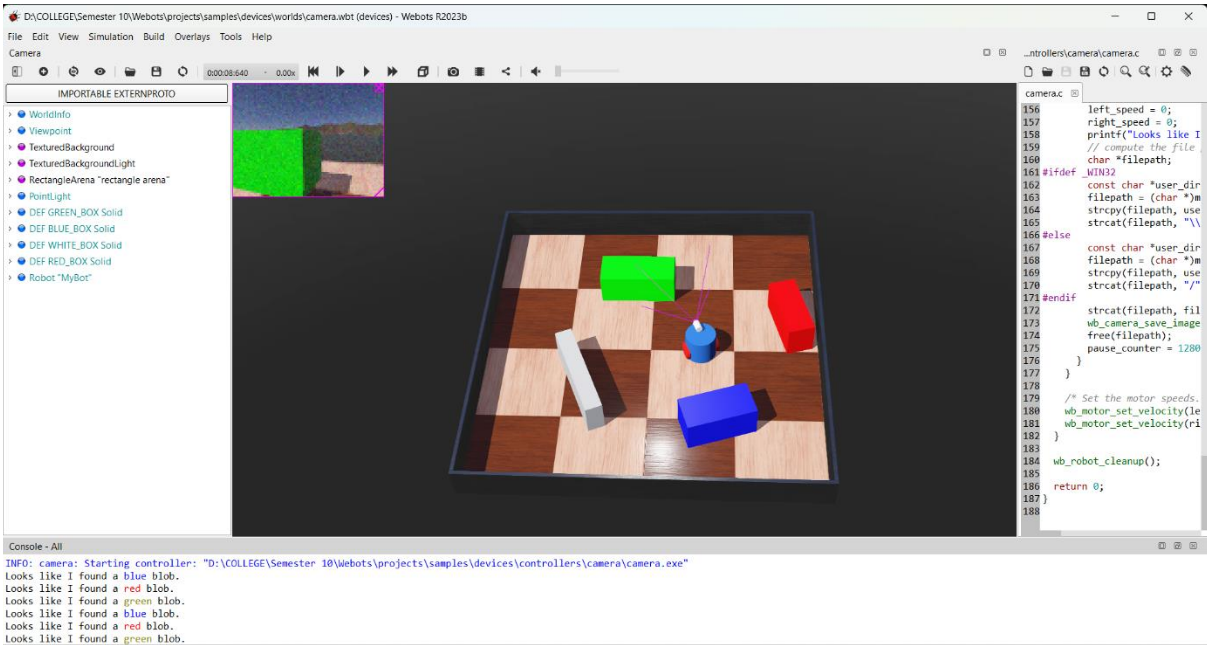
Tugas Week 10 : Simulasi Robot dengan Kamera di Webots

Simulasi 1 : Camera robot untuk mendeteksi blob warna (merah, hijau, dan biru)



```
        left_speed = 0;
        right_speed = 0;
        printf("Looks like I
        // compute the file
        char *filepath;
#ifdef _WIN32
        const char *user_dir
        filepath = (char *)m
        strcpy(filepath, use
        strcat(filepath, "\\
#else
        const char *user_dir
        filepath = (char *)m
        strcpy(filepath, use
        strcat(filepath, "/"
#endif
        strcat(filepath, fil
        wb_camera_save_image
        free(filepath);
        pause_counter = 1280
      }
    }

    /* Set the motor speeds.
    wb_motor_set_velocity(le
    wb_motor_set_velocity(ri
  }

  wb_robot_cleanup();

  return 0;
}
```

```
INFO: camera: Starting controller: "D:\COLLEGE\Semester 10\Webots\projects\samples\devices\controllers\camera\camera.exe"
Looks like I found a blue blob.
Looks like I found a red blob.
Looks like I found a green blob.
Looks like I found a blue blob.
Looks like I found a red blob.
Looks like I found a green blob.
```

Output:

```
INFO: camera: Starting controller: "D:\COLLEGE\Semester
10\Webots\projects\samples\devices\controllers\camera\camera.exe"
Looks like I found a blue blob.
Looks like I found a red blob.
Looks like I found a green blob.
Looks like I found a blue blob.
```

Berikut adalah penjelasan dari koding berikut:

Header dan Definisi

1.Header File: Kode ini mengimpor beberapa header file untuk fungsi input-output, manajemen memori, operasi string, dan fungsi Webots.

```
#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>
```

2. Definisi Warna ANSI: Mendefinisikan warna-warna ANSI untuk output berwarna di terminal.

```
#define ANSI_COLOR_RED "\x1b[31m"
#define ANSI_COLOR_GREEN "\x1b[32m"
#define ANSI_COLOR_YELLOW "\x1b[33m"
#define ANSI_COLOR_BLUE "\x1b[34m"
#define ANSI_COLOR_RESET "\x1b[0m"
```

3. Konstanta dan Enumerasi: Mendefinisikan kecepatan motor dan tipe blob warna.

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

Fungsi Utama main

1.Inisialisasi Variabel dan Perangkat: Inisialisasi perangkat kamera dan motor, serta variabel yang diperlukan.

```
WbDeviceTag camera, left_motor, right_motor;
int width, height;
int pause_counter = 0;
int left_speed, right_speed;
int i, j;
int red, blue, green;
const char *color_names[3] = {"red", "green", "blue"};
const char *ansi_colors[3] = {ANSI_COLOR_RED, ANSI_COLOR_GREEN,
ANSI_COLOR_BLUE};
const char *filenames[3] = {"red_blob.png", "green_blob.png", "blue_blob.png"};
enum BLOB_TYPE current_blob;
```

2.Inisialisasi Robot dan Perangkat: Inisialisasi robot, mengambil perangkat kamera, dan mengatur parameter kamera.

```
wb_robot_init();
const int time_step = wb_robot_get_basic_time_step();
camera = wb_robot_get_device("camera");
wb_camera_enable(camera, time_step);
width = wb_camera_get_width(camera);
height = wb_camera_get_height(camera);
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);
```

Loop Utama

1.Mengambil Gambar dari Kamera: Mengambil gambar terbaru dari kamera.

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

2. Mengelola pause_counter: Mengatur kecepatan motor berdasarkan nilai pause_counter.

```
if (pause_counter > 0)
  pause_counter--;
if (pause_counter > 640 / time_step) {
  left_speed = 0;
  right_speed = 0;
} else if (pause_counter > 0) {
  left_speed = -SPEED;
  right_speed = SPEED;
} else if (!image) {
  left_speed = 0;
  right_speed = 0;
```

3. Menganalisis Gambar: Menganalisis piksel di tengah gambar untuk mendeteksi blob warna.

```
else {
  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);
    }
  }
```

4. Mendeteksi Blob Warna: Menentukan apakah ada blob warna yang 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;
```

5. Mengatur Kecepatan Motor: Mengatur kecepatan motor berdasarkan deteksi blob warna.

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

#ifdef _WIN32
const char *user_directory =
wbu_system_short_path(wbu_system_getenv("USERPROFILE"));
filepath = (char *)malloc(strlen(user_directory) + 16);
strcpy(filepath, user_directory);
strcat(filepath, "\");
#else
const char *user_directory = wbu_system_getenv("HOME");
filepath = (char *)malloc(strlen(user_directory) + 16);
strcpy(filepath, user_directory);
strcat(filepath, "/");
#endif
strcat(filepath, filenames[current_blob]);
wb_camera_save_image(camera, filepath, 100);
free(filepath);
pause_counter = 1280 / time_step;
}
```

6. Mengatur Kecepatan Motor: Menetapkan kecepatan motor untuk gerakan robot.

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

Pembersihan dan Keluar

Pembersihan: Membersihkan semua sumber daya sebelum keluar

```
wb_robot_cleanup();
return 0;
```

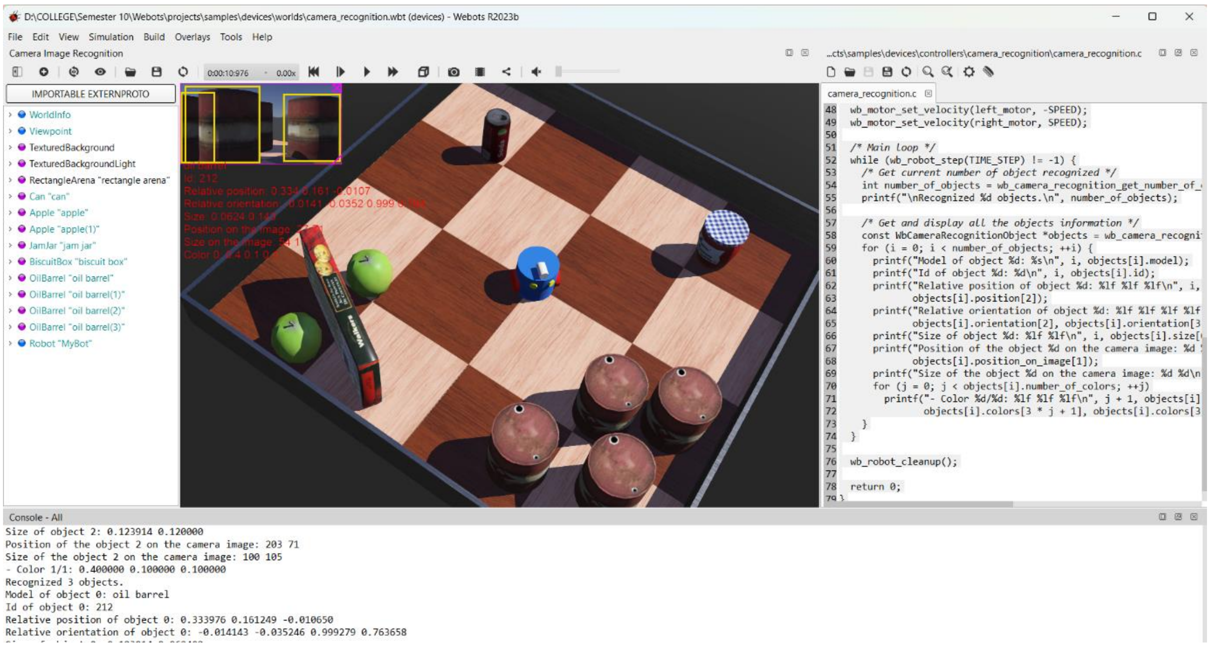
Kesimpulan**:**

- **Inisialisasi dan Pengaturan**: Robot diinisialisasi dan kamera serta motor dikonfigurasi.

- **Pengambilan Gambar dan Analisis**: Gambar diambil dari kamera dan dianalisis untuk mendeteksi warna merah, hijau, dan biru.
- **Logika Deteksi Blob**: Jika warna tertentu terdeteksi sebagai dominan, robot berhenti dan menyimpan gambar. Jika tidak ada warna dominan, robot berputar mencari blob.
- **Kontrol Motor**: Kecepatan motor diatur berdasarkan hasil analisis gambar untuk mengontrol gerakan robot.

Proses ini diulang dalam loop utama hingga simulasi selesai.

Simulasi 2 : Deteksi Objek dengan Kamera dan Pengenalan Objek pada Robot



```c
  wb_motor_set_velocity(left_motor, -SPEED);
  wb_motor_set_velocity(right_motor, SPEED);

  /* Main loop */
  while (wb_robot_step(TIME_STEP) != -1) {
    /* Get current number of object recognized */
    int number_of_objects = wb_camera_recognition_get_number_of_
    printf("\nRecognized %d objects.\n", number_of_objects);

    /* Get and display all the objects information */
    const WbCameraRecognitionObject *objects = wb_camera_recogni
    for (i = 0; i < number_of_objects; ++i) {
      printf("Model of object %d: %s\n", i, objects[i].model);
      printf("Id of object %d: %d\n", i, objects[i].id);
      printf("Relative position of object %d: %lf %lf %lf\n", i,
             objects[i].position[2]);
      printf("Relative orientation of object %d: %lf %lf %lf %lf
             objects[i].orientation[2], objects[i].orientation[3
      printf("Size of object %d: %lf %lf\n", i, objects[i].size[
      printf("Position of the object %d on the camera image: %d
             objects[i].position_on_image[1]);
      printf("Size of the object %d on the camera image: %d %d\n
      for (j = 0; j < objects[i].number_of_colors; ++j)
        printf("- Color %d/%d: %lf %lf %lf\n", j + 1, objects[i]
               objects[i].colors[3 * j + 1], objects[i].colors[3
    }
  }

  wb_robot_cleanup();

  return 0;
}
```

```
Size of object 2: 0.123914 0.120000
Position of the object 2 on the camera image: 203 71
Size of the object 2 on the camera image: 100 105
- Color 1/1: 0.400000 0.100000 0.100000
Recognized 3 objects.
Model of object 0: oil barrel
Id of object 0: 212
Relative position of object 0: 0.333976 0.161249 -0.010650
Relative orientation of object 0: -0.014143 -0.035246 0.999279 0.763658
```

Output:

```
- Color 1/1: 0.920000 0.760000 0.300000
Recognized 2 objects.
Model of object 0: apple
Id of object 0: 161
Relative position of object 0: 0.313888 -0.007743 -0.031462
Relative orientation of object 0: -0.014145 -0.030217 0.999443 0.876067
Size of object 0: 0.100000 0.100000
Position of the object 0 on the camera image: 133 85
Size of the object 0 on the camera image: 71 70
- Color 1/1: 0.590000 0.750000 0.280000
Model of object 1: biscuit box
Id of object 1: 200
Relative position of object 1: 0.342610 0.155769 0.016507
Relative orientation of object 1: 0.711595 0.123033 -0.691734 2.889798
Size of object 1: 0.205965 0.393771
Position of the object 1 on the camera image: 83 64
Size of the object 1 on the camera image: 165 127
```

Berikut adalah penjelasan dari koding tersebut:

Header dan Definisi

1.Header File: Kode ini mengimpor beberapa header file yang diperlukan untuk fungsionalitas input-output, kamera, pengenalan objek, motor, dan robot di Webots.

```
#include <stdio.h>
#include <webots/camera.h>
#include <webots/camera_recognition_object.h>
#include <webots/motor.h>
#include <webots/robot.h>
```

2.Konstanta: Mendefinisikan kecepatan motor dan langkah waktu untuk simulasi.

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

Fungsi Utama main

1.Inisialisasi Variabel dan Perangkat: Mendeklarasikan tag perangkat untuk kamera dan motor serta variabel lainnya.

```
WbDeviceTag camera, left_motor, right_motor;
int i, j;

wb_robot_init();
```

2. Inisialisasi Kamera dan Pengaktifan Pengenalan: Mengambil perangkat kamera, mengaktifkan kamera dan fitur pengenalan objek.

```
/* Get the camera device, enable it and the recognition */
camera = wb_robot_get_device("camera");
wb_camera_enable(camera, TIME_STEP);
wb_camera_recognition_enable(camera, TIME_STEP);
```

3.Inisialisasi Motor: Mengambil perangkat motor dan mengatur posisi target ke tak terbatas (untuk kontrol kecepatan).

```
/* get a handler to the motors and set target position to infinity (speed control). */
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);
```

4. Mengatur Kecepatan Motor: Mengatur kecepatan motor untuk membuat robot bergerak.

```
/* Set the motors speed */
wb_motor_set_velocity(left_motor, -SPEED);
wb_motor_set_velocity(right_motor, SPEED);
```

Loop Utama

1.Loop Utama Simulasi: Loop utama yang akan dijalankan selama simulasi aktif.

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

2. Mendapatkan Jumlah Objek yang Dikenali: Mengambil jumlah objek yang dikenali oleh kamera dan mencetaknya.

```
  /* Get current number of object recognized */
  int number_of_objects = wb_camera_recognition_get_number_of_objects(camera);
  printf("\nRecognized %d objects.\n", number_of_objects);
```

3. Mengambil Informasi Objek yang Dikenali: Mengambil informasi dari objek yang dikenali oleh kamera dan mencetak detailnya.

```
  /* Get and display all the objects information */
  const WbCameraRecognitionObject *objects =
wb_camera_recognition_get_objects(camera);
  for (i = 0; i < number_of_objects; ++i) {
    printf("Model of object %d: %s\n", i, objects[i].model);
    printf("Id of object %d: %d\n", i, objects[i].id);
    printf("Relative position of object %d: %lf %lf %lf\n", i, objects[i].position[0],
objects[i].position[1],
           objects[i].position[2]);
    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]);
    printf("Size of object %d: %lf %lf\n", i, objects[i].size[0], objects[i].size[1]);
    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]);
    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]);
  }
```

Pembersihan dan Keluar

1.Pembersihan: Membersihkan semua sumber daya sebelum keluar dari program.

```
wb_robot_cleanup();
```

2.Pengembalian Nilai: Mengembalikan nilai 0 sebagai indikator bahwa program berakhir dengan sukses.

```
return 0;
```

Penjelasan Singkat:

- **Inisialisasi dan Pengaturan**: Robot diinisialisasi, kamera diaktifkan dengan fitur pengenalan objek, dan motor diatur untuk kontrol kecepatan.
- **Loop Utama**: Dalam loop utama, robot secara berkala mengambil gambar dari kamera, menganalisis objek yang dikenali, dan mencetak informasi tentang objek tersebut, termasuk model, ID, posisi relatif, orientasi, ukuran, posisi pada gambar kamera, ukuran pada gambar kamera, dan warna objek.
- **Kontrol Motor**: Kecepatan motor diatur untuk membuat robot bergerak, meskipun dalam contoh ini, robot hanya berputar di tempat (salah satu motor bergerak maju dan yang lain bergerak mundur).
- **Output Informasi**: Informasi tentang objek yang dikenali ditampilkan di terminal, memberikan detail lengkap tentang setiap objek yang terdeteksi oleh kamera.

Proses ini diulang dalam loop utama hingga simulasi dihentikan.

Kode Simulasi 1 :

```
/*
 * Copyright 1996-2023 Cyberbotics Ltd.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *     https://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

/*
 * Description:  An example of use of a camera device.
 */

#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>

#define ANSI_COLOR_RED "\x1b[31m"
#define ANSI_COLOR_GREEN "\x1b[32m"
#define ANSI_COLOR_YELLOW "\x1b[33m"
#define ANSI_COLOR_BLUE "\x1b[34m"
#define ANSI_COLOR_MAGENTA "\x1b[35m"
#define ANSI_COLOR_CYAN "\x1b[36m"
#define ANSI_COLOR_RESET "\x1b[0m"

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

int main() {
  WbDeviceTag camera, left_motor, right_motor;
  int width, height;
  int pause_counter = 0;
  int left_speed, right_speed;
  int i, j;
  int red, blue, green;
  const char *color_names[3] = {"red", "green", "blue"};
  const char *ansi_colors[3] = {ANSI_COLOR_RED, ANSI_COLOR_GREEN,
ANSI_COLOR_BLUE};
  const char *filenames[3] = {"red_blob.png", "green_blob.png", "blue_blob.png"};
  enum BLOB_TYPE current_blob;

  wb_robot_init();

  const int time_step = wb_robot_get_basic_time_step();

  /* Get the camera device, enable it, and store its width and height */
  camera = wb_robot_get_device("camera");
  wb_camera_enable(camera, time_step);
  width = wb_camera_get_width(camera);
  height = wb_camera_get_height(camera);

  /* get a handler to the motors and set target position to infinity (speed control). */
  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);

  /* Main loop */
  while (wb_robot_step(time_step) != -1) {
    /* Get the new camera values */
    const unsigned char *image = wb_camera_get_image(camera);

    /* Decrement the pause_counter */
    if (pause_counter > 0)
      pause_counter--;

    /*
     * Case 1
     * A blob was found recently
     * The robot waits in front of it until pause_counter
     * is decremented enough
     */
    if (pause_counter > 640 / time_step) {
      left_speed = 0;
      right_speed = 0;
    }
    /*
     * Case 2
```

```
 * A blob was found quite recently
 * The robot begins to turn but don't analyse the image for a while,
 * otherwise the same blob would be found again
 */
else if (pause_counter > 0) {
  left_speed = -SPEED;
  right_speed = SPEED;
}
/*
 * Case 3
 * The robot turns and analyse the camera image in order
 * to find a new blob
 */
else if (!image) {  // image may be NULL if Robot.synchronization is FALSE
  left_speed = 0;
  right_speed = 0;
} else {  // pause_counter == 0
  /* Reset the sums */
  red = 0;
  green = 0;
  blue = 0;

  /*
   * Here we analyse the image from the camera. The goal is to detect a
   * blob (a spot of color) of a defined color in the middle of our
   * screen.
   * In order to achieve that we simply parse the image pixels of the
   * center of the image, and sum the color components individually
   */
  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);
    }
  }

  /*
   * If a component is much more represented than the other ones,
   * a blob is detected
   */
  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;

      /*
       * Case 3a
       * No blob is detected
       * the robot continues to turn
       */
      if (current_blob == NONE) {
        left_speed = -SPEED;
        right_speed = SPEED;
      }
      /*
       * Case 3b
       * A blob is detected
       * the robot stops, stores the image, and changes its state
       */
      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);
        // compute the file path in the user directory
        char *filepath;
#ifdef _WIN32
        const char *user_directory =
wbu_system_short_path(wbu_system_getenv("USERPROFILE"));
        filepath = (char *)malloc(strlen(user_directory) + 16);
        strcpy(filepath, user_directory);
        strcat(filepath, "\\");
#else
        const char *user_directory = wbu_system_getenv("HOME");
        filepath = (char *)malloc(strlen(user_directory) + 16);
        strcpy(filepath, user_directory);
        strcat(filepath, "/");
#endif
        strcat(filepath, filenames[current_blob]);
        wb_camera_save_image(camera, filepath, 100);
        free(filepath);
        pause_counter = 1280 / time_step;
      }
    }

    /* Set the motor speeds. */
    wb_motor_set_velocity(left_motor, left_speed);
    wb_motor_set_velocity(right_motor, right_speed);
  }

  wb_robot_cleanup();

  return 0;
}
```

Kode simulasi 2 :

```
/*
 * Copyright 1996-2023 Cyberbotics Ltd.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *     https://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

/*
 * Description:  An example of use of a camera device with recognition capability.
 */

#include <stdio.h>
#include <webots/camera.h>
#include <webots/camera_recognition_object.h>
#include <webots/motor.h>
#include <webots/robot.h>

#define SPEED 1.5
#define TIME_STEP 64

int main() {
  WbDeviceTag camera, left_motor, right_motor;
  int i, j;
  wb_robot_init();

  /* Get the camera device, enable it and the recognition */
  camera = wb_robot_get_device("camera");
  wb_camera_enable(camera, TIME_STEP);
  wb_camera_recognition_enable(camera, TIME_STEP);

  /* get a handler to the motors and set target position to infinity (speed control). */
  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);

  /* Set the motors speed */
  wb_motor_set_velocity(left_motor, -SPEED);
  wb_motor_set_velocity(right_motor, SPEED);

  /* Main loop */
  while (wb_robot_step(TIME_STEP) != -1) {
    /* Get current number of object recognized */
    int number_of_objects = wb_camera_recognition_get_number_of_objects(camera);
    printf("\nRecognized %d objects.\n", number_of_objects);

    /* Get and display all the objects information */
    const WbCameraRecognitionObject *objects =
wb_camera_recognition_get_objects(camera);
    for (i = 0; i < number_of_objects; ++i) {
      printf("Model of object %d: %s\n", i, objects[i].model);
      printf("Id of object %d: %d\n", i, objects[i].id);
      printf("Relative position of object %d: %lf %lf %lf\n", i, objects[i].position[0],
objects[i].position[1],
             objects[i].position[2]);
      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]);
      printf("Size of object %d: %lf %lf\n", i, objects[i].size[0], objects[i].size[1]);
      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]);
      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]);
    }
  }

  wb_robot_cleanup();

  return 0;
}
```