UNIVERSITY OF HERTFORDSHIRE

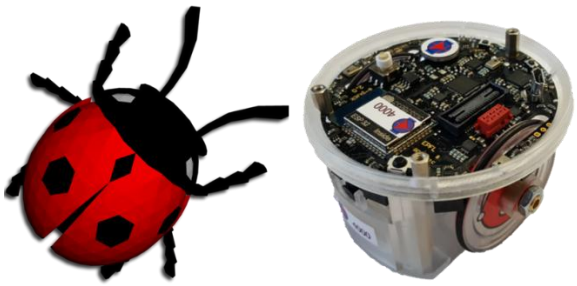
School of Engineering and Technology, PSB Academy

5FTC2068 Applied Robotics and AI Project

FT

# COURSEWORK 2

# Interim Report - Webots



By

**(SELVARAJAN SIRPY , 2431JLZO)**

**Interim Report: Webots Robot Search Operation (Part 1)**

## 1. Introduction

Robots can significantly improve safety, efficiency, and dependability whenever it is used in unfamiliar conditions environments. In this project, a **e-puck robot** is programmed to search and identify coloured blocks automatically using a enviroinment that I create using  using the **Webots** simulation. Three distinct coloured blocks—**red, blue, and green**—must be placed in the random place  for the robot to detect the object , and when it detects them, it must output the create a messages. This simulation demonstrates basic search algorithms and the use of camera sensors for colour based  object recognition.

## 2. Methodology

This project's methodology created with a methodical approach to robot simulation and programming using the Webots environment.

### 2.1 Environment Setup

The Webots simulation environment will be configured as per the assignment brief. This includes:

- **Robot:** Utilizing the **e-puck robot model**, known for its compact size and integrated sensors suitable for basic navigation and object detection tasks.


**Fig.1. E-puck robot model**

- **Blocks: Three distinct blocks – one red, one blue, and one green** – will be placed at random locations within the environment. Each of the blocks will have the length **0.3m (length) x 0.3m (width) x 0.1m (height)**. The random placement ensures the robot's search algorithm is robust and not dependent on fixed object positions.

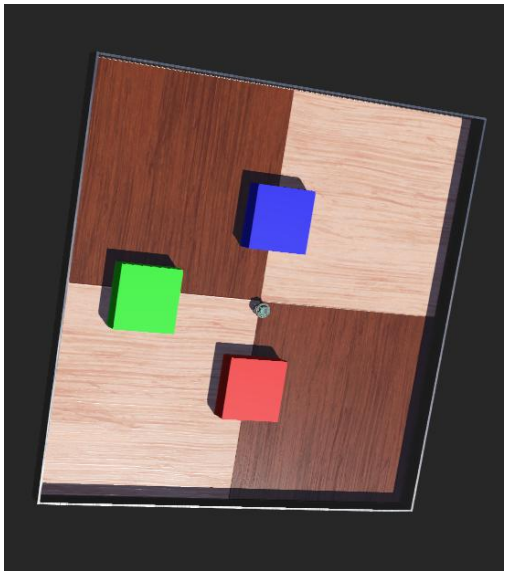


**Fig.2. 3 Blocks  in Webots Environment**

### 2.2 Robot Control and Perception

The e-puck robot's controller will be programmed in Python, leveraging Webots  API for sensor data acquisition and motor control.

- **Sensors:**
  - **Distance Sensors:** The e-puck's eight infrared distance sensors (located around its periphery) will be crucial for obstacle avoidance and basic navigation. Readings from these sensors that will inform the robot's immediate movement decisions, that make it preventing from  collisions with walls or the blocks themselves.
  - **Camera:** The robot's camera will be the primary sensor for color detection. The camera image will be processed to identify the presence of red, blue, or green pixels within its field of view.
- **Actuators:**
  - **Wheels:** The two differential drive wheels will be controlled to enable forward movement, turning, and stopping.

### 2.3 Search Strategy

A simple, yet effective, search strategy will be implemented:

- **Random Walk with Obstacle Avoidance:** The robot will primarily engage in a forward movement. When an obstacle is detected by the distance sensors, the robot will execute a turn (e.g., a random turn or a turn away from the obstacle) to avoid collision and continue its search. This ensures exploration of the environment.
- **Colour Detection Loop:** Continuously, within each simulation step, the camera image will be captured and analyse for the presence of the target colours.

### 2.4 Colour Detection Algorithm

The colour detection will involve processing the camera image data:

- The camera image will be obtained as an array of RGB pixel values.
- A simple thresholding approach will be used:
  - For **Red**: Pixels with a high red component and relatively low blue and green components will be considered.
  - For **Blue**: Pixels with a high blue component and relatively low red and green components will be considered.
  - For **Green**: Pixels with a high green component and relatively low red and blue components will be considered.
- A "dominant color" check will be performed to determine if a significant portion of the image (or a central region) matches one of the target colors, indicating a block has been found.

### 2.5 Response and State Management

Upon the first encounter of a specific colored block:

- **Message Display:** The robot will print a message to the Webots console indicating which color it has found (e.g., "I see red").
- **Encounter Summary:** After each message, the robot will provide a summary of all unique colored blocks encountered up to that point. This will be managed using boolean flags (e.g., found_red, found_blue, found_green) that are set to True once a block is detected for the first time.

## 3. Project Plan

This project will be executed in a phased approach, with Part 1 focusing on the core search and detection functionalities.

**Phase 1: Environment Setup and Basic Navigation (Current Focus - Part 1)**

- **Task 1.1:** Set up the Webots simulation environment with the e-puck robot and three colored blocks at random locations.
- **Task 1.2:** Implement basic forward movement and obstacle avoidance using distance sensors.
- **Task 1.3:** Configure and enable the e-puck's camera.
- **Task 1.4:** Develop the color detection logic to identify red, blue, and green blocks from camera input.
- **Task 1.5:** Implement the logic for displaying "I see [color]" messages upon first encounter.

- **Task 1.6:** Implement the state tracking for previously encountered blocks and display a summary after each new encounter.
- **Deliverable:** Interim Report (this document) including the controller code.

**Phase 2: Refinement and Advanced Search (Future Work)**

- **Task 2.1:** Optimize the search strategy for more efficient exploration (e.g., wall-following, spiral search).
- **Task 2.2:** Improve color detection robustness under varying lighting conditions or distances.
- **Task 2.3:** Implement a mechanism for the robot to stop or perform a specific action after finding all blocks.
- **Task 2.4:** Introduce additional environmental complexities or block types.

**Timeline (Part 1)**

| Task | Estimated Duration | Status |
|---|---|---|
| Environment Setup | 1 day | Completed |
| Basic Navigation | 1 day | Completed |
| Camera Integration | 0.5 day | Completed |
| Color Detection Logic | 1.5 days | Completed |
| Message & Summary Logic | 1 day | Completed |
| Report Generation | 1 day | Completed |

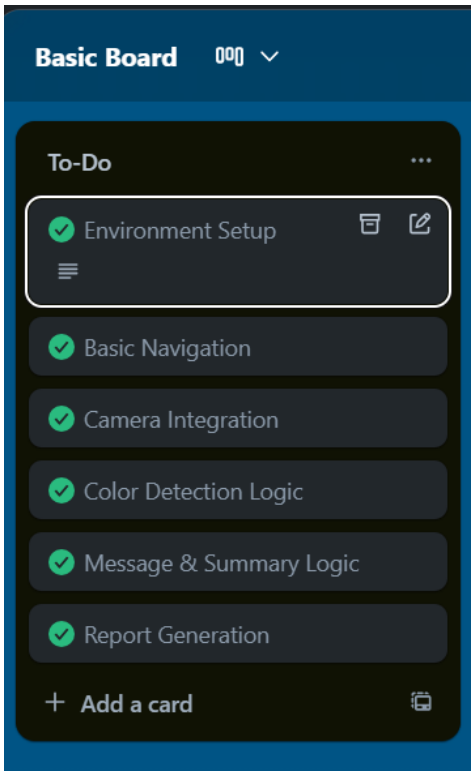



**Fig.3) Trello Board For Managing Task**

**4. Bibliography and References**


- Webots User Guide: https://cyberbotics.com/doc/guide/index
- e-puck Robot Documentation: https://www.cyberbotics.com/doc/guide/epuck?version=cyberbotics:R2019a

## 5. Appendix

The Python code for the e-puck robot controller, designed to be run within the Webots environment, is provided in the following section.

Webots Colab File : https://colab.research.google.com/drive/18bgsBPL9Gz2xeD7pRap5ZiRcnXG7Ydw3?usp=sharing

```
""" ARAIP obstacle avoidance and seeing RGB Colors """
from controller import Robot
#create a robot instance
robot=Robot()
print("Robot instance created!") #confirmation for robot created sucessfully
#timestep(how often the robot updates)
TIME_STEP= int(robot.getBasicTimeStep())
#motor names
left_motor= robot.getDevice('left wheel motor')
right_motor= robot.getDevice('right wheel motor')
#enable velocity control mode
left_motor.setPosition(float('inf'))
right_motor.setPosition(float('inf'))
#velocity control which runs the robot forever
left_motor.setVelocity(4.0)
right_motor.setVelocity(4.0)
#enable distance sensors
distance_sensors=[]
sensor_names= ['ps0', 'ps1', 'ps2', 'ps3', 'ps4', 'ps5', 'ps6', 'ps7']
for name in sensor_names:
    sensor= robot.getDevice(name)
    sensor.enable(TIME_STEP)
    distance_sensors.append(sensor)
#enable cameras
camera= robot.getDevice('camera')
camera.enable(TIME_STEP)
seen_color= set()
Obstacle_Thershold = 100
Turn_steps=25
Turn_counter=0
seen_colors = set() #keep track of colours already seen
#main loop:which runs until the simulation ends
while robot.step(TIME_STEP) !=-1:
    distances = [sensor.getValue() for sensor in distance_sensors]
    image=camera.getImage()
    width=camera.getWidth()
    height=camera.getHeight()
    x=width // 2
    y=height // 2
    #process sensor data and image
    r=camera.imageGetRed(image, width, x, y)
```

```
g=camera.imageGetGreen(image, width, x, y)
b=camera.imageGetBlue(image, width, x, y)

red_count = 0
green_count = 0
blue_count= 0
step= 5
detected_color = None
seen_color=[] # to show the color in order

if r > 100 and g < 80 and b < 80:
    detected_color = 'red'
elif g > 100 and r < 80 and b < 80:
    detected_color = 'green'
elif b > 100 and r < 80 and g < 80:
    detected_color = 'blue'

if detected_color and detected_color not in seen_colors:
    print(f"I see {detected_color}")
    seen_colors.add(detected_color)
    print("Summary: I have previously seen " + ", ".join(seen_colors))
    print(f"R={r}, G={g}, B={b}") #a debug print for seeing the camera RGB values

front_obstacle = any(sensor_value > 80 for sensor_value in distances)

if front_obstacle:
    print("Obstacle seen! Turning")
    is_turning = True
    Turn_counter = 0
    left_motor.setVelocity(4.0)
    right_motor.setVelocity(-4.0)
else:
    left_motor.setVelocity(5.0)
    right_motor.setVelocity(5.0)
```