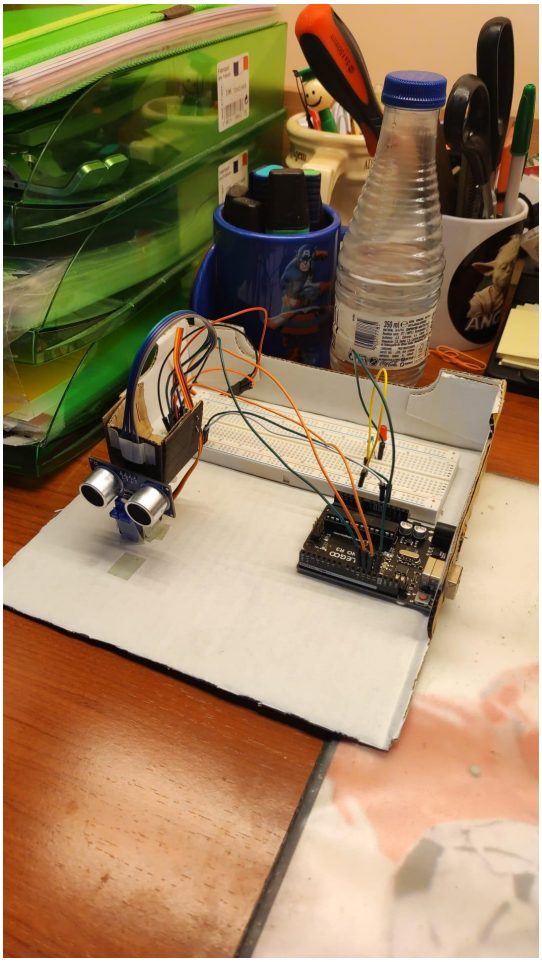
Proyecto Radar Ultrasónico "*Rasputin*"



Memoría técnica del proyecto

Ángel López Martos

18/03/2023

PDIH, Escuela Técnica Superior de Ingeniería Informática y Telecomunicaciones

Conceptos básicos

Para la realización de este proyecto, debemos de entender que el radar, se basa en microondas que se reflejan cuando encuentran un cambio de constante dieléctrica, estas ondas al reflejarse vuelven al sensor que las ha emitido y basándose en el tiempo de vuelta, el sensor es capaz de determinar la distancia a la que se encuentra el objeto contra el que ha rebotado. Este tipo de tecnología no se ve afectada de ninguna manera por variaciones de temperatura o presión en el medio ya que las ondas electromagnéticas que emite no necesitan de medio para propagarse.

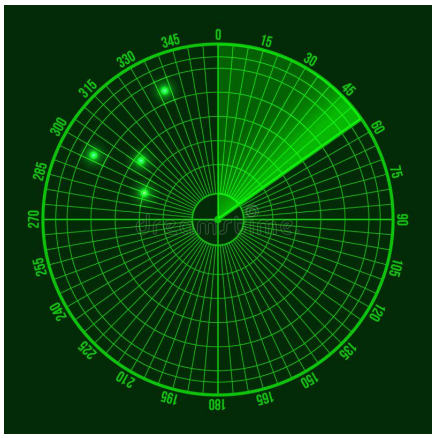
Los sensores ultrasónicos funcionan de una manera bastante similar, sin embargo, las ondas que emiten no son de microondas sino ondas mecánicas, es decir, ultrasonidos. Estas ondas se reflejan solo cuando hay un cambio de densidad en el material, vuelven al receptor y se calcula la distancia del objeto de densidad diferente de forma similar al radar. Este modelo de tecnología presenta algunos inconvenientes:

-Necesitan un medio para propagarse.

-La velocidad de propagación del sonido dependerá en parte de las condiciones del medio en el que se realicen las mediciones.

-Requiere un control genuino de la temperatura, presión, densidad, ruido externo y velocidad de propagación en el medio si se quieren tener una mediciones precisas.

Basándonos en estos conceptos, vamos a montar un sistema de “Radar” basado en ultrasonidos, el cual mostrará los resultados mediante una interfaz de control como la que se muestra:



A parte de mostrar la información clásica de un radar, iluminará distintos LEDs dependiendo del umbral de proximidad que cruce un objeto y finalmente, ante una distancia de colisión, el usuario podrá pulsar un botón sobre la interfaz gráfica con el que lanzar un ataque contra el objeto (ADS), que se efectuará con la iluminación de otro LED.

Esta tecnología está presente en muchos aparatos y sistemas que quizás no utilizamos en nuestro día a día pero que si estamos acostumbrados a describir y conocemos de su existencia tales como los radares de aplicación militar, los escáneres submarinos (sonar), detectores de movimiento y proximidad…

Materiales usados

A continuación se expone una lista con todos los materiales físicos utilizados para la realización de este proyecto:

-UNO R3 Controller Board

-Sensor Ultrasónico de compatibilidad Arduino

-Motor Servo SG90

-Protoboard 830 Tie-Points

-Female-To-Male Dupont Wires

-BreadBoard Jumper Wires

-Resistencia 220 Ohmios x 2

-LED Rojo y Azul

-Base de montaje

Para la realización del programa que se ejecuta sobre nuestro microcontrolador se han usado los siguientes recursos:

-Arduino IDE

-Tinkercad Circuits

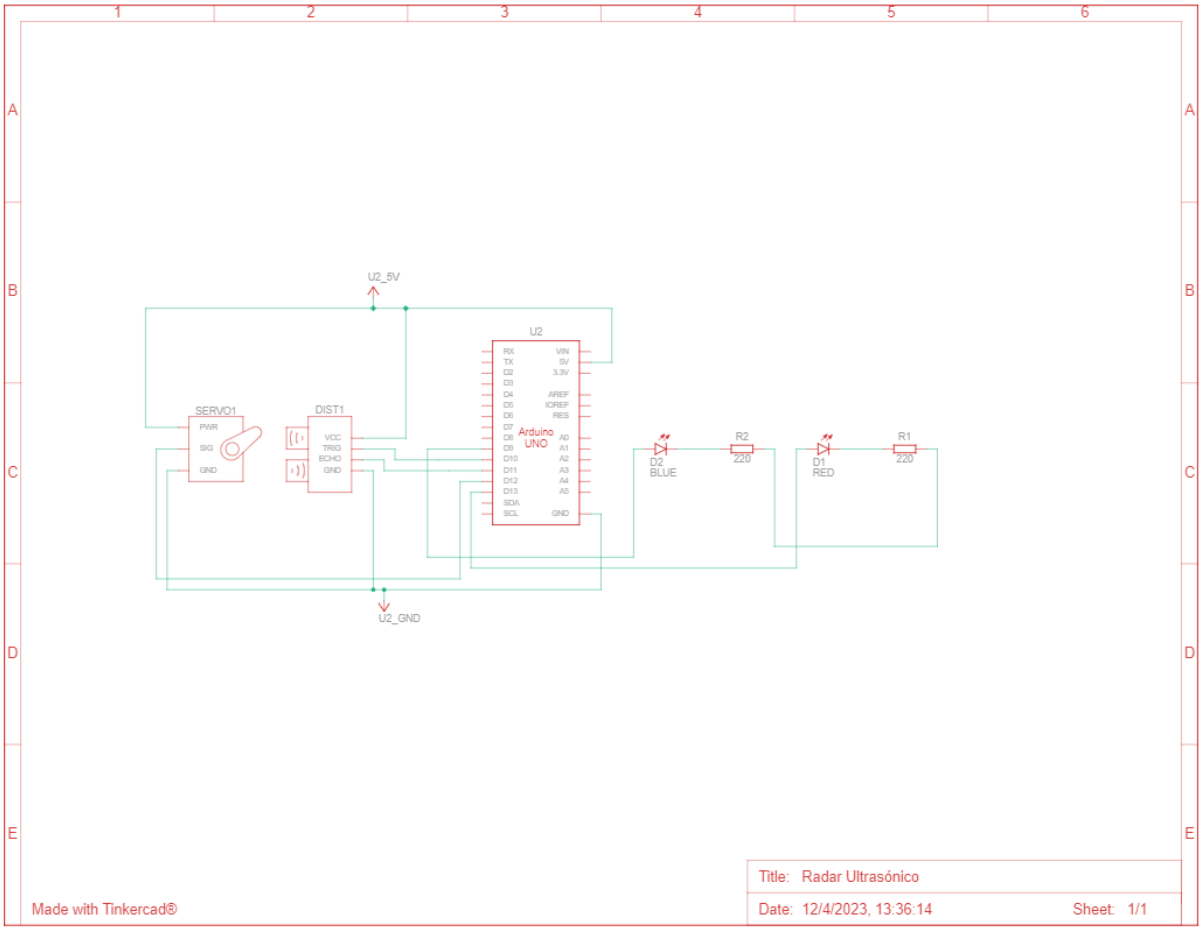
-Processing Interface

Construcción física del proyecto

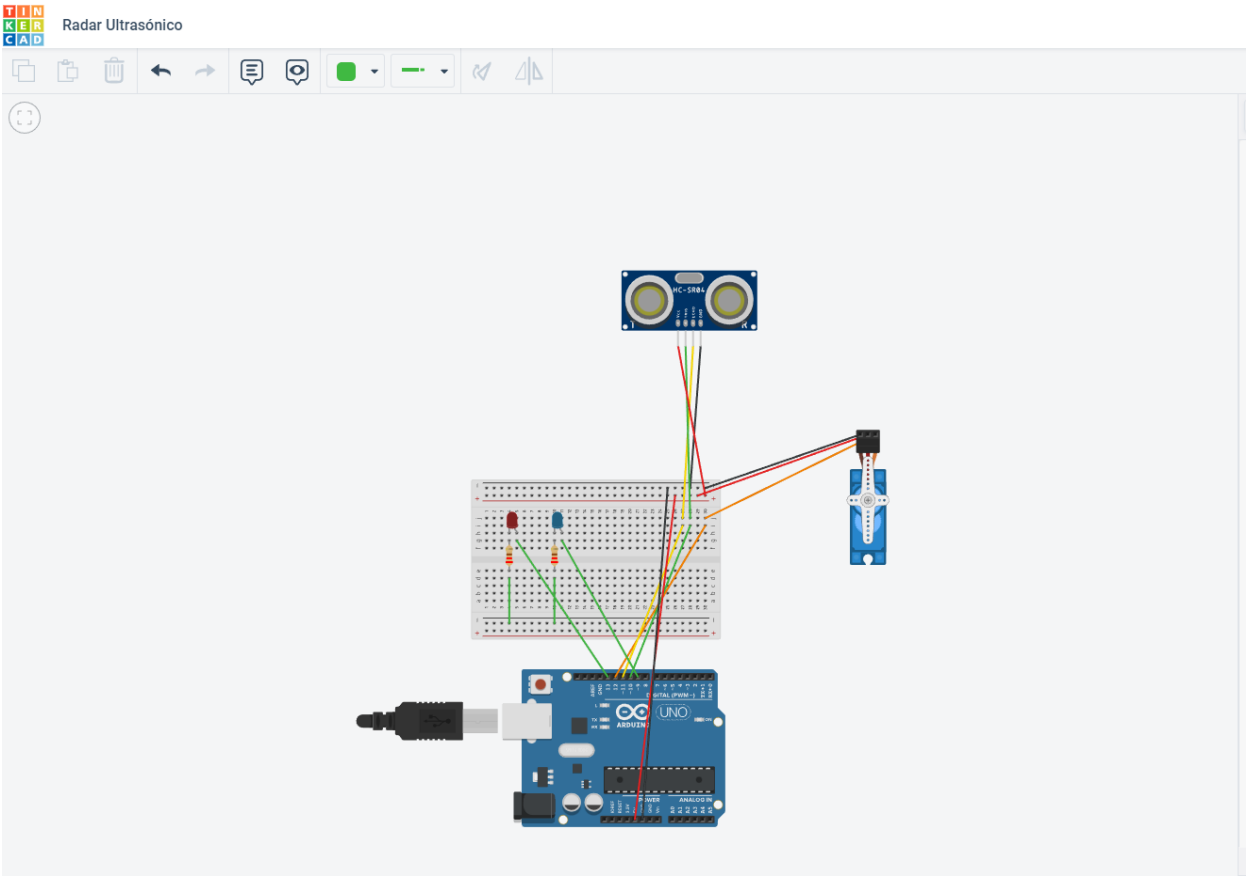
-Para comenzar con el montaje de este sistema, me he ayudado de la herramienta online de Tinkercad Circuits para hacer una primera prueba antes de montar con los componentes físicos el sistema definitivo. Con este primer montaje lo que he ensamblado ha sido el sistema estrictamente necesario para el radar (Sin circuito para LEDS) para establecer una primera funcionalidad

Cuando el circuito ya estaba listo he realizado una simulación del código en Tinkercad pero Tinkercad no me mostraba las mediciones hechas por el sensor, por lo que he pasado a un modelo físico y he probado el código en él. Una vez que esto ha funcionado, he implementado la interfaz gráfica por medio de Processing, una vez estaba listo, he añadido la funcionalidad de los LEDS tanto en el proyecto de Tinkercad como en el modelo físico y el código de ambos IDEs.

Vista Esquemática



Vista TinkerCad



Código Arduino

El código siguiente se diferencia en varias partes:

-Declaración de variables globales y de control

-Inicialización de los pines OUTPUT e INPUT, estados, comunicación serial

-Cuerpo principal; Función LOOP(), en esta función lo que vamos a incrustar son el conjunto de órdenes que va a ejecutar nuestro proyecto mientras se le suministre energía.

Como la base de este proyecto se basa en el barrido del sensor anclado a un servo y es la acción sobre la que se basa todo, nos ocupamos de esto en dos partes: Un barrido de 15 a 165 grados y otro barrido de 165 a 15. En cada uno de los grados, lo que hacemos es calcular la distancia del objeto más próximo al sensor, mediante calculate Distance, que

veremos más adelante.

A la hora de pasar los datos a Processing, necesitamos pasarlos por el puerto serial separados por un coma, que también se envía, y un punto. Todo esto se hace para que se pueda indexar, para que Processing pueda trabajar con él. Después lo que hacemos es hacer un test de distancia para iluminar el LED si se cruza un umbral de proximidad.

Ahora, comprobamos que por el puerto serial nos venga un carácter 'a', lo que significa que podemos analizar ese flujo de llegadas para cambiar el estado del led azul a led encendido.

A continuación se repite todo el proceso, para el barrido inverso

Para calcular la distancia tenemos la función calculateDistance(), con digitalWrite podemos encender y apagar el sensor para que mida la distancia, después, mide la distancia basándose en la duración del viaje del ping ida y vuelta hasta que choca con el objeto

```
#include <Servo.h>.

// Definimos los pines para el echo y el trig del sensor de ultrasonidos
const int trigPin = 10;
const int echoPin = 11;
// Definimos los pines para los leds
const int ledPin = 13;
const int ledPin2 = 9;
// Variables que vamos a utilizar para la duración y la distancia
long duration;
int distance;
bool EstadoLed = false; //variable que nos va a servir para la
iluminación de los leds

Servo myServo; // Nueva instancia del objeto Servo
```

```
void setup() {

//Definir los pines como OUTPUT e INPUT
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(ledPin, OUTPUT);
  pinMode(ledPin2, OUTPUT); //Establecemos un valor default para el led
azul
  digitalWrite(ledPin2, EstadoLed); //Establecemos un valor logico para
el led azul
  Serial.begin(9600);
  myServo.attach(12); // Definimos el pin 12 como el pin del servo
}
void loop() {
  // Rota el servo en una primera pasada de 15 a 165 grados
  for(int i=15;i<=165;i++){
  myServo.write(i);
  delay(30);
  distance = calculateDistance();// Llama a una función que calcula la
distancia respecto a los datos que llegan del sensor

  Serial.print(i); // Envia el grado actual al Serial Port
  Serial.print(","); // Envia un caracter adicional junto al valor
anterior para que Processing pueda indexarlo
  Serial.print(distance); //Manda la distancia obtenida al Serial Port
  Serial.print("."); // Envia un caracter adicional junto al valor
anterior para que Processing pueda indexarlo

  if (distance < 10) { //Si pasa un umbral, iluminamos un led
      digitalWrite(ledPin, HIGH);
    } else {
      digitalWrite(ledPin, LOW);
    }

  if(Serial.available()){ //Si se le pasa un caracter a, iluminamos o
apagamos un led
```

```
    char letra = Serial.read();
    if(letra == 'a'){
      EstadoLed = !EstadoLed;
    }

    digitalWrite(ledPin2, EstadoLed);

  }



  }
  // Repetimos el primer barrido en sentido inverso (165 a 15 grados)
  for(int i=165;i>15;i--){
    myServo.write(i);
    delay(30);
    distance = calculateDistance();
    Serial.print(i);
    Serial.print(",");
    Serial.print(distance);
    Serial.print(".");

    // Enciende el led si la distancia es menor que 10
    if (distance < 10) {
      digitalWrite(ledPin, HIGH);
    } else {
      digitalWrite(ledPin, LOW);
    }
  }
}
// Function for calculating the distance measured by the Ultrasonic
sensor
int calculateDistance(){

  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
```

```
  // Establece trigPin a HIGH durante 10 microsegundos
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH); // Lee desde echoPin y devuelve un
valor en microsegundos
  distance= duration*0.034/2; //calcular la distancia en base a la
durancion del ping
  return distance;
}
```

Código Processing

Processing lo que va a hacer es procesar las entradas de datos y en vez de realizar cálculos con ellas, va a "dibujar con ellos". Lo primero que hacemos es declarar el conjunto de variables iniciales y por supuesto un puerto del que vamos a recibir los datos.

En la función setup(), al igual que en Arduino lo que hacemos es inicializar las variables, puerto, el sistema de comunicación (Leer hasta "."), fuente de texto que usamos y la resolución de pantalla.

La siguiente función que declaramos es la función draw). que es una función maestra que se encarga de llamar a otras funciones a su cargo, de esta manera, dividimos los elementos a dibujar en la pantalla entre varias funciones. Para dibujar el radar lo que vamos a hacer es dibujar el texto, las líneas, los objetos y el propio radar por separado.

Tenemos una función llamada serialEvent() que va a leer los datos de nuestro puerto Arduino COM3 (Puede variar entre COM3 y COM4 en windows, por lo que si se usa en otro sistema operativo hay que configurarlo previa ejecución)

Más tarde en la función DrawRadar(), lo que hacemos es dibujar los elementos rectos y curvos estáticos que se muestran en la pantalla. Con drawObject() dibujamos las

colisiones que hay con el sensor y con drawLine() realizamos el efecto de barrido.

Con drawText() no solo situamos las diferentes referencias angulares y de distancia por la pantalla sino que además, decidimos que mostrar dependiendo de los datos que nos lleguen del microcontrolador. En base a la distancia que recibimos dibujamos avisos diferentes e incluso una interacción nueva, un botón con el que podemos encender uno de los LED de la protoboard.

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

Serial myPort; // Definimos el objeto Puerto
// Definimos variables iniciales
String angle="";
String distance="";
String data="";
String noObject;
float pixsDistance;
int iAngle, iDistance;
int index1=0;
int index2=0;
PFont orcFont;
boolean EstadoBoton = false;
color off = color (200,0,0);
color on = color (0,200,0);

void setup() {

 size (1920, 1080); //La resolución de pantalla se deja así, ha dado muchos problemas una menor
 smooth();
 myPort = new Serial(this,"COM3", 9600); // Empezamos la comunicación serial con el puerto vinculado a Arduino
 myPort.bufferUntil('.'); // reads the data from the serial port up to the character '.'. So actually it reads this: angle,distance.
 orcFont = createFont("Georgia", 32); //Fuente que he utilizado para el display de info
}

void draw() {

  fill(98,245,31);
  textFont(orcFont);
  noStroke();
```

```
  fill(0,4);
  rect(0, 0, width, height-height*0.065);

  fill(98,245,31); // verde militar
  // Llamada a las funciones que dibujan el radar
  drawRadar();
  drawLine();
  drawObject();
  drawText();
}

void serialEvent (Serial myPort) { // Lee datos del puerto serial COM3 (ARDUINO)
  // lee los datos provenientes del puerto y los pone en data hasta que lee un punto
  data = myPort.readStringUntil('.');
  data = data.substring(0,data.length()-1);

  index1 = data.indexOf(","); //encuentra el caracter ',' y lo pone en la variable "index1"
  angle= data.substring(0, index1); // // lee los datos desde la posición "0" hasta la
posición de la variable index1, ese es el valor del ángulo que la placa Arduino ha
enviado al puerto serie
  distance= data.substring(index1+1, data.length()); // lea los datos desde la posición
"índice 1" hasta el final de los datos, ese es el valor de la distancia

  // Convierte el tipo String a Int para poder operar con la distancia
  iAngle = int(angle);
  iDistance = int(distance);
}

void drawRadar() {
  pushMatrix();
  translate(width/2,height-height*0.074); // Mueve las coordenadas a la nueva ubicación
  noFill();
  strokeWeight(2);
  stroke(98,245,31);
  // Dibuja las líneas curvas horizontales
  arc(0,0,(width-width*0.8325),(width-width*0.8325),PI,TWO_PI);
  arc(0,0,(width-width*0.0625),(width-width*0.0625),PI,TWO_PI);
  arc(0,0,(width-width*0.27),(width-width*0.27),PI,TWO_PI);
  arc(0,0,(width-width*0.479),(width-width*0.479),PI,TWO_PI);
  arc(0,0,(width-width*0.687),(width-width*0.687),PI,TWO_PI);
  // Dibuja las líneas rectas que delimitan los ángulos
  line(-width/2,0,width/2,0);
  line(0,0,(-width/2)*cos(radians(30)),(-width/2)*sin(radians(30)));
  line(0,0,(-width/2)*cos(radians(60)),(-width/2)*sin(radians(60)));
  line(0,0,(-width/2)*cos(radians(90)),(-width/2)*sin(radians(90)));
  line(0,0,(-width/2)*cos(radians(120)),(-width/2)*sin(radians(120)));
  line(0,0,(-width/2)*cos(radians(150)),(-width/2)*sin(radians(150)));
```

```
  line((-width/2)*cos(radians(30)),0,width/2,0);
  popMatrix();
}

void drawObject() {
  pushMatrix();
  translate(width/2,height-height*0.074); // Mueve las coordenadas a la nueva ubicación
  strokeWeight(9);
  stroke(255,10,10); // red color
  pixsDistance = iDistance*((height-height*0.1666)*0.025); // Convierte la distancia de
cm a pixels
  // limiting the range to 40 cms
  if(iDistance<40){
    // Dibuja el objeto de acuerdo a la posición y ángulo, los pasa a pixeles

line(pixsDistance*cos(radians(iAngle)),-pixsDistance*sin(radians(iAngle)),(width-width
*0.505)*cos(radians(iAngle)),-(width-width*0.505)*sin(radians(iAngle)));
  }
  popMatrix();
}

void drawLine() { //Con esta función lo que hacemos es crear el efecto de pasada de una
cortina verde, mediante la generación de líneas verdes
  pushMatrix();
  strokeWeight(9);
  stroke(30,250,60);
  translate(width/2,height-height*0.074); // Mueve las coordenadas iniciales a una nueva
localización

line(0,0,(height-height*0.12)*cos(radians(iAngle)),-(height-height*0.12)*sin(radians(iAn
gle))); // dibuja la línea de acuerdo a un ángulo
  popMatrix();
}

void drawText() { // dibuja texto en la pantalla
  //Estados del objeto referenciado en el radar
  pushMatrix();
  if(iDistance>40) {
  noObject = "Out of Range";
  }
  else if(iDistance > 20 &&iDistance < 30) {
  noObject = "In Range";
  }
  else{
  noObject = "Critical Range";
  }
  fill(0,0,0);
```

```
  noStroke();
  rect(0, height-height*0.0648, width, height);
  fill(98,245,31);
  textSize(25);

  text("5cm",width-width*0.4830,height-height*0.0833);
  text("10cm",width-width*0.3854,height-height*0.0833);
  text("20cm",width-width*0.281,height-height*0.0833);
  text("30cm",width-width*0.177,height-height*0.0833);
  text("40cm",width-width*0.0729,height-height*0.0833);
  textSize(40);
  text("Object: " + noObject, width-width*0.875, height-height*0.0277);
  text("Angle: " + iAngle +" °", width-width*0.48, height-height*0.0277);
  text("Distance:   ", width-width*0.26, height-height*0.0277);

//Los avisos van cambiando dependiendo de la distancia que se registre en Arduino

  if(iDistance<40) {
  text("        " + iDistance +" cm", width-width*0.225, height-height*0.0277);
  }

  if(iDistance < 10 ){
     fill(204, 102, 0);
     textSize(40);
     text("  Too close, Too close!", width-width*0.6, height-height*0.6);

  }

//Si la distancia es menor que 5cm, activamos la opción de clicar el boton y lo
dibujamos

   if(iDistance < 5 ){
     if(EstadoBoton){
       fill(on);
     }else{
       fill(off);
     }
    ellipse(200,200,55,55);
   }


  textSize(25);
  fill(98,245,60);

//A continuación vamos a dibujar la información angular de cada línea recta del radar

translate((width-width*0.4994)+width/2*cos(radians(30)),(height-height*0.0907)-width/
```

```
2*sin(radians(30)));
  rotate(-radians(-60));
  text("30°",0,0);
  resetMatrix();

translate((width-width*0.503)+width/2*cos(radians(60)),(height-height*0.0888)-width/2
*sin(radians(60)));
  rotate(-radians(-30));
  text("60°",0,0);
  resetMatrix();

translate((width-width*0.507)+width/2*cos(radians(90)),(height-height*0.0833)-width/2
*sin(radians(90)));
  rotate(radians(0));
  text("90°",0,0);
  resetMatrix();

translate(width-width*0.513+width/2*cos(radians(120)),(height-height*0.07129)-width/
2*sin(radians(120)));
  rotate(radians(-30));
  text("120°",0,0);
  resetMatrix();

translate((width-width*0.5104)+width/2*cos(radians(150)),(height-height*0.0574)-width
/2*sin(radians(150)));
  rotate(radians(-60));
  text("150°",0,0);
  popMatrix();
}

void mouseClicked(){ //Con esta función vamos a hacer que processing envie un
caracter ‘a’ al microcontrolador cada vez que se clique dentro del boton creado

  float distancia = dist(200,200,mouseX,mouseY);
  if(distancia  < 27){
    EstadoBoton = !EstadoBoton;
    myPort.write('a');
  }

}
```

Bibliografía

Arago, Davy, and Nikola Tesla. “Electromagnetismo.” *Wikipedia*, https://es.wikipedia.org/wiki/Electromagnetismo. Accessed 12 April 2023.

“Arduino Radar.” *Hackster.io*, 25 April 2021, https://www.hackster.io/noneedforit/arduino-radar-49127a. Accessed 12 April 2023.

Nedelkovski, Dejan. “Arduino Radar Project.” *How To Mechatronics*, https://howtomechatronics.com/projects/arduino-radar-project/. Accessed 12 April 2023.

“Radar.” *Wikipedia*, https://es.wikipedia.org/wiki/Radar. Accessed 12 April 2023.

“Sonar.” *Wikipedia*, https://es.wikipedia.org/wiki/Sonar. Accessed 12 April 2023.