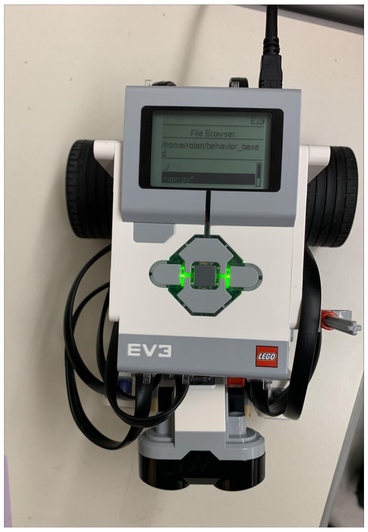
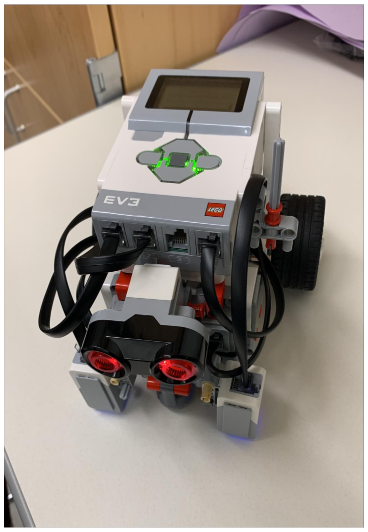
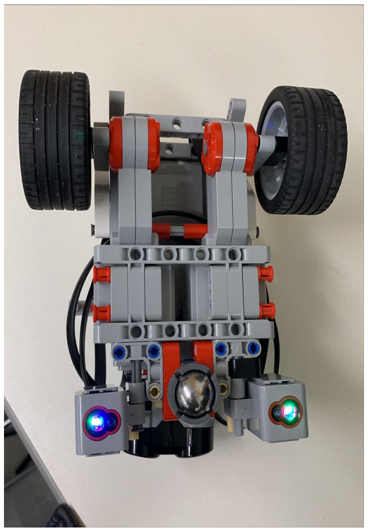
<u>Project 2 Report</u>
CSE-4360-001 Autonomous ROBOT Design

Team 15:
Amanda Whisenand
Vivek Kumar Yadav
Gabriel De Sa

Robot Description

Our robot was built using the Lego Mindstorm EV3 kit. We went with a simple unicycle type design which consists of a small base and two wheels. Each wheel has a low profile tire that is 56x28 mm and is connected to a large motor. Each large motor is connected to the main brick. The model balances itself with an omni directional ball in the rear. We went with these design choices because the unicycle type robots are relatively easy to control compared to other possible design choices.

In addition, the robot has two color sensors. One on the front right and one on the front left. These sensors are placed less than one inch above the ground and are secured in place with zero degrees of freedom. The light sensors are used in detecting and following walls as well as finding the goal and removing the object from it. On the front top of the robot, we have placed an ultrasonic sensor. This is used to periodically check if the object on the goal space is visible to the robot.

Figures 1, 2, and 3. Top, Front, and Bottom views of robot

Control System

The goal of this project was to implement a behavior-based object clearance robot that is able to move through “rooms”, avoiding walls while looking for the goal and object and removing the object from its goal space. To implement this, we created several helper functions. For general movement of the robot, we created the functions goToTarget() and rotate(). The goToTarget function takes the left and right motor, and a distance and will move the robot by the given distance. The rotate function takes the left and right motor and an angle and will rotate the robot by the given angle.

The main components for achieving the objective include wallFollow(), wander(), findObject(), goToObject(), and moveObject(). The program begins by calling the wander function which begins by making the robot choose a random direction to move until anyone of both color sensors detects a wall to follow. Once this happens, the wallFollow() function is called which directs the robot to follow the wall for a certain distance while continuously checking to make sure it is aligned with the wall. The robot will follow the wall until the wall ends or until it has traveled more than 0.5 meters. In this case the robot will turn and begin to wander again i.e. the robot moves a certain distance and makes small rotations in all directions looking for the object nearby using a sonic sensor on its front. In the event that the ultrasonic sensor picks up the object in findObject, the program will call goToObject which will move the robot forward in small increments while continuously checking that it is moving in the right direction by checking the distance returned from the ultrasonic sensor. While doing this, it also uses the color sensors to make sure it is not going to go through a wall. If for some reason the robot gets off track, or gets stuck on a wall, the program will direct the robot back into wander mode. Once the color sensor picks up on the color of the goal space, it will make a sound and then calls moveObject behavior to get the exact direction of the object and finally pushes the object until the color sensor no longer returns the goal color.

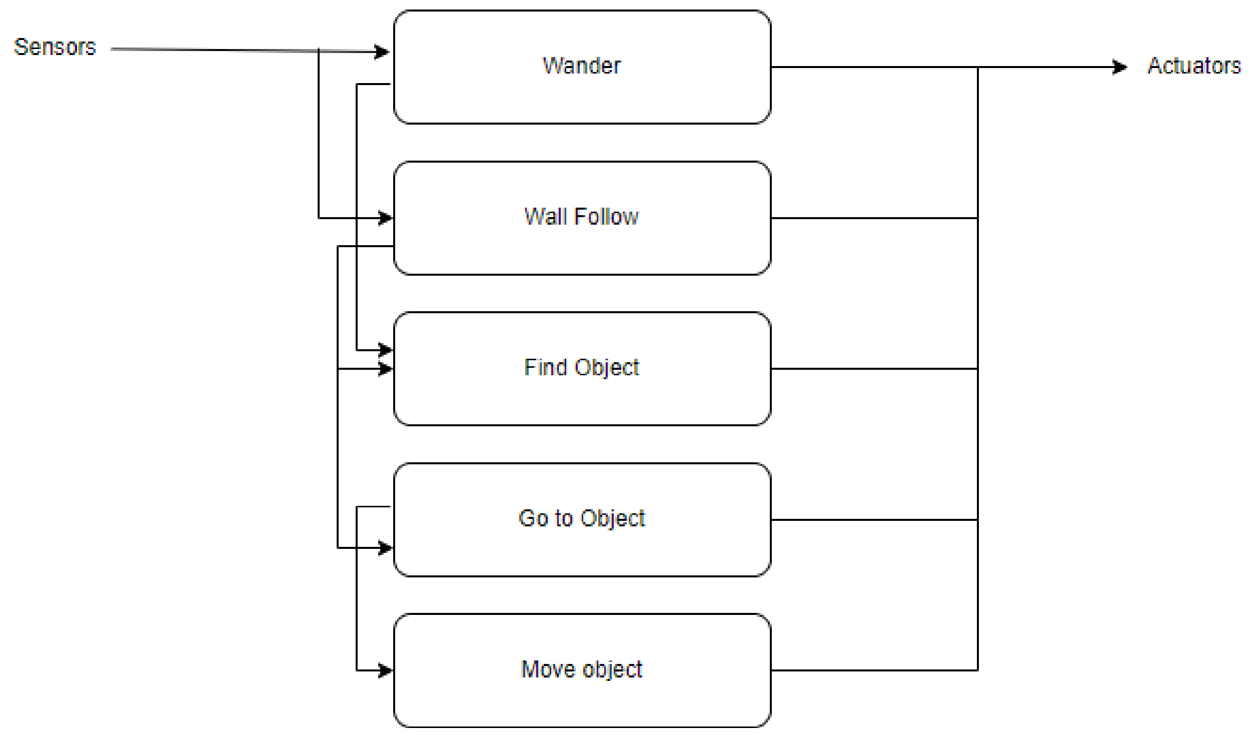


Figure 4. Behavior-Based Architecture

```python
#!/usr/bin/env pybricks-micropython
# from sys import builtin_module_names
from pybricks.hubs import EV3Brick
from pybricks.ev3devices import (Motor, TouchSensor, ColorSensor,
                                 InfraredSensor, UltrasonicSensor,
GyroSensor)
from pybricks.nxtdevices import (LightSensor)
from pybricks.parameters import Port, Stop, Direction, Button, Color
from pybricks.tools import wait, StopWatch, DataLog
from pybricks.robotics import DriveBase
from pybricks.media.ev3dev import SoundFile, ImageFile
import time
import random

# This program requires LEGO EV3 MicroPython v2.0 or higher.
# Click "Open user guide" on the EV3 extension tab for more information.
wheelCirc = (5.3975*3.14159)/100 # meters
robotCirc = (12.065*3.14159)/100 # meters
wallColor = 10
goalColor = Color.RED
ev3 = EV3Brick()


def goToTarget(right, left, target, velocity = 360):
    resetAngles(right, left)
    desired_angle = (target/wheelCirc)*360
    right.run_angle(velocity, desired_angle, Stop.HOLD, False)
    left.run_angle(velocity, desired_angle)

def resetAngles(right, left, angle = 0):
    right.reset_angle(angle)
    left.reset_angle(angle)

def rotate(right, left, angle=90, velocity = 150):
    ratio = angle/360
    dist = robotCirc * ratio
    desiredAngle = ((dist/wheelCirc) * 360 * (14/15))
    resetAngles(right, left)
    right.run_target(velocity, desiredAngle, Stop.HOLD, False)
    left.run_target(velocity, -desiredAngle)

def alignToWall(sensor, leftSensor, right, left):
    goToTarget(right, left, .05) # move forward a small distance
    if(leftSensor.reflection() > wallColor and sensor.reflection() <=
wallColor):
        goToTarget(right, left, -.075)
        rotate(right, left, 20)
        return False
    rotate(right, left, 25)
    for i in range(10):
        if(sensor.reflection() <= wallColor):
            while(sensor.reflection() <= wallColor):
                rotate(right, left, 10)
            else:
                rotate(right, left, -10)
                return True
        rotate(right, left, 10) # Rotate to find the wall
    else:
        return False

def wallFollow(sensor, leftSensor, right, left, dist):
    while sensor.reflection() <= wallColor:
        # Check inital Distance and return if followed for too long
        if dist >= 1.5:
            return 1.5
        # If we run into a perpendicular wall, follow it
        if leftSensor.reflection() <= wallColor:
            while(sensor.reflection() <= wallColor and
leftSensor.reflection() <= wallColor):
                right.run(100)
                left.run(100)
            else:
                right.stop()
                left.stop()
            goToTarget(right, left, .02)
            rotate(right, left, 60)
            while sensor.reflection() > wallColor:
                rotate(right, left, 5)
            else:
                rotate(right, left, 5)
                return wallFollow(sensor, leftSensor, right, left, 0)
        else:
            # Move along the wall
            right.run(150)
            left.run(150)
            dist = dist + .0001
    else:
        right.stop()
        left.stop()
        # check to make sure we didn't clear the right side of a wall:
        rotate(right, left, 15)
        if sensor.reflection() <= wallColor:
            rotate(right, left, -5)
            return wallFollow(sensor, leftSensor, right, left, dist)
        # check left side
        rotate(right, left, -30)
        if sensor.reflection() <= wallColor:
            rotate(right, left, 5)
            return wallFollow(sensor, leftSensor, right, left, dist)
        rotate(right, left, 15)
        # do a check that to make sure we aren't at corner
        for i in range(3):
            goToTarget(right, left, -.025)
            if leftSensor.reflection() <= wallColor:
                goToTarget(right, left, .05)
                rotate(right, left, 60)
                while sensor.reflection() > wallColor:
                    rotate(right, left, 5)
                else:
                    rotate(right, left, 5)
                    return wallFollow(sensor, leftSensor, right, left, 0)
        goToTarget(right, left, .125)
        return dist

def findObject(sonicSens, right, left):
    """Find the object by moving the vehicle in multiple directions"""
    for i in range(18):
        rotate(right, left, 20) # 20 degree increments
        dist = sonicSens.distance() / 1000
        if dist < .45:
            return dist
    return 1
        # need to avoid walls and remember direction

def moveObject(right, left, colorS, lColorS, sonicS):
    boolean = True
    # Check to see if object is in front of it
    ev3.speaker.beep()
    distance = sonicS.distance()/1000
    if(distance < .2):
        goToTarget(right,left, distance+.3)
        return
    while(boolean):
        goToTarget(right, left, .02)
        if(colorS.color() != goalColor and lColorS.color() == goalColor):
            while(colorS.color() != goalColor):
                rotate(right, left, 5)
        if(colorS.color() == goalColor and lColorS.color() != goalColor):
            while(lColorS.color() != goalColor):
                rotate(right, left, -5)
        if(colorS.color() == goalColor and lColorS.color() == goalColor):
            rotate(right, left, 60)
            shortestDist = 1
            for i in range(10):
                rotate(right, left, -10)
                if shortestDist < sonicS.distance():
                    shortestDist = (sonicS.distance()/1000)
                if shortestDist < .25:
                    goToTarget(right, left, shortestDist+.3)
                    return True



def goToObject(right, left, rightSensor, leftSensor, sonar, distance):
    # goToTarget(right, left, distance)
    moving = True
    while moving:
        if(rightSensor.reflection() <= wallColor or rightSensor.color() ==
goalColor) or (leftSensor.reflection() <= wallColor or leftSensor.color()
== goalColor):
            moving = False
        else:
            goToTarget(right, left, .05)
    else:
        if(rightSensor.reflection() <= wallColor or
leftSensor.reflection() <= wallColor):
            #wall follow
            alignToWall(rightSensor, leftSensor, right, left)
            dist = wallFollow(rightSensor, leftSensor, right, left, 0)
            return False
        else:
            # find the target within the goal, and move and knock it off.
            if(leftSensor.color() == goalColor):
                while(rightSensor.color() != goalColor):
                    rotate(right, left, 30)
            else:
                while(leftSensor.color() != goalColor):
                    rotate(right, left, -30)
            ev3.speaker.play_file(SoundFile.FANFARE)
            return moveObject(right, left, rightSensor, leftSensor, sonar)


def wander(colorS, lColorS, sonicS, right, left):
    # choose some direction
    # Keep moving in direction until
    # 1. move a certain distance
    # 2. find a wall
    # 3. find the goal space
    # needs to an ability to go a certain distance and direction but also
check for walls on the way there.
    distance = 0
    boolean = True
    direction = [0, 90, -90, 180]
    while(boolean):
        # choose some random direction to go in.
        rotate(right, left, direction[random.randint(0, 2)])
        right.run(360), left.run(360)
```

```python
        while(colorS.color() != goalColor and colorS.reflection() > wallColor and lColorS.reflection() > wallColor):
            continue
        else:
            right.stop()
            left.stop()
            if(lColorS.reflection() <= wallColor and colorS.reflection() > wallColor):
                goToTarget(right, left, -.10)
                rotate(right, left, -20)
                continue
            if(colorS.reflection() <= wallColor):
                if not alignToWall(colorS, lColorS, right, left):
                    goToTarget(right, left, -.05)
                    continue
                distance = wallFollow(colorS, lColorS, right, left, 0)
                if distance == 1.5: # we've travelled the wall for too long.
                    rotate(right, left, 90)
                    goToTarget(right, left, .1)
                    distFromObject = findObject(sonicS, right, left)
                    if distFromObject < 1:
                        if goToObject(right, left, colorS, lColorS, sonicS, distFromObject):
                            boolean = False
                        else:
                            continue
                else:  # we cleared the wall
                    goToTarget(right, left, .15)
                    rotate(right, left, -90) # turn right past wall
                    goToTarget(right, left, .15)
                    distFromObject = findObject(sonicS, right, left)
                    if distFromObject < 1:
                        if goToObject(right, left, colorS, lColorS, sonicS, distFromObject):
                            boolean = False
                        else:
                            continue
            else:
                ev3.speaker.play_file(SoundFile.FANFARE)
                goToTarget(right, left, .05)
                return moveObject(right, left, colorS, lColorS, sonicS)


# Create your objects here.
def main():
    rightMotor = Motor(Port.A, Direction.COUNTERCLOCKWISE)
    leftMotor = Motor(Port.D, Direction.COUNTERCLOCKWISE)
    colorSensor = ColorSensor(Port.S4)
    leftColorSensor = ColorSensor(Port.S2)
    sonicSensor = UltrasonicSensor(Port.S1)
    # Write your program here.
    ev3.speaker.beep()
    wander(colorSensor, leftColorSensor, sonicSensor, rightMotor,
leftMotor)
main()
```