Behavior-based Fire Alarm Robot

Aman Hogan-Bailey

Sai Karthik Reddy Muddana Laligari

Trieu Nguyen

2238-CSE-4360-001, Professor Manfred Huber

The University of Texas at Arlington

November 21st, 2023

Contents

Design

The chosen design emphasizes a balance between sensor coverage and the simplicity of a unicycle-type robot. The unicycle configuration allows for efficient maneuvering, while the placement of sensors optimizes the robot's ability to perceive and respond to its environment. This design is well-suited for applications that require a combination of collision avoidance, color detection, and distance measurement. The specific arrangement of touch, color, and ultrasonic sensors, along with the unicycle-type configuration, is tailored to enhance the robot's responsiveness and adaptability in various scenarios, aligning with the project's goals of intelligent navigation and behavior-based control.

Front Touch Sensor:

The front touch sensor is strategically placed to detect obstacles directly in front of the robot. This placement allows the robot to respond promptly when it encounters a wall or obstacle.

Left Touch Sensor:

The left touch sensor provides additional collision detection on the robot's left side. This feature enhances the robot's ability to navigate around obstacles and avoid collisions.

Color Sensor:

The color sensor is positioned in the front and pointing downward, allowing it to detect colors on the surface beneath the robot. This configuration is ideal for tasks such as fire detection, where the robot needs to identify specific colors, such as red.

Ultrasonic Sensor:

The ultrasonic sensor is located on the right side, enabling the robot to measure distances to objects on its right. This sensor enhances the robot's spatial awareness, particularly in scenarios involving navigation near walls.

Unicycle Configuration:

The unicycle-type robot with two wheels provides a simple and agile platform. This design choice facilitates quick and precise movements, making the robot responsive to the detected stimuli.

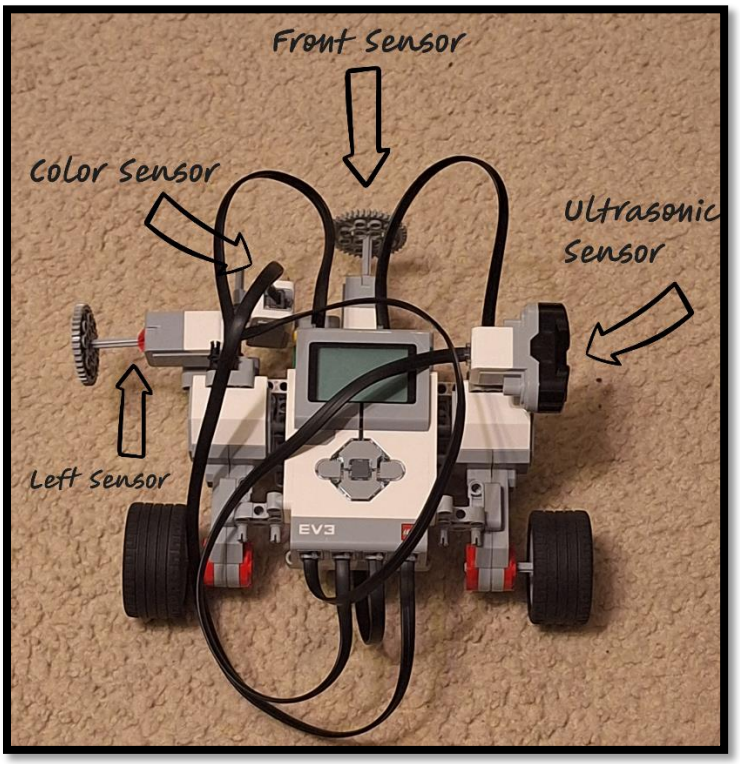


Figure 1: Final Robot Design

Movement Equation

The robot's translational movement was determined by the angle of rotation of the tires. To convert from a desired distance to the angle of rotation of the tires, we used known equations. Taking in a desired distance, we calculated the tire angle and multiplied this by the error factor to get a more accurate angle.

$$\theta = \left(\frac{X}{C}\right) 360,$$

$$\theta = \theta\gamma$$

$$\theta = angle\ of\ tire\ rotation\ [deg], \gamma = error\ factor, C = tire\ circumference$$

Turning Equation

The robot's translation movement was determined by the angle of rotation of the tires. To convert from a desired steering angle to a rotation of the tires, we used known equations. First find the arc length of the turning circle of the robot, then divide by the tire circumference, multiply buy the error factor. And then you receive the left and right rotation of the tires

$$s = 2\pi r(\frac{\theta_i}{360})$$

$$\theta_f = \left(\frac{s}{C}\right) 360$$

$$\theta_f = \theta_f\gamma$$

$$\theta_R = \theta_f, \theta_L = -\theta_f$$

$$\theta_L = rotation\ of\ left\ tire\ in\ degrees, \theta_R = rotation\ of\ right\ tire\ in\ degrees,$$

$$\theta = angle\ of\ tire\ rotation\ [deg], C = tire\ circumference, \gamma = error\ factor$$

Wall Following Equation

The angle that the robot needs to turn to make its orientation parallel to the wall is derived using the arctangent function. If D is the distance from the wall and W is the width of the robot, the equation is:

$$Angle\ to\ turn = \tan^{-1}(\frac{W}{D})$$

Behavior-Based Implementation

Priority Queue

A priority queue, implemented using the heapq module in Python, serves as the core mechanism for managing behaviors. Each behavior is an instance of the RobotBehavior class, which contains a priority level. The priority queue ensures that the highest-priority behavior is executed, allowing the robot to respond quickly to changing environmental stimuli.

Behaviors

1. Touch Behavior:
    a. Priority: 0
    b. Description:
        i. When the robot makes physical contact with an obstacle, the touch behavior is triggered.
        ii. If the front touch sensor is activated, the robot performs a backup and executes a counterclockwise turn.
        iii. If the left touch sensor is triggered, the robot executes a clockwise turn.
        iv. This behavior ensures the robot can recalibrate its position and orientation when it encounters obstacles.
2. Fire Detection Behavior:
    a. Priority: 1
    b. Description:
        i. This behavior is activated when the color sensor detects the color red, signifying the presence of a potential fire.

ii. The robot enters a state of "following fire," moving forward until the red color is no longer detected.

iii. If the robot collides with a wall or another obstacle during fire detection, it stops the behavior.

iv. Once a fire is found, the behavior is stopped, and a message is logged.

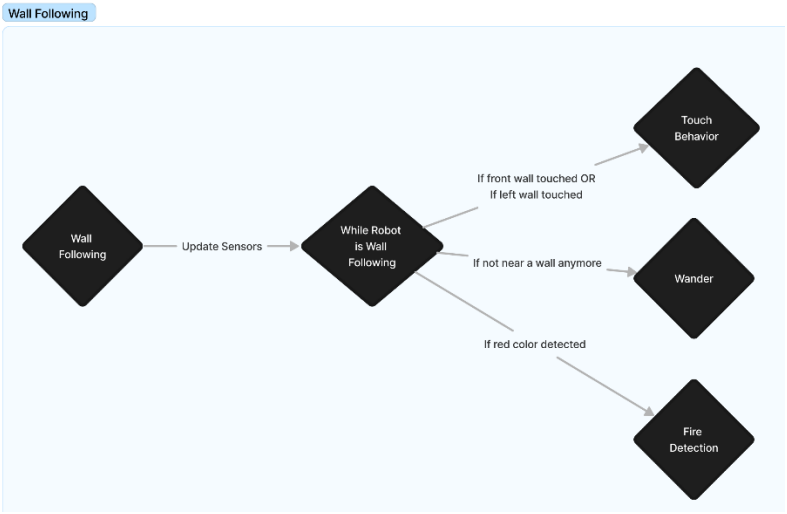
3. Wall Following Behavior:
    a. Priority: 2
    b. Description:
        i. When the robot gets close enough to a wall (within a specified distance), the wall following behavior is activated.
        ii. The robot aligns itself with the wall using calculated steering angles based on the distance to the wall.
        iii. It continues to follow the wall until a higher-priority behavior is triggered or the robot moves too far from the wall.
        iv. Collision with obstacles or detection of fire stops the wall following behavior.
4. Wander Behavior:
    a. Priority: 3
    b. Description:
        i. The default behavior, triggered when no higher-priority behaviors are active.- The robot moves forward until it detects a change in its environment.

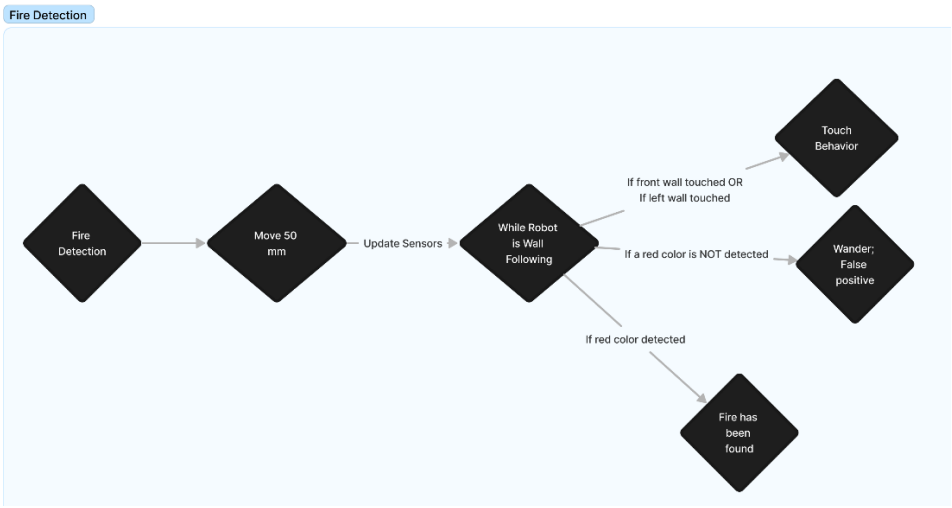
ii. The behavior has a timeout duration to prevent the robot from getting stuck in a loop.

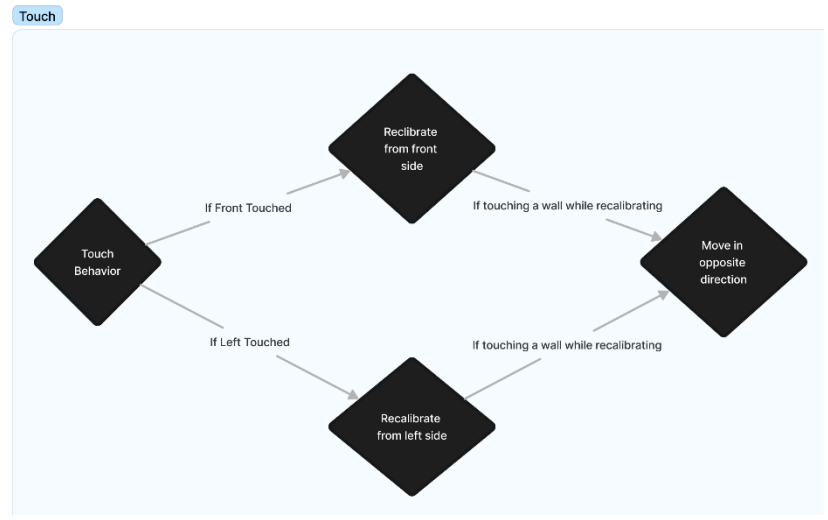
iii. Detection of obstacles, walls, or fire, as well as reaching the timeout, stops the wandering behavior.

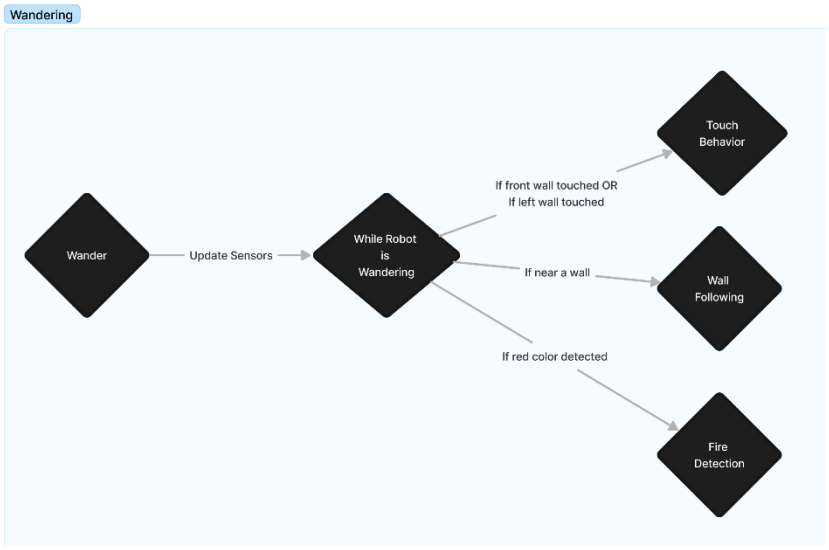
Behavior Flow

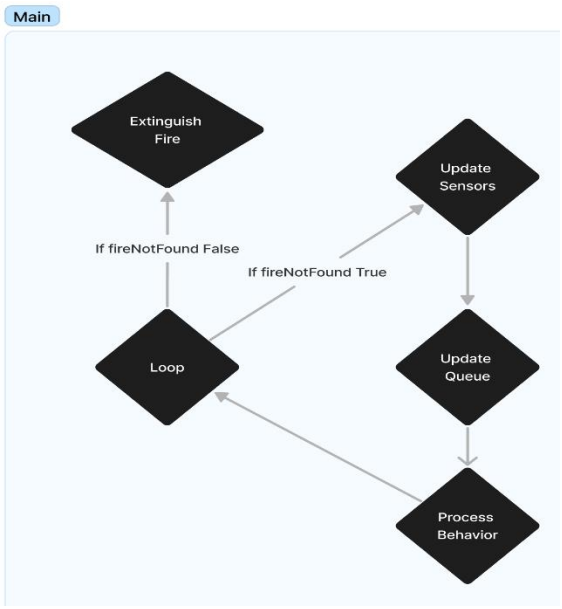
Below are diagrams of the flow of logic of each behavior:






Touch
Reclibrate from front side
If Front Touched
If touching a wall while recalibrating
Touch Behavior
Move in opposite direction
If Left Touched
If touching a wall while recalibrating
Recalibrate from left side


Wandering
Touch Behavior
If front wall touched OR If left wall touched
Wander
Update Sensors
While Robot is Wandering
If near a wall
Wall Following
If red color detected
Fire Detection


Main
Extinguish Fire
Update Sensors
If fireNotFound False
If fireNotFound True
Loop
Update Queue
Process Behavior

References

None