

Mini radar/sonar

Grupo 5

Mateo Villegas - Nicolas Paz - Julián Alba

Especialidad en Computación , Escuela Técnica N°32 D.E 14

4°3: Proyecto Informático

Gonzalo N. Consorti

Desde el 2 de Octubre hasta el (Fecha Indefinida)

Introducción del Proyecto

Nuestro proyecto consiste en un mini radar/sonar, que tiene la funcionalidad extra de medir la velocidad.

Vamos a tener que investigar bastante, para aprender a utilizar arduino.

Además necesitaremos el uso de código de processing, por lo tanto también deberemos aprender a usar este tipo de código. Lo primero que haremos es priorizar la funcionalidad básica del radar, una vez este funcione, ahí le agregaremos la funcionalidad extra de poder medir la velocidad con la que el objeto que detecta se mueve.

Una vez ya el profesor nos asignó a cada uno nuestros proyectos, hable con mis compañeros para poder tomar una idea de cómo podríamos hacer este proyecto del radar. Así que lo primero que fue comenzar con la recolección de información

Bueno para poder llevar a cabo la realización de nuestro radar, primero debo investigar, así que voy a estar dejando en esta carpeta de campo, toda la investigación que realice y además voy a ir explicando la funcionalidad, errores, el código y demás cosas que abarcare en el documento

El radar, es un sistema que usa ondas electromagnéticas para medir distancias, altitudes, direcciones y velocidades de objetos (Esta función es la característica que va a tener nuestro radar) , como aeronaves, barcos, vehículos motorizados, formaciones meteorológicas y el propio terreno.

El funcionamiento de este mismo trata de una emisión de un impulso de radio, esto se refleja en el objetivo y recibe en la misma posición del emisor. Su uso de ondas electromagnéticas con diversas longitudes de onda permite detectar objetos más allá del rango de otro tipo de emisores.

Información sacada de: <https://es.wikipedia.org/wiki/Radar>

SU HISTORIA:

Los Radares fueron introducidos durante la Segunda Guerra Mundial para medir la distancia y la velocidad de los objetos usando ondas de radio. En la guerra se utilizaba como un

sistema de alerta temprana, detectando aviones enemigos distantes, que de otro modo eran indetectables a simple vista

Este sistema se extendió a lo largo de toda la costa británica, permitiendo a los militares británicos tener la alerta temprana de cualquier bombardero enemigo entrante. Se cree que los logros de Watson-Watts han cambiado el resultado de la Segunda Guerra Mundial para el lado aliado. Watson-Watt fue nombrado caballero en 1942 y recibió la Medalla de Mérito de los Estados Unidos en 1946 por su trabajo en radar. El *radar* del término fue acuñado por la marina de guerra de los EEUU en 1939. Los sistemas del radar ahora se utilizan por todo el mundo para una serie de usos civiles y militares.

Información sacada de:

<https://www.storyboardthat.com/es/innovations/radar#:~:text=Es%20un%20sistema%20que%20se,eran%20indetectables%20a%20simple%20vista.>

Luego de esta pequeña introducción, de que es un radar, para que se utiliza y cuando se introdujeron voy a explicar nuestro objetivo de este proyecto detalladamente, para el Radar que debo elaborar, necesito realizar una interfaz de un radar promedio y que este me muestre los datos que quiero, en esta primera investigación me enfoque en recabar información del programa en el cual me muestre la interfaz del radar en pantalla.

Luego de estar buscando en algunas páginas de internet, la mayoría recomiendan que se utilice “Processing”. Estas fueron las páginas

<https://www.instructables.com/Radar-Sencillo-Con-Processing-Y-Arduino/>

<https://eloctavobit.com/proyectos-tutoriales-arduino/realizar-un-radar-con-arduino-y-processing>

Por lo tanto vamos a tener que aprender a programar en Processing, estuvimos viendo en Google cursos de como usar Processing. Adjunto el link de la página

http://www.mywonderland.es/curso_js/processing/processing.html

Además de esto estuve buscando algún video de Processing en Youtube y vi algunos cursos de Processing. Adjunto los links de los videos

<https://www.youtube.com/watch?v=W9HCmHibQxk&pp=ygUTY3Vyc28gZGUgcHJvY2Vzc2luZw%3D%3D>

<https://www.youtube.com/watch?v=60r2JKNmQIE&list=PLtyMmy0eKyqFsLPeszmz7y4>

[EznkZFJrGuu](#)

Lo que nosotros estamos buscando es que esta sea el interfaz que muestra nuestro radar:

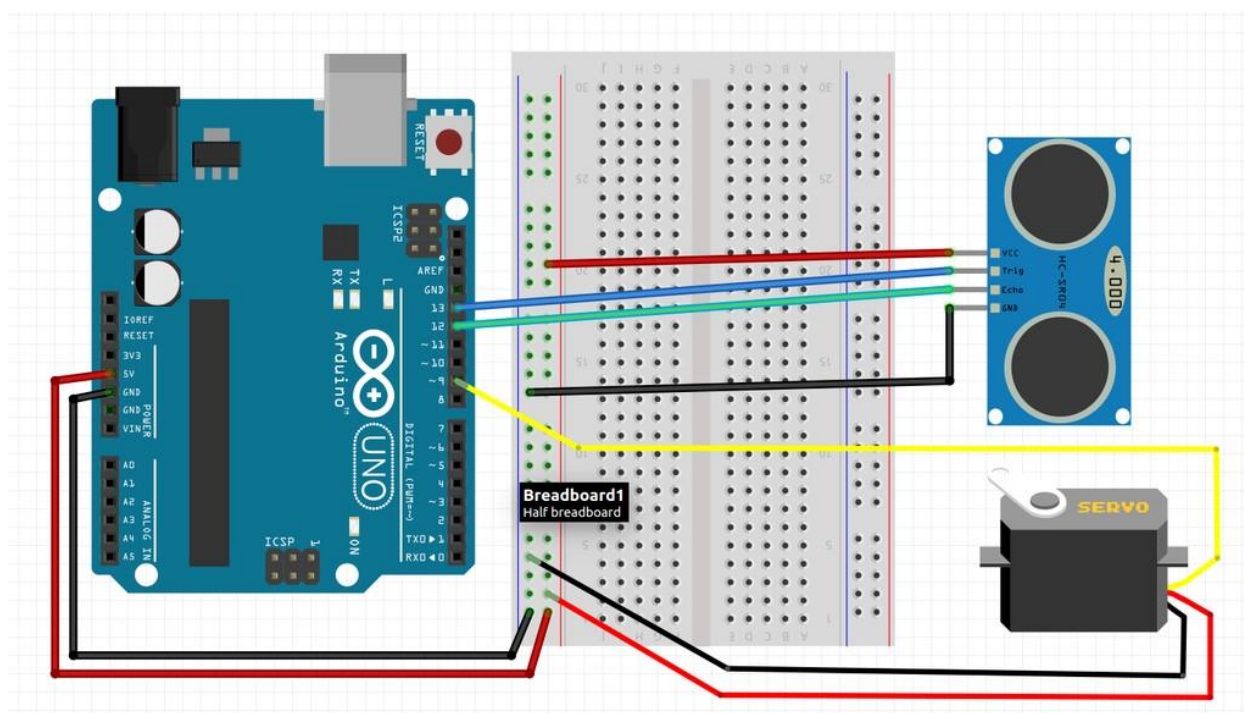


(Esta foto fue sacada de Google)

Processing es una herramienta que nos permite la programación gráfica con el lenguaje java, es decir, en nuestro caso crear un radar en tiempo real de forma gráfica. Así que con este programa que trabaja con lenguaje JAVA crearemos la interfaz que tiene un radar promedio.

Pero primero para este Radar tenemos que entender masomenos JAVA, me parece que va a ser necesario por las dudas, así que por unos días estaré viendo cursos de Processing y un poco de Arduino, por el motivo de que no estoy tan familiarizado con el tema, así que para el Arduino, recurro a los videos de las clases del profesor Consorti.

Luego de estar varios días en los que solo vi los cursos de Processing, empecé a buscar por google alguna referencia o esquema para hacer las conexiones para armar el radar.



Gracias a este esquema, se ven en pantalla de los componentes que se usarán para armar el radar:

- **Arduino Uno**
- **Sensor Ultrasónico**
- **Servo motor pequeño**
- **Cables Jumper (Macho y Hembra)**

Arduino Uno: Arduino se utilizó como un microcontrolador, cuando tiene un programa descargado desde un ordenador y funciona de forma independiente de éste, y controla y alimenta determinados dispositivos y toma decisiones de acuerdo al programa descargado e interactúa con el mundo físico gracias a sensores y actuadores.

<https://www.fundacionaquae.org/wiki/sabes-arduino-sirve/#:~:text=1.,gracias%20a%20sensores%20y%20actuadores.>

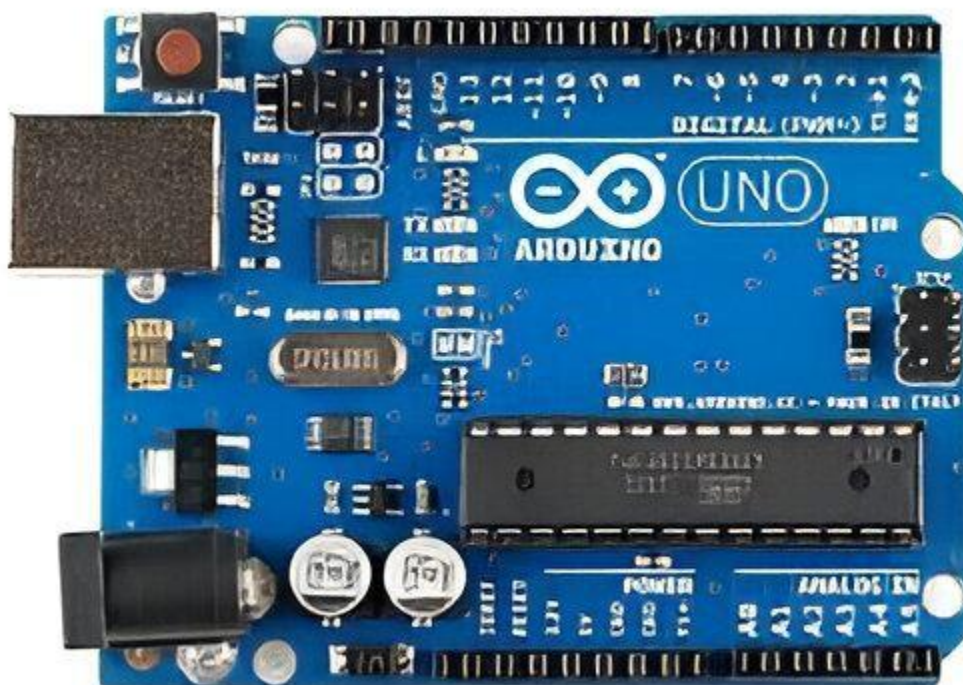


Imagen de un Arduino Uno

Sensor Ultrasónico: El sensor ultrasónico, es un sensor de distancias por ultrasonidos capaz de detectar objetos y calcular la distancia a la que se encuentra en un rango de 2 a 450 cm. El sensor funciona por ultrasonidos y contiene toda la electrónica encargada de

hacer la medición. En el trabajo el sensor será la parte fundamental del trabajo para poder hacer el radar.

A continuación les dejare las paginas en las cuales pude extraer esta información del sensor ultrasónico:

http://ceca.uaeh.edu.mx/informatica/oas_final/red4_arduino/ultrasnico.html#:~:text=Son%20detectores%20de%20proximidad%20que,la%20se%C3%B1al%20tarda%20en%20regresar.



Imagen del Sensor Ultrasónico

Servo motor pequeño: El servo es un dispositivo alimentado por corriente continua que puede controlar de modo muy exacto la posición (de 0° a 180°) o la velocidad (en revoluciones por minuto, rpm, en sentido horario o antihorario)

<https://solectroshop.com/es/blog/servomotores-como-configurarlos-para-arduino-n41#:~:text=Un%20servomotor%20es%20un%20dispositivo,el%20pin%20de%20la%20se%C3%B1al.>



Imagen del servo motor

Cables Jumper (Macho y Hembra): Estos cables nos permiten la conexión o comunicación entre diferentes dispositivos:

[https://www.codigoiot.com/base-de-conocimiento/cables-jumper-macho-hembra/#:~:text=Un%20jumper%20o%20saltador%20es,opuesto%20al%20sensor%20\(normalmente\).](https://www.codigoiot.com/base-de-conocimiento/cables-jumper-macho-hembra/#:~:text=Un%20jumper%20o%20saltador%20es,opuesto%20al%20sensor%20(normalmente).)

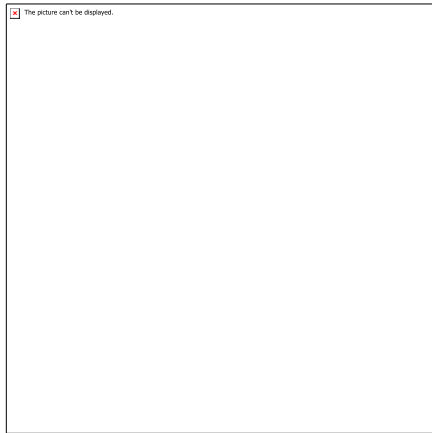


Imagen de un cable jumper (Macho y Hembra)

Empecé a buscar sobre el código, use fuentes como Google, Youtube,, en Google visite paginas para saber que era fundamental para armar el radar. una de esas fue:

<https://www.instructables.com/Radar-Sencillo-Con-Processing-Y-Arduino/>

gracias a esta página me quede con el siguiente código:

```
long microsec = ultrasonic.timing();
cmMsec = ultrasonic.convert(microsec, Ultrasonic::CM);
```

Esta parte del código o al menos escribir esto es fundamental para que el Arduino pueda leer el sensor ultrasónico: La primera línea lee el tiempo en que la señal se envía y regresa, la segunda calcula en centímetros la distancia (básicamente el tiempo multiplicado por la velocidad del sonido y luego dividido entre 2).

Esto es por una parte, aunque me falta averiguar cómo podemos hacer que el radar gire 180°.

Utilice de referencia o para hacerme una idea, esta este código:

```
a+=dir;
servo.write(a);
if(a==0)dir=5;
if(a==180)dir=-5;
```

La primera y segunda línea suma o resta dir al ángulo, y luego mueve el servo. Luego los "if" lo que hacen es determinar si viene desde la derecha o desde la izquierda y cambia la dirección de movimiento, cambiando el signo de "dir".

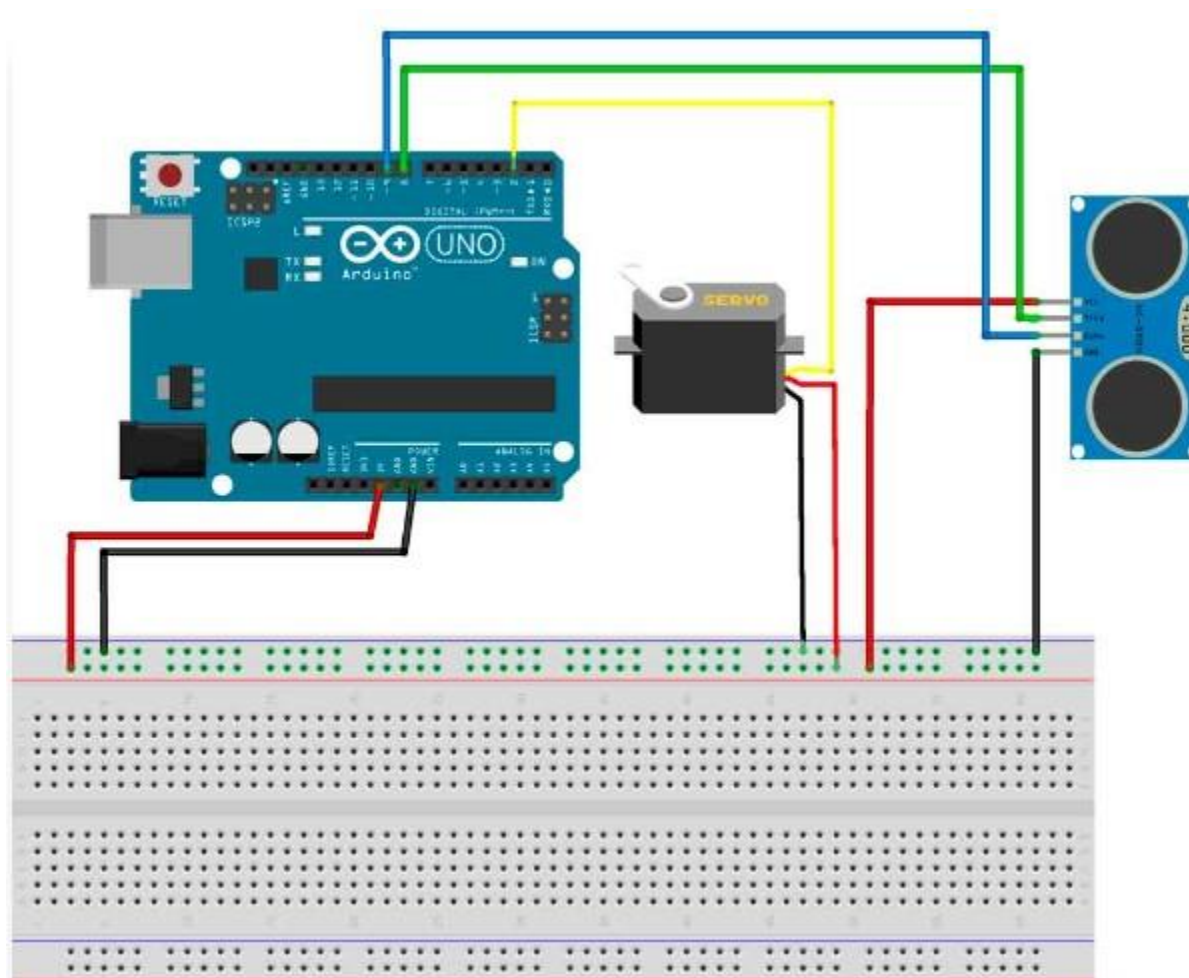
Pero siendo honesto no me convence el código o al menos pienso que existe una forma mas corta y entendible de hacerlo.

Se necesita implementar una librería llamada "Servo.h" osea es OBLIGATORIO usar la librería "Servo.h" página usada:

<https://eloctavobit.com/proyectos-tutoriales-arduino/realizar-un-radar-con-arduino-y-processing>

Gracias a esta biblioteca permite a una placa Arduino controlar servomotores RC (hobby). Los Servos integran engranajes y un eje que puede ser controlado con precisión. Los servos estándar permiten que el eje sea colocado en distintos ángulos, por lo general entre 0 y 180 grados. Los servos de rotación continua permiten la rotación del eje para ajustarse a diferentes velocidades.

Información saca de: <https://manueldelgadocrespo.blogspot.com/p/biblioteca-servo.html>



Este es un modelo es más fácil y más “claro” de entender y este usare como referencia en mi Tinkercad para posteriormente usar el código.

Imagen sacada de la página: <https://codemain.pro/radar-con-arduino-y-processing/>

Al mismo tiempo, vi una librería llamada “NewPing” la cual dispone de muchas funciones en tema de mediciones, que esto es una de las funcionalidades que debe cumplir mi Radar. Por ejemplo:

Funciones:

- **sonar.ping ([max_cm_distance]):** envía un ping y obtiene el tiempo de eco (en microsegundos) como resultado. [max_cm_distance] permite establecer opcionalmente una nueva distancia máxima.
- **sonar.ping_in ([max_cm_distance]):** envía un ping y obtiene la distancia en pulgadas enteras. [max_cm_distance] permite establecer opcionalmente una nueva distancia máxima.
- **sonar.ping_cm ([max_cm_distance]):** envía un ping y obtiene la distancia en centímetros enteros. [max_cm_distance] permite establecer opcionalmente una nueva distancia máxima.
- **sonar.ping_median (iteraciones [, max_cm_distance]):** realiza varios pings (predeterminado = 5), descarta los pings fuera de rango y devuelve la mediana en microsegundos. [max_cm_distance] permite establecer opcionalmente una nueva distancia máxima.
- **sonar.convert_in (echoTime):** convierte echoTime de microsegundos a pulgadas.
- **sonar.convert_cm (echoTime):** convierte echoTime de microsegundos a centímetros.
- **sonar.ping_timer (function [, max_cm_distance]):** envía un ping y llama a la función para probar si el ping está completo. [max_cm_distance] permite establecer opcionalmente una nueva distancia máxima.
- **sonar.check_timer ():** comprueba si el ping ha regresado dentro del límite de distancia

establecido.

- **NewPing :: timer_us (frecuencia, función):** función de llamada cada microsegundos de frecuencia.
- **NewPing :: timer_ms (frecuencia, función):** función de llamada cada milisegundos de frecuencia.
- **NewPing :: timer_stop ():** detiene el temporizador.

Estas son las funciones que dispone esta librería.

La información fue sacada de la siguiente página: <https://elctavobit.com/librerias-arduino/newping>

Se me ocurrió una idea, que hasta el momento no había hecho y fue hacer las búsquedas de google pero en inglés, y escribir algo en google en inglés hace que te aparezca información más precisa o incluso más útil que en español.

Y efectivamente funcionó, estaba buscando en como calcular la velocidad y que este dato me lo muestre en pantalla y lo que encontré fue una página en la cual desarrollaron un radar parecido al que yo iba a hacer y bueno, use la página para saber como hacer las bases del código en Processing ya que una cosa era saber la teoría y otra la práctica, en fin, a continuación adjunto el link de la página:

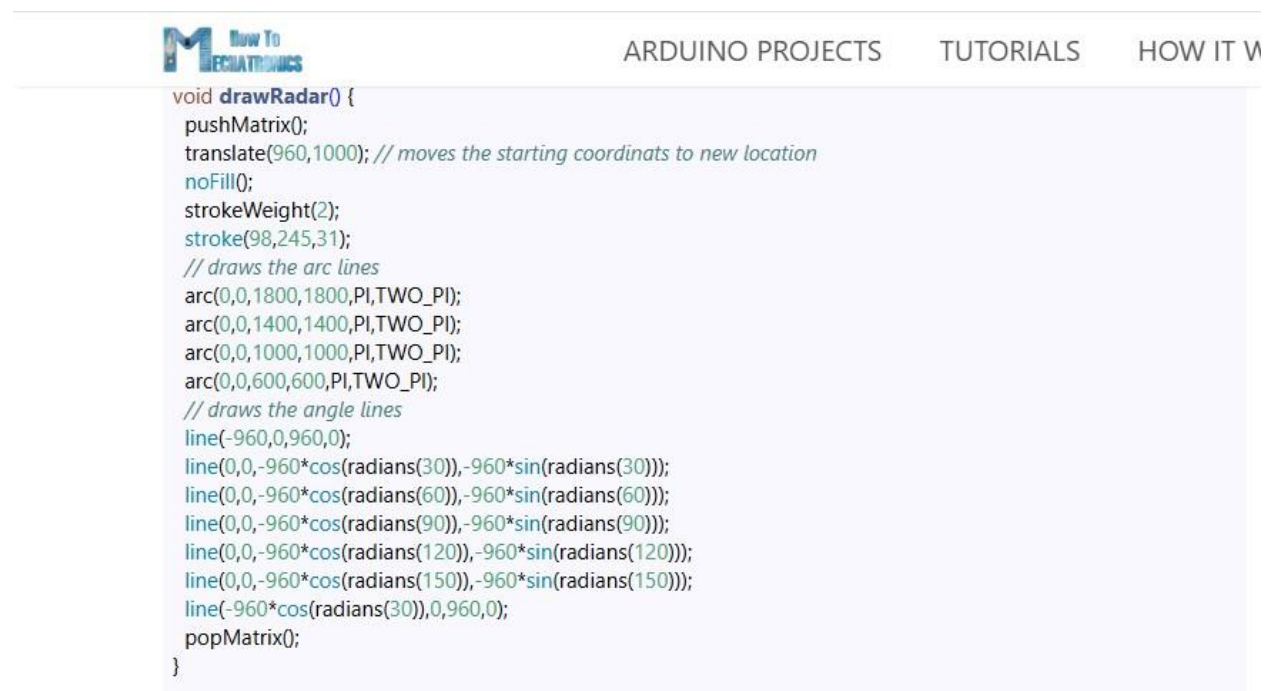
<https://howtomechatronics.com/projects/arduino-radar-project/>

Esta página fue de mucha ayuda ya que el código me sirve demasiado para saber en como empezar a crear la interfaz del radar y bueno, los videos que use al principio del proyecto los volvi a ver para asi entender mejor los comandos.

Voy a seguir investigando otras fuentes para entender bien el planteo que se debía hacer en el código de Processing, no me esperaba que iba a ser lo más complicado del proyecto

https://howtomechatronics.com/projects/arduino-radar-project/#google_vignette

https://www.hackster.io/Yug_Ajmera/radar-sonar-using-processing-3-7302c6



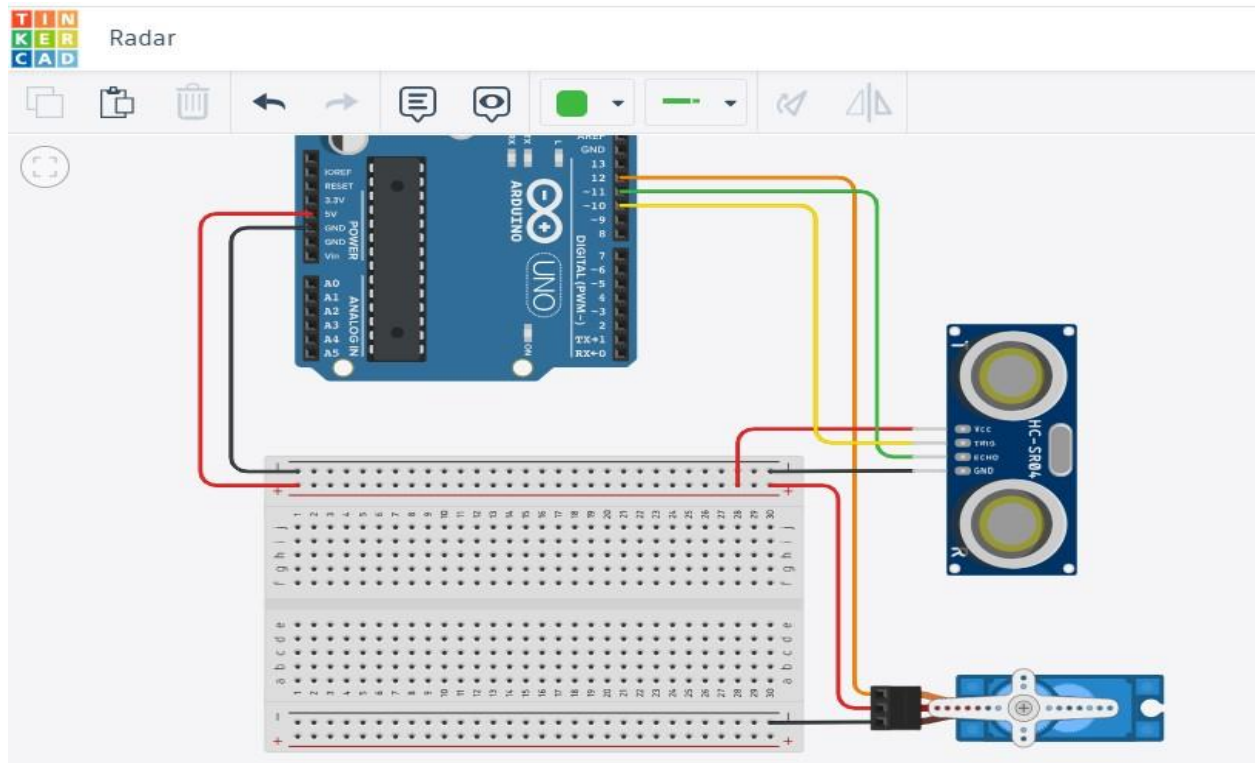
Este es otro de los Links que visite, la idea principal de buscar estos códigos es para ver alguna similitud que tienen ambos códigos, para que en el momento de hacer el código no tenga complicaciones o al menos esa es la idea.

En esta pagina pude encontrar demasiada información para hacer el código de processing, como por ejemplo imágenes de referencia para qué servía cada linea de codigo o cada void().

Segunda Etapa: Prueba y ERROR

Bueno después de toda la primer etapa, de mirar cursos, investigar en páginas y demás investigación.

Comenzamos la construcción del radar usando de referencia todo lo adjuntado hasta el momento, empezando con la maqueta en el Tinkercad, adjunto imagen:



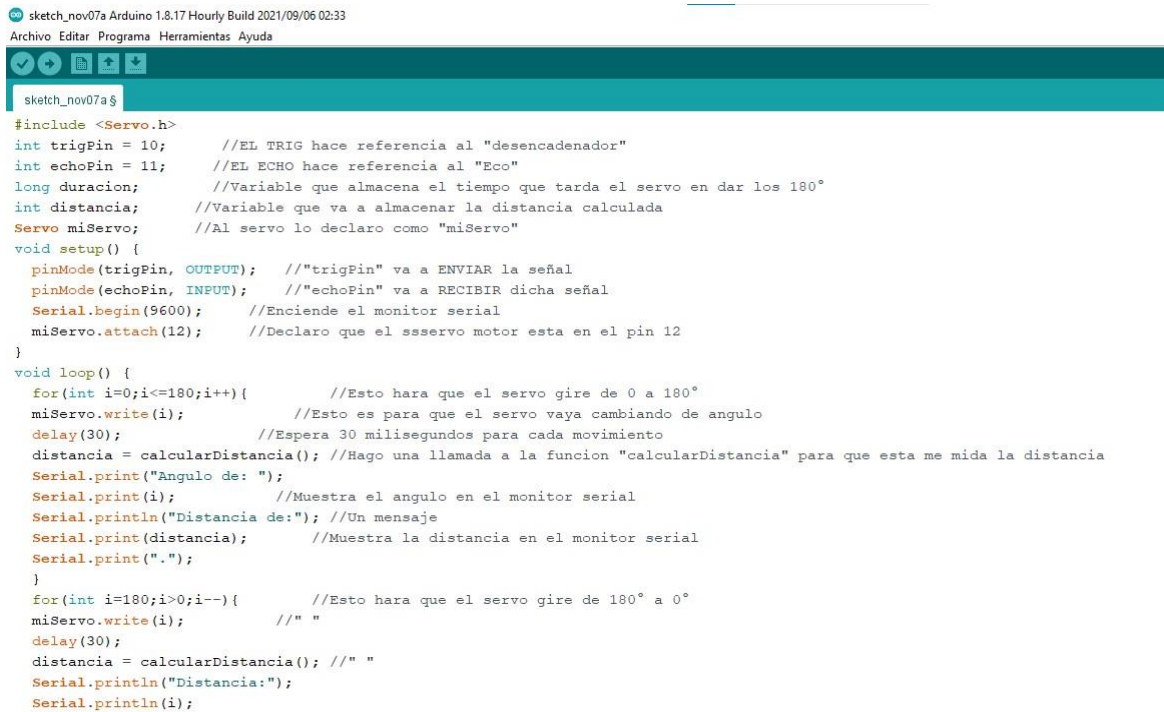
En esta imagen se puede ver como pude realizar la conexión y además como seguí bastante el ejemplo anterior, los únicos que cambios que le pude realizar a la imagen anterior, fueron los cables que les pude realizar un conexión más prolija y entendible

Una vez que ya hecha la conexión en Tinkercad, para tener una idea de cómo sería el radar, empezare con la parte del código, utilizaré arduino para poder darle una funcionalidad a nuestro proyecto.

Ahora con el apartado del código use las páginas de google que adjunte y las clases grabada del profesor

Consorti de cómo usar el sensor ultrasónico para masomenos tener una idea de cómo usarlo.

Así sería la primer parte del código hasta el momento:



```

sketch_nov07a Arduino 1.8.17 Hourly Build 2021/09/06 02:33
Archivo Editar Programa Herramientas Ayuda

sketch_nov07a$
#include <Servo.h>
int trigPin = 10;      //EL TRIG hace referencia al "desencadenador"
int echoPin = 11;      //EL ECHO hace referencia al "Eco"
long duracion;         //Variable que almacena el tiempo que tarda el servo en dar los 180°
int distancia;         //Variable que va a almacenar la distancia calculada
Servo miServo;         //Al servo lo declaro como "miServo"

void setup() {
  pinMode(trigPin, OUTPUT); // "trigPin" va a ENVIAR la señal
  pinMode(echoPin, INPUT);  // "echoPin" va a RECIBIR dicha señal
  Serial.begin(9600);       //Enciende el monitor serial
  miServo.attach(12);       //Declaro que el sservo motor esta en el pin 12
}

void loop() {
  for(int i=0;i<=180;i++){ //Esto hara que el servo gire de 0 a 180°
    miServo.write(i);      //Esto es para que el servo vaya cambiando de angulo
    delay(30);             //Espera 30 milisegundos para cada movimiento
    distancia = calcularDistancia(); //Hago una llamada a la funcion "calcularDistancia" para que esta me mida la distancia
    Serial.print("Angulo de: ");
    Serial.print(i);       //Muestra el angulo en el monitor serial
    Serial.println("Distancia de:"); //Un mensaje
    Serial.print(distancia); //Muestra la distancia en el monitor serial
    Serial.print(".");
  }
  for(int i=180;i>0;i--){ //Esto hara que el servo gire de 180° a 0°
    miServo.write(i);      // " "
    delay(30);
    distancia = calcularDistancia(); // " "
    Serial.println("Distancia:");
    Serial.println(i);
  }
}

```

En cada línea de nuestro código, deje algunas notas de que funcionalidad tiene cada una, por si no quedo

del todo claro ahora voy a explicar a detalle bien estas:

Declaración de variables y librería:

En la maqueta del Tinkercad se ve que el Pin “TRIG” lo conecte en el Pin 10, por eso es la declaración “int trigPin”. Ocurre lo mismo con el Pin ECHO, solo que en este lo conecte en el Pin 11.

El “#include <Servo.h>” es nuestra libreria, que anteriormente había comentado que SI O SI se tiene que usar para hacer este proyecto

“long duracion;” Usamos LONG para poder almacenar grandes cantidades de números, lo cual es necesario para el radar para almacenar el tiempo que se va a tomar el Servo en dar los 180°

“int distancia” Creo que no es necesario explicar pero en resumen:Esta variable va a almacenar la distancia

“Servo miServo;” Al Servo lo declaró o lo llamó como “miServo”(Por un tema de comodidad)

Explicación del void setup:

En los PinModes lo que estoy haciendo es declarar que el “trigPin” va a ENVIAR la señal y el “echoPin” va a recibir dicha señal

“Serial.begin(9600);” es para INICIAR o ENCENDER el monitor serial

“miServo.attach(12)” En la maqueta vemos que “Señal” del Servo está conectado en el Pin 12

Explicación del loop:

El primer for (for i=0; i<= 180; i++) Lo que va a hacer es GIRAR el servo de 0° a 180°.

“miServo.write(i) va a ir moviendo el servo a cada ángulo (ósea de 0 a 180°)

El “delay(30)” es para que espere 30 milisegundos para los movimientos del Radar

```

    Serial.print(distancia);
    Serial.println(".");
  }
}
int calcularDistancia(){
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duracion = pulseIn(echoPin, HIGH); //Esto medira el tiempo que tarda en recibir la señal
  distancia= duracion*0.034/2;      //El 0.034 representa la velocidad del sonido en el aire por cm por segundo
  return distancia;                //Devuelve la distancia medida
}

```

Por el momento vengo bien con el código del arduino, pero ahora viene lo más complejo y es el processing, del cual vi varios cursos más, algunos en inglés, que me resultaron bastante útiles, mientras que otros no me sirvieron de nada. Lo bueno es que encontré este manual de todos los comandos que usa el Processing con su respectiva explicación.

[http://www.tallertecno.com/recursos/Referencia Processing con Imagenes.pdf](http://www.tallertecno.com/recursos/Referencia%20Processing%20con%20Imagenes.pdf)

Fue lo más útil hasta el momento, respecto al Processing, gran parte de los comandos que utilizamos en la construcción del Radar fueron sacados de esta página

Ahora lo único que quedaría es hacer el código en Processing para posteriormente empezar a finalizar el trabajo,

Una de las cosas que no considere al empezar el trabajo es en CÓMO sabría yo si el código que estaba haciendo desde mi casa estaba bien o mal para el RADAR es lo único malo de este proyecto ya que solo puedo saber eso los Jueves (horario de Proyecto Informático).

```

sketch 241126a
float angle = 0;
float distance = 0;
PFont font;

// Simulación de obstáculos
float[] distances = new float[181];

void setup() {
  size(800, 400); // Tamaño de la ventana
  font = createFont("Arial", 16, true);

  // Inicializar obstáculos fijos
  for (int i = 0; i < distances.length; i++) {
    if (i > 50 && i < 100) {
      distances[i] = random(50, 100); // Objeto fijo en este rango
    } else {
      distances[i] = random(150, 200); // Distancia aleatoria
    }
  }
}

void draw() {
  background(0);
  fill(0, 255, 0);
  textFont(font);
  text("Simulación de masomenos como se veria el Radar", 20, 30);

  // Dibuja el radar
  translate(width / 2, height);
  stroke(0, 255, 0);
  line(0, 0, cos(radians(angle)) * distances[int(angle)] * 2, -sin(radians(angle)) * distances[int(angle)] * 2); // Línea del radar
  ellipse(0, 0, 400, 400); // Radar base

  // Marca un punto rojo en el objeto detectado
  fill(255, 0, 0);

```

Así que gracias a los tutoriales que logré ver en Youtube, descubrí que podía hacer “SIMULACROS” de mi Radar, no necesariamente me garantizara que el código esta bien, pero el funcionamiento es similar, solo que varia el diseño.

```

noStroke();
ellipse(cos(radians(angle)) * distances[int(angle)] * 2, -sin(radians(angle)) * distances[int(angle)] * 2, 10, 10); // Objeto detectado

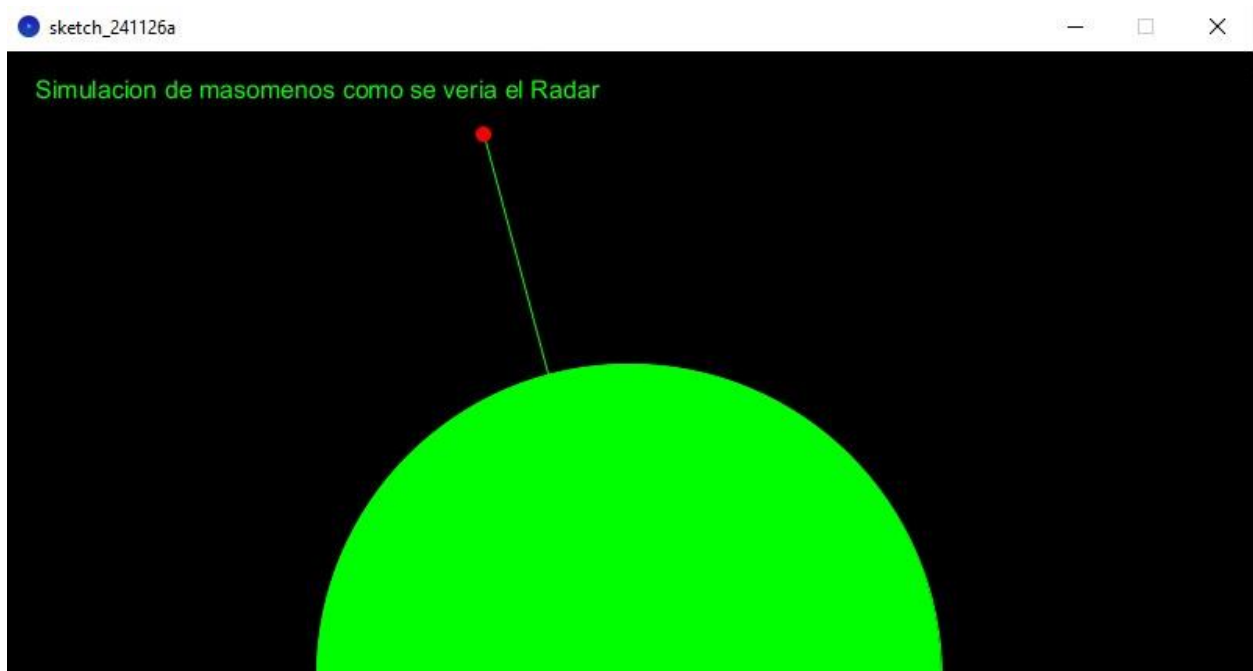
// Actualiza el ángulo
angle += 1;
if (angle >= 180) {
  angle = 0;
}

// Simula movimiento de obstáculos
if (angle % 10 == 0) {
  distances[int(angle)] = random(50, 200); // Cambia la distancia en este ángulo
}

delay(30); // Pausa para simular tiempo real
}

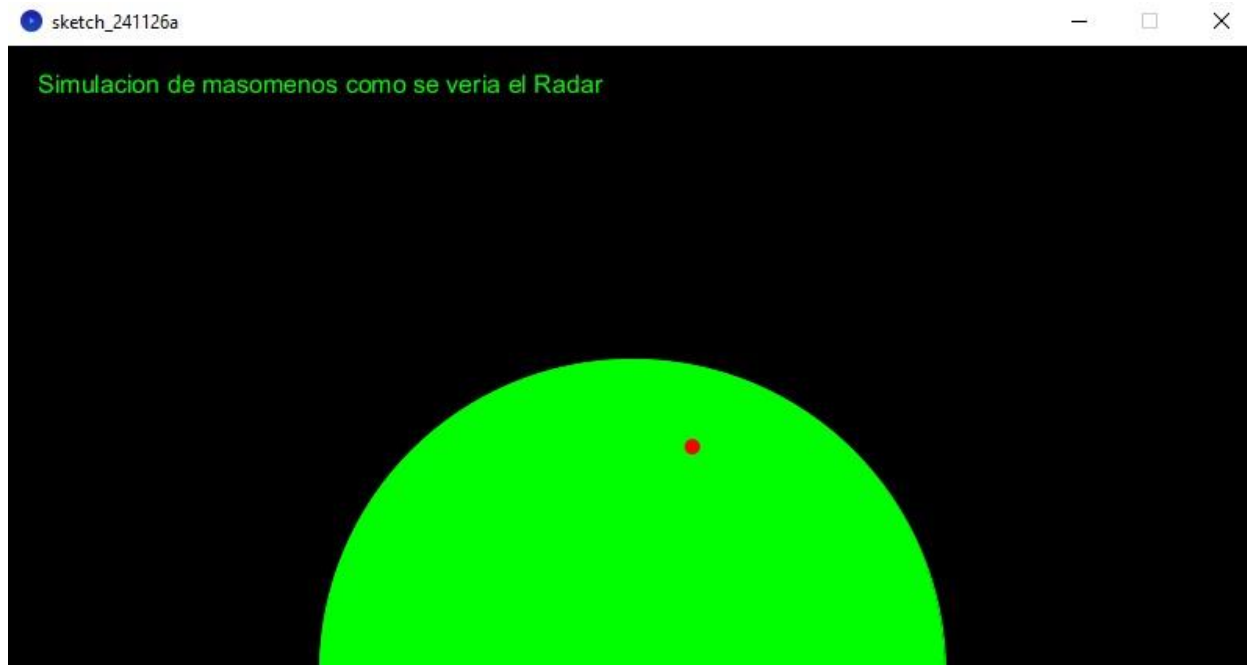
```

Para poder ahorrar tiempo hasta el jueves comencé con los simulacros de nuestro radar. Y bueno este es el Código que compile para poder hacer el simulacro del Radar y se veria asi:



Acá podemos ver la simulación de nuestro radar, donde gracias a nuestro código y luego de estar viendo tantos cursos e información en páginas, pudimos crear la primer interfaz de nuestro mini radar y detectar objetos.

Estoy muy feliz de que poco a poco y gracias a toda la investigación mía y de mis compañeros, podamos estar logrando paso a paso nuestro objetivo y poder tener nuestro propio radar hecho por nosotros.



No puedo grabar el funcionamiento completo de esta simulación, pero básicamente el Radar se movería 180° y cada cierta distancia detecta un objeto y por eso en la 2da imagen se ve el punto rojo dentro de la semicircunferencia verde (eso es porque detectó un objeto).

Se que no es bastante poco pero es lo máximo que puedo hacer desde casa, (Un consejo por si alguien desea crear un radar en proyecto informático: COMPRA POR LAS DUDAS LO QUE NECESITES PARA PROBARLO TODO EN TU CASA).

En este día, empezamos a realizar las conexiones del arduino y comprobar el código de processing que se ve así:

```

sketch 241114a
1 import processing.serial.*; // imports library for serial communication
2 import java.awt.event.KeyEvent; // imports library for reading the
3 import java.io.IOException;
4 Serial myPort; // Define el puerto serial
5 // Variables
6 String angle="";
7 String distance="";
8 String data="";
9 String noObject;
10 float pixsDistance;
11 int iAngle, iDistance;
12 int index1=0;
13 int index2=0;
14 PFont orcFont;
15 void setup() {
16
17   size (1200, 700); // Resolucion en pantalla
18   smooth();
19   myPort = new Serial(this,"COM3", 9600); // Da inicio al monitor serial
20   myPort.bufferUntil('.');
21 }
22 void draw() {
23
24   fill(98,245,31);
25   noStroke();
26   fill(0,4);
27   rect(0, 0, width, height-height*0.065);
28
29   fill(98,245,31); // color verde
30   // llamadas a las funciones
31   drawRadar();
32   drawLine();
33   drawObject();
34   drawText();
35 }
36 void serialEvent (Serial myPort) {
37   data = myPort.readStringUntil('.');
38   data = data.substring(0,data.length()-1);
39
40   index1 = data.indexOf(","); angle= data.substring(0, index1);
41   distance= data.substring(index1+1, data.length());
42
43

```

Descripción del código (primera captura):

Esta primera parte que se logra apreciar de la foto, lo primero que realicé fue en importar la librería que voy a utilizar para el proyecto y que eran necesarias para este Radar, me refiero del “import processing.serial.*;”.

Esta librería es de vital importancia para poder recibir datos de un dispositivo externo (en este caso lo necesitamos debido a que el Arduino al momento de exponerlo va a estar conectado a través de un cable conectado a la computadora, y en la computadora va a estar el código de Processing) y sin esa biblioteca, no puedo usar objetos de tipo “serial”.

A continuación tenemos la declaración de variables y implemente un “ PFont orcFont ”, lo que significa el “ PFont ” es una variable de tipografía para que pueda poner el tipo de letra que quiera y también va a ser necesario para que me haga las líneas diagonales, cuando lo logre terminar se verán las líneas a las que me refiero.

En el apartado de “ myPort = new Serial(this,”COM3”,9600); ” se refiere a que declaramos que la variable “ myPort ” va a ser la que muestre la interfaz del radar y se usa el “ COM3 ” pq es el puerto de la PC que estoy usando para el proyecto.

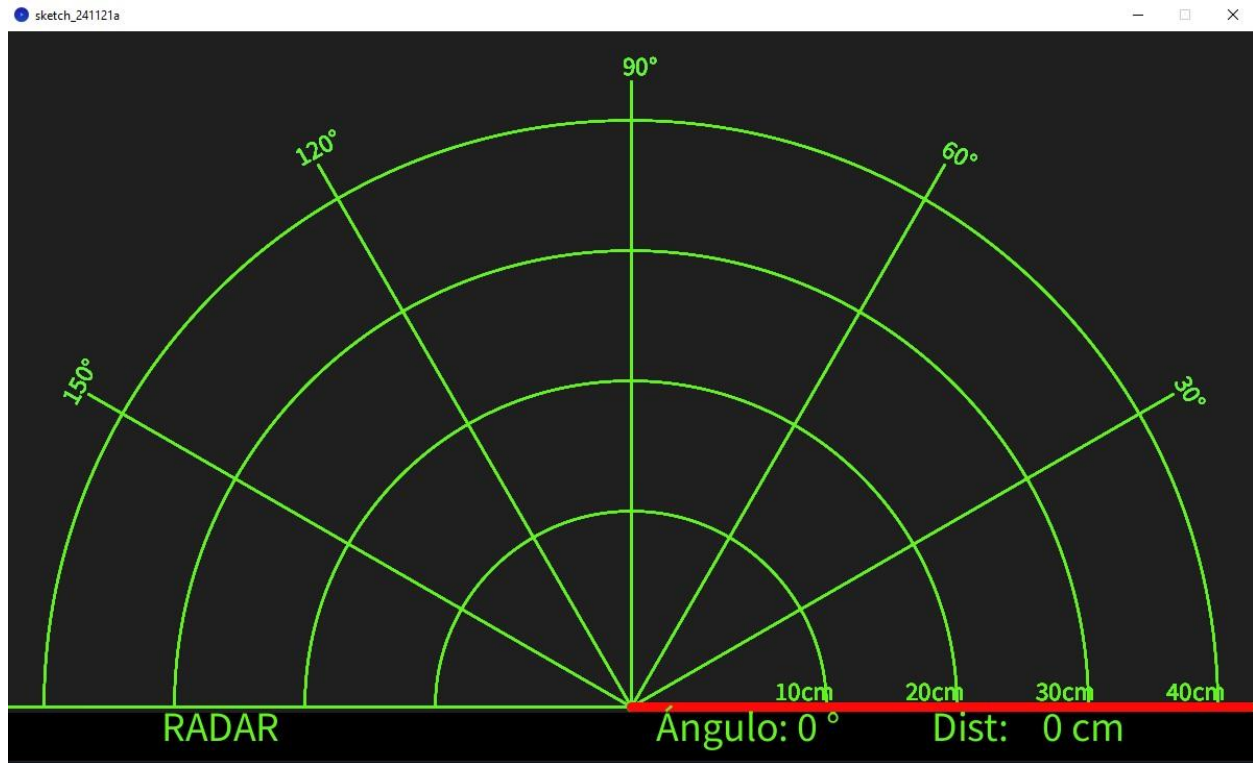
El “ noStroke(); ” es por el hecho de que en el código implementa formas geométricas (yo use el “rect” osea rectángulo) y se usa esta función para que elimine los bordes de dichas figuras (el motivo es para optimizar la interfaz).

Los “ fill (98,245,31); ” es para rellenar con color verde el rectángulo ya mencionado antes, osea el “ noStroke(); ” y el “ fill(98,245,31) ” van a hacer efecto en el rect (rectángulo).

Después se prosigue con las “ llamadas a las funciones ”, en el código llame a “ drawRadar (); drawLine (); - drawObject (); y drawText (); ”.

Recurrir a las funciones por el hecho de poder organizarme en el código, así podía asignar una tarea a cada función como por ejemplo. “ drawRadar (); ” hará el dibujo del Radar y bueno, así sucesivamente con cada

Gracias a la implementación de ese código se logra apreciar esta interfaz:



<https://howtomechatronics.com/projects/arduino-radar-project/#h-building-the-device>

En este sitio que lo compartí anteriormente, no sabía que había un apartado que explicaba en cómo pensaron el desarrollo del radar, y otros proyectos a lo que voy es en que me sirve demasiado el código fuente que usaron en cada proyecto ya que en todas las investigaciones que hice en ninguna había visto acceso a códigos fuentes de processing sin explicación, mas allá que vi cursos y otras cosas de processing

<https://www.humix.com/video/flcd124497c96593944288908250824fed49b992>

En el siguiente video explican o al menos rescato en cómo lograron hacer que el servo gire con el sensor

Debíamos darle la funcionalidad extra a nuestro radar, pero no parecía que sea algo tan complejo, la funcionalidad del radar es que pueda medir la velocidad de los objetos que detecte en base a su rango (por

el momento su rango es de 40 cm y digo por el momento por si es que funciona mal el radar o algo similar).

Específicamente no se como hacerlo, me guié con datos de Google, para ser más preciso primero me enfoque en buscar cómo se calcula la velocidad. https://morsmal.no/wp-content/uploads/2019/08/Vei_fart_tid_og_akselerasjon_spansk_UU.pdf

Mediante este link, se puede ver que en la página indica que para calcular la velocidad se necesita saber la distancia (que mi radar lo cumple) dividida por el tiempo. De todo esto iba bien del lado de la distancia ya que en pantalla aparece la distancia del objeto que detectó, el problema era el tema del TIEMPO, por que no se como aplicarlo u usarlo. Una vez encontrado la distancia y el tiempo se tiene que hacer la siguiente

$$\text{Velocidad} = \frac{\text{desplazamiento}}{\text{tiempo empleado}}$$

$$\vec{V} = \frac{\vec{d}}{t}$$

fórmula:

(El DESPLAZAMIENTO es la distancia)

<https://youtu.be/dqPzRh1JaBs?si=BHBw7ObdQhoS4enY>

En este video explica el concepto de la velocidad en un sensor ultrasónico, el sujeto utiliza 2 distancias que lo entendí como “ Distancia Inicial “ y “Distancia Final “ y esto lo va a dividir por una “ diferencia ” que este será el tiempo que se va a tardar el radar en detectar otro objeto (si es que detecta otro objeto).

Una vez ya el proyecto va encaminado voy a estar nombrando, los errores y complicaciones que se nos presentaron a la hora de crear nuestro mini radar:

El primer problema de el código de arduino solo es el mensaje del monitor serial, el problema específicamente es que me aparece así:

Distancia100.Distancia123,.....

En pocas palabras me aparece muy junto las palabras, pero supongo que es lo de menos, solo es tema del “println”. Y efectivamente si era él println, opte por usar solo serial.print

Otros de los problemas que se me presentan es en cómo hacer que se mueva el servo y que se vea en pantalla, es decir, la línea roja que aparece en la pantalla (se logra ver en las imágenes de la etapa 2) se tiene que mover hasta cumplir 180° y luego volver a 0°, así sucesivamente

Una de las primeras opciones que considere fue en revisar primero el código del Arduino y al revisar el Arduino, todo funcionaba correctamente, pero note que tal vez se podía hacer el código aún más simple y coherente por lo cual era algun error del código de Processing, posiblemente se trata en el void setup(), por que acabo de ver que no implemente algún for, que con este comando podría hacer que gire y lo detecte la app de Processing. Con ayuda de Consorti nos dijo que el error era o del Sensor o algun error de conexion, no pude verificar el error ese mismo día por el simple hecho de que el profesor de la materia ya se iba.

Opte por cambiar gran parte del código de Processing, como por ejemplo, en las librerías que implemente, vi que NO era necesario usar librerías de tipo “Java” en el Processing, esas librerías reconozco que las

saque de google, pero al entender mejor cómo funcionaba processing me di cuenta que es obsoleto para el armado del Radar o al menos algo a tener en cuenta, así que sustituí esto:

```

1  import processing.serial.*; // imports library for serial communication
2  import java.awt.event.KeyEvent; // imports library for reading the
3  import java.io.IOException;
4  Serial myPort; // Define el puerto serial
5  // Variables
6  String angle="";
7  String distance="";
8  String data="";
9  String noObject;
10 float pixsDistance;
11 int iAngle, iDistance;
12 int index1=0;
13 int index2=0;
14 PFont orcFont;
15 void setup() {
16
17   size (1200, 700); // Resolucion en pantalla
18   smooth();
19   myPort = new Serial(this,"COM3", 9600); // Da inicio al monitor serial
20   myPort.bufferUntil('.');
21 }
22 void draw() {
23
24   fill(98,245,31);
25   noStroke();
26   fill(0,4);
27   rect(0, 0, width, height-height*0.065);
28
29   fill(98,245,31); // color verde
30   // llamadas a las funciones
31   drawRadar();
32   drawLine();
33   drawObject();
34   drawText();
35 }
36 void serialEvent (Serial myPort) {
37   data = myPort.readStringUntil('.');
38   data = data.substring(0,data.length()-1);
39
40   index1 = data.indexOf(","); angle= data.substring(0, index1);
41   distance= data.substring(index1+1, data.length());
42
43

```

A este código:



```

1 import processing.serial.*; // importa la libreria para la comunicacion serial
2 // importa la libreria para el puerto serial
3
4 Serial myPort; // defino el objeto serial
5 // defino variables
6 String angle="";
7 String distance="";
8 String data="";
9 String noObject;
10 float pixsDistance;
11 int iAngle, iDistance;
12 int index1=0;
13 int index2=0;
14 PFont orcFont;
15 void setup() {
16
17   size(1200, 700); // ESTO LO CAMBIO DEPENDE MI RESOLUCION
18   smooth();
19   myPort = new Serial(this, "COM3", 9600); // inicia la comunicacion serial
20   myPort.bufferUntil('.'); // Lee los datos del puerto serial up to the character '.'. So actually it reads this: angle,distance.
21 }
22 void draw() {
23
24   fill(98, 245, 31);
25   // simulating motion blur and slow fade of the moving line
26   noStroke();
27   fill(0, 4);

```

Me pareció más cómodo la implementación de funciones ya que en clases de laboratorio nos enseñaron a usar las funciones y las llamadas a dichas funciones

En clase probé el código y al menos funcionaba y uno de los problemas más grandes que tenía en todo el proyecto fue lograr que gire o que la interfaz del radar se vea que gire los 180°, sin embargo el nuevo problema que se me presento en esto, fue que por alguna extraña razón empezaba a girar desde los 10° e ignoraba los 0°, erróneamente pense que el error era un `if idistance < 40`, cuando en realidad hacía referencia a que si un objeto se detectaba a menos de 40 cm, tenía que aparecer en pantalla que estaba en el rango y lo tenía que indicar con unas “ manchas “ rojas en la interfaz de processing, así que ese no era la forma de solucionar el error, sinceramente había ocasiones en las que funcionaba a la perfeccion y otras veces andaba mal, así que para despejar dudas, iré con el profesor a despejar estas dudas:

¿Dónde está el error? - ¿Qué modificaciones le puedo hacer a mi código?. O al menos esas son las dudas que se me plantean ahora mismo.

Dado que no pudimos hablar con el profesor, no me queda otra que arriesgar e intentar complementar el código con lo que tenga, sin saber si esta bien o mal

Hasta el momento mi codigo no está del todo listo, lo que rescato o al menos destaco es que:

1. Dibuja la Interfaz gráfica de un radar
2. Gira los 180° e regresa a su valor inicial que son 0°
3. Detecta un objeto (aunque este lo detecta con un delay de 10 segundos aproximadamente)

Por ultimo nos faltaría agregarle la funcionalidad extra, que es la medición de velocidad del objetivo que detecta el radar.