EE-555 – AUTOMATION LAB

A Lab Project Report on maze solving using LEGO EV3 ROBOT.

Submitted by

Mallavaram Sai Harini	214102508
Mallemputi Aravind	214102509
Ritika Jaiswal	214102510

Guide by

Dr. Hanumant Singh Shekhawat

Dept. of Electronics and Electrical Engineering

IIT Guwahati

Code:

```
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()
left_motor=Motor(Port.B)
right_motor=Motor(Port.C)
cs_front=ColorSensor(Port.S1)
cs_right=ColorSensor(Port.S4)
robot=DriveBase(left_motor, right_motor, wheel_diameter=30.1, axle_track=161)
ref_low=0
ref_high=20
# Write your program here.
ev3.speaker.beep()
n=0
ref=0
sideangle=10
def stop_robot():
robot.stop()
left_motor.brake()
right_motor.brake()
# while(True):
# if(cs_front.reflection()>ref_high and cs_right.reflection()>ref_high):
# robot.straight(182)
# stop_robot()
# robot.turn(90)
# robot.straight(50)
# stop_robot()
# break
```

```
# else:
# robot.drive(20,0)
n=0
while True:
#
if ref==0:
n=0
#if(cs_front.reflection()>ref_low and cs_front.reflection()<ref_high and
cs_right.reflection()>ref_low and cs_right.reflection()<ref_high ):
if(cs_front.reflection()>ref_low and cs_front.reflection()<ref_high):
if(cs_right.reflection()>ref_low and cs_right.reflection()<ref_high):
print("1111111111")
stop_robot()
robot.straight(-5)
stop_robot()
robot.turn(-90-ref)
print(-90-ref/2)
print(ref)
ref=0
else:
print("EEEEEEEEEEEEE")
robot.straight(200)
stop_robot()
robot.turn(90-ref)
ref=0
robot.straight(30)
# if(cs_right.reflection()>ref_low and cs_right.reflection()<ref_high):
# print("2222222222222")
# #d move back and turn left
# robot.straight(-50)
elif(cs_front.reflection()>ref_high and cs_right.reflection()>ref_low and
cs_right.reflection()<ref_high ):
#print("22222222222")
# if (ref>0 or ref<0):
# n=n+1
# if n>5000:
# robot.turn(-ref)
```

```
# ref=0
robot.drive(80,0)
elif(cs_front.reflection()>ref_low and cs_front.reflection()<ref_high and
cs_right.reflection()>ref_high ):
print("33333333333")
stop_robot()
robot.straight(-50)
stop_robot()
robot.turn(90-ref)
ref=0
elif(cs_front.reflection()>ref_high and cs_right.reflection()>ref_high):
robot.straight(18)
robot.turn(sideangle)
ref=ref+sideangle
if(cs_right.reflection()>ref_low and cs_right.reflection()<ref_high):
#robot.drive(80,0)
pass
#wait()
else:
robot.turn(-sideangle)
robot.turn(-sideangle)
ref=ref-2*sideangle
if(cs_right.reflection()>ref_low and cs_right.reflection()<ref_high):
pass
else:
print("44444444444")
print("RRRRR")
print(ref)
stop_robot()
robot.turn(-ref)
robot.straight(175)
stop_robot()
robot.turn(90)
ref=0
robot.straight(70)
stop_robot()
```

Challenges faced and their solution:

Challenges	Solutions
Fixing reference was difficult as there was no gyroscope sensor. Due to this after multiple turns, there was a deviation in the position of the robot concerning the initial reference.	We measured deviation and compensated it.
The resolution of the color sensor was very poor.	So we made a design that can measure the color perfectly. If the reflection value of floor and maze is more, it is detecting the maze properly.