# **Carpetas de Campo**

Alumno: Felipe luchetta

Grupo: Felipe luchetta, Alejandro Alarcon, Thiago Arellano, Dayana Vila

Profesor: Gonzalo Consorti

Materia:

Año: 4° 1

**Proyecto: Radar**

**Descripción:**

El radar es un dispositivo de uso fácil el cual su función principal es el detectar un objeto o movimiento a una cierta distancia utilizando el sensor ultrasónico que detecta la posición de una entidad y el servomotor para expandir la distancia de visión.



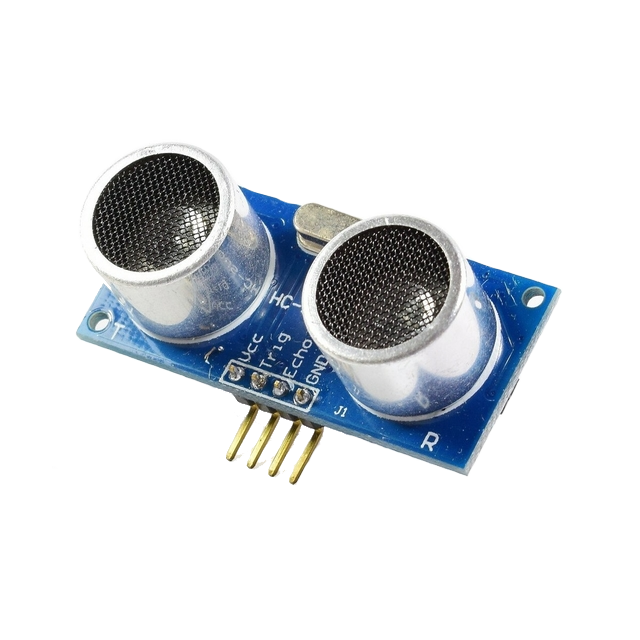
25 / 10 / 2024

**investigación:**

Empezamos a investigar sobre el radar: como se arma, los materiales y el código.

**Los materiales:**

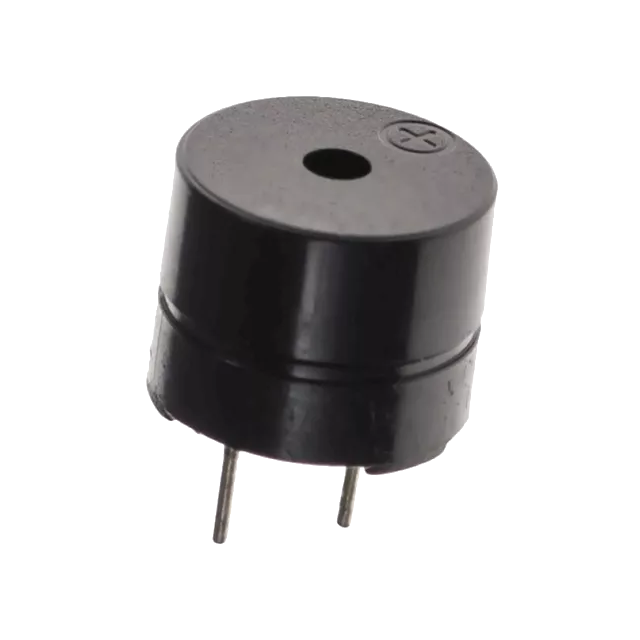
* **Sensor de proximidad**



<https://www.keyence.com.mx/ss/products/sensor/sensorbasics/ultrasonic/info/#:~:text=Como%20su%20nombre%20lo%20indica,la%20emisi%C3%B3n%20y%20la%20recepci%C3%B3n.>

El sensor de proximidad es un dispositivo que detecta la presencia de objetos cercanos sin necesidad de contacto físico. Los sensores de proximidad pueden ser de distintos tipos, como inductivos, capacitivos, fotoeléctricos y magnéticos. Pero el sensor que utilizamos es el sensor ultrasónico que miden la distancia mediante el uso de ondas ultrasónicas.

* **buzzer**



<https://osakaelectronicsltda.com/blog/biblioteca/que-es-un-buzzer?srsltid=AfmBOoq4HltZQFvNlzCQNYYTF-qLdBa70l8n4P4O1rw0WIYW5NXebI1J>

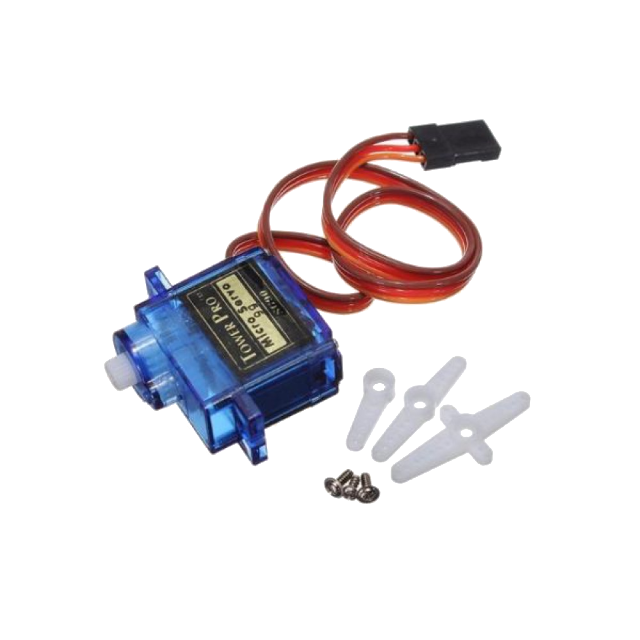
**Que es:**

Un [buzzer](https://osakaelectronicsltda.com/componentes-electronicos/buzzer-zumbador-activo-tmb12a12-12v.html) es un dispositivo transductor electroacústico que emite un sonido, ya sea continuo o intermitente, al recibir una señal eléctrica. Hay distintos tipos de buzzer: el buzzer Piezoeléctrico y el Magnético.

**Uso:**

El buzzer que vamos a utilizar es un Buzzer Piezoeléctrico, el cual va a emitir un sonido cuando detecte un objeto dentro del rango del sensor ultrasónico. Este sonido servirá como una señal de alerta para notificar al usuario sobre la presencia de un objeto cercano.

* **micro servo motor**



<https://www.egasen.com/es/blog/noticias/que-es-servomotor-para-que-se-utiliza>

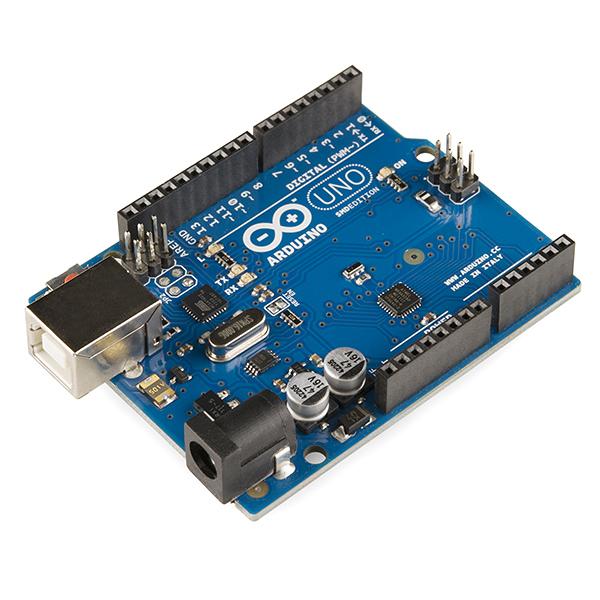
**Que es:**

Es un dispositivo que permite controlar con precisión la posición y movimientos de su eje. Esto significa que este se puede mover en un ángulo, posición y a una velocidad determinada en cada momento.

**Uso:**

Micro servomotor es una de las partes más importantes de la estructura del radar ya que permite el movimiento del sensor ultrasónico, lo que a su vez le posibilita detectar objetos en diferentes ángulos alrededor de su entorno. que permite que el servomotor amplíe la zona de visión del sensor para que cubra un área más amplia.

* **arduino**



<https://es.wikipedia.org/wiki/Arduino>

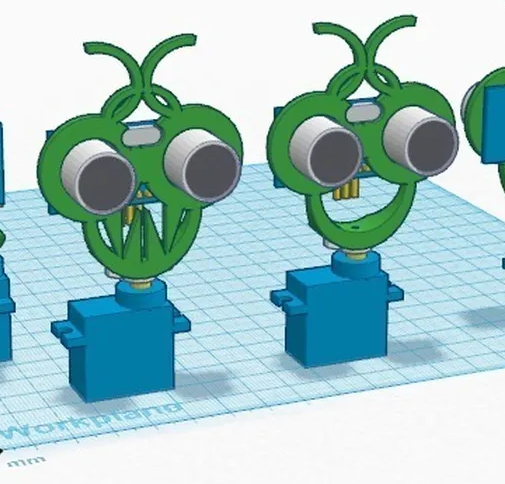
**Que es:**

Arduino es una placa de circuitos de código abierto que permite crear objetos electrónicos interactivos, utilizando sus pines para enviar señales a otros componentes y darles órdenes según el código programado.

**Uso:**

La placa Arduino UNO va a actuar como el cerebro del proyecto, controlando todas las funciones digitales y virtuales definidas en el código. Además, se encargará de establecer las conexiones necesarias entre los cables, la placa de pruebas, el sensor ultrasónico, el servomotor y el buzzer a través de sus pines.

* Soporte 3D para el sensor ultrasónico



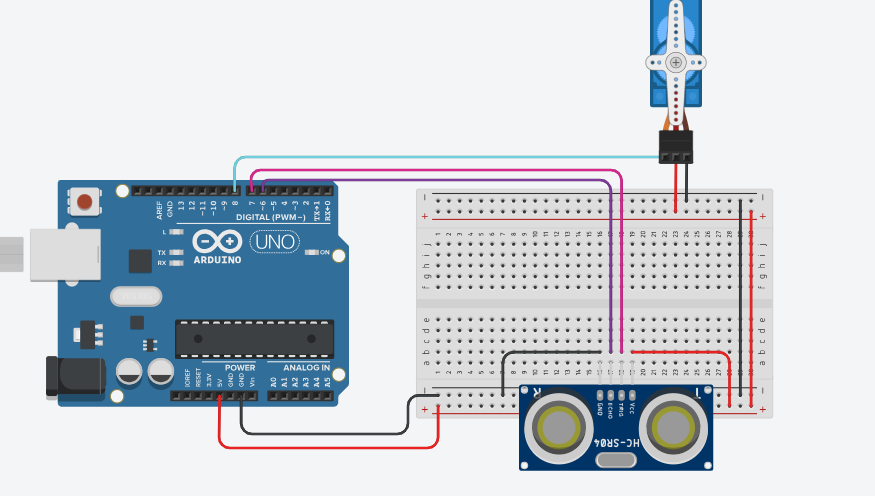
<https://cults3d.com/es/modelo-3d/artilugios/sr04-servo-mount-bugz>

Buscamos un soporte adecuado para que el servomotor sujete el sensor ultrasónico de manera que no nos estorbe y se dañen, evitando que los componentes sufran daños. Para encontrar los modelos 3D correctos, investigamos varias opciones que pudieran funcionar y que no fueran demasiado grandes. Finalmente, encontramos un modelo compacto, fácil de usar y que encaja perfectamente con nuestras necesidades, lo que facilita su implementación y mejora la eficiencia del proyecto.

No pudimos usarlo porque la impresora 3D estaba ocupada y no iba a estar disponible a tiempo para la exposición. Entonces, le preguntamos al profesor qué podríamos hacer, y nos dijo que no era necesario utilizarlo en el proyecto para la exposición. En su lugar, uno de los miembros de nuestro equipo tomará el radar y lo girará mientras está sentado en una silla.

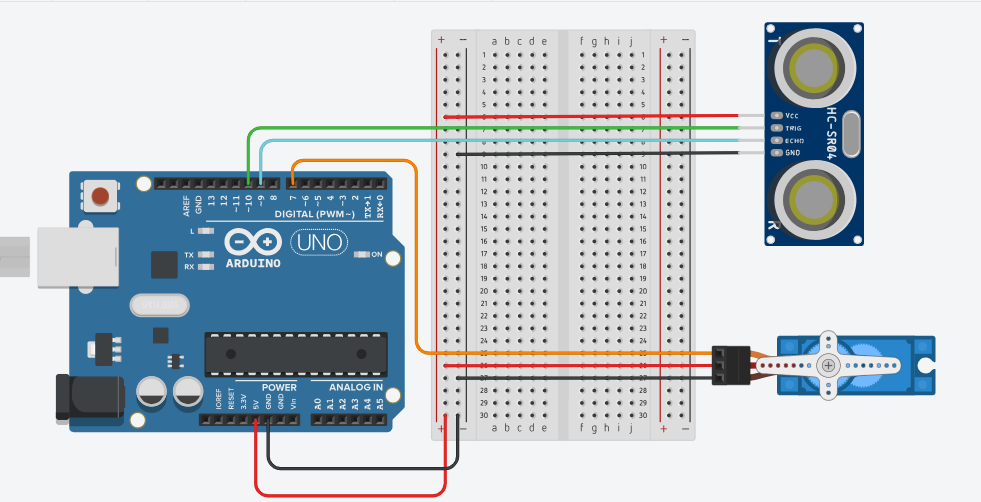
**armado del proyecto:**

**prueba número 1**

****

Utilizamos un modelo simple para empezar y tener una idea inicial de cómo podríamos empezar utilizando el sensor ultrasónico y el micro servo motor para ver su funcionamiento pero como no era suficiente para tener una imagen tan clara utilizamos otro modelo.

**prueba número 2**

****

En la segunda prueba del cuerpo del radar utilizamos los mismos componentes pero cambiamos algunas cosas simples pero que nos iba a ayudar para más adelante como que el servo motor y el sensor ultrasónico estén más sueltos para que tengan mayor movilidad. nos olvidamos de agregar el buser.

**explicación del diseño:**

En el presente proyecto, emplearemos el pin 10 del Arduino para generar un pulso ultrasónico, mientras que el pin 9 se utilizará para recibir dicho pulso. Esta disposición facilitará la medición de distancias a través de un sensor ultrasónico. Asimismo, el servomotor será controlado mediante el pin 7, el cual se encargará de regular sus movimientos. Aunque el sensor ultrasónico y el servomotor estén conectados de forma independiente y en las mencionadas ubicaciones, esta es solo una configuración ejemplar.

**código arduino:**

Para comprender correctamente las tareas que debíamos realizar, resultó fundamental familiarizarnos con el funcionamiento básico del servomotor. Para ello, recurrimos a diversas guías y fuentes de información, entre las cuales algunos compañeros compartieron un trabajo práctico que habían realizado en conjunto, lo que nos permitió tener una idea más clara de lo que se necesitaba hacer. Un breve repaso nos indicó que este dispositivo cuenta con tres pines digitales, cada uno con una función específica:

1. Tierra: conectado al cable negro, su función es asegurar el retorno de la corriente al circuito.
2. Potencia: conectado al cable rojo, se enlaza a una fuente de alimentación adecuada para el servomotor.
3. Señal: representado por el cable naranja, es el pin encargado de recibir las órdenes del Arduino para controlar el movimiento del servomotor.

Para poder controlar el servomotor mediante nuestro código, es imprescindible incluir la biblioteca **Servo**, la cual simplifica tanto su configuración como su manejo. Esto se logra mediante la siguiente línea de código:

| #include <Servo.h> |
| --- |

Una vez incluida la biblioteca, tenemos acceso a diversas funciones que nos permiten interactuar con el servomotor. Una de las principales es la siguiente:

**Servo.attach(pin):** Esta función se emplea para vincular el servomotor con un pin digital del Arduino. Al invocar **Servo.attach()** y especificar el número del pin como parámetro, el Arduino queda preparado para controlar el servomotor a través de ese pin. Esta conexión es fundamental para que el Arduino pueda enviar señales de Modulación por Ancho de Pulso (PWM) al servomotor, lo que posibilita su movimiento y control. Sin esta función, no sería viable operar el servomotor desde el código.

**Servo.write(ángulo):** Esta función permite definir el ángulo exacto al que debe posicionarse el servomotor. El parámetro de entrada es un valor en grados, donde generalmente 0° representa la posición más a la izquierda, 90° el centro y 180° la posición más a la derecha, dependiendo de las características del servomotor. Por ejemplo, al utilizar **Servo.write(90)**, el servomotor se posicionará en su punto medio. Esta función es ideal para situaciones en las que se requiere controlar el movimiento en ángulos preestablecidos.

**Servo.writeMicroseconds(valor):** Para configuraciones más avanzadas, esta función permite modificar el ancho del pulso enviado al servomotor en pocos segundos. Es particularmente útil cuando se busca una mayor precisión o se trabaja con servomotores que requieren rangos de pulso personalizados. Normalmente, los valores oscilan entre 1000 microsegundos (posición mínima) y 2000 microsegundos (posición máxima). El ajuste directo del ancho de pulso permite un control más detallado en comparación con los ángulos predefinidos.

**Servo.read():** La función **Servo.read()** se emplea para obtener el ángulo actual del servomotor. Es especialmente útil en proyectos donde se necesita monitorear o registrar en tiempo real la posición del servo, ya sea para validaciones o ajustes automáticos dentro del sistema. Esta función facilita la sincronización de las acciones del servomotor con otros componentes del proyecto, como sensores o mecanismos de retroalimentación.

**Servo.detach():** Finalmente, **Servo.detach()** se utiliza para liberar el control del servomotor. Al invocar esta función, se interrumpe la comunicación entre el Arduino y el servomotor, lo que detiene el envío de señales PWM al servo. Esta opción resulta útil cuando se desea apagar el servomotor para ahorrar energía, detener su movimiento o liberar el pin digital asignado para utilizarlo en otras funciones. Cabe destacar que, al desconectar el servomotor, éste permanecerá en su última posición.

una vez explicado todo esto definiremos los pines del sensor ultrasónico que tales son:

| const int trigPin = 10; const int echo = 11; |
| --- |

Para comenzar con la configuración, primero es necesario declarar la librería correspondiente al servomotor creando el objeto myServo que nos permitirá controlar el servomotor. Esta declaración se realiza fuera de la función setup(). Dentro de setup(), se establece la conexión del servomotor con el Arduino mediante myServo.attach(12), asignando el pin digital 12 al servomotor. Además, se configuran los pines para el buzzer, trigg y echo, y se inicializa la comunicación serial con Serial.begin(9600), lo cual es útil para visualizar los datos en la pantalla.

Para permitir que el servomotor gire, es necesario implementar un bucle dentro de la función loop(). Un ejemplo de esto es el siguiente fragmento de código:

| for (int i = 15; i <= 180; i++) {  myServo.write(i);  delay(30); } |
| --- |

Este bucle for comienza con el valor de i igual a 15, lo que representa el ángulo inicial del servomotor. A medida que el valor de i aumenta hasta 180, el ángulo del servomotor se ajusta gradualmente mediante la función myServo.write(i). El uso de delay(30) asegura que el movimiento del servomotor sea más fluido, ya que introduce una pequeña pausa entre cada ajuste del ángulo.

Para que el servomotor cumpla su función dentro de un sistema de radar, es necesario calcular la distancia a la que se encuentra un objeto. Para ello, hemos creado una función específica que mide la distancia utilizando el sensor ultrasónico:

| int calculardistancia() {  digitalWrite(trigPin, LOW);   delayMicroseconds(2);   digitalWrite(trigPin, HIGH);   delayMicroseconds(10);   digitalWrite(trigPin, LOW);   duracion = pulseIn(echoPin, HIGH);   distancia = duracion \* 0.034 / 2;  return distancia; } |
| --- |

La función calculardistancia() envía un pulso de 10 microsegundos a través del pin trigPin para activar el sensor. Luego, utiliza la función pulseIn() para medir el tiempo que tarda la señal en regresar al pin echoPin, lo que nos da la duración del pulso. Esta duración se convierte en distancia utilizando la velocidad del sonido (0.034 cm/µs) y se divide entre 2 para compensar el recorrido de ida y vuelta de la señal. El valor calculado se devuelve como resultado.

Una vez que la distancia es calculada, podemos agregar la lógica para que el buzzer emita sonidos dependiendo de la proximidad de un objeto detectado. Para ello, se incorpora una condición dentro del bucle for que controla el movimiento del servomotor. El buzzer emitirá un tono más agudo y rápido si la distancia es menor a 10 cm, lo que indica una proximidad alta. Si la distancia está entre 20 y 40 cm, el tono será menos agudo, pero aún perceptible:

| if (distancia < 10) {  tone(5, 500, 100);  } else if (distancia > 20 && distancia < 40) {  tone(5, 200, 100); } |
| --- |

Es importante mencionar que, antes de esta condición, la distancia se establece como el valor retornado por la función calculardistancia(), es decir, distancia = calculardistancia(). Además, se utilizan varias funciones Serial.print() para imprimir en el monitor serial la proximidad de los objetos detectados:

| Serial.print(i);  Serial.print(",");  Serial.print(distancia);  Serial.print("."); |
| --- |

Aún falta una parte del control del servomotor: el movimiento de vuelta. El bucle for que hemos utilizado para mover el servomotor de 15° a 180° debe replicarse para el movimiento en reversa. Esto se logra con el siguiente código, donde la variable de ángulo comienza en 180 y disminuye hasta 15:

| for (int i = 180; i > 15; i--) {  myServo.write(i);   delay(30); } |
| --- |

El propósito de este bucle for es definir el ángulo al que debe orientarse el servomotor. La variable entera i se inicializa en 15, lo que representa el ángulo inicial del servomotor. A medida que i aumenta hasta 180, el servomotor cambia su posición de acuerdo con el valor de i, que es actualizado mediante la función myServo.write(i). Para asegurar que el movimiento sea suave, se incorpora un delay que introduce una pequeña pausa entre cada cambio de posición, lo que proporciona mayor fluidez en la rotación del servomotor que sostiene el sensor.

Sin embargo, para que el sistema funcione como un mini radar, es necesario calcular la distancia a la que se encuentra un objeto. Para ello, se diseñó una función específica que se encarga de medir esa distancia.

| int calculardistancia() {  digitalWrite(trigPin, LOW);   delayMicroseconds(2);   digitalWrite(trigPin, HIGH);   delayMicroseconds(10);   digitalWrite(trigPin, LOW);   duracion = pulseIn(echoPin, HIGH);   distancia = duracion \* 0.034 / 2;  return distancia; } |
| --- |

Esta función se encarga de emitir un pulso de 10 microsegundos a través de los pines trig mediante los comandos digitalWrite(), lo que activa el sensor ultrasónico. A continuación, se utiliza la función pulseIn() para medir el tiempo que tarda el pulso en regresar al pin echo. Esta medición de tiempo se almacena en la variable duración. Posteriormente, se calcula la distancia a partir del tiempo registrado, utilizando la velocidad del sonido (0.034 cm/µs) y dividiendo entre 2, ya que la señal realiza un recorrido de ida y vuelta. Finalmente, la distancia calculada se devuelve como resultado de la función.

Una vez que esta función está implementada, podemos incorporar un buzzer que emita un sonido cuando el sensor detecte un objeto o movimiento. Para ello, se agrega el buzzer dentro del bucle for. Es necesario definir una condición para que el buzzer emita diferentes tonos dependiendo de la distancia medida. Si la distancia es menor a 10 cm, el buzzer emitirá un tono agudo y rápido, indicando una cercanía mayor. Si la distancia se encuentra entre 20 y 40 cm, el tono será menos agudo, pero aún llamativo. Este comportamiento se implementa con el siguiente bloque de código:

| if (distancia < 10) {  tone(5, 500, 100);  } else if (distancia > 20 && distancia < 40) {  tone(5, 200, 100); } |
| --- |

Cabe señalar que, antes de este bloque de código, la variable distancia se asigna con el valor retornado por la función calculardistancia(), es decir, distancia = calculardistancia(). Además, se incluyen varias funciones Serial.print() para imprimir la distancia calculada en el monitor serial, permitiendo visualizar en tiempo real la proximidad del objeto detectado:

| Serial.print(i);  Serial.print(",");  Serial.print(distancia);  Serial.print("."); |
| --- |

A pesar de haber avanzado en el código, aún falta un bucle for adicional para completar el movimiento del servomotor. El primer bucle solo permite que el servomotor se mueva en una dirección. El segundo bucle, similar al primero, se encarga de invertir el movimiento del servomotor, comenzando desde 180° y regresando hasta 15°. Esto se logra modificando la variable del ángulo y utilizando un decremento en el valor de i:

| for (int i = 180; i > 15; i--) {  myServo.write(i);   delay(30); } |
| --- |

Este bucle completa el recorrido del servomotor en ambas direcciones, lo que permite un barrido continuo para el mini radar.

Con el código de Arduino completo, el siguiente paso es desarrollar la interfaz gráfica para el mini radar, la cual permitirá representar visualmente las mediciones y facilitar la interacción con el sistema.

**Código Processing:**

Arduino Processing es un lenguaje de programación y un entorno de desarrollo integrado (IDE) de código abierto basado en Java. Está diseñado principalmente para creativos y diseñadores que trabajan en proyectos audiovisuales o multimedia. Su principal ventaja es que permite crear gráficos en 2D y 3D, así como texturas y otros elementos visuales, lo que lo hace ideal para el desarrollo de interfaces gráficas y visualizaciones interactivas.

Una vez que el software ha sido instalado, el siguiente paso consiste en investigar cómo desarrollar el código necesario para la interfaz gráfica del radar. Junto con mis compañeros, realizamos una búsqueda en línea, explorando diversos videos y páginas que parecían relevantes. A partir de esta investigación, encontramos varias fuentes que resultaron ser de gran utilidad para aprender las funciones y comandos básicos que necesitábamos para implementar la interfaz en Processing.

<https://www.youtube.com/watch?v=N8qPvS2YTec>