Rares Teodor Popa -> Mapping Robot

Introducere

Prezentarea pe scurt a proiectului vostru:

- Robotul foloseste 3 senzori de distante pe care ii invarte pentru a crea o imagine a camerei in care se afla
- Ideea originala a fost robotul roomba care face un proces asemanator pentru a scana casa si a afla drumul pe care il va urma
- Rezultatul acestuia poate fi folosit de alte echipamente din casa carora o mapare a unei camare le-ar folosi in calculele pe care le realizeaza, precum un termostat, pentru a cunoaste zonele unde caldura nu ar trebui sa ajunga, sau un aparat de aer conditionat, etc.

Descriere generală

Prima data sunt initializati toti senzorii, modulul sd si servo-ul. Servo-uleste rotit de un numar limitat de ori, iar la fiecare pasi sunt preluate distantele oferite de senzorii de distanta si salvate rezultatele pe cardul micro sd.

Hardware Design

Piese folosite:

- x3 Senzori ultrasonici de distanta HC-SR04
- x1 Micro Servo SG90
- x1 Breadboard
- x1 Modul Slot Card Compatibil cu MicroSD
- x1 Placa Arduino Uno

Software Design

Descrierea codului aplicaţiei (firmware):

- mediu de dezvoltare: Arduino IDE
- scripturi externe: Am folosit un script de python extern pentru a mapa rezultatele oferite de robot
- functii implementate: getDistance - preia distanta data de un senzor, getModifications - parcurge toti senzorii de distanta, preia distanta si marcheaza obstacolul
- functionare cod:
  - In setup sunt initializate cardul micro SD, senzorii de distanta si servo-ul
  - Se roteste servo-ul pana la 180 de grade, la fiecare pas sunt preluate distantele din senzori
  - se calculeaza locatia reala a obiectului oferit de senzorul de distanta
  - si apoi locatia este salvata in card

```
#include <SPI.h>
#include <SD.h>
#include <Servo.h>

const int TURN_TAKES = 2;
const int DIST_TAKES = 1;
const int ROOM_MATRIX_SIZE = 800;
const int MAX_GRADE = 180;
const int NO_MOD = MAX_GRADE * 4;
const int SDPin = 10;
const int servoPin = 9;
const int echoPins[] = { 2, 4, 6 };
const int trigPins[] = { 3, 5, 7 };

char *roomMapName = "room_map.txt";
File roomMapFile;
struct point {
  double x, y;
};

double getDistance(int dir) {
  double mean = 0;
  for (int i = 0; i < DIST_TAKES; i++) {
    digitalWrite(trigPins[dir], HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPins[dir], LOW);
    mean += pulseIn(echoPins[dir], HIGH) / 58.0;
  }
  mean /= DIST_TAKES;
  return mean;
}
```

```
void getModifications(int grade) {
  double dist[3];
  struct point location;
  unsigned long fileLocation;
  for (int dir = 0; dir < 3; dir++ ) {
    dist[dir] = getDistance(dir);
    Serial.print(dir);
    Serial.print(" ");
    Serial.println(dist[dir]);
    if (dist[dir] > 400 || dist[dir] < 2) {
      continue;
    }
    location.x = 400 + cos(radians(grade + dir * 120)) * (dist[dir] + 2.5);
    location.y = 400 + sin(radians(grade + dir * 120)) * (dist[dir] + 2.5);
    if (location.x >= ROOM_MATRIX_SIZE || location.y >= ROOM_MATRIX_SIZE ||
location.x < 0 || location.y < 0 )
      continue;
    roomMapFile.print((int) location.x, 10);
    roomMapFile.print(" ");
    roomMapFile.println((int) location.y, 10);
  }
}

void setup() {
  int i, j, k, grade;
  Servo servo;
```

```
  // initialize Serial Monitor
  Serial.begin(9600);
  while (!Serial) {
    ; // wait for serial port to connect. Needed for native USB port only
  }
```

```
  Serial.println("Started Initialization..");
```

```
  // Initialize SD card and get room map file
  if (!SD.begin(10)) {
    Serial.println("initialization failed!");
    while (1);
  }
  // delete old files as we need new ones to operate
  if (SD.exists(roomMapName)) {
    SD.remove(roomMapName);
  }
  // create starting matrix of the room with no walls or obstacles
  roomMapFile = SD.open(roomMapName, O_RDWR | O_CREAT);
```

```
  Serial.println("File initiated");
```

```
  // Initialize Servo Motor
```

```
  servo.attach(9);
  servo.write(0);
```

```
  // Initialize the Ultrasonic distance senzors
  for (i = 0; i < 3; i++) {
    pinMode(trigPins[i], OUTPUT);
    pinMode(echoPins[i], INPUT);
  }
```

```
  for (i = 0; i < TURN_TAKES; i++) {
    // First turn
    for (grade = 0; grade < MAX_GRADE; grade++) {
      servo.write(grade);
      delay(15);
      getModifications(0.75 * grade);
    }
```

```
    // Second turn
    for (grade = MAX_GRADE; grade > 0; grade--) {
      servo.write(grade);
      delay(15);
      getModifications(0.8 * grade);
    }
  }
```

```
  roomMapFile.flush();
  roomMapFile.close();
  Serial.println("Finished mapping");
}
```

```
void loop() {
}

import matplotlib.pyplot as plt
mapFile = open("ROOM_MAP.TXT", "r")
xPoints = []
yPoints = []
for line in mapFile.readlines():
    x = int(line.split(" ")[0])
    y = int(line.split(" ")[1])
    xPoints.append(x)
    yPoints.append(y)
plt.plot(xPoints, yPoints, 'o')
plt.show()
```

Rezultate Obţinute

Care au fost rezultatele obţinute în urma realizării proiectului vostru.

Concluzii

Download

O arhivă (sau mai multe dacă este cazul) cu fişierele obţinute în urma realizării proiectului: surse, scheme, etc. Un fişier README, un ChangeLog, un script de compilare şi copiere automată pe uC crează întotdeauna o impresie bună .

Fişierele se încarcă pe wiki folosind facilitatea **Add Images or other files**. Namespace-ul în care se încarcă fişierele este de tipul **:pm:prj20??:c?** sau **:pm:prj20??:c?:nume_student** (dacă este cazul). **Exemplu:** Dumitru Alin, 331CC → **:pm:prj2009:cc:dumitru_alin**.

Jurnal

Puteți avea și o secțiune de jurnal în care să poată urmări asistentul de proiect progresul proiectului.

Bibliografie/Resurse

Listă cu documente, datasheet-uri, resurse Internet folosite, eventual grupate pe **Resurse Software** şi **Resurse Hardware**.

Export to PDF