IoT Rendszerek

Tartalom

Név: Sümegi Bence

Az ötlet rövid leírása:

A projekt célja egy ultrahangos (szonár alapú) objektumdetektáló rendszer fejlesztése, amely képes egy meghatározott tartományon belül az akadályok felderítésére és távolságuk meghatározására.

Hozzávalók és költségvetés

Alkatrészlista költségvetéssel:

- mikrokontroller
- szervó motor
- ultrahangos szenzor

További elkészítendő:

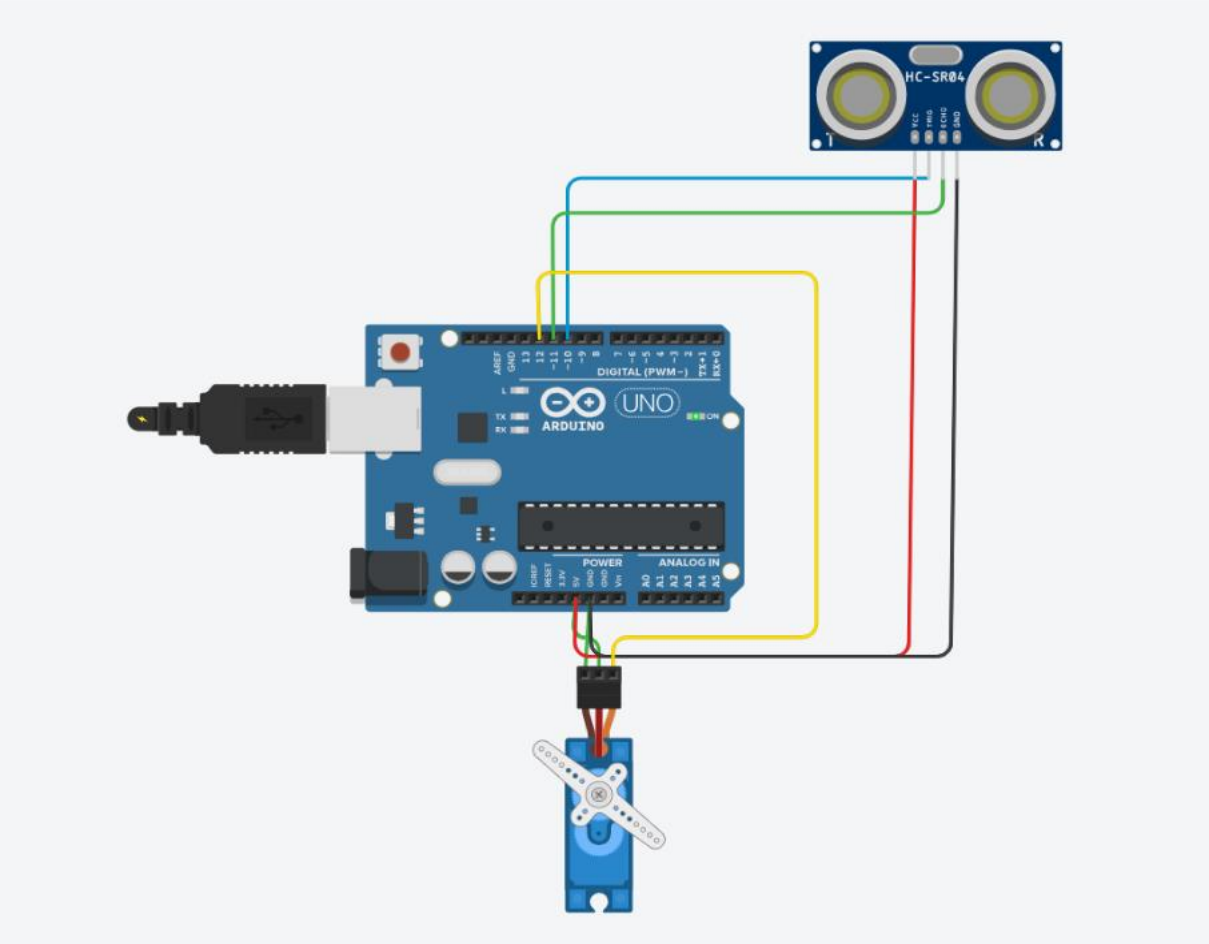
- program (Arduino IDE, Processing IDE)

Működési elv

A rendszer működésének alapja két kulcsfontosságú egység, az ultrahangos távolságmérő és a szervomotor precíz együttműködése. A folyamat lelke a HC-SR04 szenzor, amely a vezérlőegység utasítására emberi fül számára hallhatatlan, 40 kHz-es hangimpulzusokat bocsát ki. Amikor ezek a hullámok egy akadályba ütköznek, visszaverődnek, a szenzor pedig érzékeli a visszhangot. Mivel a hang terjedési sebessége ismert, a vezérlőprogram a jel kibocsátása és visszaérkezése között eltelt időből pontosan kiszámítja a tárgy távolságát. Ezt a mérési folyamatot egészíti ki a térbeli tájékozódásért felelős szervomotor, amely a kapott vezérlőjel hosszától függően – jellemzően 1 és 2 ezredmásodperc közötti tartományban – áll be a kívánt szögbe. A motor belső elektronikája folyamatosan felügyeli a tengely pozícióját, így biztosítva a stabil iránytartást. A két eszköz összehangolt munkájának köszönhetően a szervomotor folyamatosan pásztázza a környezetet, miközben a szenzor rendszeres időközönként méréseket végez, lehetővé téve az akadályok valós idejű, radarszerű feltérképezését a megadott hatósugáron belül.

Kapcsolási rajz

1. Kép Arduino Kapcsolási rajz



Forrás: TinkerCad

Arduino IDE Kód

```
  delay(10);

  // Measure distance
  int distance = CalculateDistance();

  // Output via serial
  SerialOutput(motorAngle, distance);

  // Update angle
  motorAngle += motorRotateAmount;

  // Reverse direction at limits
  if (motorAngle <= minimumAngle || motorAngle >= maximumAngle) {
    motorRotateAmount = -motorRotateAmount;
  }
}

int CalculateDistance() {
  // Send ultrasonic pulse
  digitalWrite(TriggerPin, LOW);
  delayMicroseconds(2);
  digitalWrite(TriggerPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(TriggerPin, LOW);

  // Listen for echo with timeout (25ms = ~4m)
  long duration = pulseIn(EchoPin, HIGH, 25000);

  // If no echo received
  if (duration == 0) return -1;

  // Distance in cm (Sound speed = 343 m/s)
  float distance = duration * 0.01715; // more precise constant

  return int(distance);
}

void SerialOutput(int angle, int distance) {
  Serial.print(angle);
  Serial.print(",");
  Serial.println(distance);
}
```

```
#include <Servo.h>   // <-- Missing include

const int TriggerPin = D2;
const int EchoPin = D1;

const int motorSignalPin = D4;
const int startingAngle = 90;

const int minimumAngle = 6;
const int maximumAngle = 175;

const int rotationSpeed = 1;

Servo motor;

void setup() {
  pinMode(TriggerPin, OUTPUT);
  pinMode(EchoPin, INPUT);

  motor.attach(motorSignalPin);

  Serial.begin(9600);

  // Make sure trigger pin starts LOW
  digitalWrite(TriggerPin, LOW);
  delay(50);
}

void loop() {
  static int motorAngle = startingAngle;
  static int motorRotateAmount = rotationSpeed;

  // Move servo
  motor.write(motorAngle);
  delay(10);
```

```
}

void serialEvent(Serial myPort) {
  try {
    String data = myPort.readStringUntil('\n');
    if (data == null) return;
    int comma = data.indexOf(",");
    if (comma == -1) return;
    String angle = data.substring(0, comma).trim();
    String distance = data.substring(comma + 1).trim();
    iAngle = StringToInt(angle);
    iDistance = StringToInt(distance);
  } catch (Exception e) {
    println("Serial error: " + e);
  }
}
// ------------------ RADAR DRAW ------------------
void DrawRadar() {
  pushMatrix();
  translate(width/2, 0.926 * height);
  noFill();
  strokeWeight(2);
  stroke(98, 245, 31);
  DrawRadarArcLine(0.9375);
  DrawRadarArcLine(0.7300);
  DrawRadarArcLine(0.5210);
  DrawRadarArcLine(0.3130);
  final int halfWidth = width/2;
  line(-halfWidth, 0, halfWidth, 0);
  for (int angle = 30; angle <= 150; angle += 30)
    DrawRadarAngledLine(angle);
  popMatrix();
}
void DrawRadarArcLine(final float coef) {
```

```
import processing.serial.*;
import java.awt.event.KeyEvent;
import java.io.IOException;

Serial myPort;
PFont orcFont;
int iAngle;
int iDistance;

void setup() {
  size(1000, 500);
  smooth();

  orcFont = createFont("Arial", 30, true);
  textFont(orcFont);

  myPort = new Serial(this, "COM4", 9600);
  myPort.clear();
  myPort.bufferUntil('\n');
}

void draw() {
  fill(98, 245, 31);
  textFont(orcFont);
  noStroke();

  // background fade effect
  fill(0, 4);
  rect(0, 0, width, 0.935 * height);
  fill(98, 245, 31);
  DrawRadar();
  DrawLine();
  DrawObject();
  DrawText();
```

Processing Kód

```
void DrawRadarArcLine(final float coef) {
  arc(0, 0, coef * width, coef * width, PI, TWO_PI);
}
void DrawRadarAngledLine(final int angle) {
  line(0, 0, (-width/2) * cos(radians(angle)),
             (-width/2) * sin(radians(angle)));
}
// ------------------- OBJECT DRAW ------------------------
void DrawObject() {
  pushMatrix();
  translate(width/2, 0.926 * height);
  strokeWeight(9);
  stroke(255, 10, 10);
  if (iDistance > 0 && iDistance <= 40) {  // full sensor range
    int pixDist = int(iDistance * 0.020835 * height);
    float cx = cos(radians(iAngle));
    float cy = sin(radians(iAngle));
    int x1 = int(pixDist * cx);
    int y1 = int(-pixDist * cy);
    int x2 = int(0.495 * width * cx);
    int y2 = int(-0.495 * width * cy);
    line(x1, y1, x2, y2);
  }
  popMatrix();
}
// ------------------- SWEEP LINE ------------------------
void DrawLine() {
  pushMatrix();
  strokeWeight(9);
  stroke(30, 250, 60);
  translate(width/2, 0.926 * height);
  float angle = radians(iAngle);
  int x = int(0.88 * height * cos(angle));
  int y = int(-0.88 * height * sin(angle));

line(0, 0, x, y);
  popMatrix();
}
 // ------------------- TEXT DRAW ------------------------
void DrawText() {
  pushMatrix();
  fill(0, 0, 0);
  noStroke();
  rect(0, 0.9352 * height, width, height);
  fill(98, 245, 31);
  textSize(25);
  text("10cm", 0.6146 * width, 0.9167 * height);
  text("20cm", 0.7190 * width, 0.9167 * height);
  text("30cm", 0.8230 * width, 0.9167 * height);
  text("40cm", 0.9271 * width, 0.9167 * height);
  textSize(40);
  // STATUS
  if (iDistance == 0) {
    text("Object: No Echo", 0.125 * width, 0.9723 * height);
  } else {
    text("Object: In Range", 0.125 * width, 0.9723 * height);
  }
  // ANGLE
  text("Angle: " + iAngle + "°", 0.52 * width, 0.9723 * height);
  // DISTANCE
  text("Distance: " + iDistance + " cm", 0.74 * width, 0.9723 * height);
  // Angle labels (30°, 60°, 90°, 120°, 150°)
  textSize(25);
  fill(98, 245, 60);
  drawAngleLabel(30);
  drawAngleLabel(60);
  drawAngleLabel(90);
  drawAngleLabel(120);
  drawAngleLabel(150);
  popMatrix();
}
void drawAngleLabel(int ang) {
  resetMatrix();
  float x = 0.5006 * width + width/2 * cos(radians(ang));
  float y = 0.9093 * height - width/2 * sin(radians(ang));
  translate(x, y);
  rotate(-radians(ang - 90));
  text(ang + "°", 0, 0);
}
int StringToInt(String s) {
  int val = 0;
  for (int i = 0; i < s.length(); i++) {
    char c = s.charAt(i);
    if (c >= '0' && c <= '9')
      val = val * 10 + (c - '0');
  }
  return val;
}
```

Fejlesztési lehetőségek

- Szenzor stabilabb rögzítése a szervó motorra.
- 360 fokos szkennelés

Önreflexió

Az IoT tárgy legfontosabb tanulsága számomra a szoftver és a fizikai világ összekapcsolása volt. A radaros projekt során megtapasztaltam, hogy a precíz működéshez nem elég a jó kód, a hardveres időzítésen és a türelmes tesztelésen is sok múlik. A működő rendszer látványa végül magabiztossá és motiválttá tett az elektronikai fejlesztések terén.