University of Colorado - Boulder

CSCI 3303
Intro to Robotics — Fall 2024

e-Puck Robot State Machine Lab Report

Sam Walker

Nikolaus Correll

Friday, October 4, 2024

College of Engineering & Applied Science
UNIVERSITY OF COLORADO BOULDER

Introduction

This project demonstrates the use of a finite state machine (FSM) controller for an e-puck robot tasked with obstacle avoidance and specific turn maneuvers within a simulated Webots environment. The robot is placed in a controlled 3m x 3m area with two obstacles on opposite sides. The objective is for the robot to perform collision avoidance, make a 180-degree turn upon reaching the first obstacle, and then turn right upon reaching the second obstacle, before stopping.

This report includes a screenshot of the world setup, a drawing of the state machine, and code outlining the controller's behavior. A reflection discusses challenges faced during implementation, including state transitions and state management.

Simulation Setup

World Setup Screenshot

The environment consists of a 3m x 3m floor area with walls surrounding it. An e-puck robot is positioned at the center, with two obstacles: one in front of the robot, at a distance of at least 15cm, and the other positioned further than 15cm, directly behind the robot. The figure below shows the simulation setup:

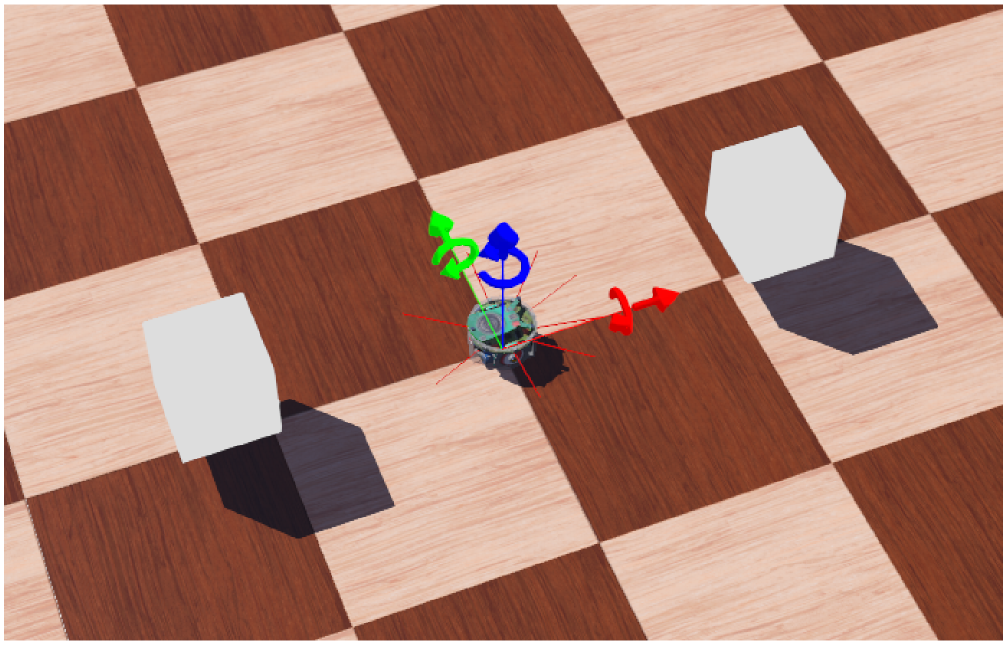


Figure 1: Webots Environment Showing e-Puck Robot and Obstacles

Finite State Machine (FSM)

State Machine Diagram

The FSM for this project is illustrated in the following state diagram. Each state and transition is labeled according to the implemented behavior, and the diagram reflects the use of counters to manage repetitive returns to the forward state.

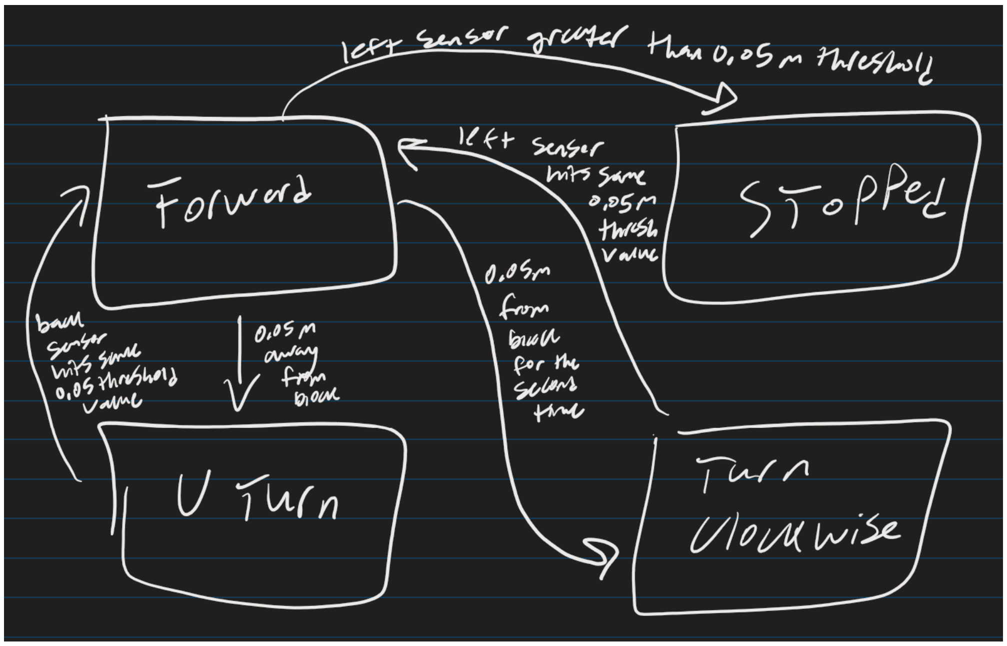
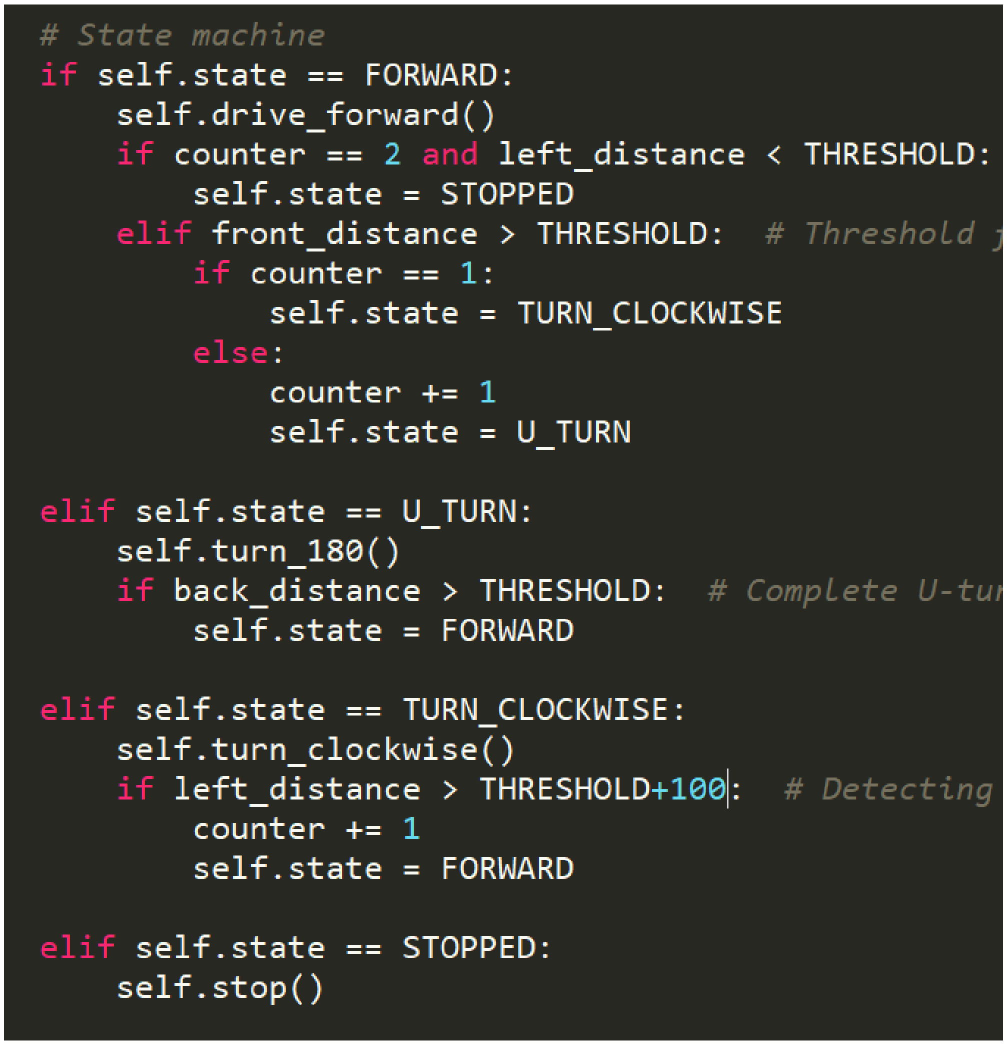

left sensor greater than 0.05m threshold
Forward
Stopped
left sensor hits same 0.05m thresh value
0.05m from block for the second time
back sensor hits same 0.05 threshold value
0.05m away from block
U Turn
Turn Clockwise

Figure 2: State Machine Diagram for e-Puck Controller

State Machine Code

The Python code implementing the FSM in Webots is provided below. Each state corresponds to a specific behavior, such as moving forward, making a U-turn, or turning right, with transitions occurring based on sensor feedback from the robot's proximity sensors.



```
# State machine
if self.state == FORWARD:
    self.drive_forward()
    if counter == 2 and left_distance < THRESHOLD:
        self.state = STOPPED
    elif front_distance > THRESHOLD:  # Threshold f
        if counter == 1:
            self.state = TURN_CLOCKWISE
        else:
            counter += 1
            self.state = U_TURN

elif self.state == U_TURN:
    self.turn_180()
    if back_distance > THRESHOLD:  # Complete U-tur
        self.state = FORWARD

elif self.state == TURN_CLOCKWISE:
    self.turn_clockwise()
    if left_distance > THRESHOLD+100:  # Detecting
        counter += 1
        self.state = FORWARD

elif self.state == STOPPED:
    self.stop()
```

Figure 3: Code Implementing the FSM for e-Puck Controller

Reflection on Implementation

This lab assignment provided valuable experience with finite state machines, sensor integration, and control logic for reactive robotic behavior. The primary challenge was managing the robot's repeated returns to the *Forward* state, particularly when executing sequences involving close transitions between behaviors. After several iterations, I employed counters to limit the re-entry into the *Forward* state as a workaround.

While this approach is not ideal for larger or more complex state machines, it was effective within the scope of this project. Counters provide a simplistic solution but lack scalability and are prone to failure if the robot's state transitions become more complex. In future implementations, a more robust solution, possibly with additional control variables, would better handle state dependencies.

Additional challenges included:

- **Sensor Threshold Tuning**: Setting appropriate threshold values for sensor activation was critical for successful state transitions. Fine-tuning these thresholds to accurately detect obstacles without premature transitions required multiple test runs and parameter adjustments.
- **Debugging Complex Behaviors**: Debugging involved tracking sensor values at each state to ensure correct behavior. The implementation of counters helped isolate and control transitions effectively within the limited complexity of this FSM.

Conclusion

In conclusion, the e-puck robot successfully completed the task by navigating through states defined within the FSM, handling collision avoidance, and performing specific turns based on proximity sensor feedback. Although counters were necessary to manage state transitions, this approach sufficed for the given task but revealed limitations in scalability for more intricate FSMs. Future work could include implementing more scalable state management strategies to enhance the controller's flexibility and reliability.

The project highlighted the importance of sensor threshold tuning and robust state management, providing hands-on experience with FSMs in robotic control.