**Arduıno code**

#include <HCSR04.h>

#include <Servo.h>

UltraSonicDistanceSensor distanceSensor(6, 7); // 1. sensör nesnesi oluşturuldu

UltraSonicDistanceSensor distanceSensor2(5, 4); // 2. sensör nesnesi oluşturuldu

Servo servoMotor; // Servo nesnesi oluşturuldu

int delayTime = 5; // Servonun 1 derece farka ulaşması için beklenen süre

long d; // 1. sensörden ölçülen mesafe

long d2; // 2. sensörden ölçülen mesafe

void setup() {

Serial.begin(9600); // Seri iletişimi 9600 baud hızında başlat

servoMotor.attach(2); // Servo, Dijital PWM pin 2'ye bağlandı

servoMotor.write(0); // Servo 180 dereceye döndürüldü

delay(1000); // Servo, 180 dereceye ulaşması için bekledi

servoMotor.write(180); // Servo 0 dereceye döndürüldü

delay(1000); // Servo, 0 dereceye ulaşması için bekledi

}

void loop() {

for (int i = 1; i < 180; i++) { // Servo 180 derece ileri hareket ettirildi

readSensors(); // Sensörler okundu

Serial.print(i); // Açı bilgisi seri port üzerine gönderildi

Serial.print(","); // Virgül seri port üzerine gönderildi

Serial.print(d); // 1. sensörden ölçülen mesafe seri port üzerine gönderildi

Serial.print(","); // Virgül seri port üzerine gönderildi

Serial.println(d2); // 2. sensörden ölçülen mesafe seri port üzerine gönderildi (satır sonu)

servoMotor.write(i); // Servo belirtilen açıya getirildi

delay(delayTime); // Servo belirtilen açıya ulaşana kadar beklendi

}

for (int i = 180; i > 1; i--) { // Servo 180 derece geri hareket ettirildi

readSensors(); // Sensörler okundu

Serial.print(i); // Açı bilgisi seri port üzerine gönderildi

Serial.print(","); // Virgül seri port üzerine gönderildi

Serial.print(d); // 1. sensörden ölçülen mesafe seri port üzerine gönderildi

Serial.print(","); // Virgül seri port üzerine gönderildi

Serial.println(d2); // 2. sensörden ölçülen mesafe seri port üzerine gönderildi (satır sonu)

servoMotor.write(i); // Servo belirtilen açıya getirildi

delay(delayTime); // Servo belirtilen açıya ulaşana kadar beklendi

}

}

void readSensors() {

d = distanceSensor.measureDistanceCm(); // 1. sensörden mesafe ölçüldü

d2 = distanceSensor2.measureDistanceCm(); // 2. sensörden mesafe ölçüldü

}

**PROCESSİNG CODE :**

**import processing.serial.\*;**

**import static javax.swing.JOptionPane.\*;**

**Serial myPort; // Serial nesnesi oluşturuldu**

**String serialin;**

**int data[] = new int[360]; // 360 elemanlı bir 'data' dizisi oluşturuldu, RADAR verileri için**

**PFont f; // Yazı tipi nesnesi tanımlandı**

**final boolean debug = true; // Debug modu**

**void setup() {**

**String COMx, COMlist = "";**

**size(1280, 720); // Pencere boyutu ayarlandı**

**f = createFont("Verdana", 32, true); // Yazı tipi oluşturuldu**

**textFont(f, 20); // Yazı tipi boyutu ve özellikleri belirlendi**

**frameRate(60); // Çalışma hızı ayarlandı**

**for (int i = 0; i < 360; i++) {**

**data[i] = 0; // 'data' dizisinin her elemanı sıfırlandı, başlangıç değeri**

**}**

**try {**

**if (debug) printArray(Serial.list()); // Debug modunda seri port listesi yazdırıldı**

**int i = Serial.list().length; // Mevcut seri port sayısı alındı**

**if (i != 0) {**

**if (i >= 2) {**

**// Hangi portun kullanılacağını belirlemek için kullanıcıdan girdi alındı**

**for (int j = 0; j < i; ) {**

**COMlist += char(j+'a') + " = " + Serial.list()[j];**

**if (++j < i) COMlist += ", ";**

**}**

**COMx = showInputDialog("Doğru COM portu hangisi? (a,b,..):\n"+COMlist); // Kullanıcıya hangi COM portunun doğru olduğunu sormak için iletişim kutusu gösterildi**

**if (COMx == null) exit(); // Eğer kullanıcı seçim yapmazsa uygulamadan çıkılır**

**if (COMx.isEmpty()) exit(); // Eğer kullanıcı seçim yapmazsa uygulamadan çıkılır**

**i = parseInt(COMx.toLowerCase().charAt(0) - 'a') + 1; // Seçilen port numarası alındı**

**}**

**String portName = Serial.list()[i-1]; // Seçilen seri portun adı alındı**

**if (debug) println(portName); // Debug modunda seçilen portun adı yazdırıldı**

**myPort = new Serial(this, portName, 9600); // Seçilen port üzerinden seri port oluşturuldu ve iletişim hızı belirlendi**

**myPort.bufferUntil('\n'); // Veriler CR/LF (Satır başı karakteri) gelene kadar tampona alındı**

**} else {**

**println("Cihaz PC'ye bağlı değil"); // Cihazın PC'ye bağlı olmadığına dair bir uyarı mesajı yazdırıldı**

**exit(); // Uygulamadan çık**

**}**

**} catch (Exception e) {**

**println("COM portu mevcut değil (başka bir program tarafından kullanılıyor olabilir)"); // COM portunun kullanılamaz olduğuna dair bir uyarı mesajı yazdırıldı**

**println("Hata:", e); // Oluşan hatayı yazdır**

**exit(); // Uygulamadan çık**

**}**

**}**

**void draw() {**

**// Arayüzün oluşturulması**

**background(26, 26, 36, 200); // Arka plan rengi ayarlandı**

**textSize(18); // Yazı tipi boyutu ayarlandı**

**stroke(255, 255, 255, 150); // Kenar rengi ayarlandı**

**fill(255, 50, 200, 200); // Dolgu rengi ayarlandı**

**text("Arduino RADAR 2D Görselleştirme", 20, 710); // Başlık yazısı ve konumu ayarlandı**

**text(hour(), 1050, 710); // Saat bilgisi yazıldı**

**text(":", 1075, 710); // Saat ve dakika arası ":" işareti yazıldı**

**text(minute(), 1085, 710); // Dakika bilgisi yazıldı**

**text(":", 1110, 710); // Dakika ve saniye arası ":" işareti yazıldı**

**text(second(), 1120, 710); // Saniye bilgisi yazıldı**

**fill(36, 255, 100, 200); // Daire dolgu rengi ayarlandı**

**strokeWeight(3); // Kenar kalınlığı ayarlandı**

**// RADAR dairelerinin oluşturulması**

**circle(640, 360, 700); // Ana daire**

**circle(640, 360, 600);**

**circle(640, 360, 500);**

**circle(640, 360, 400);**

**circle(640, 360, 300);**

**circle(640, 360, 200);**

**circle(640, 360, 100);**

**// RADAR verilerinin işlenmesi ve görselleştirilmesi**

**for (int i = 0; i < 360; i++) {**

**// Noktanın pozisyonunu mesafeye göre hesapla**

**float distance = map\_values(data[i]);**

**float x = 640 + distance \* cos(radians(i));**

**float y = 360 + distance \* sin(radians(i));**

**// Mesafeye göre boyutu hesapla**

**float size = 10; // Varsayılan boyut**

**if (distance < 40) {**

**size = map(distance, 0, 40, 20, 50); // Yakın mesafeler için boyutu küçült**

**} else if (distance >= 40 && distance < 80) {**

**size = map(distance, 40, 80, 10, 20); // Orta mesafeler için boyutu ayarla**

**} else {**

**size = map(distance, 80, 200, 5, 10); // Uzak mesafeler için boyutu küçült**

**}**

**// Mesafeye göre renk değerini hesapla**

**int colorValue = color(0, 255, 0); // Uzak mesafeler için varsayılan renk (Yeşil)**

**if (distance < 40) {**

**colorValue = color(255, 0, 0); // Yakın mesafeler için Kırmızı**

**} else if (distance >= 40 && distance < 80) {**

**colorValue = lerpColor(color(255, 0, 0), color(255, 255, 0), map(distance, 40, 80, 0, 1));**

**// Orta mesafeler için kırmızıdan sarıya doğru ara renkler**

**} else {**

**colorValue = color(0, 0, 0); // Uzak mesafeler için siyah**

**}**

**// Mesafeyi temsil etmek için doldurulmuş bir daire çiz**

**noStroke();**

**fill(colorValue);**

**ellipse(x, y, size, size); // Mesafeye göre dairelerin boyutunu ayarla**

**}**

**// Seri porttan gelen verilerin işlenmesi**

**while (myPort.available() > 0) {**

**serialin = myPort.readStringUntil(10);**

**try {**

**String serialdata[] = splitTokens(serialin, ",");**

**if (serialdata[0] != null) {**

**serialdata[0] = trim(serialdata[0]);**

**serialdata[1] = trim(serialdata[1]);**

**serialdata[2] = trim(serialdata[2]);**

**int i = parseInt(serialdata[0]);**

**data[179-i] = parseInt(serialdata[1]);**

**data[(179-i)+180] = parseInt(serialdata[2]);**

**}**

**} catch (java.lang.RuntimeException e) {**

**// Veri okunurken oluşabilecek hata durumunda hata yakalanır**

**// Hata durumunda bir şey yapmaz, işlem devam eder**

**}**

**}**

**}**

**// Mesafe değerlerini yeniden haritalama (yeniden boyutlandırma) fonksiyonu**

**float map\_values(float x) {**

**float in\_min = 0, in\_max = 200, out\_min = 0, out\_max = 700;**

**return (x - in\_min) \* (out\_max - out\_min) / (in\_max - in\_min) + out\_min;**

**}**