SYMBIOSIS INSTITUTE OF TECHNOLOGY, PUNE
DEPARTMENT OF ROBOTICS AND AUTOMATION ENGINEERING

ACADEMIC YEAR: 202425 SEM: V

ASSIGNMENT NO: 5

NAME:Akshaj Chainani
PRN: 22070127008
BATCH: RA1

Aim:

To write a controller program for an ePuck or a four-wheeled robot that allows the robot to detect objects using a camera sensor. The robot should stop or change direction upon detecting an object based on image processing.

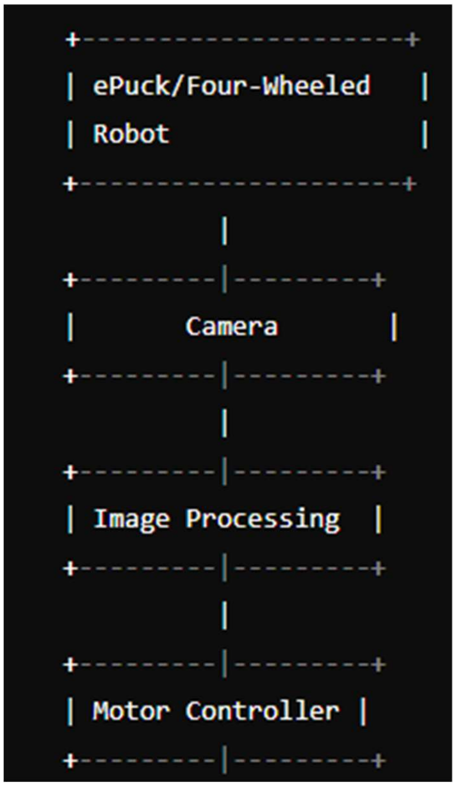
Apparatus:

Webots Simulator – A multi-robot simulator.
ePuck Robot / Four-Wheeled Robot – The physical model of the robot used for simulation.
Camera Sensor – Mounted on the robot for object detection.
Motor Actuators – For movement of wheels (left and right motors).
Keyboard – For controlling the robot using arrow keys or WASD.
Laptop/PC – To run the Webots simulation and controller code.

Theory

1. Object Detection with Image Processing:
Object detection is a process of identifying objects within an image or a video stream. The camera mounted on the robot captures real-time images of its environment. Image processing techniques, such as analyzing pixel intensity and color values (RGB), are used to detect objects.
In this context, the camera captures the RGB values of pixels in the image. The sum of these values is used as an intensity measure to determine the presence of an object. A threshold value is defined to separate object pixels from the background.
2. Robot Control:
The robot is equipped with two wheels controlled by motor actuators (left and right). These motors can be set to different speeds to allow the robot to move forward, backward, and turn left or right.
Keyboard input is used to control the movement, while the camera continuously monitors for objects in the environment.
3. Key Concepts:
Webots API: A C programming interface that provides control over robot hardware such as motors, sensors, and cameras.
Motor Control: Speed is set to control direction and movement.
Image Processing: Using RGB values of camera images to detect objects.
Object Detection Threshold: Defined based on intensity sum (R+G+B) to differentiate objects from the background.

Block Diagram:



```
+--------------------+
| ePuck/Four-Wheeled  |
| Robot               |
+--------------------+
          |
+---------|---------+
|       Camera      |
+---------|---------+
          |
+---------|---------+
| Image Processing  |
+---------|---------+
          |
+---------|---------+
| Motor Controller |
+---------|---------+
```

Program Code

~~~
```c
#include <webots/robot.h>
#include <webots/motor.h>
#include <webots/keyboard.h>
#include <webots/camera.h>
#include <stdio.h>

#define TIME_STEP 64
#define MAX_SPEED 6.28
#define INTENSITY_THRESHOLD 200

// Function to detect objects
int detect_object(const unsigned char *image, int width, int height) {
  int object_detectdwed = 0;

  // Loop through all pixels in the image
  for (int y = 0; y < height; y++) {
    for (int x = 0; x < width; x++) {
      int r = (int)wb_camera_image_get_red(image, width, x, y);
      int g = (int)wb_camera_image_get_green(image, width, x, y);
      int b = (int)wb_camera_image_get_blue(image, width, x, y);
~~~

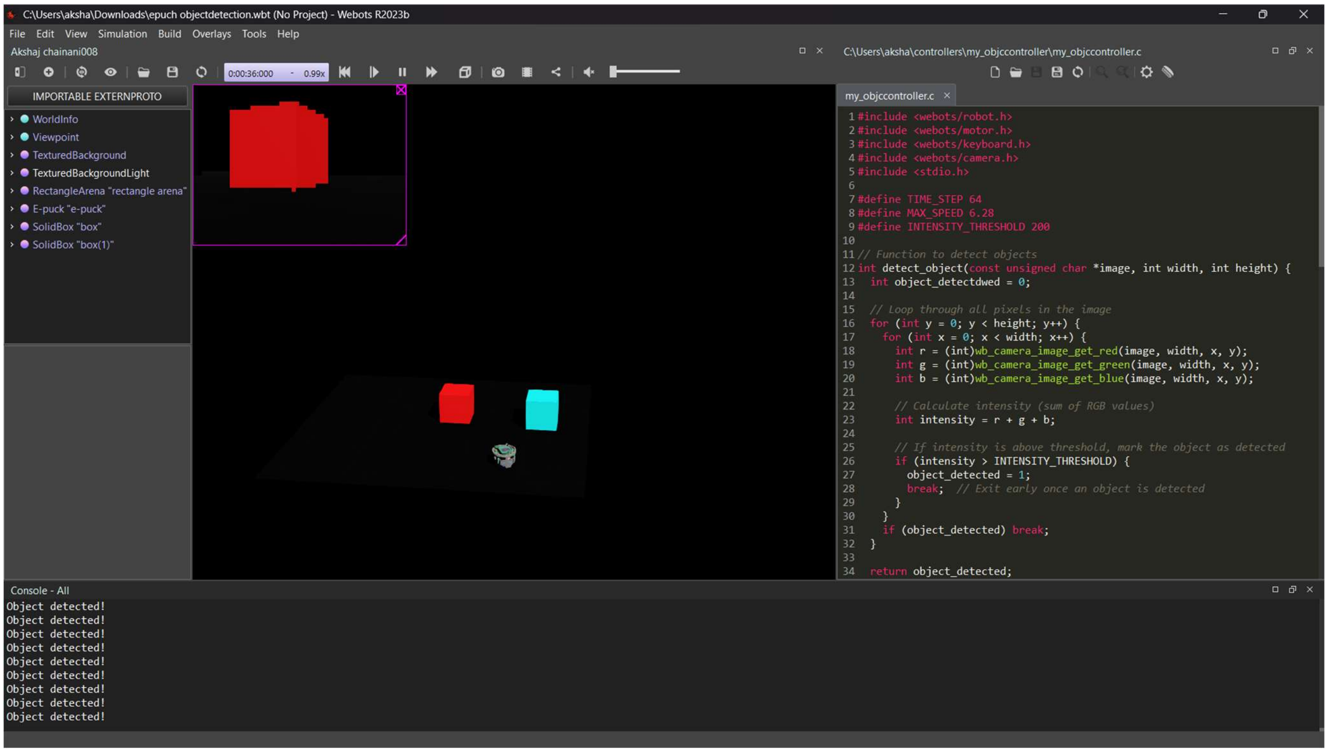
```
      // Calculate intensity (sum of RGB values)
      int intensity = r + g + b;

      // If intensity is above threshold, mark the object as detected
      if (intensity > INTENSITY_THRESHOLD) {
        object_detected = 1;
        break;  // Exit early once an object is detected
      }
    }
    if (object_detected) break;
  }

  return object_detected;
}

int main(int argc, char **argv) {
  // Initialize Webots API
  wb_robot_init();

  // Get motor devices
  WbDeviceTag left_motor = wb_robot_get_device("left wheel motor");
  WbDeviceTag right_motor = wb_robot_get_device("right wheel motor");

  // Set motors to velocity control mode
  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);

  // Enable keyboard input
  wb_keyboard_enable(TIME_STEP);

  // Enable the camera
  WbDeviceTag camera = wb_robot_get_device("camera");
  wb_camera_enable(camera, TIME_STEP);
  int width = wb_camera_get_width(camera);
  int height = wb_camera_get_height(camera);

  // Main loop
  while (wb_robot_step(TIME_STEP) != -1) {
    // Get keyboard input
    int key = wb_keyboard_get_key();

    // Control motors based on key press (arrow keys and WASD keys)
    if (key == WB_KEYBOARD_UP || key == 'W') {
      wb_motor_set_velocity(left_motor, MAX_SPEED);
      wb_motor_set_velocity(right_motor, MAX_SPEED);
    } else if (key == WB_KEYBOARD_DOWN || key == 'S') {
      wb_motor_set_velocity(left_motor, -MAX_SPEED);
      wb_motor_set_velocity(right_motor, -MAX_SPEED);
    } else if (key == WB_KEYBOARD_LEFT || key == 'A') {
      wb_motor_set_velocity(left_motor, -MAX_SPEED / 2);
      wb_motor_set_velocity(right_motor, MAX_SPEED / 2);
    } else if (key == WB_KEYBOARD_RIGHT || key == 'D') {
      wb_motor_set_velocity(left_motor, MAX_SPEED / 2);
      wb_motor_set_velocity(right_motor, -MAX_SPEED / 2);
    } else {
      wb_motor_set_velocity(left_motor, 0.0);
      wb_motor_set_velocity(right_motor, 0.0);
    }

    // Get camera image and detect objects
    const unsigned char *image = wb_camera_get_image(camera);
    if (detect_object(image, width, height)) {
      printf("Object detected!\n");
    } else {
      printf("No object detected.\n");
    }
  }

  // Cleanup Webots API
  wb_robot_cleanup();

  return 0;
}```
```

WBT file:

```
#VRML_SIM R2023b utf8

EXTERNPROTO
    "https://raw.githubusercontent.com/cyberbotics/webots/R2023b/projects/objects/backgrounds/protos/TexturedBackground.proto"
EXTERNPROTO
    "https://raw.githubusercontent.com/cyberbotics/webots/R2023b/projects/objects/backgrounds/protos/TexturedBackgroundLight.proto"
EXTERNPROTO
    "https://raw.githubusercontent.com/cyberbotics/webots/R2023b/projects/objects/floors/protos/RectangleArena.proto"
EXTERNPROTO
    "https://raw.githubusercontent.com/cyberbotics/webots/R2023b/projects/robots/gctronic/e-puck/protos/E-puck.proto"
EXTERNPROTO
    "https://raw.githubusercontent.com/cyberbotics/webots/R2023b/projects/objects/solids/protos/SolidBox.proto"
EXTERNPROTO
    "https://raw.githubusercontent.com/cyberbotics/webots/R2023b/projects/appearances/protos/Parquetry.proto"
```

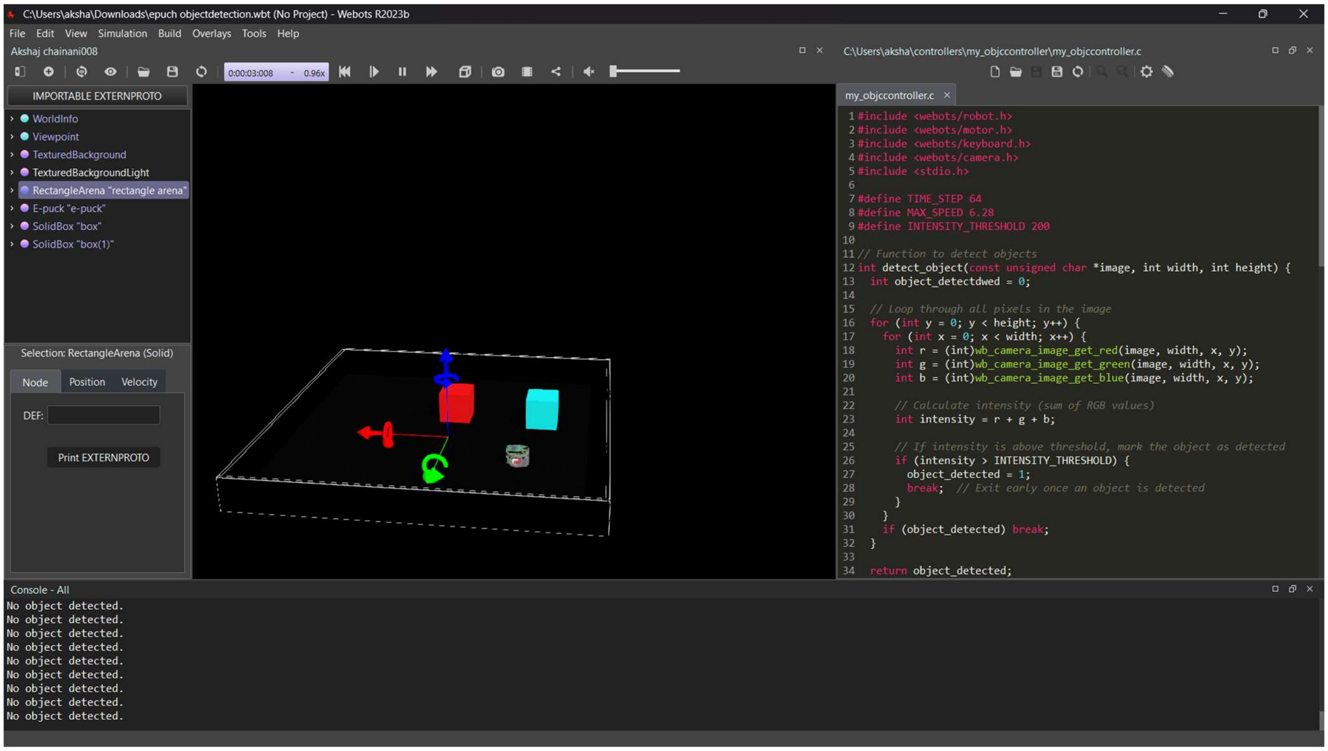
```
EXTERNPROTO "https://raw.githubusercontent.com/cyberbotics/webots/R2023b/projects/appearances/protos/BrushedAluminium.proto"

WorldInfo {
  title "Akshaj chainani008"
}
Viewpoint {
  orientation 0.15484665892775729 0.1740176221923562 -0.9724918402667598 1.4820670023134566
  position -0.4707697185697835 2.2124172855552793 1.0898386735232568
}
TexturedBackground {
  texture "noon_stormy_empty"
  skybox FALSE
}
TexturedBackgroundLight {
}
RectangleArena {
  floorAppearance Parquetry {
    type "chequered"
    colorOverride 0 0 0
    textureTransform TextureTransform {
    }
  }
  wallAppearance BrushedAluminium {
    colorOverride 0 0 0
    textureTransform TextureTransform {
    }
    IBLStrength 0
  }
}
E-puck {
  hidden position_0_0 24.013386579127427
  hidden position_0_1 39.27363839911135
  hidden linearVelocity_0 -1.601889214441066e-16 -8.453626491053946e-15 1.0421293457814803e-15
  hidden angularVelocity_0 -1.0225754789626036e-14 -2.941828992967705e-15 -3.542942049380251e-15
  hidden rotation_1 0 -1 0 1.1193546495909183
  hidden linearVelocity_1 -4.288471187117767e-16 -1.140232879051753e-14 1.3424822938305199e-16
  hidden angularVelocity_1 4.906402797753751e-13 -2.209328978697124e-14 -3.630029715007651e-15
  hidden rotation_2 0 1 0 1.5745265560338297
  hidden linearVelocity_2 -9.557117746426761e-17 -5.413512558555329e-15 -1.3833994767579334e-15
  hidden angularVelocity_2 6.13281777092501e-13 -2.3049662216398075e-14 -3.6292751723981164e-15
  translation -0.2397513500162066 0.1503414653141316 -6.396199578766713e-05
  rotation 5.216801992817007e-12 2.6416872661740185e-10 1 3.1028464379964618
  controller "my_objccontroller"
}
SolidBox {
  translation 1.219635284080379e-18 -0.16 0.0496076
  rotation -0.217295380716767 2.6072195038756203e-15 -0.9761058946237109
        1.96196366461998e-16
  size 0.1 0.1 0.1
  appearance PBRAppearance {
    baseColor 1 0 0
    baseColorMap ImageTexture {
      url [
        ""
      ]
    }
    roughness 0.5
    metalness 0
  }
  physics Physics {
  }
}
SolidBox {
  translation -0.2899999981570614 -0.13999999815706124 0.04960760000000001
  rotation 0.4007195223190498 -0.36674863973469807 0.8395947234738562
        1.7597537291982753e-16
  name "box(1)"
  size 0.1 0.1 0.1
  appearance PBRAppearance {
    baseColor 0 1 1
    baseColorMap ImageTexture {
      url [
        ""
      ]
    }
    roughness 0.5
    metalness 0
  }
  physics Physics {
  }
}
```

Snapshots


C:\Users\aksha\Downloads\epuch objectdetection.wbt (No Project) - Webots R2023b
File Edit View Simulation Build Overlays Tools Help
Akshaj chainani008
0:00:36:000 0.99x
IMPORTABLE EXTERNPROTO
WorldInfo
Viewpoint
TexturedBackground
TexturedBackgroundLight
RectangleArena "rectangle arena"
E-puck "e-puck"
SolidBox "box"
SolidBox "box(1)"
C:\Users\aksha\controllers\my_objccontroller\my_objccontroller.c
my_objccontroller.c
Console - All

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

#define TIME_STEP 64
#define MAX_SPEED 6.28
#define INTENSITY_THRESHOLD 200

// Function to detect objects
int detect_object(const unsigned char *image, int width, int height) {
  int object_detectdwed = 0;

  // Loop through all pixels in the image
  for (int y = 0; y < height; y++) {
    for (int x = 0; x < width; x++) {
      int r = (int)wb_camera_image_get_red(image, width, x, y);
      int g = (int)wb_camera_image_get_green(image, width, x, y);
      int b = (int)wb_camera_image_get_blue(image, width, x, y);

      // Calculate intensity (sum of RGB values)
      int intensity = r + g + b;

      // If intensity is above threshold, mark the object as detected
      if (intensity > INTENSITY_THRESHOLD) {
        object_detected = 1;
        break;  // Exit early once an object is detected
      }
    }
    if (object_detected) break;
  }

  return object_detected;
```

```
Object detected!
Object detected!
Object detected!
Object detected!
Object detected!
Object detected!
Object detected!
Object detected!
Object detected!
```


C:\Users\aksha\Downloads\epuch objectdetection.wbt (No Project) - Webots R2023b
File Edit View Simulation Build Overlays Tools Help
Akshaj chainani008
0:00:03:008 0.96x
IMPORTABLE EXTERNPROTO
WorldInfo
Viewpoint
TexturedBackground
TexturedBackgroundLight
RectangleArena "rectangle arena"
E-puck "e-puck"
SolidBox "box"
SolidBox "box(1)"
Selection: RectangleArena (Solid)
Node
Position
Velocity
DEF:
Print EXTERNPROTO
C:\Users\aksha\controllers\my_objccontroller\my_objccontroller.c
my_objccontroller.c
Console - All

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

#define TIME_STEP 64
#define MAX_SPEED 6.28
#define INTENSITY_THRESHOLD 200

// Function to detect objects
int detect_object(const unsigned char *image, int width, int height) {
  int object_detectdwed = 0;

  // Loop through all pixels in the image
  for (int y = 0; y < height; y++) {
    for (int x = 0; x < width; x++) {
      int r = (int)wb_camera_image_get_red(image, width, x, y);
      int g = (int)wb_camera_image_get_green(image, width, x, y);
      int b = (int)wb_camera_image_get_blue(image, width, x, y);

      // Calculate intensity (sum of RGB values)
      int intensity = r + g + b;

      // If intensity is above threshold, mark the object as detected
      if (intensity > INTENSITY_THRESHOLD) {
        object_detected = 1;
        break;  // Exit early once an object is detected
      }
    }
    if (object_detected) break;
  }

  return object_detected;
```

```
No object detected.
No object detected.
No object detected.
No object detected.
No object detected.
No object detected.
No object detected.
No object detected.
No object detected.
```

Learning Outcome:

Understanding of Object Detection: The student will learn how to detect objects using image processing techniques in real-time.

Robot Control Using Motors: Students will learn how to control motor actuators in Webots for various movements of a robot.

Keyboard Input Handling: Understanding the handling of user inputs for controlling robot motion.

Integration of Camera and Motors: Students will learn how to integrate camera sensors and motor actuators for autonomous behavior.