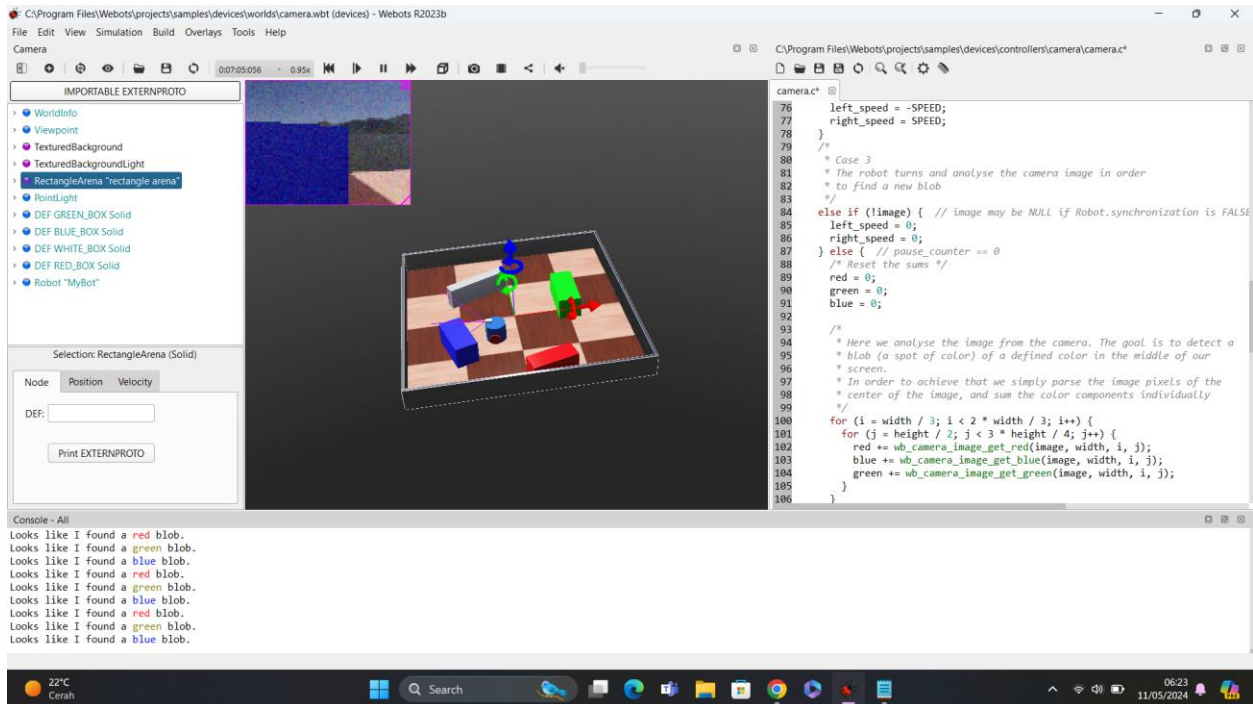


NAMA : ENRICO ANDRESON PATTIPEILOHY

NIM : 1103204206

Demo kamera deteksi warna



```
#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_BLUE "\x1b[34m"
```

```
#define ANSI_COLOR_RESET "\x1b[0m"
```

```
#define SPEED 4
```

```

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

int main() {

    // Deklarasi variabel dan perangkat yang dibutuhkan

    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};

    enum BLOB_TYPE current_blob;

    // Inisialisasi robot

    wb_robot_init();

    const int time_step = wb_robot_get_basic_time_step();

    // Mengakses dan mengaktifkan kamera

    camera = wb_robot_get_device("camera");

    wb_camera_enable(camera, time_step);

    width = wb_camera_get_width(camera);

    height = wb_camera_get_height(camera);

    // Mendapatkan akses ke motor dan mengatur posisi target ke tak terhingga (kontrol kecepatan)

    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

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

        // Mendapatkan citra terbaru dari kamera

        const unsigned char *image = wb_camera_get_image(camera);

```

```

// Mengurangi nilai pause_counter
if (pause_counter > 0)
    pause_counter--;

// Kasus 1: Blob ditemukan baru-baru ini, robot menunggu di depannya sampai pause_counter berkurang cukup
if (pause_counter > 640 / time_step) {
    left_speed = 0;
    right_speed = 0;
}

// Kasus 2: Blob ditemukan agak baru-baru ini, robot mulai berbelok tetapi tidak menganalisis citra untuk
// sementara waktu
else if (pause_counter > 0) {
    left_speed = -SPEED;
    right_speed = SPEED;
}

// Kasus 3: Robot berputar dan menganalisis citra dari kamera untuk menemukan blob baru
else if (!image) {
    left_speed = 0;
    right_speed = 0;
} else { // pause_counter == 0

    // Mereset jumlah warna
    red = 0;
    green = 0;
    blue = 0;

    // Menganalisis citra dari kamera untuk mendeteksi blob warna di tengah-tengah citra
    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);
        }
    }
}

```

```

// Jika komponen warna lebih dominan dari yang lain, blob terdeteksi
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;

// Kasus 3a: Tidak ada blob yang terdeteksi, robot terus berputar
if (current_blob == NONE) {
    left_speed = -SPEED;
    right_speed = SPEED;
}

// Kasus 3b: Blob terdeteksi, robot berhenti, menyimpan citra, dan mengubah keadaan untuk sementara waktu
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);

    // Menyimpan citra ke file
    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;
}
}

// Mengatur kecepatan motor

wb_motor_set_velocity(left_motor, left_speed);

wb_motor_set_velocity(right_motor, right_speed);
}

// Membersihkan sumber daya

wb_robot_cleanup();

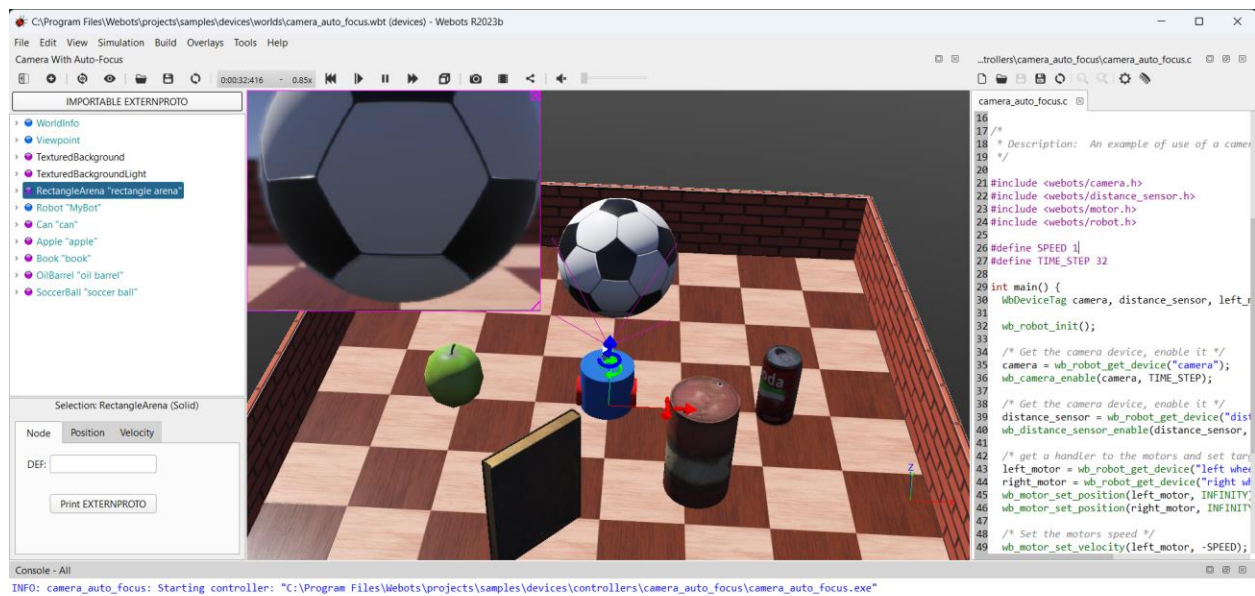
return 0;
}

```

- Inisialisasi: Program melakukan inisialisasi robot dan perangkat yang diperlukan seperti kamera dan motor.
- Loop Utama: Program memasuki loop utama dimana robot beroperasi secara terus menerus.
- Pembacaan Citra: Citra dibaca dari kamera untuk analisis lebih lanjut.
- Analisis Citra: Citra dianalisis untuk mendeteksi blob warna di tengah-tengah citra.
- Tanggapan terhadap Deteksi Blob: Berdasarkan hasil analisis citra, program memberikan respons yang sesuai. Jika tidak ada blob yang terdeteksi, robot akan terus berputar. Jika blob terdeteksi, robot akan berhenti, menyimpan citra, dan mengubah keadaan untuk sementara waktu.
- Pengaturan Kecepatan Motor: Program mengatur kecepatan motor berdasarkan respons yang diberikan.
- Pembersihan: Setelah selesai, program membersihkan sumber daya yang digunakan.

program ini mengimplementasikan kontrol sederhana berdasarkan analisis citra dari kamera untuk menanggapi deteksi blob warna di lingkungan simulasi robot.

Webots device camera auto fokus



```
#include <webots/camera.h>
```

```
#include <webots/distance_sensor.h>
```

```
#include <webots/motor.h>
```

```
#include <webots/robot.h>
```

```
#define SPEED 1
```

```
#define TIME_STEP 32
```

```
int main() {
```

```
    // Deklarasi variabel dan perangkat yang dibutuhkan
```

```
    WbDeviceTag camera, distance_sensor, left_motor, right_motor;
```

```
    // Inisialisasi robot
```

```
    wb_robot_init();
```

```
    /* Get the camera device, enable it */
```

```
    // Mendapatkan perangkat kamera dan mengaktifkannya
```

```
    camera = wb_robot_get_device("camera");
```

```
    wb_camera_enable(camera, TIME_STEP);
```

```
    /* Get the distance sensor device, enable it */
```

```
    // Mendapatkan perangkat sensor jarak dan mengaktifkannya
```

```

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

/* Get a handler to the motors and set target position to infinity (speed control) */
// Mendapatkan akses ke motor dan mengatur posisi target ke tak terhingga (kontrol kecepatan)
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);

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

/* Main loop */
// Loop utama
while (wb_robot_step(TIME_STEP) != -1) {
    // Mendapatkan jarak objek dari sensor jarak
    const double object_distance = wb_distance_sensor_get_value(distance_sensor) / 1000;
    // Mengatur jarak fokus kamera berdasarkan jarak objek
    wb_camera_set_focal_distance(camera, object_distance);
}

// Membersihkan sumber daya
wb_robot_cleanup();

return 0;
}

```

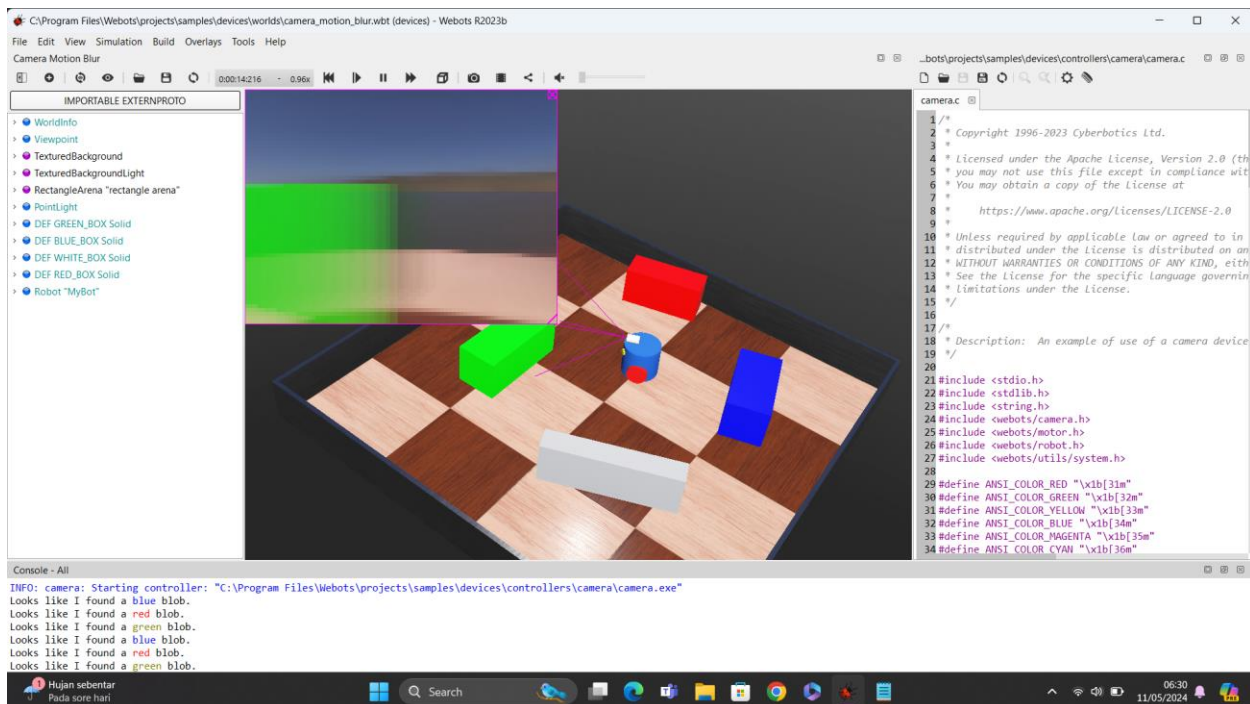
Inisialisasi: Program melakukan inisialisasi robot dan perangkat yang diperlukan seperti kamera, sensor jarak, dan motor.

- Loop Utama: Program memasuki loop utama dimana robot beroperasi secara terus menerus.

- Pengaturan Perangkat: Program mendapatkan akses dan mengaktifkan perangkat seperti kamera dan sensor jarak.
- Pengaturan Kecepatan Motor: Kecepatan motor diatur untuk menggerakkan robot.
- Pembacaan Sensor: Data jarak dari sensor jarak dibaca untuk analisis lebih lanjut.
- Kontrol Fokus Kamera: Jarak fokus kamera diatur berdasarkan data yang diperoleh dari sensor jarak.
- Pembersihan: Setelah selesai, program membersihkan sumber daya yang digunakan.

program ini menggunakan sensor jarak untuk mengatur fokus kamera berdasarkan jarak objek yang terdeteksi, sambil menggerakkan robot menggunakan motor.

Webots device camera motion blur



```
#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() {

    // Deklarasi variabel dan perangkat yang dibutuhkan

    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;

    // Inisialisasi robot

    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 */

    // Mendapatkan perangkat kamera, mengaktifkannya, dan menyimpan lebar dan tingginya

    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) */
// Mendapatkan akses ke motor dan mengatur posisi target ke tak terhingga (kontrol kecepatan)
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 */
// Loop utama
while (wb_robot_step(time_step) != -1) {
    /* Get the new camera values */
    // Mendapatkan nilai baru dari kamera
    const unsigned char *image = wb_camera_get_image(camera);

    /* Decrement the pause_counter */
    // Mengurangi nilai 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
    */

    // Kasus 1: Blob ditemukan baru-baru ini, robot menunggu di depannya sampai pause_counter berkurang cukup
    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

*/

```

// Kasus 2: Blob ditemukan agak baru-baru ini, robot mulai berbelok tetapi tidak menganalisis citra untuk sementara waktu

```

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

*/

```

// Kasus 3: Robot berputar dan menganalisis citra dari kamera untuk menemukan blob baru

```

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 */

    // Mengatur ulang jumlah

    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

```

```

*/
// Di sini kita menganalisis citra dari kamera. Tujuannya adalah mendeteksi
// blob (titik warna) dari warna tertentu di tengah layar.
// Untuk mencapainya, kita hanya mengurai piksel citra dari
// tengah citra, dan menjumlahkan komponen warnanya secara individual
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
*/
// Jika satu komponen jauh lebih dominan dari yang lain,
// blob terdeteksi
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

```

```

*/
// Kasus 3a: Tidak ada blob yang terdeteksi, robot terus berputar
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
*/
// Kasus 3b: Blob terdeteksi, robot berhenti, menyimpan citra, dan mengubah keadaan untuk sementara waktu
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
    // menghitung jalur file di direktori pengguna
    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
#endif

```

```

    strcat(filepath, filenames[current_blob]);

    wb_camera_save_image(camera, filepath, 100);

    free(filepath);

    pause_counter = 1280 / time_step;
}
}

/* Set the motor speeds */

// Mengatur kecepatan motor
wb_motor_set_velocity(left_motor, left_speed);
wb_motor_set_velocity(right_motor, right_speed);
}

// Membersihkan sumber daya
wb_robot_cleanup();

return 0;
}

```

- Inisialisasi: Program melakukan inisialisasi robot dan perangkat yang diperlukan seperti kamera dan motor.
- Loop Utama: Program memasuki loop utama dimana robot beroperasi secara terus menerus.
- Pembacaan Citra: Citra dibaca dari kamera untuk analisis lebih lanjut.
- Analisis Citra: Citra dianalisis untuk mendeteksi blob warna di tengah-tengah citra.
- Tanggapan terhadap Deteksi Blob: Berdasarkan hasil analisis citra, program memberikan respons yang sesuai. Jika tidak ada blob yang terdeteksi, robot akan terus berputar. Jika blob terdeteksi, robot akan berhenti, menyimpan citra, dan mengubah keadaan untuk sementara waktu.
- Pembersihan: Setelah selesai, program membersihkan sumber daya yang digunakan.

program ini mengimplementasikan kontrol sederhana berdasarkan analisis citra dari kamera untuk menanggapi deteksi blob warna di lingkungan simulasi robot.