Utkarsh pandey - 22BRS1111

Hrijumana Dubey – 22BRS1187

Ankit kumar – 22BRS1130

Lab-9

**Use o/p of lab8 and find shortest path between source and destination**

1.Arduino Code

#include <NewPing.h>

// === Ultrasonic Sensor ===

#define TRIG\_PIN A5

#define ECHO\_PIN A4

#define MIN\_DISTANCE 2 // cm

#define MAX\_DISTANCE 200 // cm

NewPing sonar(TRIG\_PIN, ECHO\_PIN, MAX\_DISTANCE);

// === Motor Pins (for rotation) ===

#define IN1 5

#define IN2 6

#define IN3 7

#define IN4 8

#define ENA 9

#define ENB 11

// === Mapping Settings ===

#define STEPS 8 // number of rotation steps (360° / 8 = 45° per step)

#define CELL\_SIZE 20 // cm per grid cell

void setup() {

Serial.begin(9600);

pinMode(IN1, OUTPUT);

pinMode(IN2, OUTPUT);

pinMode(IN3, OUTPUT);

pinMode(IN4, OUTPUT);

pinMode(ENA, OUTPUT);

pinMode(ENB, OUTPUT);

}

void loop() {

for (int i = 0; i < STEPS; i++) {

// Read ultrasonic sensor

int distance = sonar.ping\_cm();

if(distance < MIN\_DISTANCE) distance = MIN\_DISTANCE;

if(distance > MAX\_DISTANCE) distance = MAX\_DISTANCE;

// Convert distance to grid cells relative to car

int obsX = round((distance \* cos(i \* 2 \* 3.14159 / STEPS)) / CELL\_SIZE);

int obsY = round((distance \* sin(i \* 2 \* 3.14159 / STEPS)) / CELL\_SIZE);

// Send obstacle coordinates to Raspberry Pi

Serial.print(obsX);

Serial.print(",");

Serial.print(obsY);

Serial.print(",");

Serial.println(distance);

// Rotate car by one step

rotateStep(360.0 / STEPS);

}

while (1); // Stop after one full 360° scan

}

// Rotate car in place by given degree

void rotateStep(float degree) {

analogWrite(ENA, 150);

analogWrite(ENB, 150);

digitalWrite(IN1, HIGH);

digitalWrite(IN2, LOW);

digitalWrite(IN3, LOW);

digitalWrite(IN4, HIGH);

int delayTime = degree \* 10; // adjust for your kit

delay(delayTime);

stopMotors();

}

void stopMotors() {

analogWrite(ENA, 0);

analogWrite(ENB, 0);

}

2.Raspberry pi code

lab9

import numpy as np

import matplotlib.pyplot as plt

import heapq

import serial

import time

import math

# === CONFIG ===

MAP\_FILE = "lab8\_map.csv"

START = (10, 10)

GOAL = (15, 18)

PORT = "/dev/ttyACM0"

BAUD = 9600

CELL\_SIZE = 20

MOVE\_DELAY = 1.0

# === LOAD MAP ===

grid = np.loadtxt(MAP\_FILE, delimiter=",")

rows, cols = grid.shape

# === HELPER FUNCTIONS ===

def heuristic(a, b):

return math.hypot(a[0]-b[0], a[1]-b[1])

def get\_neighbors(node):

neighbors = []

directions = [(-1,0),(1,0),(0,-1),(0,1),(-1,-1),(-1,1),(1,-1),(1,1)]

for dr, dc in directions:

nr, nc = node[0]+dr, node[1]+dc

if 0 <= nr < rows and 0 <= nc < cols:

if grid[nr][nc] == 0:

neighbors.append((nr,nc))

return neighbors

def a\_star(start, goal):

open\_set = []

heapq.heappush(open\_set, (0, start))

came\_from = {}

g\_score = {start:0}

f\_score = {start:heuristic(start, goal)}

while open\_set:

\_, current = heapq.heappop(open\_set)

if current == goal:

path = []

while current in came\_from:

path.append(current)

current = came\_from[current]

path.append(start)

path.reverse()

return path

for neighbor in get\_neighbors(current):

cost = 1 if abs(neighbor[0]-current[0]) + abs(neighbor[1]-current[1]) == 1 else 1.414

tentative\_g = g\_score[current] + cost

if neighbor not in g\_score or tentative\_g < g\_score[neighbor]:

came\_from[neighbor] = current

g\_score[neighbor] = tentative\_g

f\_score[neighbor] = tentative\_g + heuristic(neighbor, goal)

heapq.heappush(open\_set, (f\_score[neighbor], neighbor))

return None

def get\_angle(curr, next\_cell):

dy = next\_cell[0]-curr[0]

dx = next\_cell[1]-curr[1]

angle = math.degrees(math.atan2(dy, dx))

return angle

# === CONNECT TO ARDUINO ===

ser = serial.Serial(PORT, BAUD, timeout=1)

time.sleep(2)

# === RUN A\* ===

path = a\_star(START, GOAL)

if path is None:

print("No path found!")

exit()

# === SEND PATH COMMANDS ===

current\_angle = 0

for i in range(1, len(path)):

curr = path[i-1]

next\_cell = path[i]

target\_angle = get\_angle(curr, next\_cell)

rotate\_angle = target\_angle - current\_angle

current\_angle = target\_angle

# Rotate

ser.write(f"ROTATE {int(rotate\_angle)}\n".encode())

time.sleep(MOVE\_DELAY)

# Move forward 1 cell

ser.write(f"MOVE {CELL\_SIZE}\n".encode())

time.sleep(MOVE\_DELAY)

# === VISUALIZE PATH ===

for step in path:

grid[step[0]][step[1]] = 2

plt.figure(figsize=(6,6))

cmap = plt.cm.get\_cmap("gray\_r", 3)

plt.imshow(grid, cmap=cmap, origin="lower")

plt.scatter(START[1], START[0], c='green', marker='o', label='Start')

plt.scatter(GOAL[1], GOAL[0], c='red', marker='x', label='Goal')

plt.legend()

plt.title("Lab 9: A\* Path + Smart Car Commands")

plt.show()

Github link for the implementation :