Utkarsh pandey - 22BRS1111

Hrijumana Dubey – 22BRS1187

Ankit kumar – 22BRS1130

Lab-8

**Create a map data structure with details**

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

import serial

import numpy as np

import matplotlib

matplotlib.use("TkAgg")

import matplotlib.pyplot as plt

PORT = "/dev/ttyACM0" # Arduino port

BAUD = 9600

CELL\_SIZE = 20

MAX\_DISTANCE\_CM = 200

# Grid setup

num\_cells\_side = int(MAX\_DISTANCE\_CM / CELL\_SIZE)

GRID\_WIDTH = num\_cells\_side \* 2 + 1

GRID\_HEIGHT = num\_cells\_side \* 2 + 1

carX = GRID\_WIDTH // 2

carY = GRID\_HEIGHT // 2

grid = np.zeros((GRID\_HEIGHT, GRID\_WIDTH))

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

plt.ion()

fig, ax = plt.subplots()

try:

while True:

line = ser.readline().decode().strip()

if not line:

continue

try:

obsX\_str, obsY\_str, distance\_str = line.split(",")

obsX = int(obsX\_str)

obsY = int(obsY\_str)

distance = int(distance\_str)

if distance < 2 or distance > MAX\_DISTANCE\_CM:

continue

gridX = carX + obsX

gridY = carY + obsY

if 0 <= gridX < GRID\_WIDTH and 0 <= gridY < GRID\_HEIGHT:

grid[gridY][gridX] = 1

ax.clear()

ax.imshow(grid, cmap="gray\_r", origin="lower")

ax.plot(carX, carY, "ro")

ax.set\_title("Lab 8: Single-Position Occupancy Grid")

plt.draw()

plt.pause(0.05)

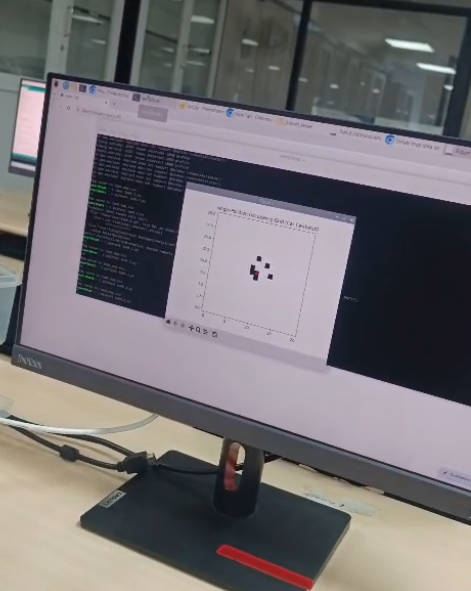
except ValueError:

continue

except KeyboardInterrupt:

np.savetxt("lab8\_map.csv", grid, fmt='%d', delimiter=",")

print("Map saved to lab8\_map.csv")



Github link for the implementation :