_sensor_enable(us[i], TIME_STEP);
Sensor jarak kiri dan kanan diaktifkan untuk mendeteksi jarak ke objek.
coefficients: Koefisien yang digunakan untuk mengontrol motor berdasarkan nilai sensor jarak.
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);

Motor diatur ke mode kontrol kecepatan (posisi tak terbatas).

```
while (wb_robot_step(TIME_STEP) != -1) {
```

Loop utama berjalan selama simulasi aktif, dengan langkah waktu sesuai TIME_STEP.

```
const unsigned char *image = wb_camera_get_image(camera);
for (i = 0; i < 2; i++)
  us_values[i] = wb_distance_sensor_get_value(us[i]);
```

Kamera mengambil gambar, dan nilai jarak dari sensor dibaca.

```
for (i = 0; i < 2; i++) {
  speed[i] = 0.0;
  for (k = 0; k < 2; k++)
    speed[i] += us_values[k] * coefficients[i][k];
}
```

Kecepatan motor kiri dan kanan dihitung berdasarkan nilai sensor jarak dan coefficients.

```
for (y = 0; y < height; y++) {
  for (x = 0; x < width; x++) {
    r = wb_camera_image_get_red(image, width, x, y);
    g = wb_camera_image_get_green(image, width, x, y);
    b = wb_camera_image_get_blue(image, width, x, y);
    if (r > THRESHOLD && g < THRESHOLD && b < THRESHOLD) { ... }
    else if (r < THRESHOLD && g > THRESHOLD && b < THRESHOLD) { ... }
    else if (r < THRESHOLD && g < THRESHOLD && b > THRESHOLD) { ... }
  }
}
```

wb_camera_image_get_*: Mendapatkan nilai intensitas warna (merah, hijau, biru) pada setiap piksel gambar.

Warna blob ditentukan berdasarkan intensitas RGB dibandingkan dengan THRESHOLD.

```
ANSI_CLEAR_CONSOLE();
for (i = 0; i < 3; i++)
  printf("last %s blob seen at (%d,%d) with an angle of %f\n",
         (i == GREEN) ? "Green" :
         (i == RED)   ? "Red" :
                        "Blue",
         color_index[i][X], color_index[i][Y],
         coord2D_to_angle((double)(color_index[i][X] + width / 2),
(double)(color_index[i][Y] + height / 2)));
```

Informasi tentang blob terakhir yang terlihat (koordinat dan sudut relatif terhadap kamera) dicetak ke konsol.

```
wb_motor_set_velocity(left_motor, 3.0 + speed[LEFT]);
wb_motor_set_velocity(right_motor, 3.0 + speed[RIGHT]);
```

Kecepatan motor diperbarui berdasarkan nilai sensor jarak dan kecepatan dasar (3.0).