

# Dunaújvárosi Egyetem Bánki Donát Technikum

## Projekt feladat dokumentáció

### Tartalom

Az ötlet rövid leírása:.....	1
Hozzávalók és költségvetés .....	2
Működési elv.....	2
Kapcsolási rajz.....	2
Kód példa .....	3
Fejlesztési lehetőségek .....	5

Tantárgy neve: IoT

Projekt tervezői: Fogas Bence

Projekt címe: Radar

Osztály: 13.B

Dátum: 2025.12.09.

## Az ötlet rövid leírása:

A projekt célja egy sonar alapú radar készítése, amely alkalmas egy előre beállított rádiuszban érzékelni a tárgyakat.

## Hozzávalók és költségvetés

Alkatrészlista költségvetéssel:

- mikrokontroller
- szervó motor
- ultrahangos szenzor

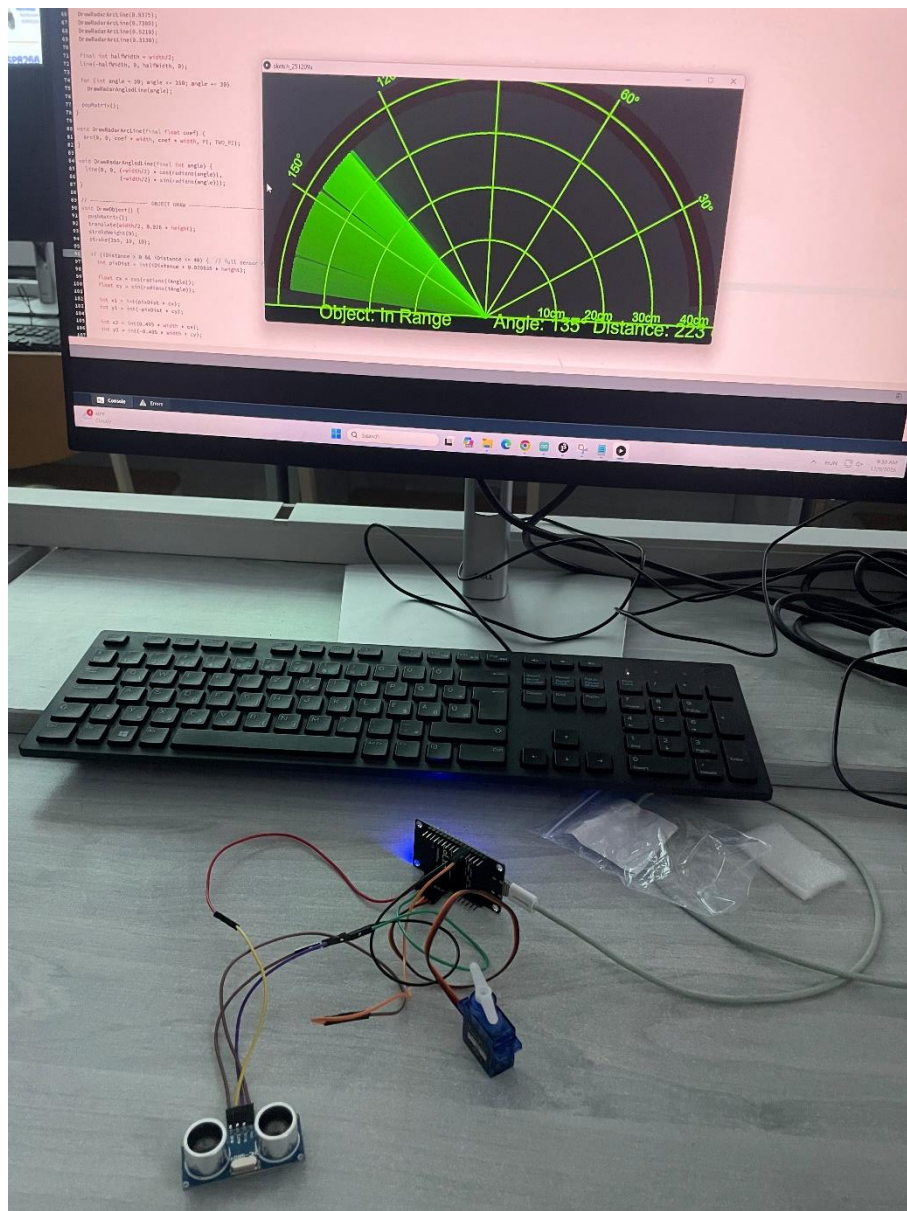
További elkészítendő:

- program (Arduino IDE, Processing IDE)

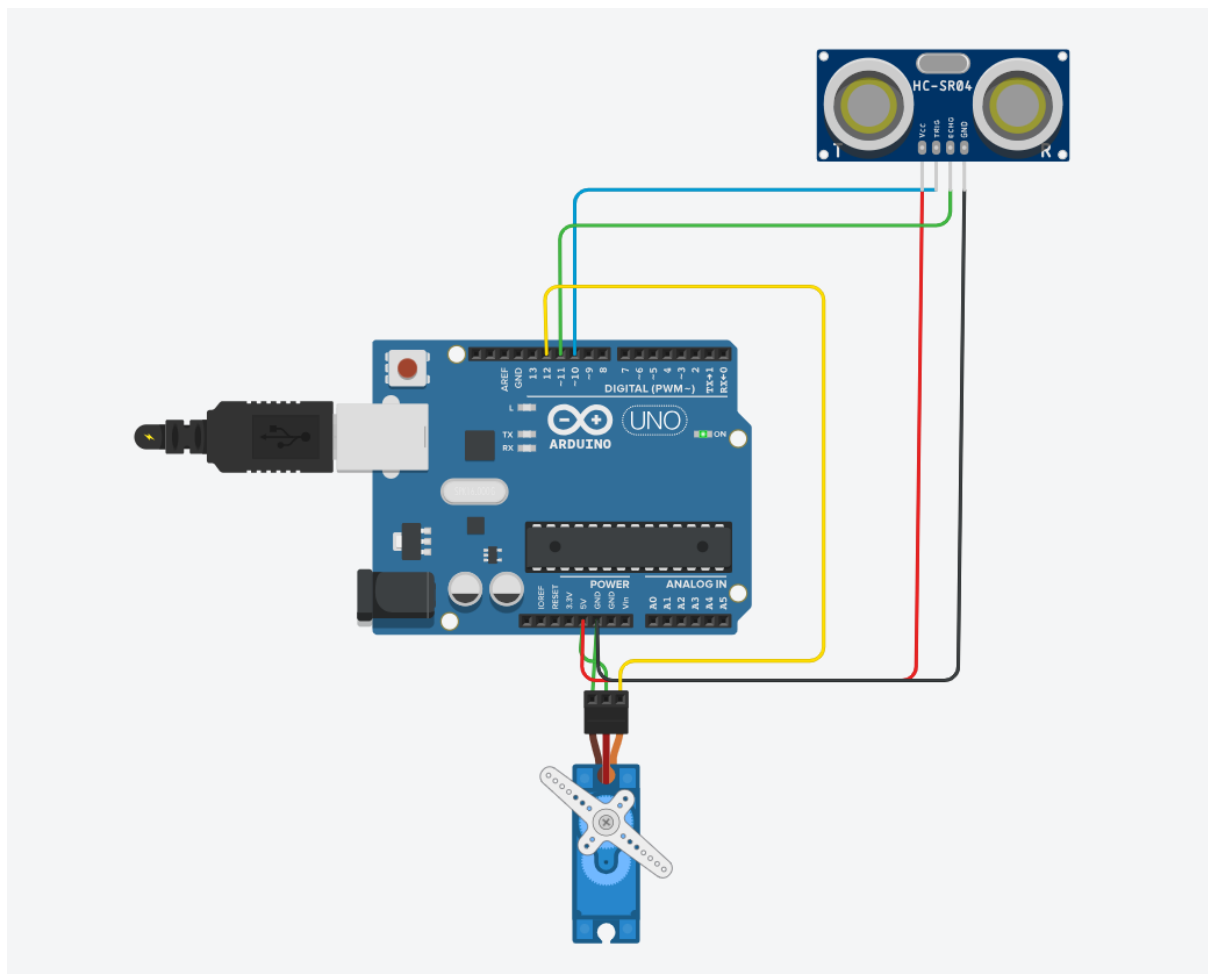
## Működési elv

Az ultrahangos szenzor és a szervómotor működése szorosan összefügg, amikor például akadályérzékelő vagy követőrendszer készül. Az ultrahangos távolságmérő szenzor – legelterjedtebb típus például a HC-SR04 – hangon kívüli frekvenciájú (általában 40 kHz-es) impulzusokat bocsát ki a trig jel hatására. Ezek a hanghullámok terjednek a levegőben, majd visszaverődnek egy akadály felületéről. A szenzor echo kimenete jelzi vissza a mikrovezérlőnek, hogy mennyi idő telt el az impulzus kibocsátása és visszaérkezése között. Mivel a hang terjedési sebessége a levegőben nagyjából 343 m/s, az időmérésből egyszerű számítással meghatározható a tárgy távolsága. A szenzor tehát valójában nem a távolságot „méri közvetlenül”, hanem az időt, amelyből a távolságot a vezérlőprogram számolja ki.

A szervómotor ezzel szemben egy apró, beépített elektronikával ellátott motor, amely pontos szögelfordulásra képes. A motor három vezetéken keresztül kapja a működéséhez szükséges jeleket: táp, föld, valamint egy vezérlő PWM jel. A szervó a PWM jel kitöltési tényezőjéből határozza meg, milyen szögbe forduljon el. Például egy 1 ms-os impulzus nagyjából 0° körüli állást, míg egy 2 ms-os impulzus 180° körüli pozíciót eredményezhet, a típusától függően. A szervó folyamatosan próbálja elérni és megtartani a neki megadott pozíciót, amit a belső visszacsatoló potencióméter segít neki.



## Kapcsolási rajz



## Arduino IDE Kód

```
#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);

    // 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(TriigerPin, LOW);

    delayMicroseconds(2);

    digitalWrite(TriigerPin, HIGH);

    delayMicroseconds(10);

    digitalWrite(TriigerPin, 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);

}

```

## Processing kód

```

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();
}

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) {
    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);

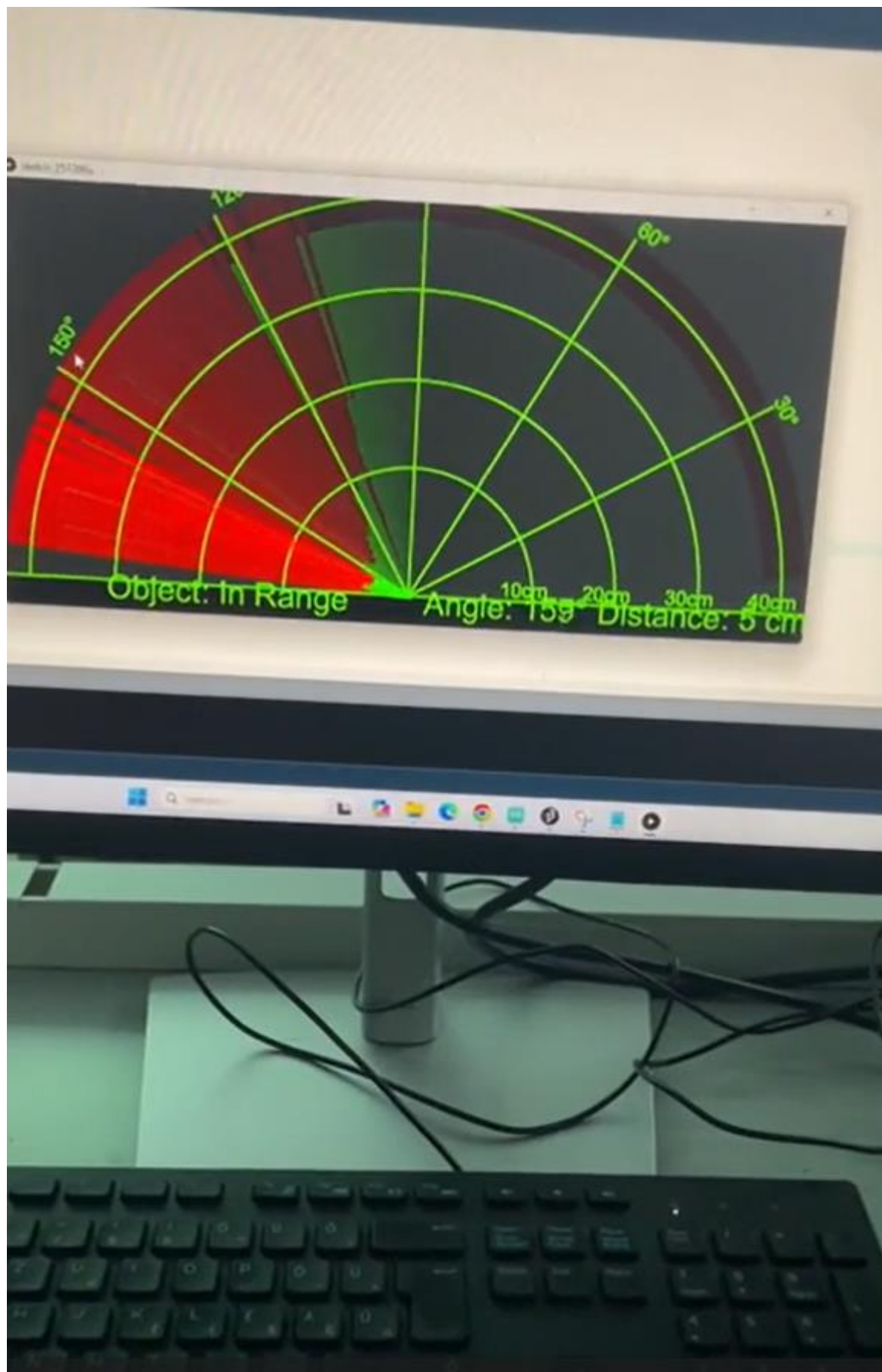
  popMatrix();
}

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

Ez a projekt rendkívül tanulságos volt számomra. Jelentősen fejlődött a problémamegoldó képességem, hiszen sok nehézséggel találkoztam a munka során. A következtetéseket levonva és a különböző dokumentációkat felhasználva sikerült a problémákat kiküszöbölni. Mivel a további tanulmányaim során is szeretnék mikrovezérlőkkel foglalkozni, örömmre szolgált egy ilyen projektben részt venni.