

Mini radar/sonar

Grupo

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

Esta es mi carpeta de campo para nuestro proyecto del Radar, en donde separe por “etapas” cada paso que hice del proyecto, contando desde los errores que tuve, la investigación que realice y los avances que se lograron durante el transcurso del proyecto, debo decir que no pude agregar la funcionalidad por el hecho de que no comprendí las explicaciones de Youtube y bueno no se pudo concretar la idea.

El problema más grande de todo este proyecto fue el movimiento del servo, lo tuve que solucionar de forma independiente ya que con el profesor de Base de datos no logramos solucionarlo.

Explore las funcionalidades que disponía Processing ya que es un programa que fue relevante para crear la interfaz del radar, y usando muchos cursos y algunos videos que mandó el profesor, logre hacer el código en Arduino.

Y agregue masomenos una explicación de que es un radar y un poco de historia para hacerse una idea el por que y cuando llegaron los radares al mundo que hoy en día lo usan los aviones, submarinos,etc.

Etapas 1: Investigación

Este es el primer día de investigación para el proyecto, en estos meses estaré mandando el progreso que haga para armar un RADAR, es lo que tengo que exponer.

Un radar cuyo acronimo es (RAdio Detection And Ranging) es un sistema que usa ondas electromagnéticas para medir distancias, altitudes, direcciones y velocidades de objetos (Esta es la otra función que deberá hacer mi Radar) , como aeronaves, barcos, vehículos motorizados, formaciones meteorológicas y el propio terreno.

Su funcionamiento se basa en emitir un impulso de radio, que se refleja en el objetivo y se recibe típicamente en la misma posición del emisor. El uso de ondas electromagnéticas con diversas longitudes de onda permite detectar objetos más allá del rango de otro tipo de emisores.

LINK: <https://es.wikipedia.org/wiki/Radar>

Los Radares se introdujeron durante la Segunda Guerra Mundial para medir la distancia y la velocidad de los objetos usando ondas de radio. En la guerra sirvió como un sistema de alerta temprana, detectando aviones enemigos distantes, que de otro modo eran indetectables a simple vista

LINK:

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

Una vez explicado masomenos que es un radar y cuando se introdujeron es hora de explicar masomenos el objetivo de este proyecto, para el Radar que tengo que elaborar, tengo que realizar una interfaz de un radar promedio y que este me muestre los datos que quiero, en este primer dia me enfoque en investigar el programa en el cual me muestre la interfaz del radar en pantalla.

Recurrí a Google, y todas recomendaban usar “Processing”. Adjunto los links de 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>

Así que no tengo más remedio que aprender a programar en Processing, busque en Google cursos de como usar Processing. Adjunto el link de la página

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

A su vez busqué algún curso 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](#)

La idea principal o al menos el “objetivo” es lograr que el radar se vea así:



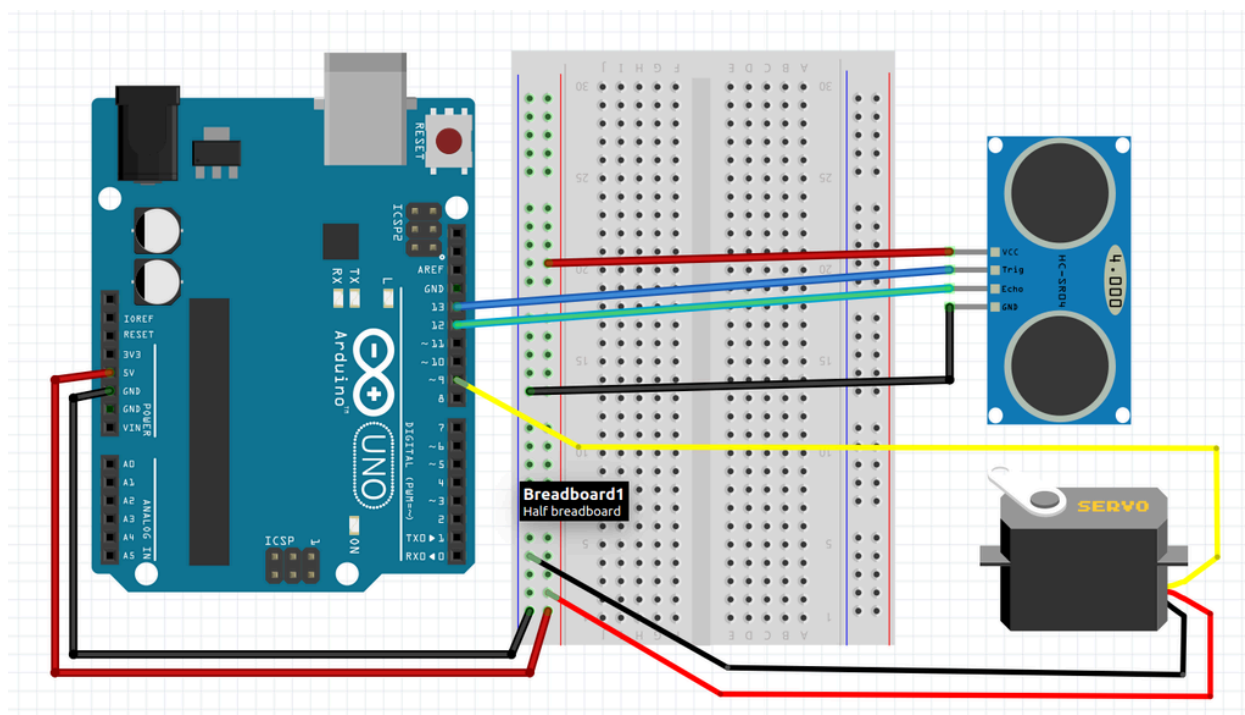
(Foto sacada de Google)

Processing es una herramienta a través de la cual podemos hacer 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 vamos a hacer la interfaz que tiene un radar promedio.

Pero primero para este Radar tengo que entender masomenos JAVA o nose, yo creo que va a ser necesario por las dudas, así que por unos días estare 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 Consorti.

Después de tomarme 2 días en los que solo vi los cursos de Processing, empecé a buscar por google alguna “maqueta” o esquema para hacer las conexiones para armar el radar.



Y con 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 utiliza 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.>

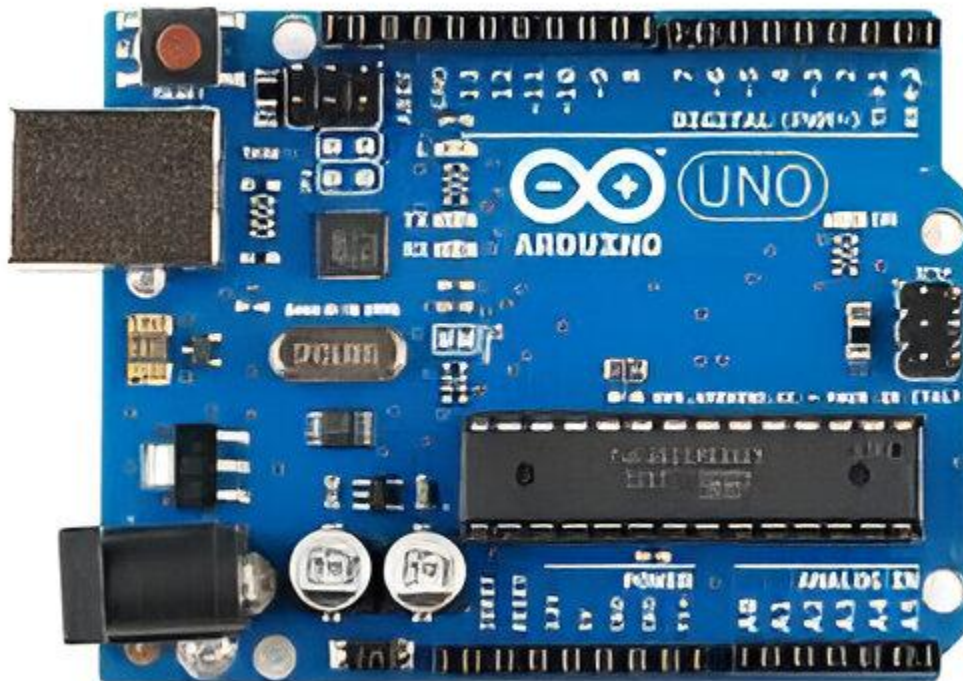


Imagen de un Arduino Uno

Sensor Ultrasónico: 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.

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: 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): 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)

En esta ocasión empecé a buscar sobre el código, use fuentes como Google, Youtube, bueno en Youtube solo fueron los cursos que adjunte anteriormente, en Google visite paginas para saber de que era fundamental para armar el radar.

una de esas es:

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

que de esta página me quede con el siguiente código:

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

Este 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 lo que no sé es cómo hacer que el radar gire 180°.

Como 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.

Y al final tuve razón 😊, solo que se necesita implementar una librería llamada "Servo.h" osea es

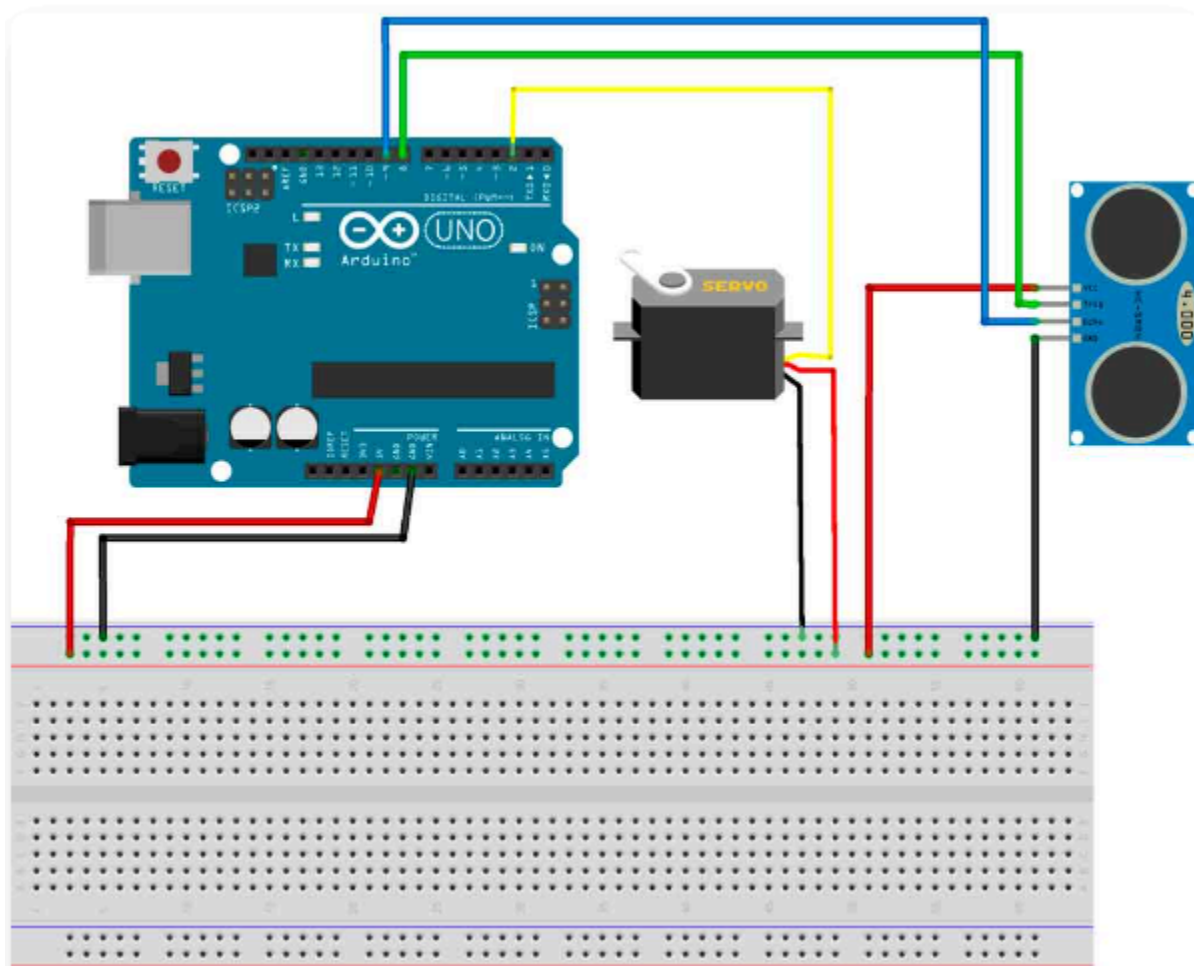
OBLIGATORIO usar la librería "Servo.h"

página usada:

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

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.

Esta Info esta sacada de: <https://manueldelgadocrespo.blogspot.com/p/biblioteca-servo.html>



Este es un modelo 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.

Son las funciones que dispone esta librería.

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

Otra de las alternativas que no había considerado hasta la fecha, 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 sucedió esto, 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 muy útil 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.

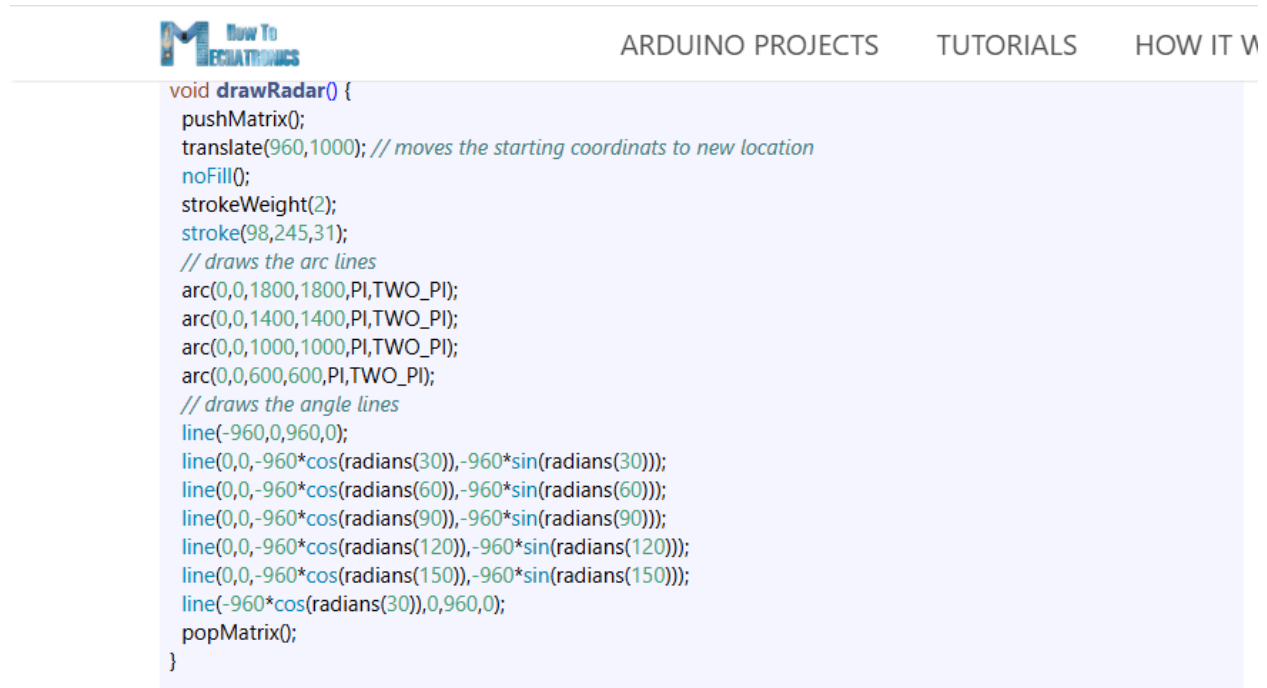
Seguí investigando otras fuentes para entender bien el planteo que se debía hacer en el código de Processing, no me esperaba que iba a ser lo más complicado del proyecto

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

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

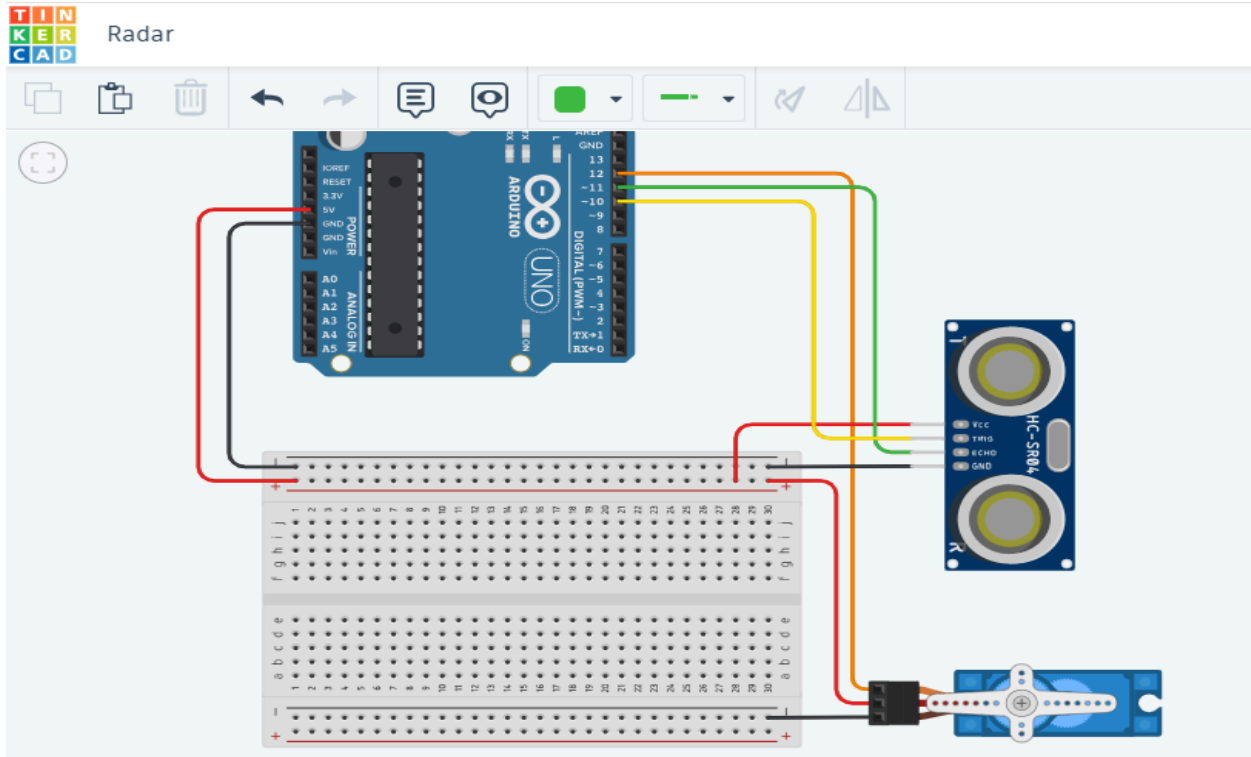
Este es otro de los Links que visite, la idea principal de buscar estos códigos es para ver alguna similitud que tienen ambos códigos, para que en el momento de hacer el código no tenga complicaciones o al menos esa es la idea.

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().



Etapas 2: Prueba y ERROR

Ahora, después de cursos, visitar páginas y etc. Empieza la construcción del radar usando de referencia todo lo adjuntado hasta el momento, empezando con la maqueta en el Tinkercad, adjunto imagen:



Como se ve en la imagen segui el ejemplo de la maqueta anterior q mande, solo que en esta se ve mejor o al menos más entendible las conexiones, tal vez había otra forma de hacerlo mejor, pero fue hasta ahí donde llego mi intelecto 😊

Ahora con el apartado del código recopilé y use las páginas de google que adjunte y la clase grabada de Consorti de como usar el sensor ultrasónico para masomenos como usarlo (sobre la clase grabada me hice apuntes por que sino no me acordaba).

Para que, me quede masomenos asi el codigo:

```
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);
  }
}
```

Para cada línea de código se usaron comentarios para explicar masomenos para que sirve cada instrucción, aunque no se ve tan visible, aca un explicación:

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 librería, que anteriormente había comentado que SI O SI se tiene que usar para hacer este proyecto

“long duracion;” Se usa 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;” Acá 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 disancia medida
}

```

Del lado del Processing me informe un poco más, en lugar de cursos de Youtube que algunos me sirvieron y otros fueron una pérdida de tiempo, 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

Alta pagina.

Gran parte de los comandos que se utilizaron en la construcción del Radar fueron sacados de esta página

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

PEEEERO, 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 Informatico). 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 está bien, pero el funcionamiento es similar, solo que varía el diseño.

```

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);

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

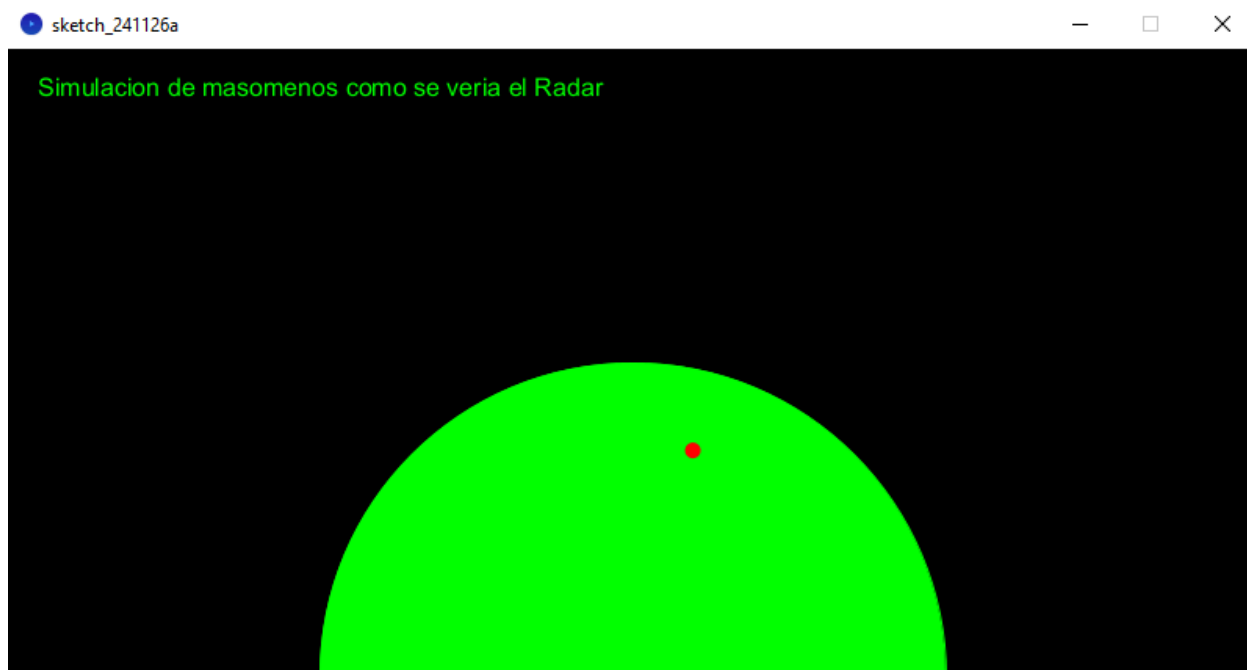
  // Actualiza el ángulo
  angle += 1;
  if (angle >= 180) {
    angle = 0;
  }

  // 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
}

```

Y bueno este es el Código que compile para poder hacer el simulacro del Radar y se veria asi:



No puedo mandar una grabación del 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 pq detectó un objeto).

Se que no es mucho, pero es lo maximo que puedo hacer desde casa, (Recomendación para alguien que quiera hacer un Radar o algún proyecto de Proyecto Informático: COMPRA POR LAS DUDAS LO QUE NECESITES PARA PROBARLO TODO EN TU CASA).

En este dia, 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

```

Desafortunadamente para mostrar el código o el progreso hasta el momento se tiene que mandar fotos que ocupan ese tamaño en la hoja.

Descripcion del codigo (primera captura):

En fin, 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 SI o SI necesaria para 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.

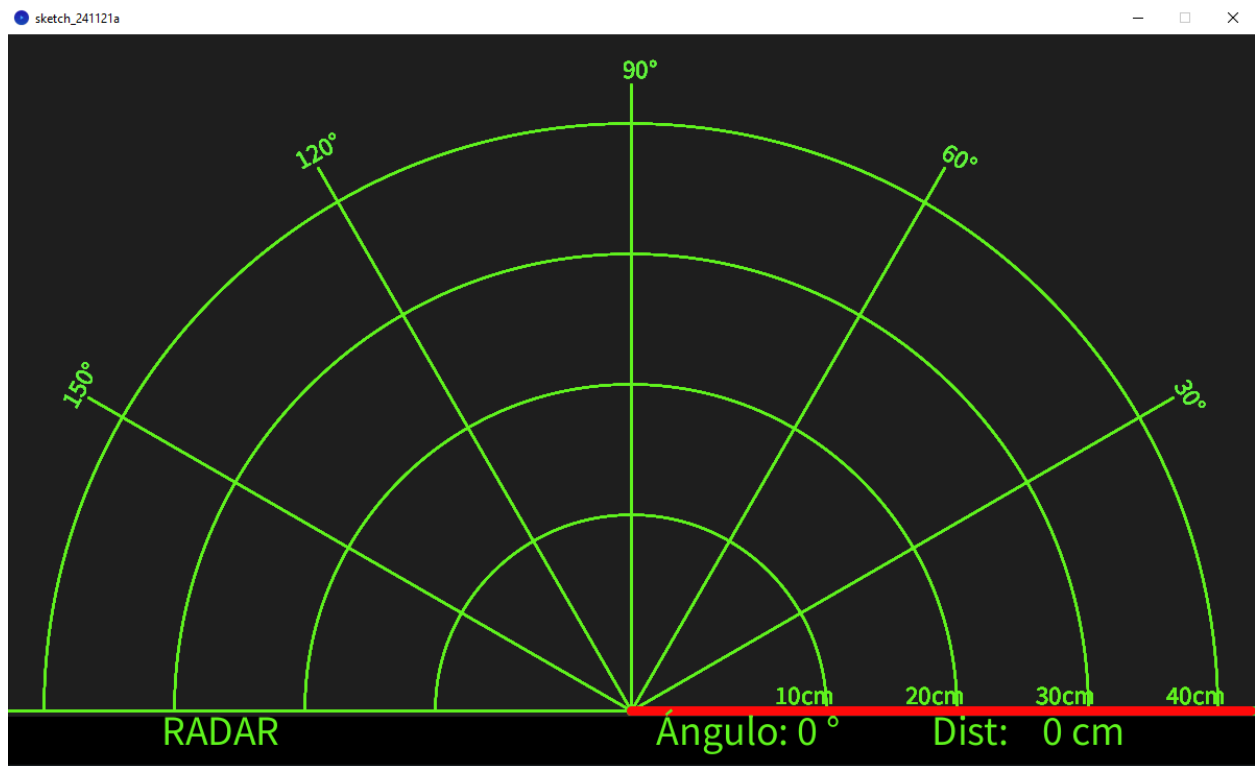
El “noStroke();” es por el hecho de que en el código implementa formas geométricas (yo use el “rect” osea rectángulo) y se usa esta función para que elimine los bordes de dichas figuras (el motivo es para optimizar la interfaz).

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

Con este 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 sabia que habia 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 explicacion, mas alla 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

Me faltaba la funcionalidad, pero no consideraba 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 de las cosas que voy a probar va a ser combinar el codigo del video ya que funcionaba correctamente en la forma de como calcula la velocidad

Etapas 3: Errores e Complicaciones del 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 algún 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 algún error de conexión, 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

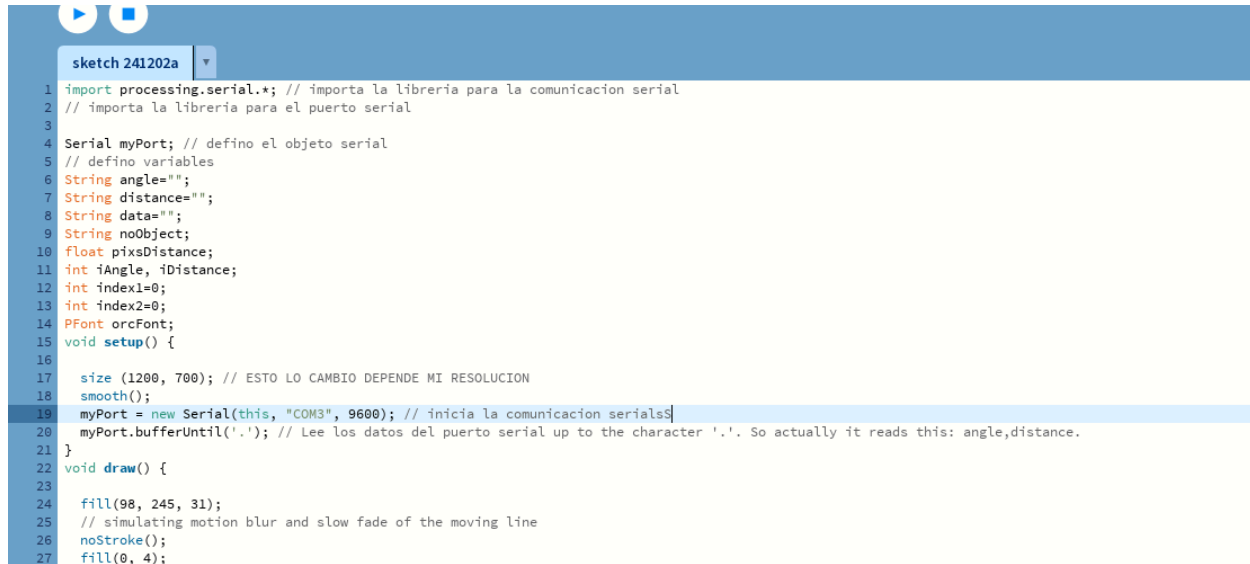
armado del Radar o al menos algo a tener en cuenta, así que sustituí esto:

```

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

```

a esto:



```

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);

```

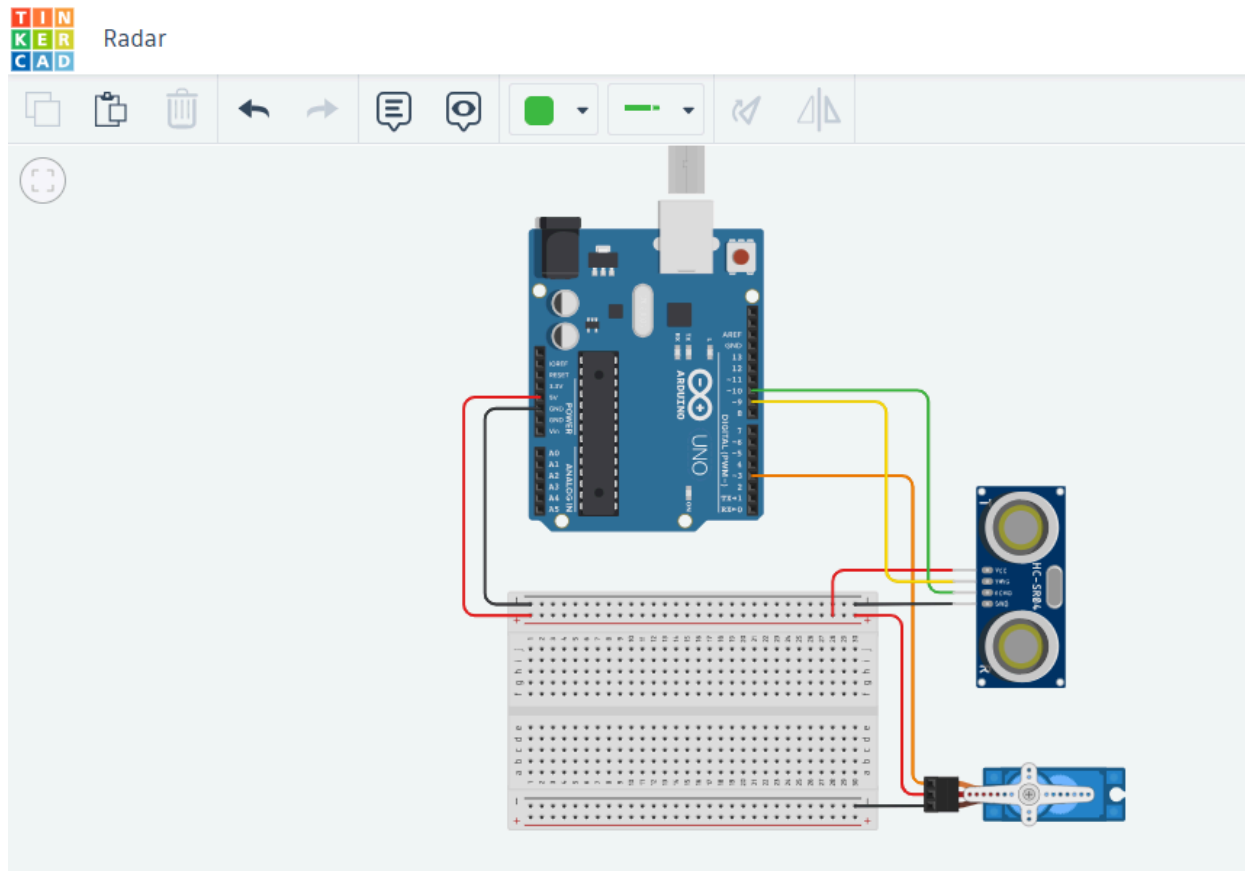
Me parecio mas comodo la implementación de funciones ya que en clases de laboratorio nos enseñaron a usar las funciones y las llamadas a dichas funciones

En clase probe el codigo 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, asi 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 pude 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

Al borde de entregar el proyecto se presento un problema con el servo motor ya que este dejo de girar repentinamente, antes podia hacer su giro (0° a 180°), no es un error de conexion dado que conecte todo segun la maqueta, adjunto la imagen de las conexiones.



El error podía ser de los propios materiales, revise que no se haya roto alguno u algo similar pero no fue asi, asi que pense q se trataba del código o del propio COM, pero el error estaba en el código, ya que no había establecido bien los for (No se por que motivo se modifiko el for) y lo malo de este es que el servo seguía girando de a 2° cuando en realidad debería moverse de forma continua, esto podia ser por temas de la fuente de alimentación. Este dato me lo planteo google

servo motor pq gira lento solucion

X | 🔊 | 🌐 | 🔍

mas preguntas :

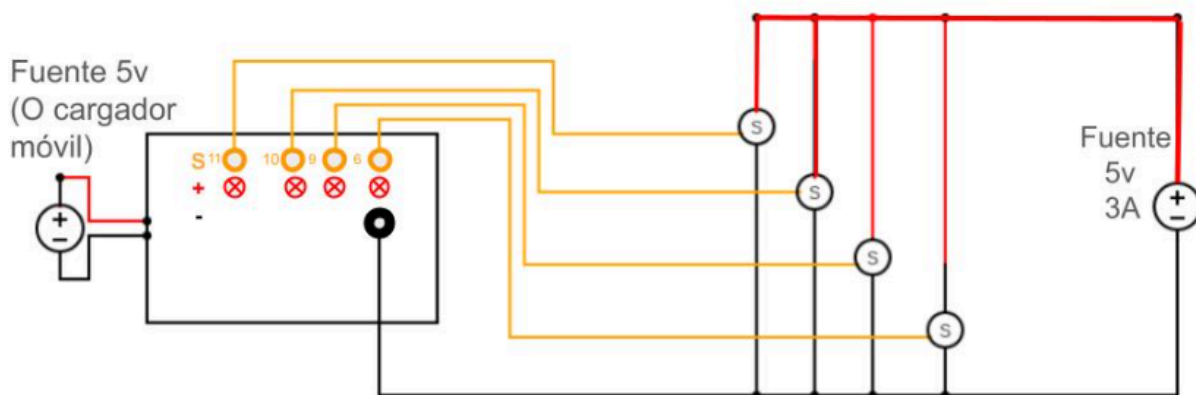
¿Cómo hacer que el servo sea más rápido?

Para cambiar la velocidad del servo, deberá **ajustar el ancho de pulso de la señal PWM**. Si el servo se mueve actualmente a una velocidad lenta, por ejemplo, puede aumentar la velocidad disminuyendo el ancho de pulso por debajo de 1,5 milisegundos. Esto hará que el servo se mueva más rápido. 1 ago 2023



Lunye Electric Motors and Drives Supplier
<https://www.lunye.com> > news > faq > how-to-change-s...

Este problema lo consulte con un tecnico del tema, aunque este me intento dar una solucion al problema con un esquema de esta magnitud:



El negativo de la fuente de 5v 3A puede ir en cualquiera de los pines del modulo que tenga indicado el signo -

En los pines de + señalizados con una X no conectar nada.

La idea es usar la fuente externa para alimentar los servos.

Y el shield y el arduino servirán para enviar las señales a los servos.

El Shield y el arduino acoplados pueden ser alimentados con otra fuente, que no tiene que ser necesariamente potente mientras tenga 5v y 1A.

A decir verdad la imagen que me mostró no la entendí, pero me planteo que lo que podía hacer era una fuente de 5V que tenga suficiente poder para mover el servo. VCC y GND de la fuente a los VCC y GND del servo.

Cable de señal a Arduino.

Tierra de la fuente y tierra del arduino puenteadas

Esas fueron las especificaciones que me dio en texto, y bueno, mi idea era conseguir un cable de alimentación que suministre al servo motor ya que no giraba como se esperaba.

En youtube, pude encontrar algunas opciones o ayudas para lograr que gire como se esperaba el servo motor. Adjunto los LINKS.

<https://youtu.be/z61xkhJrRKw?si=f1ff2yNGa34eVv9l>

aca te explicaba el sujeto en cómo hacer que el servo girase en distintas velocidades

<https://youtu.be/HQqwLuJ1pbo?si=IaT2cXCkm2T5enkN>

y ante mi frustración al ver que el servo NO giraba como antes, me puse a buscar como se conectaba por si es que yo lo estaba haciendo mal

```
void loop() {
  for(int i=0;i<=180;i+=5){
    myServo.write(i);
    delay(0);
    distance = calculateDistance();
    Serial.print(i);
    Serial.print(",");
    Serial.print(distance);
    Serial.print(".");
  }
  for(int i=180;i>0;i-=5){
    myServo.write(i);
    delay(0);
    distance = calculateDistance();
    Serial.print(i);
    Serial.print(",");
    Serial.print(distance);
  }
}
```

Gracias a los videos pude crear este código que es funcional para que pueda girar el servo motor 180°, el problema sigue siendo la velocidad de su traslado, pero al menos es un avance.

Recurrir a pedir ayuda al profesor de Base de datos y creo q a su asistente también (no se quien era el otro docente que me ayudó), pero estuvimos viendo lo comentado anteriormente, sobre el traslado del radar que giraba de forma discontinua, primero probamos con un código de arduino simple para girar el servo motor, pero el resultado fue el mismo (se trasladaba los 180° de forma lenta), luego Damian se encargó en usar un transformador de 5V para darle mas potencia al servo motor y lo unico que cambio fue que el movimiento del servomotor fue un poco más rápido y continuo pero no es lo que buscaba.

Probe de forma independiente (osea en mi casa) empezar desde 0 las conexiones, recurri a conectar la señal del servomotor a A0.

```

#include <Servo.h>
const int trigPin = 9;
const int echoPin = 10;
Servo servo1;

void setup() {

    pinMode(trigPin, OUTPUT);
    pinMode(echoPin, INPUT);
    servo1.attach (A0);
    Serial.begin(9600);
}

void loop() {
    int i;
    for(i=0;i<180;i++)
    {
        servo1.write(i);
        delay (5);           //VELOCIDAD DEL SERVOMOTOR
    }
    for(i=180;i>0;i--)
    {
        servo1.write(i);
        delay (5);           //VELOCIDAD DEL SERVOMOTOR
    }

    digitalWrite(trigPin, LOW);
    delayMicroseconds(2);
    digitalWrite(trigPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPin, LOW);

    long duration = pulseIn(echoPin, HIGH);

```

Esa es una imagen de como era mi planteó en un principio para que el servo se moviera de forma continua, y funcionó dentro de todo, solo que la conexión A0 no suele usarse para un trabajo de esta magnitud, así que luego lo cambió al pin 11.

Por otro lado para el sensor ultrasónico decidí hacer lo mismo, hacer un código aparte para ver si funcionaba o no, implemente un código simple, mando la imagen:

```
const int trigPin = 9;
const int echoPin = 10;

void setup() {

  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);

  Serial.begin(9600);
}

void loop() {

  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  long duration = pulseIn(echoPin, HIGH);

  float distance = duration * 0.034 / 2;
  Serial.print("Distancia: ");
  Serial.print(distance);
  Serial.println(" cm");

  delay(500);
}
```

este código lo que hace es que el sensor ultrasónico pueda detectar los objetos o al menos mediante el monitor serial muestra la distancia en la que está un objeto (tiene un rango de más o menos 800 cm).

Al ver que funcionaban ambos códigos, opte por unir ambos código (lo único que cambié fue la señal del servomotor) y de paso lo uni con los códigos que tenia y mostré anteriormente y bueno, quedo así:

sketch_dec4b.ino

```

1  #include <Servo.h>.
2  const int trigPin = 10;
3  const int echoPin = 11;
4  long duracion;
5  int distancia;
6  Servo myServo;
7  void setup() {
8      pinMode(trigPin, OUTPUT);
9      pinMode(echoPin, INPUT);
10     Serial.begin(9600);
11     myServo.attach(12);
12 }
13 void loop() {
14     // rota el servo de 0 a 180°
15     for(int i=0;i<=180;i++){
16         myServo.write(i);
17         delay(30);
18         distancia = calcularDistancia();// hacemos una llamada a la funcion que mide la distancia
19     }

```

Declaración de las variables y el ciclo for.

sketch_dec4b.ino

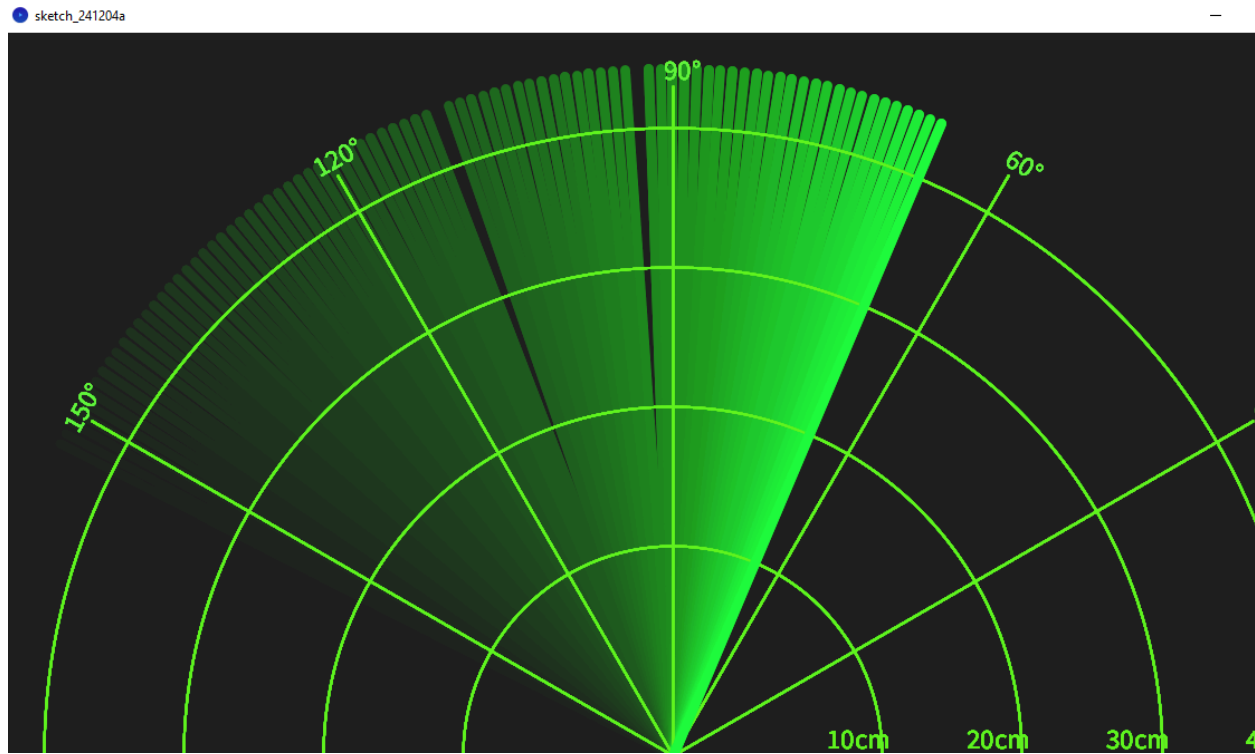
```

25     // regresa de 180 a 0°
26     for(int i=180;i>0;i--){
27         myServo.write(i);
28         delay(30);
29         distancia = calcularDistancia();
30         Serial.print(i);
31         Serial.print(",");
32         Serial.print(distancia);
33         Serial.print(".");
34     }
35 }
36 // Funcion que calcula la distancia del sensor
37 int calcularDistancia(){
38
39     digitalWrite(trigPin, LOW);
40     delayMicroseconds(2);
41     digitalWrite(trigPin, HIGH);
42     delayMicroseconds(10);
43     digitalWrite(trigPin, LOW);

```

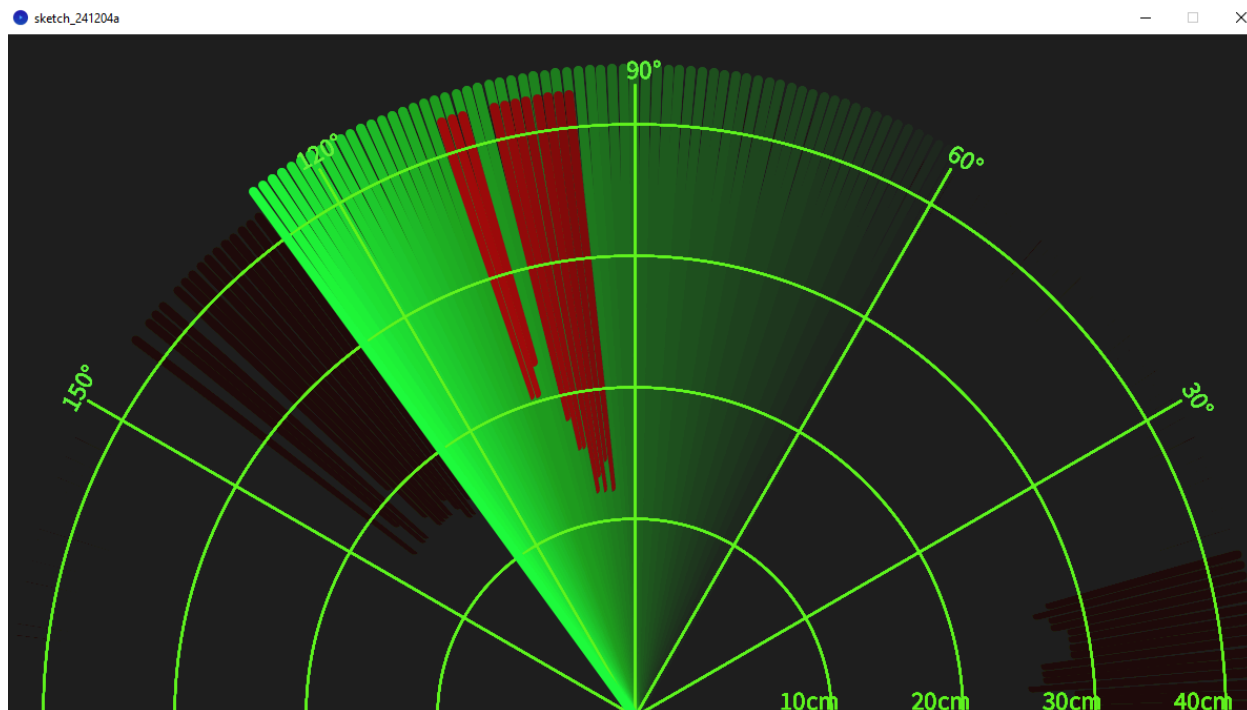
Y por último el 2do for y llamada a la función que calcula la distancia obtenida

Con todo este código, después de muchos intentos logre que el processing me muestre bien la interfaz y que me detecte los objetos que estén a 40 cm de distancia del sensor ultrasónico.



es la interfaz de processing del radar en funcionamiento, por el momento no detecta nada por el hecho de que no hay nada a su alcance (40 centímetros)

Si detecta un objeto, la pantalla aparecera en rojo, por ejemplo:



Hasta el momento mi código 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 que esté en su rango de 40 centímetros

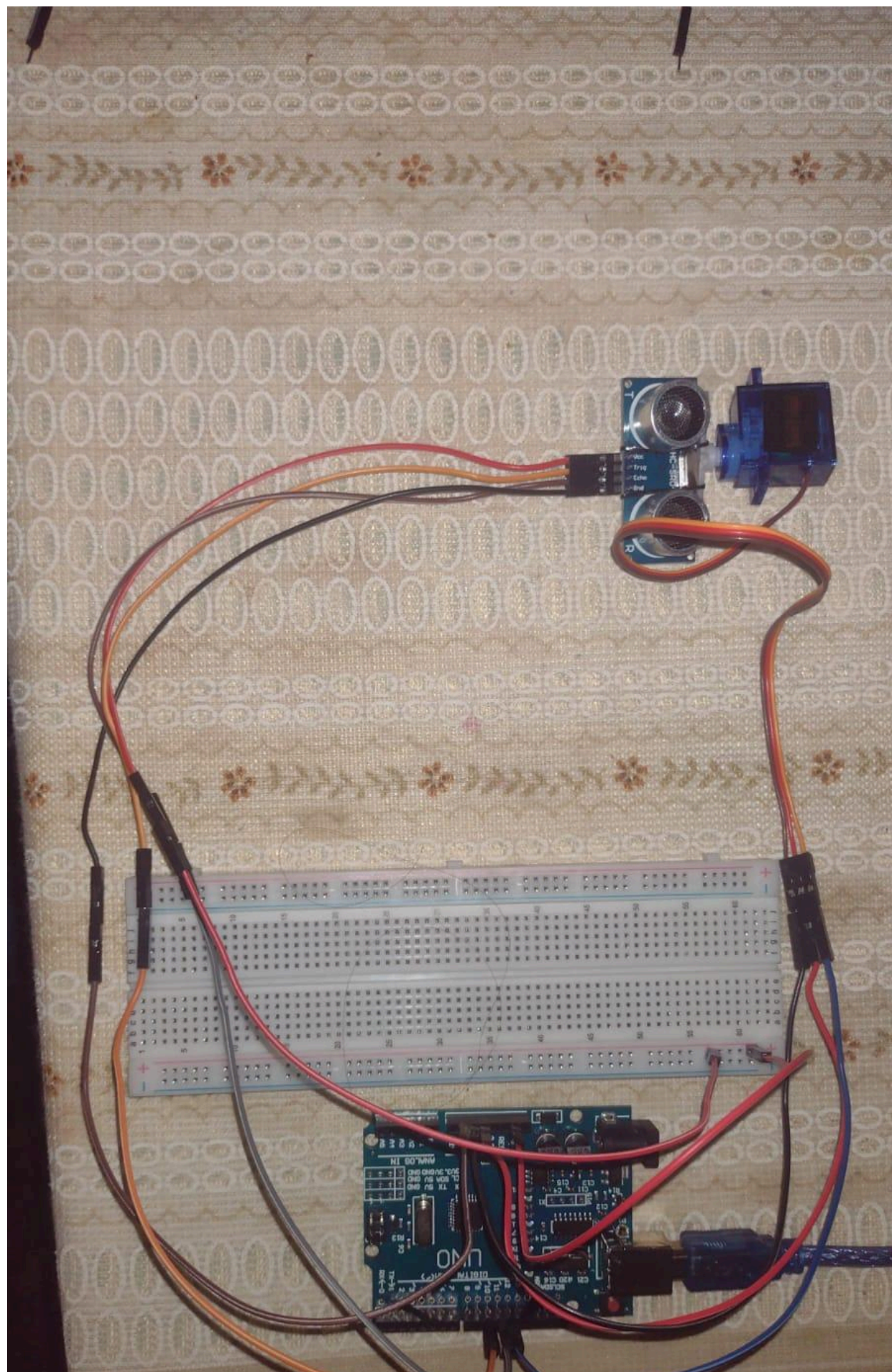
Hasta el momento elaboré un 75% del proyecto, el problema es ahora la funcionalidad.

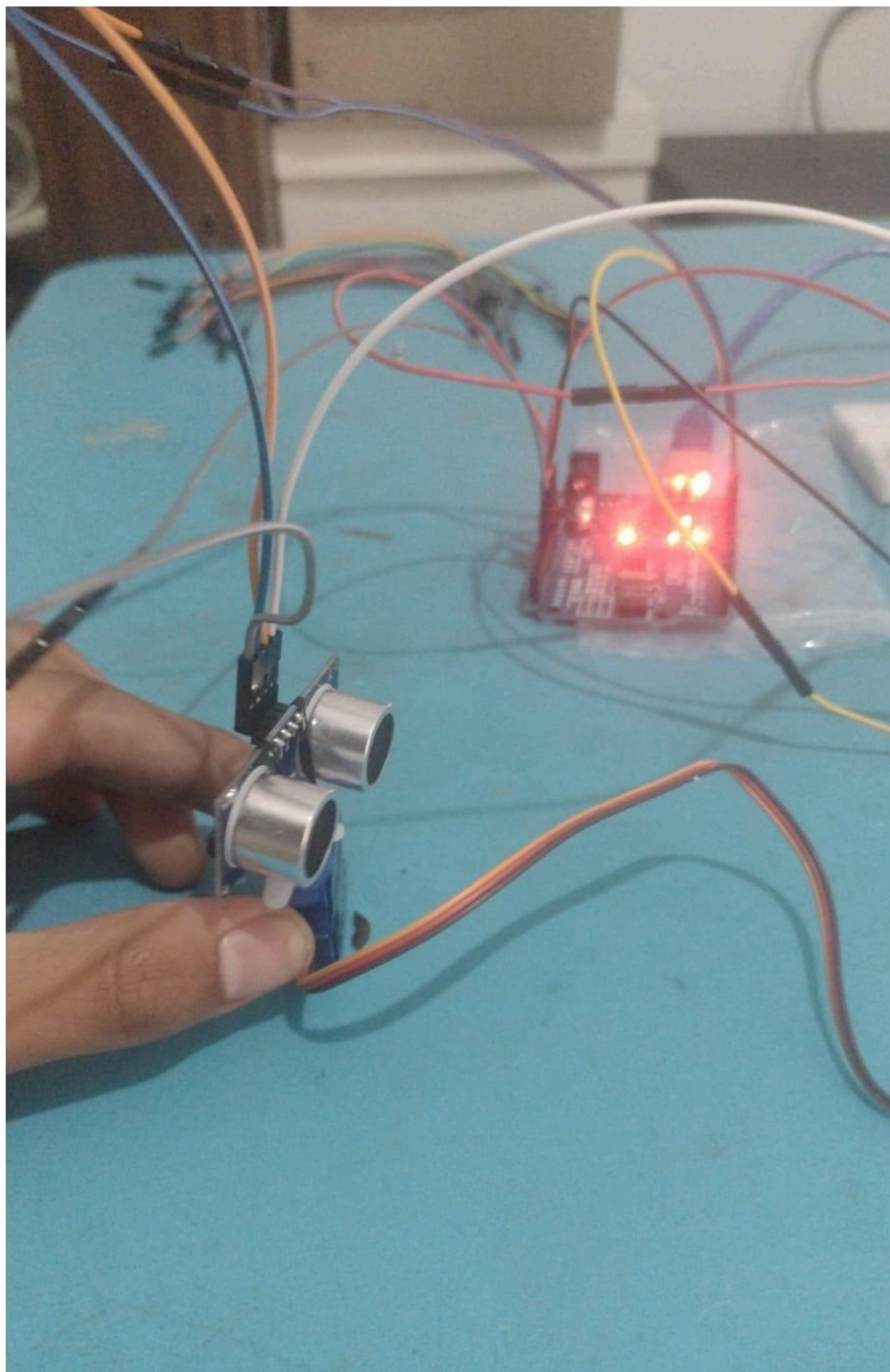
Sobre esto, anteriormente investigue en como calcular la velocidad de un objeto e inclusive planteé un código en Arduino funcional que detectaba la velocidad de un objeto, lo que no se aun es en cómo lo puedo implementar en el Processing

Etapla 4: Conclusiones e

Implementaciones finales

El proyecto lo finalizo o lo doy por concluido hasta acá, logré que se vea como un radar, con su interfaz y que pueda captar objetos, pero no pude con su funcionalidad de calcular la velocidad, por el tiempo que queda, no me voy a arriesgar en probar mas codigo pq no quiero correr el riesgo de que este mal el codigo o algo similar. Y todo esto fue por falta de conocimiento e por que no entendía los videos de Youtube, a lo mejor conversando con el profesor de la materia seguramente habría tenido algo, pero no fue el caso







Y ahí mando lo que se logró desde mi casa, primero las conexiones, luego el sensor y por último lo que muestra en la pantalla de Processing.