Benjamín Ibañez Luna

Grupo 2

-Radar

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

4to 2da: Proyecto Informático

Gonzalo Consorti

24/9/24 - 2/12

**CARPETA DE CAMPO**

**OBJETIVOS**

En este breve espacio brindaré los objetivos del proyecto llevado a cabo por el grupo 2 (RADAR). Los integrantes del mismo son: Felipe Kuo Lee, Jonathan Vera, Martín Lopez Schillaci y Benjamín Ibañez.

Los objetivos son los siguientes:

* Poder realizar el proyecto y cumplir con lo solicitado, siempre y cuando no estemos muy estresados por el trabajo que conlleva dicho proyecto.
* Divertirse(obligatorio)
* Aprobar(opcional)

**INTRODUCCIÓN**

Arrancaré explicando los componentes utilizados, y el arduino. Parte Técnica: Descripción de los Componentes y Circuitos

#### Arduino Uno

#### El Arduino Uno es el cerebro del proyecto. Esta placa de desarrollo es ideal para proyectos de electrónica y programación debido a su simplicidad y flexibilidad. El modelo que usamos es un Arduino Uno con un microcontrolador ATmega328P. Se utiliza para recibir los datos del sensor ultrasónico, controlar el servomotor y gestionar la salida del buzzer. La comunicación con la interfaz gráfica de Processing se realiza a través de un puerto serie.

#### Sensor Ultrasónico (HC-SR04)

El sensor ultrasónico HC-SR04 es un dispositivo que utiliza ondas ultrasónicas para medir la distancia a un objeto. El sensor tiene dos pines principales: el Trigger y el Echo. El primero emite una señal ultrasónica, y el segundo recibe la señal reflejada del objeto. El tiempo que tarda en regresar la señal se convierte en distancia, que es lo que utilizamos para la visualización en el radar.

#### Servo Motor

El servomotor utilizado en el proyecto es responsable del movimiento de la "mirada" del radar. El servo se conecta al Arduino y se mueve en un rango de 0 a 180 grados. A medida que el servo gira, el sensor ultrasónico emite pulsos a través de sus pines de Trigger y Echo. El código controla la posición del servo, lo que permite escanear diferentes ángulos de manera secuencial. Esto imita la acción de un radar real que barre el área en busca de objetos.

#### Buzzer

El buzzer se utiliza para emitir un sonido cuando el radar detecta un objeto dentro de un rango específico. En este caso, el buzzer se activa cuando la distancia medida es menor a 20 cm, lo que indica que hay un objeto cerca del radar. Esto agrega una capa de interacción al proyecto, proporcionando una respuesta auditiva inmediata a la detección de objetos.

#### Interfaz Gráfica con Processing

#### La interfaz gráfica se diseñó en Processing para mostrar los resultados de las mediciones de distancia en tiempo real. Processing es un entorno de programación flexible y accesible, ideal para proyectos visuales. En el proyecto, la interfaz muestra un círculo que representa el radar, y a medida que el sensor ultrasónico detecta el área, los datos de distancia se actualizan en el gráfico. La interfaz también permite ver los valores numéricos de la distancia medida y proporciona retroalimentación visual sobre la detección de objetos.

Martes 15/10/24

En el día de la fecha decidí empezar a realizar mi carpeta de campo. En realidad empezamos el día martes 24/9/24, pero ese martes no había pc 's disponibles para poder empezar la carpeta de campo, así que busqué información desde mi celular. El martes siguiente no hubo clases, y el pasado falté por turno médico. En fin. Por algún lado se empieza.

Mi compañero Martin Lopez Schillaci me comentó que el martes pasado se pusieron de acuerdo para hacer el radar en 180° grados, ya que 360° era mucho trabajo y sería bastante complejo. El 24/9/24 estuve investigando en una página qué componentes y qué información íbamos a necesitar.

Encontré una con un proyecto semejante al que queremos hacer. Adjunto el link:

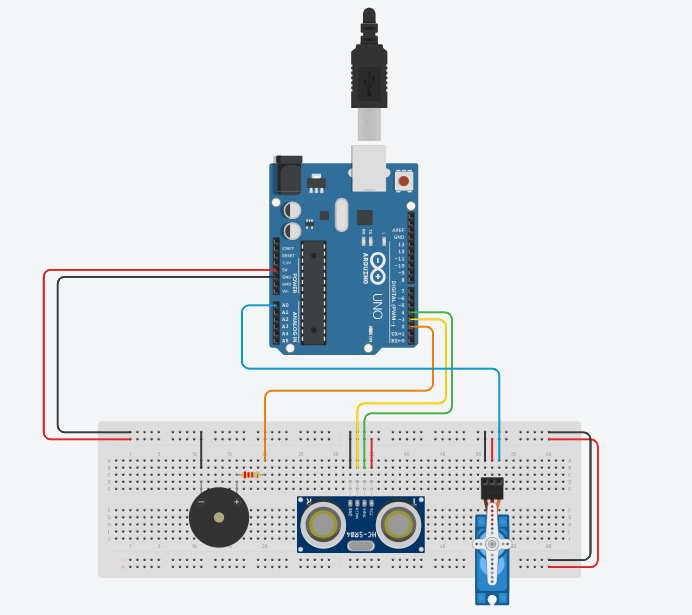
<https://www.instructables.com/Radar-Gr%C3%A1fico-Con-Arduino-Y-Processing/>

**LISTA** **DE** **COMPONENTES:**

* Arduino UNO
* Un sensor ultrasónico
* Un servo motor
* Un buzzer
* Un protoboard
* Una gran cantidad de cables

A partir de esos elementos mencionados podemos darnos una idea sobre cómo llevar a cabo la parte del hardware del RADAR.

Adjunto una imágen del proyecto realizado en tinkercad:



Para la parte del software del proyecto vamos a utilizar dos herramientas diferentes, el **IDE** de **ARDUINO** y el **IDE** de **PROCESSING.** Vamos a utilizar los códigos de la página como referencia.

Ahora voy a empezar a realizar el código en tinkercad del radar teniendo en cuenta el material encontrado. Listo los materiales que voy usando:

* Arduino UNO
* Resistencia OHM (220)
* Sensor ultrasónico
* Buzzer
* Servo motor

A la vez voy a estar repasando cómo se usaban los componentes, ya que me olvidé. Tengo que entender cómo se usa el servomotor, debido a que hoy por hoy no lo vimos con el profesor Consorti.

Hoy quisimos empezar con la parte física, pero no pudimos debido a que no teníamos el servo motor.

Martes 22/10

En el día de la fecha voy a continuar de forma individual la parte del código del radar. Jony y Feli van a hacer la parte física del proyecto. Hoy Marto faltó por segundo martes consecutivo. Me dijo que se sentía mal.

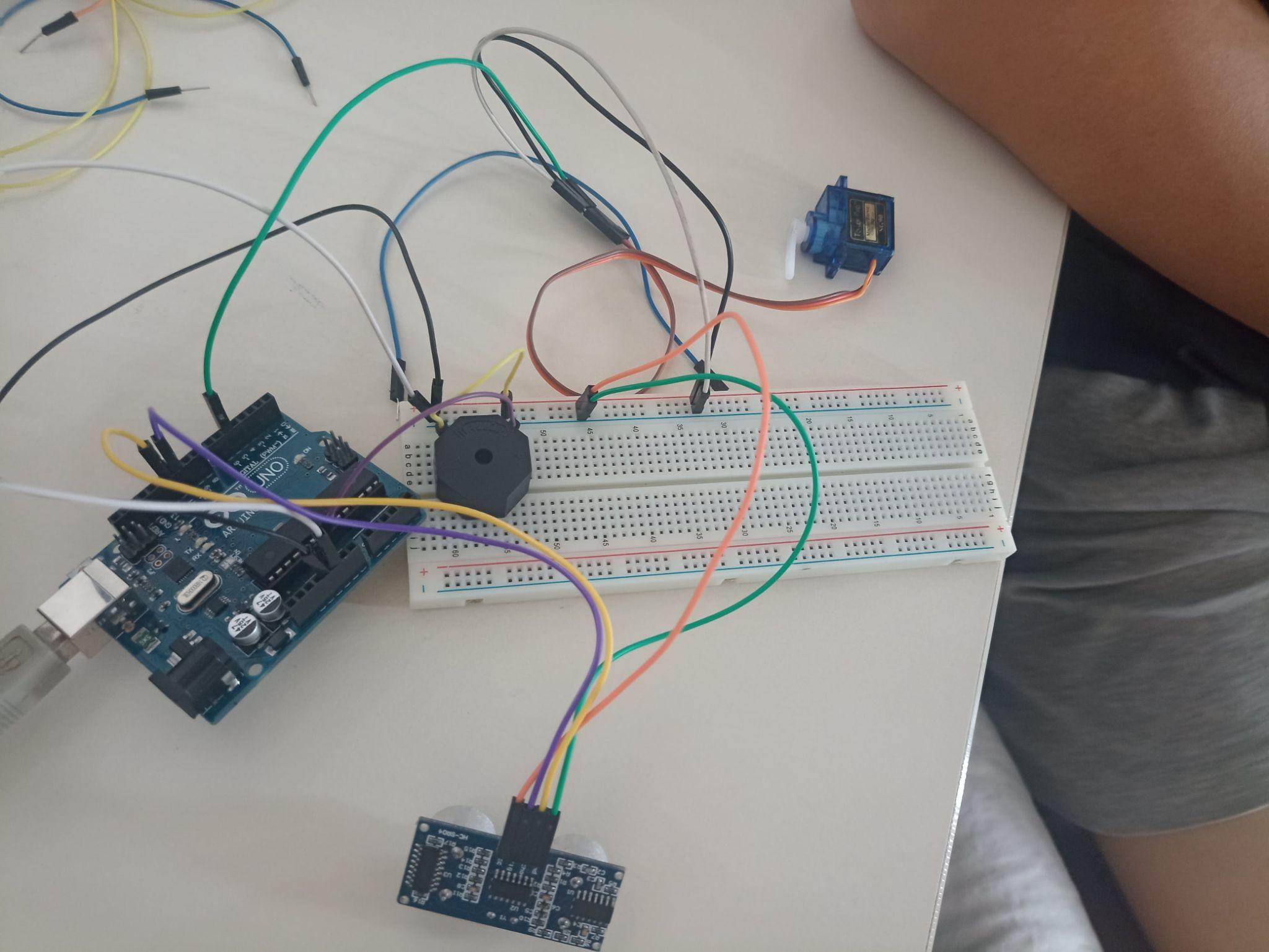
Busqué un código sobre el servomotor y encontré lo siguiente:

| for (pos = 0; pos <= 180; pos += 1) {  servo\_9.write(pos);  delay(15); // Wait for 15 millisecond(s)  }  for (pos = 180; pos >= 0; pos -= 1) {  servo\_9.write(pos);  delay(15); // Wait for 15 millisecond(s)  } } |
| --- |

Ahora voy a ver cómo adaptar este código con lo que ya empecé en tinkercad. Utilicé una IA para darme una idea de cómo sería el código. Lo que usé fue la parte del ciclo FOR, ya que intenté hacerlo por mi propia cuenta, pero no pude. Adjunto parte del FOR:

void loop() {  
 for (pos = 0; pos <= 180; pos += 1) {  
 servo\_9.write(pos);  
 delay(15);  
 float distancia = medirDistancia();  
 Serial.print("El objeto se encuentra a: ");  
 Serial.print(distancia);  
 Serial.println(" cm");  
 if (distancia > 0 && distancia < 20) { // Si la distancia es menor a 20 cm  
 digitalWrite(buzzer, HIGH);   
 } else {  
 digitalWrite(buzzer, LOW);   
 }  
 }   
 for (pos = 180; pos >= 0; pos -= 1) {  
 servo\_9.write(pos);  
 delay(15);  
 float distancia = medirDistancia();  
 Serial.print("El objeto se encuentra a: ");  
 Serial.print(distancia);  
 Serial.println(" cm");  
 if (distancia > 0 && distancia < 20) {  
 digitalWrite(buzzer, HIGH);  
 } else {  
 digitalWrite(buzzer, LOW);  
 }  
 }  
}

Lo probé y está bien. Jony y Felipe ya hicieron la parte física del proyecto. Quedó así:



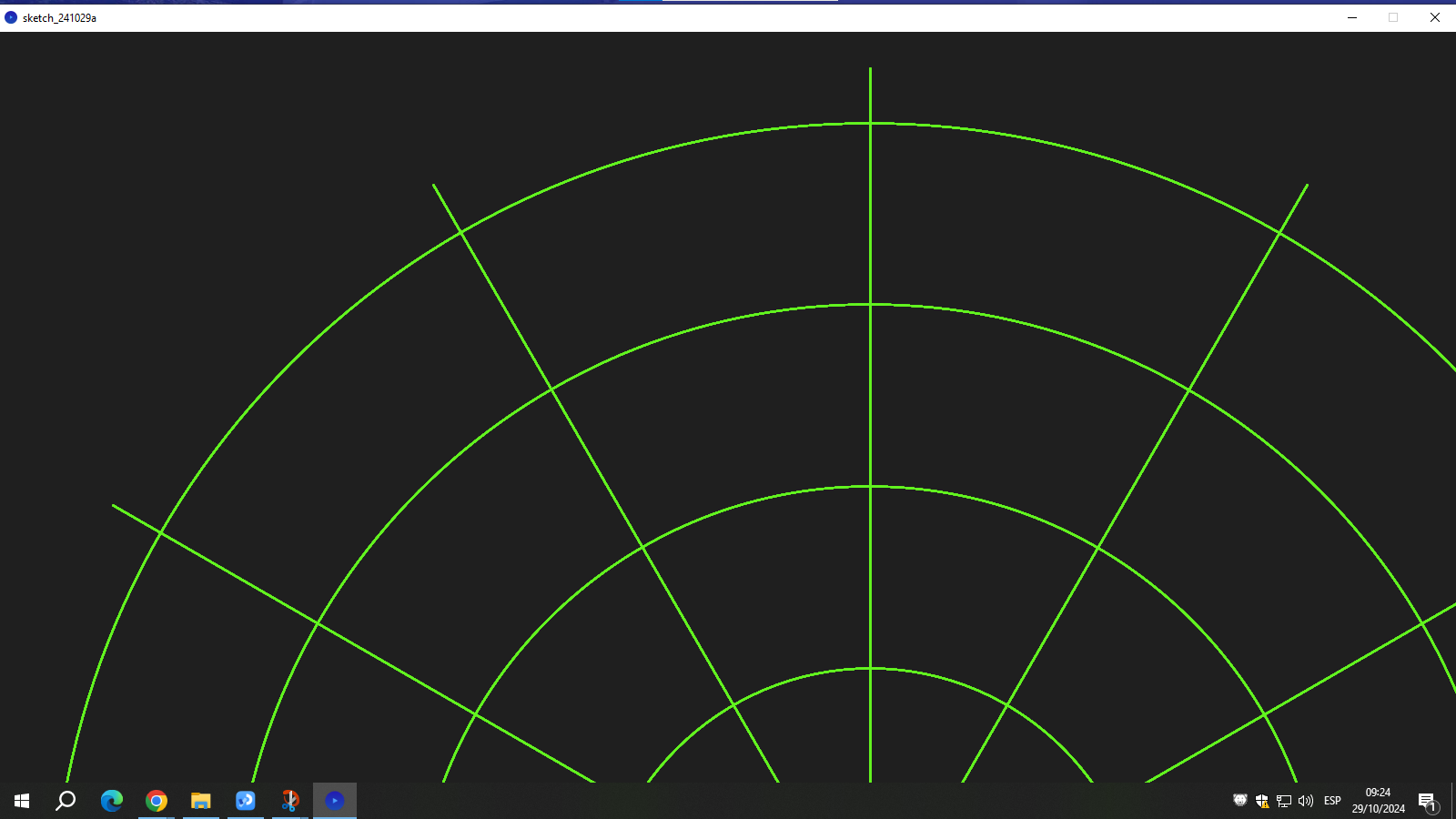
Martes 29/10/24

En el día de la fecha voy a empezar a ver el tema del processing y de la interfaz gráfica del radar. Tuve que descargar el programa desde la página inicial. Al final Marto se decidió por hacer el código. Me dijo que tuvo que modificar algo importante. Él había hecho un código semejante al mío, en donde el servo motor giraba y el sensor ultrasónico hacía un pequeño ruido cuando detectaba un objeto a cierta distancia. Consorti nos dijo que estaba mal, que lo que había que hacer era que al detectar algo (presencia, un objeto) el buzzer tenía que hacer un sonido. Ambos hicimos mal el código. Igualmente me dijo que él se encargaba, que yo me fijé en hacer lo de la interfaz gráfica.

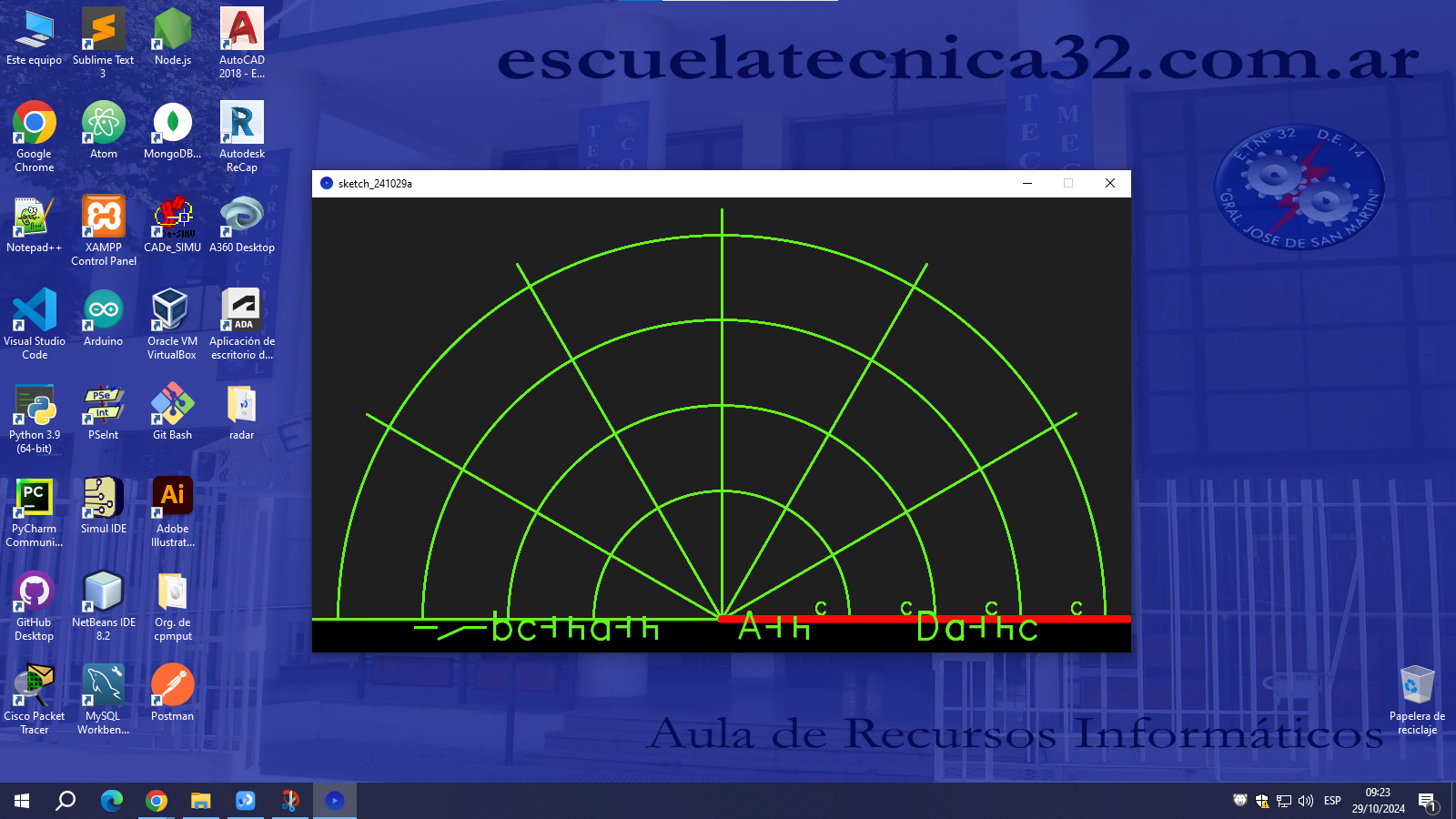
Empecé a fijarme lo de la interfaz gráfica y lo del processing. Copié el mismo código de la página con la cual me estoy ayudando. Abrí el proccesing y ejecuté el código. Me tiró el siguiente error:



Al ejecutarlo directamente no me abría la interfaz gráfica. La aplicación se me cerraba automáticamente. Le pregunté a Consorti y me dijo que era porque no tenía conectado el Arduino a la PC. Al conectarlo y ejecutarlo me funcionó la interfaz gráfica. Luego lo ejecuté y me quedó así:



Como estaba medio rara la resolución, la modifiqué con la ayuda de mi amigo Ignacio Alvarez. Fuimos probando poco a poco, hasta que la cambiamos de 1920,1080 a 900, 500. Quedó de la siguiente manera:



Al final Marto me dijo que trabajemos con mi código. Le modifiqué lo siguiente:

if (distancia > 0 && distancia < 20) { // Si la distancia es menor a 20 cm  
 tone(buzzer, 255); // Activa el buzzer  
 } else{  
 noTone(buzzer); // Apaga el buzzer  
 }  
 }

Esto nos dijo Consorti que estaba mal, ya que el buzzer tenía que hacer ruido siempre y cuando se detecte un objeto. Como la distancia máxima que logra detectar es 333.3, le modificamos lo siguiente:

if (distancia > 0 && distancia < 333.3) { // Si la distancia es menor a 20 cm  
 tone(buzzer, 255); // Activa el buzzer  
 } else{  
 noTone(buzzer); // Apaga el buzzer  
 }

Básicamente lo que modificamos fue que el buzzer se active siempre y cuando detecte algún objeto.

Luego de esto, decidimos hacer el primer test físico. Felipe se encargó de conectar todo. Son las 10:06, y necesitamos la certificación de Consorti para ver si todo está bien conectado así ya podemos empezar.

Son las 10:14, recién vino Consorti. Nos dijo que estaba todo bien conectado, así que empezamos a testear todo. Martin Romero me explicó cómo utilizar la aplicación del arduino, ya que no sabía cómo poder trabajar con ella.

Nos dijo que estaba todo bien. Empezamos a probarlo, y tuvimos varios errores: para empezar, el sensor ultrasónico no detectaba la distancia, así que llamamos a Consorti para que nos ayude. No andaba porque el sensor que estábamos usando estaba quemado (creo), así que el profe lo cambió por otro. Luego de arreglar eso, empezó a detectar distancia, pero ahora el buzzer no emitía ningún sonido. Fue una buena y una mala a la vez. El profe se dio cuenta que era porque un cable que estaba conectado al buzzer no andaba, vaya a saber Dios por qué. Al modificar todo eso, lo probamos y andaba. Con mi grupo quedamos felices, ya que lo que estuvimos haciendo funcionó de manera correcta.

A pesar de todo, Consorti nos explicó lo siguiente: si yo hago que el buzzer se active ruido cuando la distancia sea entre 0 y 333,3 cm, siempre hará ruido. Por lo tanto, nos dijo que tenemos que ver nosotros cuál es el rango para que se prenda el buzzer. Hubo una mini confusión, pero al final el buzzer se activa con el rango que nosotros queramos, no cuando detecta movimiento. Consorti me explicó que el sensor solamente detecta distancia, y no movimiento. Me confundí con ese concepto. Mi idea era que el buzzer no se active cuando no haya movimiento, pero no se puede eso, ya que el sensor detecta MOVIMIENTO y no distancia. En fin. Confusiones personales.

Con Marto tenemos la idea de que en cierto rango de distancia, se prenda el buzzer en determinado pitido y con cierto delay de por medio. Es una idea que quedará pendiente para la clase que viene.

También quisimos probar la interfaz gráfica, pero me tiraba un error. Ya había terminado la hora, así que decidimos verlo bien la clase que viene.

Hoy fue un día largo lleno de trabajo. Nos queda ver la interfaz gráfica, el martes que viene o durante el resto de la semana lo veo bien.

5/11

En el día de la fecha voy a ver lo de la interfaz gráfica, y también voy a cambiar lo de los pitidos del buzzer. Voy a empezar modificando lo del buzzer. Con Marto nos pusimos a debatir sobre los pitidos, para ver en qué rango hace ruido y en qué rango no. Básicamente en determinado rango va a hacer una mini pausa con un delay (dependiendo de la distancia dicho delay durará más o menos). Antes de probar todo en formato físico, empezamos a probarlo en el tinkercad. Consorti nos dijo que usemos la función map, y no sabemos cómo hacerlo. Lo llamamos para que nos ayude. Vino y nos modificó el código. Quedó de la siguiente manera:

ANTES:

if (distancia < 25){  
 tone(buzzer, 1000);

int pitido = map( distancia , 5 , 200 ,1 , 1000);  
 delay ( pitido );  
 Serial.println(pitido);  
 noTone(buzzer);  
 }

DESPUÉS:

if (distancia < 25)  
 {

tone(buzzer, 1000);  
 int pitido = map( distancia , 5 , 200 ,1 , 1000);  
 delay ( pitido );  
 Serial.println(pitido);  
 noTone(buzzer);

Básicamente lo que queríamos hacer es que dependiendo de la distancia, haya un pitido de mayor frecuencia o no. Luego de esto tuvimos un problema porque el pitido del buzzer no cambiaba, hasta que encontramos la vuelta. Al volver del recreo vamos a probarlo en el arduino físico, ya que el simulador tenía problemas.

Volvimos al recreo y conectamos todo. Esperamos la confirmación de Consorti para ver si todo estaba bien. Luego de 1 hora esperándolo, nos dijo que sí. Al probar todo tuvimos el siguiente problema: el buzzer no funcionaba, no hacía ruido y tampoco devolvía el valor del map. Vamos a consultarlo con él lo antes posible.

Luego de millones de pruebas, pudimos arreglar lo del buzzer. Lo que todavía no funciona es que el buzzer no hace los pitidos que queríamos hacer que tenga. La clase que viene lo vemos bien con la interfaz gráfica y todo. Lo bueno es que aunque sea empezó a hacer ruido el buzzer, cosa que antes no.

Hoy no hice casi nada, me quedó pendiente lo de la interfaz gráfica. Tengo que probar todo junto, tanto los pitidos como la interfaz gráfica a la vez. Una vez esté todo bien buscamos un diseño gráfico que quede bien fachero, así no queda todo conectado así nomás.

12/11

En el día de la fecha vamos a seguir el proceso de los pitidos de los buzzers. Felipe se está encargando de la conexión, cuando termine y vamos a seguir con los testeos. Además de ello voy a fijarme el tema del processing, y como vincularlo con el RADAR.

Felipe terminó de conectar todo y no funcionó. Llamamos a Consorti para averiguar por qué sucedió esto. Al final el error venía por los cables.

Luego de miles de pruebas, el buzzer logró hacer los pitidos que tanto queríamos hacer. El código quedó así:

if( dist >= 35 && dist <= 200)  
 {  
 int pitido = map( dist , 35 , 350 ,50 , 1000);  
 tone(buzzer,500, pitido);  
 delay(pitido);  
 noTone(buzzer);  
 }else if(dist<35) {  
 tone(buzzer,500);  
 }else{  
 noTone( buzzer );  
 }

Ahora vamos a ver qué onda con el processing y cómo hacer que se vincule con el arduino.

Acá voy hacer un breve resumen sobre las estructuras básicas del código y elementos específicos del processing:

**Importación de Librerías**:

* processing.serial.\* para la comunicación serial.
* java.awt.event.KeyEvent para eventos de teclado.
* java.io.IOException para manejar errores de entrada/salida.

**Variables**:

* Se definen varias variables para almacenar datos del ángulo y la distancia, además de los objetos Serial y PFont.

**Configuración Inicial (**setup()**)**:

* Establece el tamaño de la ventana de visualización y la comunicación serial a través de COM3 a una velocidad de 9600 baudios.
* Carga la fuente para el texto que se mostrará en pantalla.

**Bucle Principal (**draw()**)**:

* Dibuja y actualiza continuamente el radar y los objetos en la pantalla.
* Incluye funciones para dibujar el radar, las líneas, los objetos detectados, y el texto en pantalla.

**Eventos Seriales (**serialEvent()**)**:

* Lee datos del puerto serial cuando llegan (señalados por un punto .), los procesa y extrae el ángulo y la distancia.

**Funciones de Dibujo**:

* drawRadar(): Dibuja la estructura del radar con arcos y líneas.
* drawObject(): Dibuja los objetos detectados por el radar si están dentro de un rango especificado.
* drawLine(): Dibuja la línea del escáner del radar según el ángulo recibido.
* drawText(): Muestra en pantalla los datos de ángulo, distancia y si el objeto está dentro del rango.

Me la pasé toda la clase viendo cómo vincular el processing y el arduino. No encontré la forma para poder hacerlo, la clase que viene me voy a fijar bien qué hacer. No queda mucho tiempo para terminar el proyecto. Me estoy empezando a desesperar. No quiero llevarme la materia sinceramente.

19/11

En el día de la fecha voy a continuar con el tema del processing y el arduino. Además de ello tengo que ver el tema del servomotor, y cómo puedo hacer para que vaya girando y detecte movimiento. Aproveché el tiempo para buscar información sobre el uso del mismo, y lo que encontré fue lo siguiente:

Un servomotor es un dispositivo que convierte señales eléctricas en movimiento controlado y preciso. En el contexto de Arduino, el servomotor se controla enviando pulsos eléctricos a través de un pin digital. Este es el proceso básico:

### Componentes del servomotor:

1. Motor de corriente continua (DC): Proporciona el movimiento.
2. Circuito de control: Traduce las señales eléctricas en órdenes para el motor.
3. Potenciómetro: Mide la posición actual del eje.
4. Eje o brazo**:** Realiza el movimiento rotativo.

### Funcionamiento básico:

1. Señal PWM (Modulación por ancho de pulso):
   * El servomotor utiliza pulsos PWM para determinar la posición del eje.
   * Una señal típica tiene un período de 20 ms (50 Hz).
   * El ancho del pulso varía entre 1 ms y 2 ms:
     + 1 ms → Ángulo mínimo (0°).
     + 1.5 ms → Ángulo medio (90°).
     + 2 ms → Ángulo máximo (180°).
2. Retroalimentación:
   * El potenciómetro dentro del servomotor compara la posición actual con la deseada y ajusta el motor para alcanzar el ángulo especificado.

### Conexión al Arduino**:**

* Cable rojo: Alimentación (5V o 6V).
* Cable negro/marrón: Tierra (GND).
* Cable amarillo/naranja: Señal PWM (conectado a un pin digital del Arduino).

### Programación en Arduino**:**

* Se utiliza la librería Servo.h, que simplifica el control del servomotor.

Un ejemplo básico es el siguiente:

| #include <Servo.h>  Servo servoMotor; // Crear un objeto Servo  void setup() {  servoMotor.attach(9); // Conectar el servomotor al pin digital 9 }  void loop() {  servoMotor.write(0); // Mover a 0 grados  delay(1000); // Esperar 1 segundo  servoMotor.write(180); // Mover a 180 grados  delay(1000); // Esperar 1 segundo } |
| --- |
|  |

Luego me puse a ver vídeos sobre cómo vincular el procesing con el arduino, y no pude. Le pregunté a Consorti y me explicó que hay que laburar en el proccesing con el “Serial.Write”, y tengo que mandarle información del Arduino al Procesing con dicho código.

Pasamos de esta interfaz gráfica:



A esta:



Ahora sí pudimos hacer que se vincule el Arduino con el processing. Con Marto tuvimos la idea en colocar en la parte inferior del processing un mensaje en el cual nos diga del ángulo en el cual está el servomotor, a qué distancia y si detecta movimiento, o no. Me queda pendiente realizar eso.

25/11

Hoy me puse a fijarme en diversos vídeos cómo mandar la información de Arduino al Processing. Lo que nos queda por hacer es poder vincular el arduino con el processing, la parte del servomotor y hacer un diseño que quede bien fachero. Nada más que eso. Sinceramente, creo que llegamos. Se entrega la semana que viene, confío en que terminemos.

No encontré la forma de hacerlo. Le pedí ayuda a Copilot para ver cómo hacerlo.

26/11

Últimos días del proyecto. Nos faltan varias cosas por hacer, el processing que me tiene harto, el servomotor y el diseño. Con Marto y Felipe decidimos venir el jueves a la tarde para poder continuar el proyecto y al menos llegar a terminarlo. Hoy vamos a hacer las pruebas del processing, y si llegamos vamos a ver lo del servomotor. El diseño quedará para otro día.

Logré poder hacer la vinculación entre el processing y el Arduino. Me queda pendiente hacer lo de los pitidos, ya que muestra que hay un movimiento en la interfaz gráfica, pero no hay pitidos. Quedó de la siguiente manera:



Ahora estoy viendo el tema de los pitidos. Tengo que hacer unos pequeños cambios en el código de processing, y en teoría funciona.

28/11

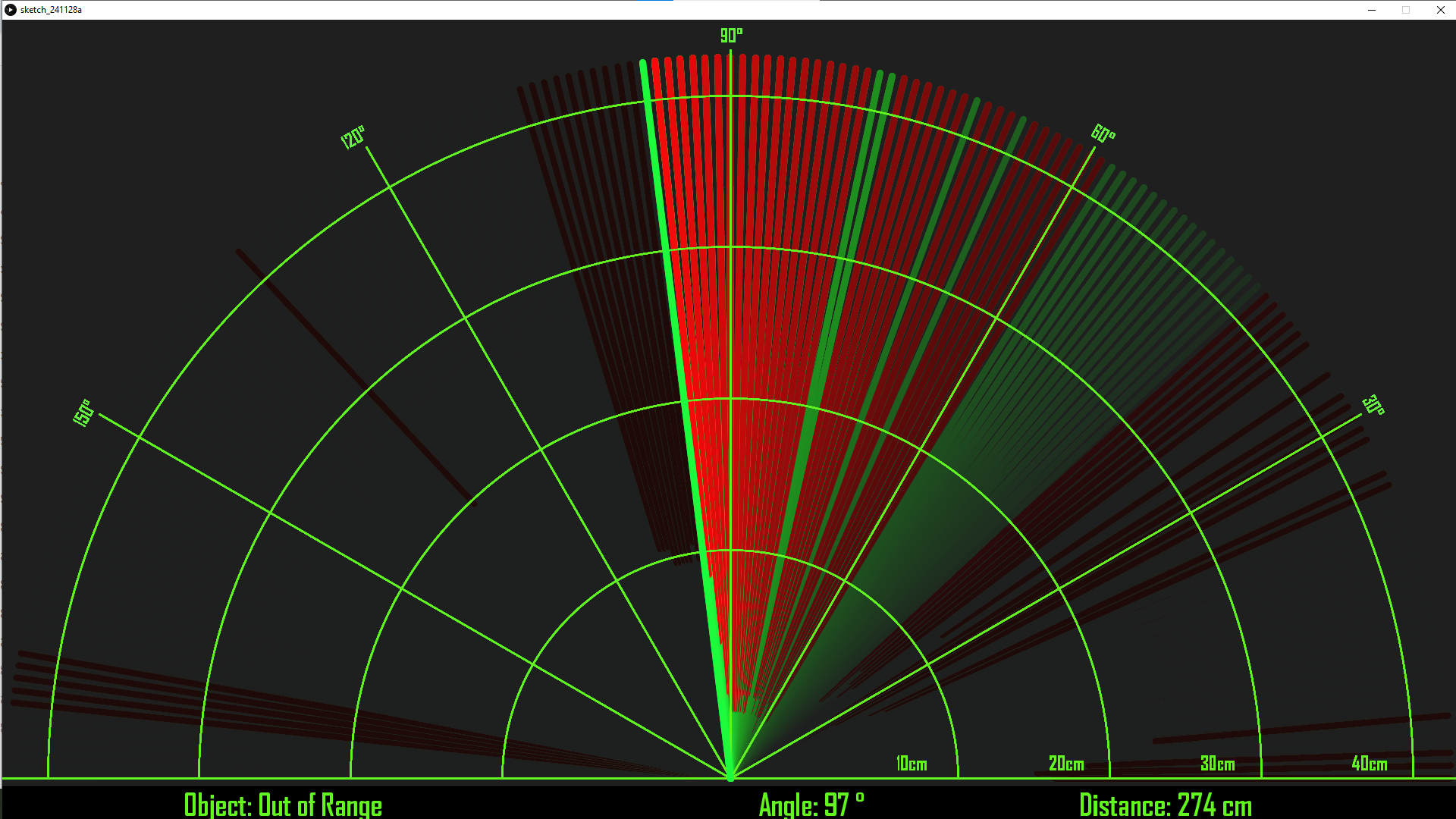
En el día de la fecha, venimos a la tarde para continuar con el proyecto. Nos queda pendiente finalizar lo del processing, empezar lo del servomotor y hacer el diseño. Me parece que no llegamos a hacer el diseño, vamos a priorizar el processing y el servomotor. Espero que terminemos. Vamos a comenzar viendo lo del proccesing.

Comenzamos la clase realizando la conexión de los componentes. Ya conectamos todo, y el sensor creo que anda mal. Me estoy empezando a estresar y recién empezó la clase. No sé qué hacer.

Tuvimos varios problemas con los cables, las conexiones y el processing. Teníamos el problema de que cuando se realizaba el giro de 0° a 180° grados, al intentar hacer la otra vuelta no podíamos, debido a un error en el código. Le pedimos ayuda a Consorti,ya que no sabíamos qué hacer. Vino, y nos modificó el código. Nosotros tuvimos varios errores.

Al final el código quedó así:

| }  for(int i=180;i>0;i--){  myServo.write(i);  delay(30);  distance = calculateDistance();  Serial.print(i);  Serial.print(",");   Serial.print(distance);  Serial.print(".");   if(distance<20){  tone(8, 1500, 100);  }else if(distance>20 && distance<40){  tone(8, 1000, 100);  noTone(buzzer);  }else if(distance > 40 && distance < 100){  tone(8, 500, 100);  noTone(buzzer);  }else if(distance>100 && distance<200){  tone(8, 250, 100);  noTone(buzzer);  }else{  noTone(8);  } |
| --- |
|  |
|  |
| Ahora vamos a ver cómo queda esto en el processing. Quedó de la siguiente manera: |
|  |



Me gustó como quedó. Le añadimos los mensajes dichos anteriormente, y ahora sí detecta la distancia.

Ahora voy a explicar el código de Arduino, explicando void por void:

| // Importa la librería Servo #include<Servo.h>. // Define los pines del sensor ultrasónico Trig y Echo const int trigPin = 10; const int echoPin = 11;  const int buzzer = 8;  // Variables de duración y distancia long duration; int distance; Servo myServo; // Crea el objeto myServo |
| --- |

Arrancaré explicando lo que está afuera de las funciones:

* Librería Servo.h:

Esta librería permite controlar un motor servo mediante diversos comandos simples, como asignarle un ángulo en específico para que pueda realizar su función de la manera adecuada.

* Constantes trigPin y echoPin:

const int trigPin, echoPin = pin 10 y pin 11;

pin 10: define el pin que envía la señal ultrasónica del sensor

pin 11: define el que recibe la señal reflejada (echo)

* Constante buzzer:

const int buzzer = 8;

pin 8: es el pin conectado al zumbador, es decir, el buzzer, que emite sonidos en función de la distancia medida por el sensor ultrasónico

long duration: almacena el tiempo (en microsegundos) que tarda el pulso ultrasónico en regresar el sensor.

int distance: guarda la distancia calculada en centímetros a partir de la variable explicada anteriormente, es decir, la variable “duration”.

* Objeto myServo:

Servo myServo: crea un objeto para controlar el servomotor, permitiendo asignar diferentes posiciones específicas (en un ángulo del 0° al 180°, tal y como es el proyecto del radar)

Ahora voy a explicar la función void setup():

| void setup() { pinMode(trigPin, OUTPUT); // Setea el pin trigPin como un Output pinMode(echoPin, INPUT); // Setea el pin echoPin como un Input Serial.begin(9600); myServo.attach(12); // Define en que pin esta conectado el Servo motor pinMode(buzzer, OUTPUT); //Setea el pin 8 del buzzer como output } |
| --- |

Antes de empezar, voy a explicar qué es la función del void setup:

¿Qué es void setup?

Es la sección del código de Arduino que se ejecuta una sola vez al iniciar el , ya sea al encender la placa o al presionar el botón de reinicio. Su función es configurar el hardware y prepararlo para que funcione de la manera adecuada en el programa.

Explicación del contenido del void setup:

El sensor tiene dos pines claves:

* Trig (disparador): permite enviar pulsos ultrasónicos provenientes del sensor de distancia.
* Echo (eco): recibe la señal reflejada

En el void setup, el pin del Trig se define como una salida, es decir, como un OUTPUT, ya que enviará señales, mientras que el pin del Echo se define como una entrada, ya que recibirá las señales de regreso, ambos pines provenientes del sensor ultrasónico.

Inicialización de la comunicación serial:

Inicializa la comunicación serial a 9600 baudios, permitiendo enviar datos al monitor serial, es decir, la consola del Arduino. Sin este código no se podrán realizar los diversos testeos del proyecto necesarios para llevar el mismo adelante.

En este caso, el serial se utiliza para monitorear la distancia y el ángulo del radar en tiempo real.

Configuración del servomotor:

El servomotor es el encargado de mover el radar en diferentes ángulos (de 0° a 180° y viceversa) .

Para que el Arduino pueda controlarlo, es necesario asociar el servomotor a uno de sus pines. En este caso, se conecta al pin 12. La función “myServo.attach(12)” indica al Arduino que ese pin controlará al servo.

Preparación del buzzer:

El buzzer es un dispositivo que emite sonidos, y en este proyecto lo hace con diferentes frecuencias disponibles dependiendo de la distancia detectada por el sensor ultrasónico. Para que el mismo funcione, se configura como salida, es decir, como OUTPUT, ya que el Arduino le enviará señales para generar los diversos tonos.

¿Por qué es importante el void setup?

* Inicialización de hardware: sin esta configuración, los componentes detectados no funcionarán correctamente.
* Preparación para el bucle principal(loop): establece las condiciones iniciales para que el programa ejecute sus tareas respectivas de una manera efectiva, sin errores de por medio.
* Evita problemas: configurar pines de entrada y salida garantiza que no haya problemas eléctricos en los pines de la protoboard.

Resumen del void setup en este proyecto:

* Prepara el sensor ultrasónico para medir distancias
* Establece la comunicación con la computadora para monitorear datos
* Configura el servomotor para que el mismo se pueda mover en diversos ángulos.
* Deja listo al buzzer para emitir sonidos según las distancias detectadas.

Ahora voy a explicar el void loop:

¿Qué es void loop?

Es la sección del código de Arduino que se ejecuta de manera continua una vez que el programa ha terminado de correr el void setup. Es el “cerebro” del programa, y contiene las principales instrucciones que el Arduino repetirá indefinidamente.

Explicación del contenido del void loop:

1- Movimiento del servomotor:

* El radar utiliza un servomotor para moverse en un rango de 0° a 180° (ida) y 180° a 0° (vuelta). Este movimiento permite al sensor ultrasónico medir distancias en diferentes direcciones.
* En el primer bucle, el servo se mueve en pasos de 1 en 1, empezando por 0° hasta 180°. En el segundo, regresa en pasos de 1 en 1 también, empezando por 180° hasta 0°.
* Cada movimiento incluye una mini pausa de 30 milisegundos para darle un tiempito al servomotor a ajustarse físicamente al nuevo ángulo correspondiente.

2- Movimiento del servomotor:

* En cada posición angular del servomotor, el programa utiliza el sensor ultrasónico para calcular la distancia hasta el objeto más cercano.
* Esta distancia se calcula enviando un pulso ultrasónico desde el sensor, que rebota en el objeto y regresa. El tiempo que tarda este eco se convierte en centímetros utilizando la siguiente fórmula:

Distancia = (Tiempo x 0.034 ) / 2

* Esta función es llamada cada vez que el servomotor se mueve a un nuevo ángulo

3- Envío de datos al monitor serial:

* El programa envía dos datos claves al monitor serial:
* El ángulo actual del servomotor
* La distancia detectada en ese ángulo
* Estos datos se formatean como “ángulo, distancia ” para poder facilitar su lectura y análisis.

4- Comportamiento del buzzer según la distancia:

* Dependiendo de la distancia medida, el buzzer emite tonos en diferentes frecuencias:
* Menos de 20 cm : un tono agudo
* Entre 20 cm y 40 cm: un tono medio
* Entre 40cm y 200cm: un tono bajo
* Más de 200 cm: el buzzer se apaga

Detalles clave del void loop:

1. El ciclo del radar:

* El servomotor permite permite un barrido de 180° en ida y vuelta, cubriendo un campo visual campo
* Esto simula un radar que puede detectar objetos en un rango grande de posiciones.

1. Cálculo y respuesta y tiempo real:

* En cada posición, el sensor mide la distancia y el programa responde inmediatamente con un sonido en específico, indicando la proximidad del objeto.

1. Visualización de datos:

* Los datos enviados al monitor serial son útiles para entender el comportamiento del sistema, ya que muestran tanto el ángulo como la distancia detectada.

¿Por qué es importante el void loop en este proyecto?

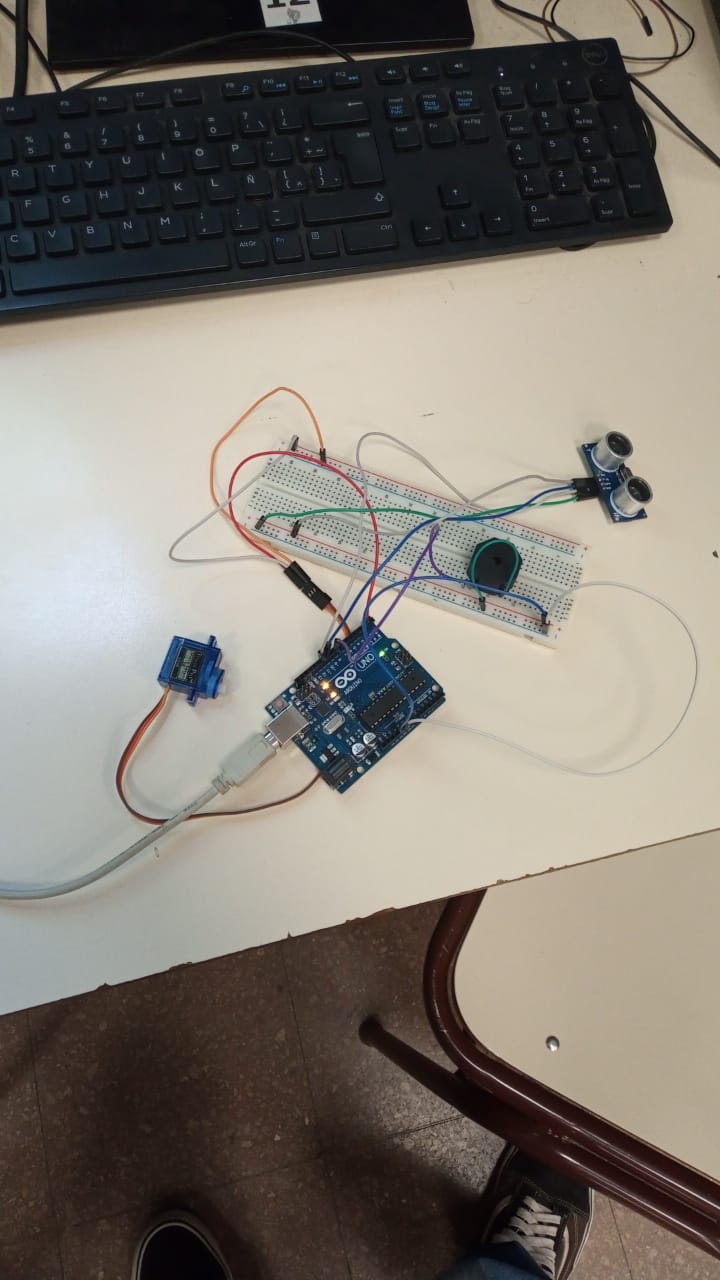
* Automatización: el void loop permite que el radar funcione continuamente, detectando y actualizando distancias en tiempo real.
* Interactividad: el buzzer proporciona retroalimentación inmediata basada en las mediciones, lo que es útil para aplicaciones prácticas.
* Monitoreo: La transmisión de datos al monitor serial permite al usuario observar cómo se comporta el sistema y ajustar parámetros si es necesario.

Resumen del void loop en este proyecto:

1. El servomotor se mueve en ambos sentidos para cubrir un rango de 180°.
2. El sensor ultrasónico mide distancias continuamente.
3. Los datos se envían al monitor serial para visualización.
4. El buzzer emite sonidos según la distancia detectada, ayudando a identificar objetos cercanos.

Bueno, ya terminé de explicar el código. Nos quedó pendiente el tema del servomotor, pero no pudimos usarlo ya que todos los que hay disponibles no funcionan. Los únicos que sí andan los tiene el grupo del jardín inteligente. Nos dijo Consorti que no pasaba nada, ya que esto no dependía de nosotros, sino que no había componentes necesarios para poder mejorar esto.

El hardware quedó así:



También queríamos realizar hoy el diseño físico, para que no presentemos todo así nomás, pero no pudimos llegar. Consorti nos dijo que el lunes él iba a estar disponible en el colegio, así que vendremos con Marto para hacer el diseño del proyecto. Me dijo que tenía una idea de una torre de control, el lunes lo veremos bien. Necesitaré cartón o algo para poder llevar la idea a cabo. Queda poco tiempo. Ya tenemos la mayoría del proyecto hecho, nos queda solamente la parte del diseño físico y ya estaría. Espero que todo el esfuerzo valga la pena.

2/12

En el día de la fecha vine a realizar a la mañana el diseño físico. Traje 3 cajas de cartón, me parece que traje de más. Empezamos a conectar todos los componentes. De nuevo tuvimos la problemática de los cables. Estuvimos como 40 minutos viendo qué cable andaba mal, hasta que pudimos encontrar la solución. Luego de esto, Marto le preguntó por las dudas a Consorti si hacía falta entregar un diseño físico, y el profe dijo que no, que con el diseño 3d estaba bien. Básicamente, traje los cartones para nada, pero no pasa nada.

Marto se puso a hacer el diseño 3D. Con Felipe nos pusimos a seguir realizando pruebas y a disfrutar de nuestro proyecto.

Marto de paso le preguntó de nuevo, por las dudas, a Consorti de si hacía falta el servomotor, y ahora nos dijo que sí. Así que me puse manos a la obra para ver qué servomotor andaba, y cuál no. Después de varias pruebas, ningún servomotor andaba. Le comenté esto a Consorti, vino y me dijo que para empezar cada servo estaba mal conectado. Google me mintió JAJSJAS. Ahora Consorti se puso a probar uno por uno. Como vió que ninguno andaba, nos explicó que quizá era porque el Arduino mandaba muy poca señal, y capaz no podía hacer que el servo tenga la energía necesaria para que pueda arrancar, así que sacó una batería suya, que se ve que es personal, y probó un servo. De increíble manera, cada servomotor volvió a andar, gracias a la batería. La verdad me puse feliz, ya que por mi parte quería que el servomotor esté incluido en el proyecto, ya que era me hacía ilusión ver como el radar giraba de forma “dependiente”, todo gracias al servo. Luego de esto, Consorti nos dijo que mañana, el día de la presentación, había 2 formas de presentar el proyecto: una era que un compañero se quede dando vueltas en una silla mientras mostraba el proyecto, y otra era que consigamos un material que logre que el servo se quede “pegado” al sensor, para así poder realizar el giro de 0° a 180° sin ningún inconveniente. Por decisión de todos, nos decidimos por la segunda opción. Al parecer Consorti notó que estábamos medio perdidos en conseguir un material, nos brindó un mini material que podía hacer que el servo gire junto al sensor. Adjuntaría una imagen sobre cómo quedó todo, pero me colgué. Cuando llegue a casa, tengo que seguir la presentación que hoy empezó a hacer Marto. Veré en qué puedo ayudar.

**CONCLUSIÓN**

Bueno, en este espació daré mi sincera opinión sobre todo. Me gustó mucho hacer este proyecto, nunca había trabajado ni con processing, ni con Arduino físico, y tampoco con protoboard. Es una experiencia la cual me gustó poder hacer. Fue un estrés constante la conexión, había veces donde los cables funcionaban, otras no, a veces el sensor le pintaba funcionar, otras no, y así con el buzzer, y también con el servo. Fue un trabajo el cual me costó mucho esfuerzo, y más que nada mucha paz mental. Casi siempre me surgía el miedo de no poder llegar a terminarlo por diversas razones, pero al momento que estoy escribiendo esto, a las 22:53 del 1/12, creo que el proyecto está aprobado. Espero no mufarla jajaj. Fue todo bastante difícil, pero el esfuerzo valió la pena. Me gustó como quedó el proyecto. Una lástima que no llegamos a hacer el diseño 3D, o que tampoco pudimos hacer un diseño físico. No me gustó entregar el trabajo así nomás, con los cables, la protoboard y el arduino así nomás, pero bueno, por el tiempo no llegamos. Sinceramente creo que si, nos poníamos las pilas de verdad, terminábamos antes y hasta llegábamos a hacer el diseño ese el cual siempre quise. La idea de Marto de la torre de control me pareció muy copada, una lástima que no pudimos ejecutarla. A mi parecer, lo más difícil fue hacer la carpeta de campo, ya que a mí no suele nacer explicar mucho y mucho sobre algo en específico. Ayer tenía 24 páginas, y hoy llegué a las 38, tuve que ponerme las pilas por la regla de las 50 páginas como mínimo. Me costó bastante poder hacer más páginas. No llegué a las 50, pero me quedo conforme que llegué a las 38, aún sabiendo que me cuesta explicar bastante. Igualmente, a mi no me nacía documentar todo lo que iba haciendo cada clase en la carpeta de campo, hubieron ciertas problemáticas que ni siquiera pude anotar, ya que no estoy acostumbrado a escribir y escribir. Quizá llegaba a 45 páginas como mucho, la verdad no lo sé. En el proyecto hice bastante, seguro podía llegar a las 50, pero bueno, es lo que toca.

En un futuro me gustaría hacer un proyecto un poco más complejo que este, ya que me pareció interesantísimo esto sobre Arduino y demás. La electrónica me gusta bastante, no sé porque no estoy en una escuela donde den dicha especialidad, pero bueno, aquí estamos, en computación.

Bueno, me voy a dejar de contar cosas sobre mi vida que no aportan nada a la carpeta de campo. Ahora voy a agradecer a diversos compañeros, empezando por:

* Mi amigo Adrian Mamani. Me ayudó bastante con el tema del código, tanto del processing como del Arduino, también en la parte física. Las veces que le pedí ayuda siempre estuvo dispuesto a colaborar, así que le agradezco por todo lo que me ayudó.
* Al profesor Consorti. Nos ayudó un montón en todo, en la parte física, en el código. Tuvo que bancar mil veces que nosotros siempre tengamos los mismos problemas de conexión, así que le pido perdón por ser tan molesto siempre con el mismo problema, y agradecerle por su ayuda,
* A mí mismo, por más que se me pasó por la cabeza dejar el proyecto así nomás sin terminar todo. Pude aguantar bastante la verdad, el hecho que se repita siempre el mismo problema con el hardware me daban ganas de romper todo.
* A Felipe Kuo Lee, siempre que le pedía que me realice la conexión o que había algún problema con ello él estaba ahí.

Martin Schilacci, Marto. Él también me ayudó a no pegarme un tiro antes, mil gracias. Además de ello, colaboró en la parte física, en el funcionamiento del processing y en el Arduino.

Y creo que no tengo que agradecerle a alguien más. Mil disculpas si esta carpeta de campo llega a alguien que me ayudó bastante y me olvidé de su presencia. Ya doy por finalizada la carpeta de campo. Son las 23:14, a esta hora la terminé.