MAZE SOLVING USING LEGO EV3 ROBOT

Submitted to:

Dr. Hanumant Singh Shekhawat

Dept. of Electronics and Electrical Engineering

Submitted by:

Sushovan Chaki (214102511)

CODE IMPLEMENTED -

```
#!/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()


# Write your program here.
# Initialize the motors.
left_motor = Motor(Port.B)
right_motor = Motor(Port.C)

# Initialize the color sensor.
light_sensor_front = ColorSensor(Port.S3)
light_sensor_right = ColorSensor(Port.S4)

# Initialize the drive base.
robot = DriveBase(left_motor, right_motor, wheel_diameter=30.1,
axle_track=161)

ref_low=0
ref_high=10
ref_rlow=0
ref_rhigh=10
ht = 20
lt = 4
ref = 0
r = 6

while (True):
    print(light_sensor_front.reflection())
    print(light_sensor_right.reflection())
    if(light_sensor_front.reflection()<ref_high and
light_sensor_front.reflection()>ref_low):
        print("111111")
```

```
        if(light_sensor_right.reflection()<ref_high and
light_sensor_right.reflection()>ref_low):
            print("222222")
            robot.stop()
            robot.straight(10)
            robot.turn(-90)
            if(light_sensor_front.reflection()<ref_high and
light_sensor_front.reflection()>ref_low):
                print("33333")
                robot.stop()
                robot.straight(10)
                robot.turn(-90)
        else:
            print("44444")
            robot.stop()
            robot.straight(10)
            robot.turn(90)
    else:
        print("555555")
        if(light_sensor_right.reflection()<ref_rhigh and
light_sensor_right.reflection()>ref_rlow):
            robot.drive(80,0)
        else:
            robot.stop()
            robot.turn(5)
            robot.straight(10)
            if(light_sensor_right.reflection()<ht and
light_sensor_right.reflection()>lt):
                robot.stop()
                robot.straight(-10)
                robot.stop()
                robot.turn(-5)
                robot.stop()
                robot.turn(-ref)
                #rotate 90 right
                robot.straight(175)
                robot.turn(90)
                robot.straight(80)
                ref = 0
            else:
                robot.stop()
                robot.straight(-10)
                robot.stop()
                robot.turn(-5)
               #when there is no obstacle in right
                robot.stop()
                #robot will turn 5 deg left
                robot.turn(-r)
```

```
                #modification of reference position
                ref = ref-r
                print("888888")
                #again check right obstacle
                if(light_sensor_right.reflection()<ref_rhigh and
light_sensor_right.reflection()>ref_rlow):
                    #if right obstacle is there then stop and return to the
loop
                    robot.stop()
                else:
                    #nothing detected of left turn
                    #stop the robot
                    robot.stop()
                    #10 degree right turn
                    robot.turn(2*r)
                    #modification of position
                    ref = ref+2*r
                    print("888888")
                    robot.stop()
```

DIFFICULTIES FACED AND IT'S SOLUTION-

DIFFICULTIES	SOLUTION
Calibration of Color Sensors	We spent a lot of time for calibrating the light sensor properly. To set the proper upper bound and lower bound we connected the pc in the robot and tried to assess the proper intensity range for the light sensor.
Keep the robot in the track	For a long run our robot was going out of track. So, we made an algorithm with help of which our robot was getting back on track whenever it was going out of track. Whenever there is no

	track, our robot stops and searches for track whether it is on left direction or right direction and then proceeds.
Difficulty faced in iteration	To find any specific algorithm to differentiate between turn and out of track we implemented an algorithm with which whenever it turns little bit and adjusts to be in the track
Differentiating between out of track and right turn	As the previous algorithm was failed, we have made a new algorithm to overcome this obstacle. We made another algorithm where if the robot finds nothing in right side it will go little straight and then if it detects the black strip then it is a sharp turn otherwise it is a wall.