A TASK 1.1

Light Sensor Testing Value 5mm

Experiment	value 1	value 2	value 3	value 4	value 5	value 6	value 7	mean value	standard deviation
White	96	96	95	97	95	96	97	96	0.756
Blue	6	4	5	8	4	5	5	5.286	1.38

Light Sensor Testing Value 25mm

Experiment	value 1	value 2	value 3	value 4	value 5	value 6	value 7	mean value	standard deviation
White	17	17	16	17	16	17	17	16.714	0.488
Blue	1	1	1	0	1	1	1	0.857	0.377

Light Sensor Testing Value 50mm

Experiment	value 1	value 2	value 3	value 4	value 5	value 6	value 7	mean value	standard deviation
White	5	6	5	5	6	5	6	5.429	0.534
Blue	0	0	0	0	0	0	0	0	0

Light Sensor Testing Value 100mm

Experiment	value 1	value 2	value 3	value 4	value 5	value 6	value 7	mean value	standard deviation
White	1	1	1	1	1	1	1	1	0
Blue	0	0	0	0	0	0	0	0	0

Gyro Sensor Testing Value									
Experiment	value 1	value 2	value 3	value 4	value 5	value 6	value 7	mean value	standard deviation
90 degrees	90	90	89	90	91	90	89	89.857	0.690
180 degrees	179	180	182	182	180	179	181	180.429	1.273
270 degrees	271	271	270	270	270	272	270	270.571	0.787
360 degrees	360	360	361	360	362	360	360	360.429	0.787
720 degrees	720	720	719	720	719	719	720	719.571	0.535
1080 degrees	1078	1078	1080	1080	1080	1080	1080	1079.429	0.976

Ultrasonic Sensor Testing Value									
Experiment	value 1	value 2	value 3	value 4	value 5	value 6	value 7	mean value	standard deviation
5 mm	32	31	31	32	32	32	32	31.714	0.487
50 mm	49	49	50	51	51	49	51	50	1.000
100 mm	112	109	109	112	109	112	112	110.714	1.603
250 mm	251	251	251	251	251	251	251	251	0.000
infinity mm	2550	2550	2550	2550	2550	2550	2550	2550	0.000

TASK 1.2

The sensors as shown by the mean values and the standard deviation all have margins of error in particular situations. The light and ultrasonic sensors are more accurate given a wider distance from their targets. The closer they get, the more difficult it is to derive an exact measurement. The gyro sensor on the other hand never gives up an actual figure and always twitches between + or – values. The sensors are less reliable for accurate values but the values can be approximated. Also, some of the sensors are defective and produce even worse reliable values.

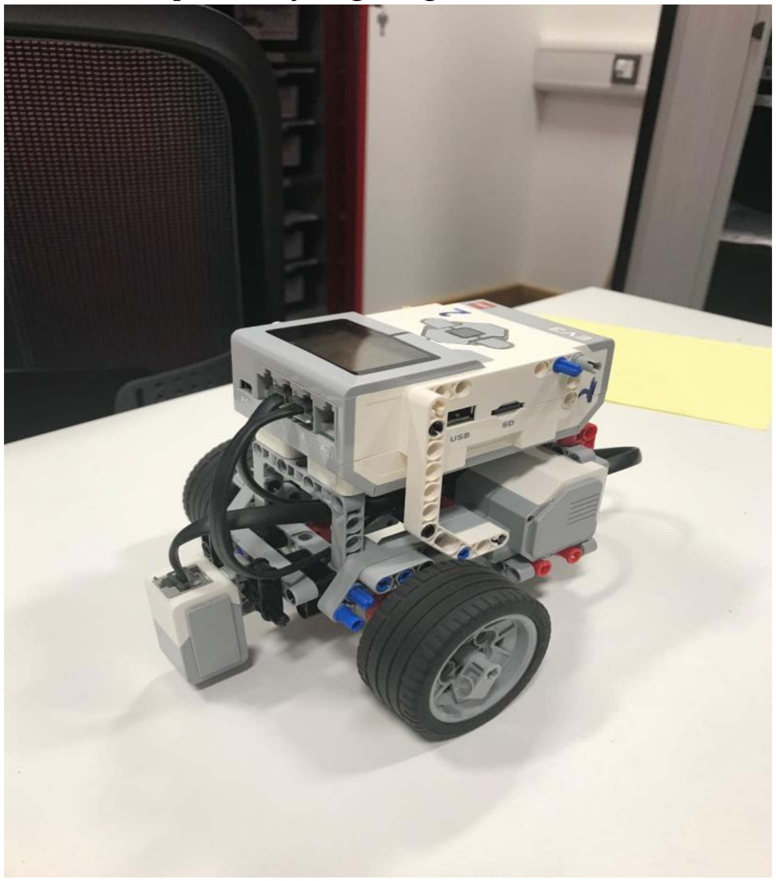
B TASK 2.1

Motor Testing Values straight line									
Experiment	value 1	value 2	value 3	value 4	value 5	value 6	value 7	mean value	standard deviation
slow 100 cm	110	70	86	93	86	213	213	124.429	61.646
medium 500 cm	32	35	39	139	51	52	132	68.571	46.378
fast 1500 cm	31	54	82	60	76	69	97	67	21.307

TASK 2.2

Based on the values and the findings as well as the standard deviation and mean, the motors are more reliable to follow the straight line when slow than at medium or fast speed. Several other factors such as the road smoothness and the lightening came to play when following the straight line. The faster the motors go, the earlier they deviate from the straight white line, the slower they go, the more likely they get to the end of the straight line. At slow speed the motors followed the line to the end. As the speed increased the distance it covered decreased.

C TASK 3

1. The selected sensor was the light/colour sensor. The sensor was selected to detect the white spot and then rotate 180 degrees back to find the other white spot and keep rotating between these spots for a maximum of twelve times. The degree rotation was done by the drivebase with the turn() function. This was preferred as after several tries with the gyro, there was no possibility of getting an accurate turn.
2. 
3. The algorithmic idea is to find the white spot rotate 180 in search of the other white spot. This leaves the robot alternating between both spots
4. Pseudocode

```
# Creating objects
SET ev3 TO EV3Brick()

# Initialize the motors
SET left_motor TO Motor(Port.A)
SET right_motor TO Motor(Port.B)

# Initialize the colour sensor
SET sensorColor1 TO ColorSensor(Port.S1)

# Initialize the drive base
SET myRobot TO DriveBase(left_motor, right_motor,
   wheel_diameter=55.5, axle_track=104)

SET colour TO sensorColor1.color()

WHILE True:
        myRobot.drive(250,0)
        SET colour TO sensorColor1.color()

        IF colour != colour.WHITE:
                myRobot.turn(210)
                myRobot.reset()

        SET colour TO sensorColor1.color()

```

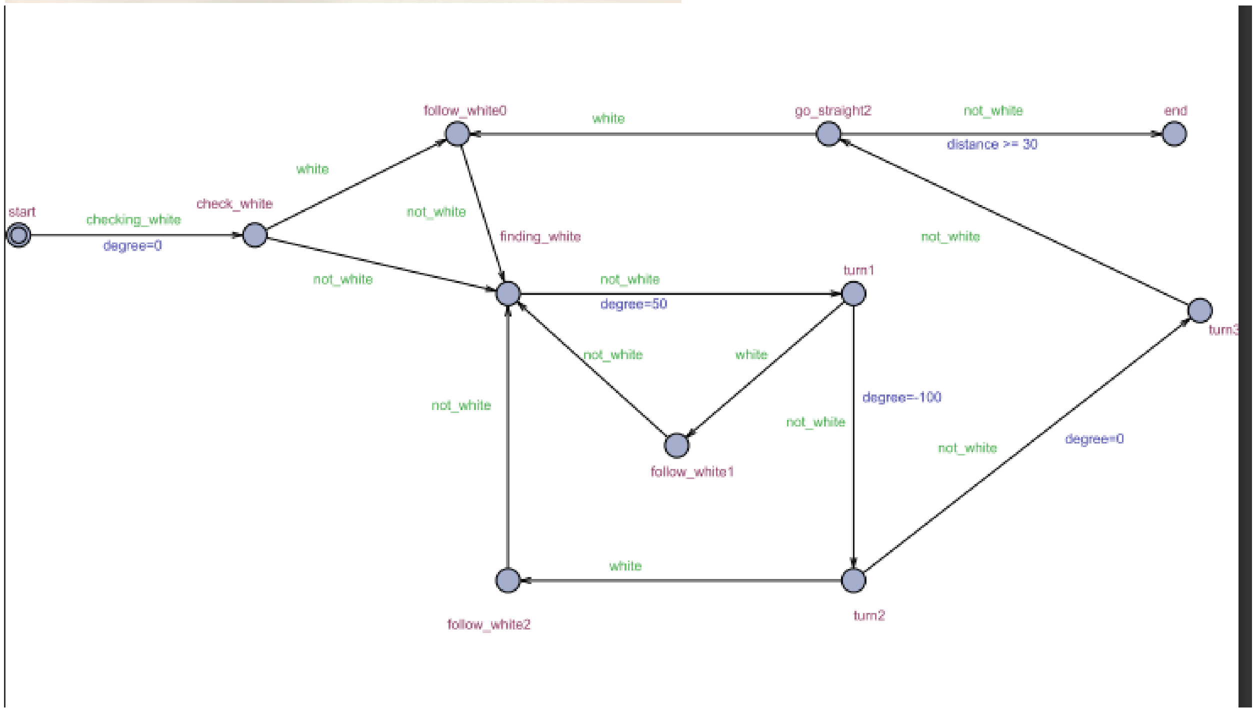
5. MicroPython Code Listing

```
#!/usr/bin/env pybricks-micropython
from pybricks.hubs import EV3Brick
from pybricks.ev3devices import (Motor, TouchSensor, ColorSensor,
                                 InfraredSensor, UltrasonicSensor, GyroSensor)
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


# This program requires LEGO EV3 MicroPython v2.0 or higher.
# Click "Open user guide" on the EV3 extension tab for more information.


# Create your objects here.
ev3 = EV3Brick()

#sensorGyro3 = GyroSensor(Port.S3)

left_motor = Motor(Port.A)
right_motor = Motor(Port.B)

# Initialize the color sensor.
Colour = ColorSensor(Port.S1)
```

```
# Initialize the drive base.
myRobot = DriveBase(left_motor, right_motor, wheel_diameter=55.5, axle_track=104)
myRobot.settings(50, 50, 60, 50)
# write program

# LINE FOLLOWER CODE

# Sets the colour
r = Colour.reflection()


#runs the motor until it reaches a white spot and turns 180 degrees then resets
the angle
while(True):
    myRobot.drive(250,0)
    colour = sensorColor1.color()

# rotates to find the next white spot
    if(colour != colour.WHITE):
        myRobot.turn(210)
        myRobot.reset()


    colour = sensorColor1.color()
    #print(colour)
```

D TASK 4

1.



2.

3. **Pseudocode**

```
# Initialize the motors
SET left_motor TO Motor(Port.B)
SET right_motor TO Motor(Port.C)

# Initialize the colour and ultrasonic sensor
SET Colour TO ColorSensor(Port.S3)
SET Distance TO UltrasonicSensor(Port.S1)

# Initialize the drive base
SET myRobot TO DriveBase(left_motor, right_motor,
    wheel_diameter=55.5, axle_track=104)

SET d TO Distance.distance()
SET r TO Colour.reflection()

WHILE True:
        WHILE r >= 42:
                SET follow TO (60 - Colour.reflection())
                myRobot.drive(50, follow)
                SET r TO Colour.reflection()

                WHILE r < 42:
                        myRobot.turn(58)
                        SET r TO Colour.reflection()
                        wait(300)
                        SET r TO Colour.reflection()

                        WHILE r < 42:
                                myRobot.turn(-100)
                                SET r TO Colour.reflection()
                                wait(300)
                                SET r TO Colour.reflection()

                                IF r < 42:
                                        myRobot.turn(44)
                                        SET r TO Colour.reflection()
                                        myRobot.straight(250)
                                        SET r TO Colour.reflection()
                                        wait(100)
                                        SET r TO Colour.reflection()
myRobot.stop()
```

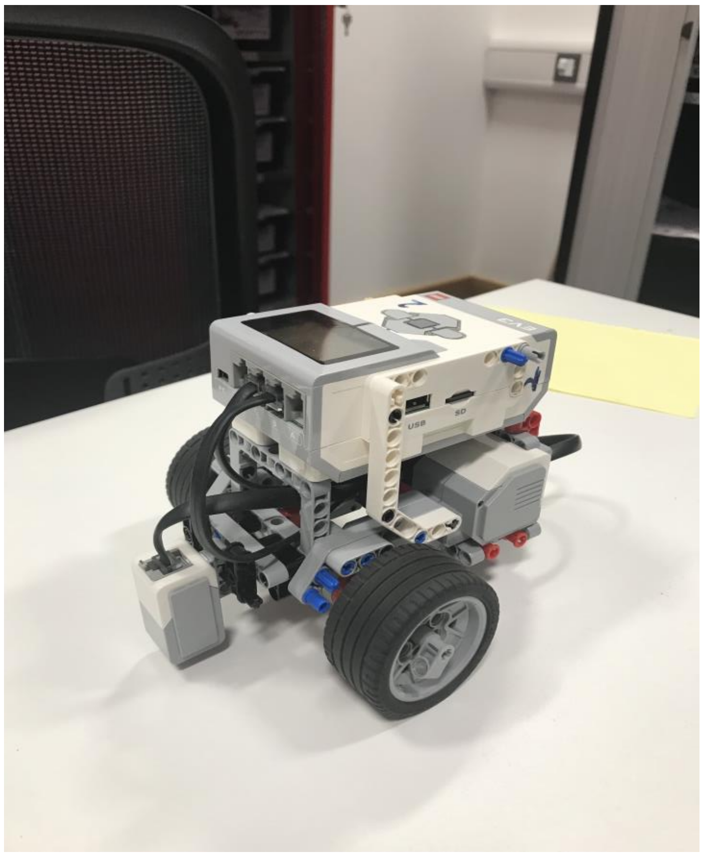
4. MicroPython Code Listing

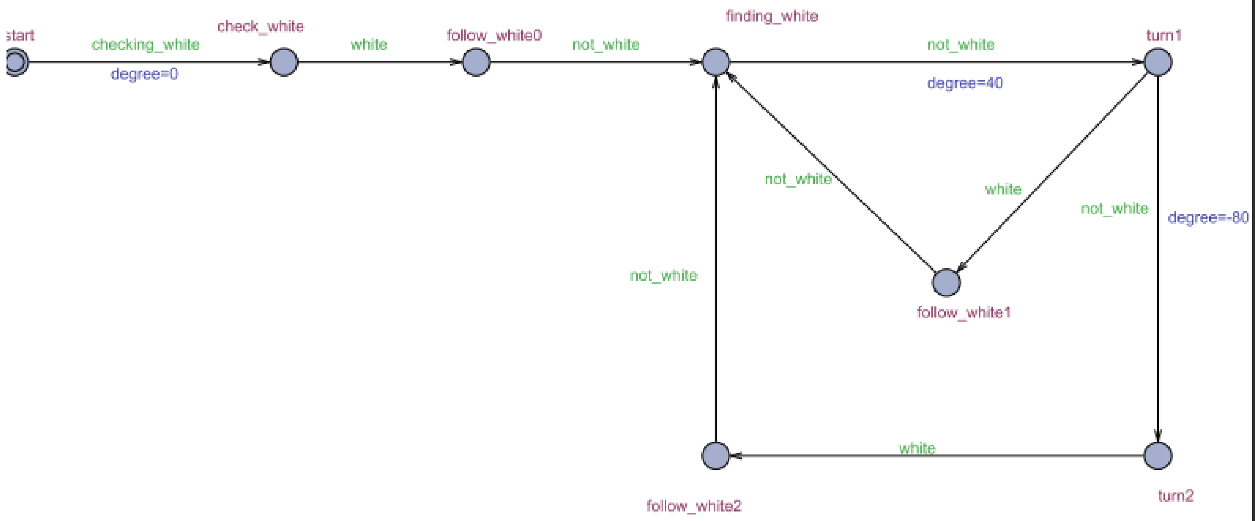
```
#!/usr/bin/env pybricks-micropython
from pybricks.hubs import EV3Brick
from pybricks.ev3devices import (Motor, TouchSensor, ColorSensor,
                                 InfraredSensor, UltrasonicSensor, GyroSensor)
from pybricks.parameters import Port, Stop, Direction, Button, Color
from pybricks.tools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

```
# This program requires LEGO EV3 MicroPython v2.0 or higher.
# Click "Open user guide" on the EV3 extension tab for more information.


# Initialize the motors.
left_motor = Motor(Port.B)
right_motor = Motor(Port.C)

# Initialize the color sensor.
Colour = ColorSensor(Port.S3)
Distance = UltrasonicSensor(Port.S1)
# Initialize the drive base.
myRobot = DriveBase(left_motor, right_motor, wheel_diameter=55.5, axle_track=104)
# myRobot.settings(150, 100, 60, 90)
# write program

# sets an initial reflection value
r = Colour.reflection()

# with the value, the loop is either entered or not and the robot drives at a
correction angle of the reflection
while r >= 42:
    follow = (60 - Colour.reflection())
    myRobot.drive(50, follow)
    r = Colour.reflection()
    print(r)

# when the reflection shows out of white line, the robot rotates to find white
    while r < 42:
        myRobot.turn(58)
        r = Colour.reflection()
        print(r)
        wait(100)
        r = Colour.reflection()

# if white is not found the first time the robot rotates the other way to check
for white
        while r < 42:
            myRobot.turn(-100)
            r = Colour.reflection()
            print(r)
            wait(100)
            r = Colour.reflection()
```

```
# if there is still no white the robot returns to original position and moves forward signaling a gap
            if r < 42:
                myRobot.turn(44)
                r = Colour.reflection()
                print(r)
                myRobot.straight(250)
                r = Colour.reflection()
                wait(100)
                r = Colour.reflection()

myRobot.stop()
```

E TASK 5

1. 



2.

3. Pseudocode

```
# Creating objects
SET ev3 TO EV3Brick()

# Initialize the motors
SET left_motor TO Motor(Port.A)
SET right_motor TO Motor(Port.B)

# Initialize the colour sensor
SET Colour TO ColorSensor(Port.S1)

# Initialize the drive base
SET myRobot TO DriveBase(left_motor, right_motor, wheel_diameter=55.5, axle_track=104)

SET r TO Colour.reflection()

WHILE True:
        WHILE r >= 10:
                SET follow TO (30 - Colour.reflection()) * 2
                myRobot.drive(250, follow)
                SET r TO Colour.reflection()

                WHILE r < 10:
                        myRobot.turn(10)
                        SET follow TO (30 - Colour.reflection()) * 2
                        myRobot.drive(50, follow)
                        SET r TO Colour.reflection()
                        wait(300)

                        WHILE r < 10:
                                myRobot.turn(-10)
```

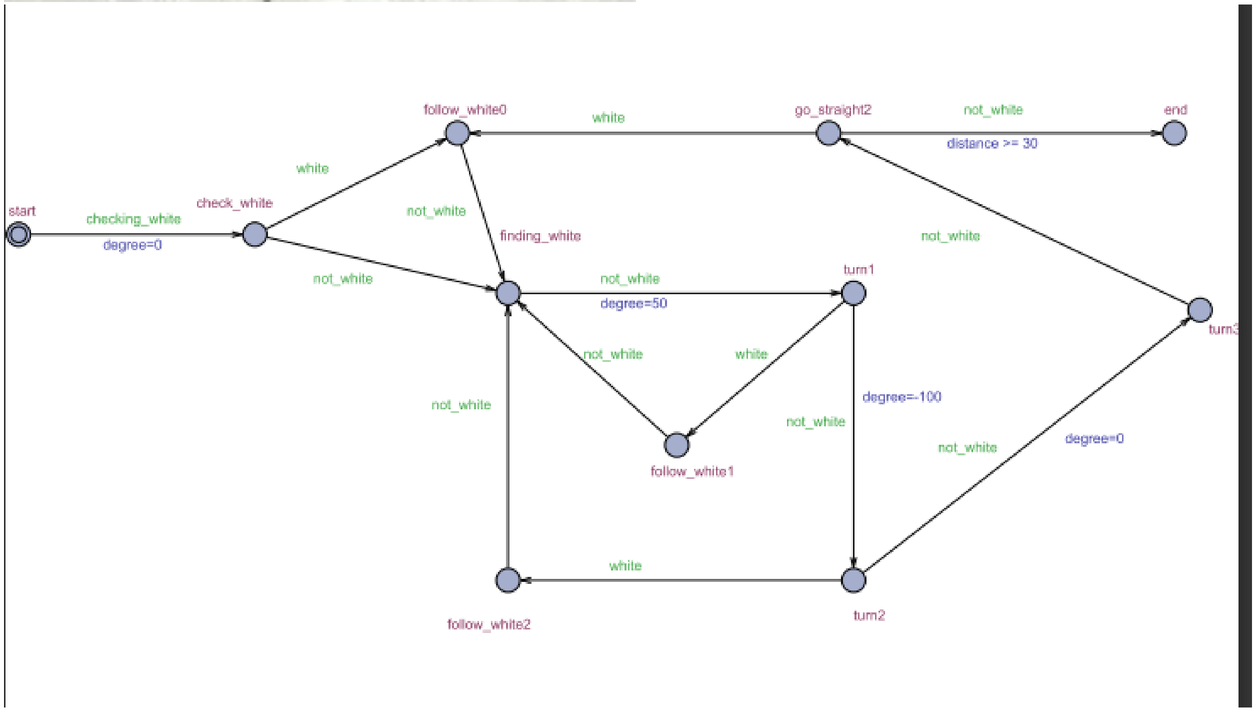
```
            SET follow TO (30 - Colour.reflection()) * 2
            myRobot.drive(50, follow)
            SET r TO Colour.reflection()
            wait(300)
```

4. MicroPython code listing

```python
#!/usr/bin/env pybricks-micropython
from pybricks.hubs import EV3Brick
from pybricks.ev3devices import (Motor, TouchSensor, ColorSensor,
                                 InfraredSensor, UltrasonicSensor, GyroSensor)
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


# This program requires LEGO EV3 MicroPython v2.0 or higher.
# Click "Open user guide" on the EV3 extension tab for more information.


# Create your objects here.
ev3 = EV3Brick()

#sensorGyro3 = GyroSensor(Port.S3)

left_motor = Motor(Port.A)
right_motor = Motor(Port.B)

# Initialize the color sensor.
Colour = ColorSensor(Port.S1)

# Initialize the drive base.
myRobot = DriveBase(left_motor, right_motor, wheel_diameter=55.5, axle_track=104)
myRobot.settings(50, 50, 60, 50)
# write program

# LINE FOLLOWER CODE

# Sets the colour
r = Colour.reflection()

# Enters the loop and checks for colour
while True:
    # if the colour is white with the reflection and it follows the colour with a correction angle
    while r >= 10:
        follow = (30 - Colour.reflection()) * 2
        myRobot.drive(250, follow)
        r = Colour.reflection()

        # if the colour is not white depending on the reflection it turns looking to find white
        while r < 10:
            myRobot.turn(10)
            follow = (30 - Colour.reflection()) * 2
            myRobot.drive(50, follow)
            r = Colour.reflection()
            wait(300)
            while r < 10:
                myRobot.turn(-10)
                follow = (30 - Colour.reflection()) * 2
                myRobot.drive(50, follow)
                r = Colour.reflection()
                wait(300)

        print(r)
```

F TASK 6

1.

2. 

3. **Pseudocode**

```
# Initialize the motors
SET left_motor TO Motor(Port.B)
SET right_motor TO Motor(Port.C)

# Initialize the colour and ultrasonic sensor
SET Colour TO ColorSensor(Port.S3)
SET Distance TO UltrasonicSensor(Port.S1)

# Initialize the drive base
SET myRobot TO DriveBase(left_motor, right_motor, wheel_diameter=55.5, axle_track=104)

SET d TO Distance.distance()
SET r TO Colour.reflection()

WHILE True:
    myRobot.drive(150,0)
    SET r TO Colour.reflection()
    SET d TO Distance.distance()

    IF (r >= 40 or d < 200):
        myRobot.straight(180)
        myRobot.turn(-110)
        SET r TO Colour.reflection()
        SET d TO Distance.distance()

        WHILE d <= 200:
            myRobot.turn(215)
            SET d TO Distance.distance()

            IF d <= 200:
                myRobot.turn(-110)
                SET d TO Distance.distance()

            ELSE:
                myRobot.drive(150,0)

    ELSE:
        myRobot.drive(150,0)
```

4. MicroPython Code Listing

```python
#!/usr/bin/env pybricks-micropython
from pybricks.hubs import EV3Brick
from pybricks.ev3devices import (Motor, TouchSensor, ColorSensor,
                                 InfraredSensor, UltrasonicSensor, GyroSensor)
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
```

```
# This program requires LEGO EV3 MicroPython v2.0 or higher.
# Click "Open user guide" on the EV3 extension tab for more information.


# Initialize the motors.
left_motor = Motor(Port.B)
right_motor = Motor(Port.C)

# Initialize the color sensor.
Colour = ColorSensor(Port.S3)
Distance = UltrasonicSensor(Port.S1)
# Initialize the drive base.
myRobot = DriveBase(left_motor, right_motor, wheel_diameter=55.5, axle_track=104)
# myRobot.settings(150, 100, 60, 90)
# write program

d = Distance.distance()
r = Colour.reflection()

# enters the loop to begin drive
while True:
    myRobot.drive(150, 0)
    r = Colour.reflection()
    print(r)
    d = Distance.distance()

      # checks for the distance between obstacles both left and right at every
white line
      # if the distance is close it turns if not and there is no obstacle it
drives forward
      # it then navigates through the maze at every white line checking for
obstacles
    if (r >= 40 or d < 200):
        myRobot.straight(180)
        myRobot.turn(-110)
        r = Colour.reflection()
        print(r)
        d = Distance.distance()
        print(d)

        while d <= 200:
            myRobot.turn(215)
            d = Distance.distance()
            print(d)
            if d <= 200:
                myRobot.turn(-110)
                d = Distance.distance()
                print(d)

            else:
                myRobot.drive(150, 0)

# if no white line or obstacle then the robot keeps driving
    else:
        myRobot.drive(150, 0)


            else:
                myRobot.drive(150, 0)


    else:
        myRobot.drive(150, 0)
```