



Universidad Nacional Autónoma de México

Colegio de Ciencias y Humanidades Plantel Naucalpan

Mecatrónica Basica

Implementación y Diseño de Sistemas Automatizados

Radar Detector de Objetos

Hernández Espinoza Gabriel Emanuel

01/06/2023

Índice

Introducción.....	1
Justificación.....	2
¿Qué es la Ingeniería Mecatrónica y para qué sirve?.....	3
¿Qué es Arduino y para qué sirve?.....	5
¿Qué relación hay entre Arduino y Mecatrónica?.....	6
Radar detector de Objetos.....	7
Conclusiones.....	20
Referencias	22

Introducción

Actualmente vivimos en un mundo donde estamos en constante contacto con la tecnología, un mundo donde conforme pasan los días se busca que dicha tecnología crezca y evolucione, para que posteriormente pueda ser aplicada a diferentes aspectos de nuestra vida cotidiana, como por ejemplo, a la hora de ir al mercado, las puertas que nos reciben se abren solas, al registrar el código de barras de un producto en el cajero, todos los datos de este aparece en pantalla. En la elaboración de automóviles, quienes se encargan de hacerlos no son personas sino máquinas, donde el ser humano simplemente se encarga de verificar y supervisar que dicho proceso sea correcto y este en orden, son unos ejemplos donde dichos procesos están automatizados, donde se busca reducir costos, obra humana, así como sustituir al ser humano en actividades peligrosas para el mismo, donde se puedan perder vidas y se puedan mejorar y acelerar dichos procesos.

Todo esto es gracias a diversas ciencias como la electrónica, programación, mecánica entre otras, dónde la Ingeniería Mecatrónica es una de las tantas ingenierías que abarcan el estudio de múltiples ramas científicas.

¿Pero te has preguntado cómo es que todo lo anteriormente dicho es posible?, ¿Cómo es que enlazamos el mundo físico con el mundo digital?, ¿Cómo es que las máquinas saben que hacer o cómo actuar bajo ciertas premisas?, muchas veces las personas no se preguntan cómo es que funcionan las cosas porque como dichas cosas forman parte cotidiana de su vida, no se dan cuenta que están en contacto con un mundo lleno de maravillas, pero cuánto estás ciencias y campos de estudios recién surgían, era todo un misterio que muy pocas personas entendían o sabían explicar.

Justificación

La razón del presente trabajo es explicar cómo sirve y aplicamos la Ingeniería Mecatrónica en diversos aspectos de nuestras vidas. En este caso la elaboración de un radar por medio de un sensor.

Para empezar, un radar se define como un instrumento de detección de objetos que es empleado en aeronáutica, navegación, etc , que sirve para indicar la presencia de un objeto y determinar la distancia a la que se halla. Tienen múltiples aplicaciones, por ejemplo, los carros autónomos usan sensores para detectar todo lo que les rodea en el entorno en el que se encuentran. Otro ejemplo hablando de esto en la industria automotriz son los carros que tienen un sistema de autofrenado o frenado de emergencia al detectar personas, ya que al ir manejando a una cierta velocidad este mismo se detiene cuando detecta una figura humana enfrente de este gracias a sensores que trabajan en conjunto así como con inteligencia artificial.

Otro ejemplo serían los sensores de fuego que nos avisan de posibles incendios con la detección de temperatura, las puertas del supermercado que se abren solas, también usan sensores y con un poco de programación, que estás se abran solas al detectarnos.

Barcos, aviones y submarinos también hacen uso de sensores y radares. ¿Pero qué diferencia hay entre un sensor y un radar?, porque suenan como si fueran lo mismo, pero no es así ya que un radar utiliza ondas de radio para detectar objetos distantes y medir la distancia a la que se encuentran, mientras que los sensores responden a diferentes cambios físicos o químicos en su entorno y generan señales medibles relacionadas con estas variables. Ambos dispositivos tienen aplicaciones diversas y desempeñan un papel fundamental en diversas industrias y campos de la tecnología.

¿Qué es la Ingeniería Mecatrónica y para qué sirve?

Mecatrónica o ingeniería mecatrónica es una disciplina que combina elementos de la ingeniería mecánica, electrónica entre otras múltiples ramas de la ingeniería así como la programación. Su objetivo es diseñar, desarrollar y operar sistemas inteligentes y automatizados que integren componentes mecánicos y electrónicos, con la finalidad de crear productos y procesos más eficientes y avanzados reduciendo costos, así como el riesgo que hay para un ser humano en ciertas actividades.

Los ingenieros mecatrónicos trabajan en el diseño y fabricación de una amplia variedad de productos y sistemas, tales como robots industriales, dispositivos médicos, sistemas de control y automatización en la industria, vehículos autónomos, sistemas de producción automatizados, entre otros.

La ingeniería mecatrónica es un campo multidisciplinario que abarca una amplia gama de aplicaciones y sectores, incluyendo la industria manufacturera, la robótica, la automatización, la electrónica, la industria automotriz, la aeroespacial, la medicina y la investigación como se dijo anteriormente.

¿Pero cuáles son las principales áreas de trabajo o aplicación de la ingeniería mecatrónica? Las principales áreas de trabajo y aplicación de la ingeniería mecatrónica incluyen:

1. Robótica: En el cual se diseñan y programan robots para diversas tareas industriales y de servicio, como ensamblaje, soldadura, exploración, limpieza, entre otros.
2. Automatización y control: Donde se busca la creación de sistemas automatizados que optimicen procesos y mejoren la productividad en la industria.
3. Sistemas electrónicos: Donde se diseñan y desarrollan sistemas electrónicos integrados en productos y maquinarias.
4. Instrumentación y sistemas de medición: Creación de dispositivos de medición y control para diversos propósitos.

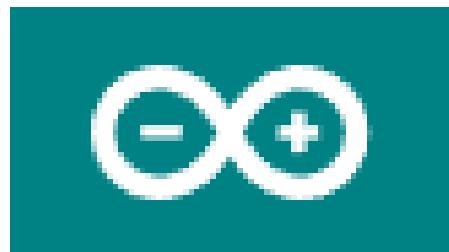
En resumen, la ingeniería mecatrónica es una disciplina que combina conocimientos y habilidades de diferentes áreas de la ingeniería, lo que permite a los profesionales de este campo abordar y resolver problemas complejos relacionados con la

automatización, la robótica y la integración de sistemas mecánicos y electrónicos. Sus aplicaciones tienen un amplio alcance y contribuyen al avance tecnológico y a la mejora de procesos en diversas industrias.

En este trabajo nos enfocaremos principalmente en el 4 y último punto que se presentó, pero antes debemos aprender, entender y tener en cuenta unos cuantos conceptos más y herramientas que se utilizaron para este proyecto.

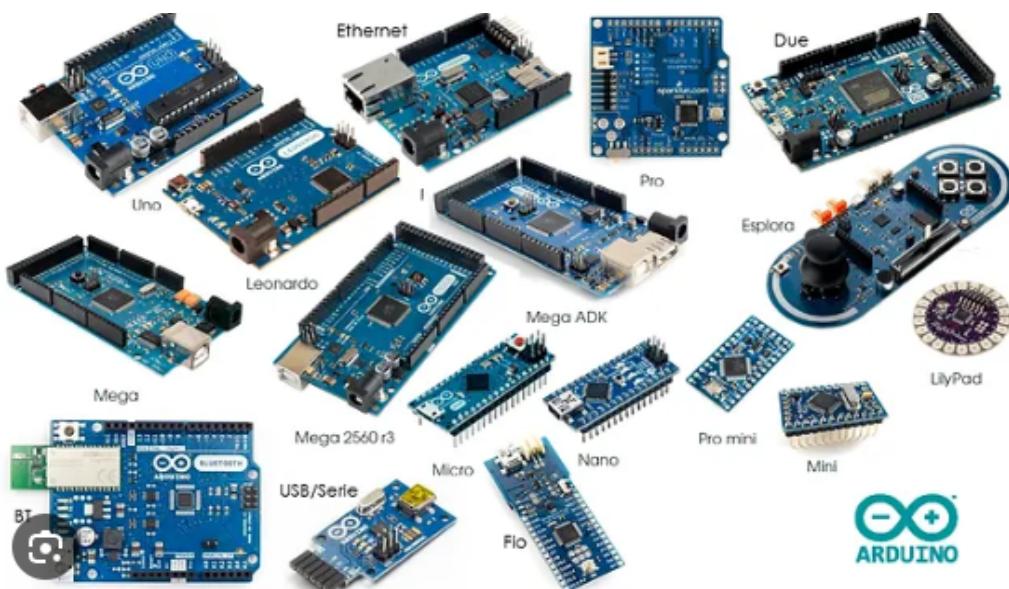
¿Qué es Arduino y para qué sirve?

Arduino es una plataforma de hardware y software de código abierto diseñada para facilitar la creación de proyectos electrónicos interactivos. Consiste en una placa de desarrollo con un microcontrolador (Una computadora en pequeña escala) y un entorno de programación que permite crear proyectos electrónicos .



Software. (s. f.). Arduino. <https://www.arduino.cc/en/software>

Las placas Arduino vienen en diferentes tamaños y configuraciones, pero todas ellas ofrecen un conjunto de pines de entrada/salida (E/S) que permiten conectar sensores, actuadores y otros dispositivos electrónicos.



Tech, L. (2019, 8 marzo). Qué es arduino Lozury Tech.

Las principales características de Arduino son que es una plataforma de código abierto, lo que significa que el diseño y el software son de libre acceso y pueden ser modificados y distribuidos por cualquier persona, así mismo que es una combinatoria de programación y electrónica haciendo uso de componentes de hardware y software.

¿Qué relación hay entre Arduino y Mecatrónica?

Arduino tiene una relación estrecha con la mecatrónica, ya que es una herramienta muy utilizada en esta disciplina para el diseño y desarrollo de sistemas mecatrónicos.

La mecatrónica combina la ingeniería mecánica, electrónica y de control, junto con la informática, para crear sistemas inteligentes y automatizados. En esta área, los ingenieros mecatrónicos diseñan y desarrollan sistemas que integran componentes mecánicos y electrónicos, así como sistemas de control y software para lograr un funcionamiento eficiente y preciso.

Arduino es una plataforma de desarrollo que se adapta muy bien a la mecatrónica debido a varias razones, pero las principales son:

1. Versatilidad: Arduino ofrece una amplia gama de placas y sensores que pueden utilizarse para interactuar con el entorno físico. Los sensores permiten medir variables como la temperatura, la humedad, la luz, la posición, el movimiento, entre otros, lo que es fundamental para muchos sistemas mecatrónicos.
2. Facilidad de uso: El entorno de programación de Arduino es fácil de aprender y usar, lo que permite a los ingenieros mecatrónicos desarrollar rápidamente prototipos y sistemas de control para sus proyectos.
3. Bajo costo: Arduino es una plataforma de código abierto y las placas y componentes son muy baratos.
4. Comunidad activa: La comunidad de Arduino es grande y activa, con una gran cantidad de recursos, tutoriales y proyectos disponibles en línea. Esto facilita encontrar soluciones a sus desafíos y aprender de otros miembros de la comunidad.

En la mecatrónica, Arduino se utiliza para una amplia variedad de aplicaciones, como el control de robots, sistemas de automatización y control, sistemas de monitoreo y adquisición de datos, dispositivos de electrónica interactiva, entre otros. Su capacidad para interactuar con el mundo físico y su facilidad de uso lo convierten en una herramienta valiosa para prototipar y desarrollar sistemas mecatrónicos de manera rápida y eficiente.

Radar detector de Objetos

Para la elaboración de dicho proyecto se hizo uso de los siguientes materiales:

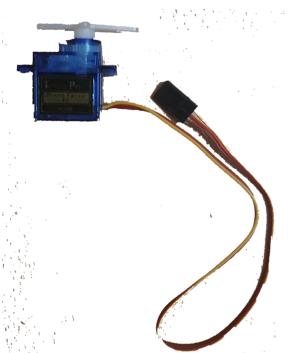
Arduino Mega 2560 (Placa en la que conectaremos el software y hardware)



Cable USB (Para conectar la placa Arduino con la computadora)



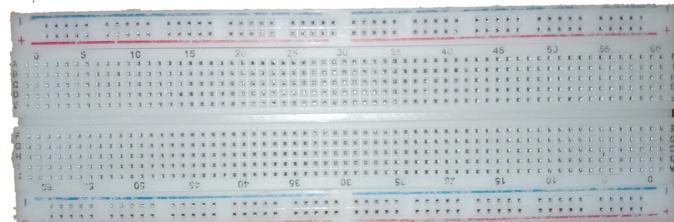
ServoMotor SG90 (Para una rotación de 180 grados)



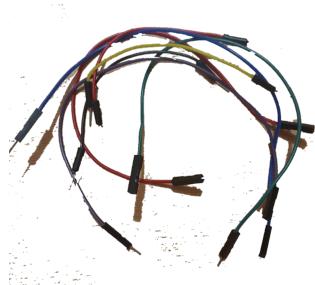
Sensor Ultrasonido HC-SR04 (El cual detectara los objetos)



Protoboard (Donde se fijarán los componentes)

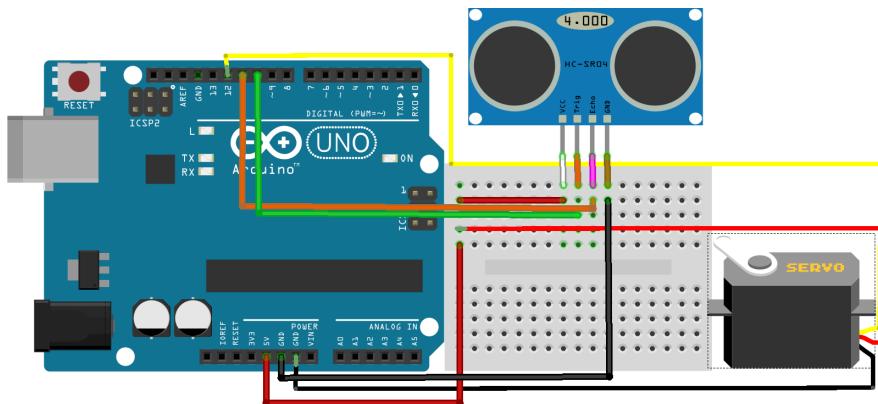


Cables (Para conectar los componentes)



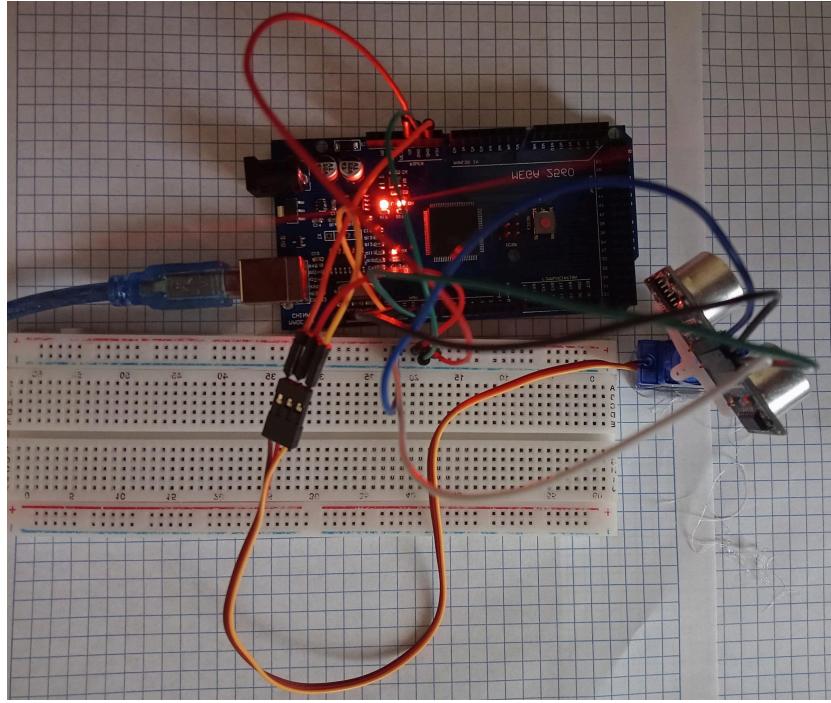
También debemos de tener instalados lo que son los softwares de “Arduino” donde haremos todo lo que es la parte de programación y “Processing” la cual nos ayudará con la interfaz gráfica del radar.

Ahora, el primer paso es conectar todos los componentes físicos (hardware). Para facilitar visualmente dicho proceso se muestra el siguiente diagrama:



Esquema de circuito electrónico para la elaboración de un radar con arduino hecho en Fritzing

El circuito ya armado quedaría de la siguiente manera:



Una vez hecho esto, debemos de codificar el siguiente código en Arduino

```
MECBAS §
#include <Servo.h>
const int trigPin = 10;
const int echoPin = 11;
long duration;
int distance;
Servo myServo;
void setup()
{
    pinMode(trigPin, OUTPUT);
    pinMode(echoPin, INPUT);
    Serial.begin(9600);
    myServo.attach(12);
}
void loop()
{
    for (int i = 15; i <= 165; i++) {
        myServo.write(i);
        delay(30);
        distance = calculateDistance();
```

```

        Serial.print(i);
        Serial.print(",");
        Serial.print(distance);
        Serial.print(".");
    }

    for (int i = 165; i > 15; i--) {
        myServo.write(i);
        delay(30);
        distance = calculateDistance();
        Serial.print(i);
        Serial.print(",");
        Serial.print(distance);
        Serial.print(".");
    }
}

int calculateDistance()
{
    digitalWrite(trigPin, LOW);
    delayMicroseconds(2);

    digitalWrite(trigPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPin, LOW);
    duration = pulseIn(echoPin, HIGH);
    distance = duration * 0.034 / 2;
    return distance;
}

```

En dicho código se incluye la librería #include<Servo.h>" que permite controlar el servomotor conectado al Arduino, se definen dos constantes para almacenar los números de los pines conectados al sensor de ultrasonido.

“trigPin” se utilizará como pin de salida para enviar la señal de disparo al sensor y “echoPin” como pin de entrada para recibir la señal de eco desde el sensor. También se declaran dos variables globales para almacenar la duración del eco recibido por el sensor y la distancia calculada; y se crea una variable llamada “myServo” de tipo “Servo”, que se utilizará para controlar el servo motor.

MECBAS §

```
#include <Servo.h>

const int trigPin = 10;
const int echoPin = 11;
long duration;
int distance;
Servo myServo;
```

Después sigue el método “void setup()” en el cual la función “setup()”, se configuran los pines “trigPin” y “echoPin” como salida y entrada. Además, se inicia la comunicación serie con una velocidad de 9600 baudios en lo que es “Serial.begin(9600)”. Por último, en “myServo.attach(12)” quiere decir que se adjuntará el servo motor al pin 12 de la tarjeta “Arduino”.

```
void setup()
{
    pinMode(trigPin, OUTPUT);
    pinMode(echoPin, INPUT);
    Serial.begin(9600);
    myServo.attach(12);
}
```

Después en el bucle principal “loop()”, el servomotor se mueve desde 15 grados hasta 165 grados en incrementos de un grado. En cada iteración, se llama a la función “calculateDistance()” para medir la distancia utilizando el sensor de

ultrasonido. Luego, los valores del ángulo y la distancia se envían a través del puerto serie “Serial.print()”

```
void loop()
{
    for (int i = 15; i <= 165; i++) {
        myServo.write(i);
        delay(30);
        distance = calculateDistance();
        Serial.print(i);
        Serial.print(",");
        Serial.print(distance);
        Serial.print(".");
    }
    for (int i = 165; i > 15; i--) {
        myServo.write(i);
        delay(30);
        distance = calculateDistance();
        Serial.print(i);
        Serial.print(",");
        Serial.print(distance);
        Serial.print(".");
    }
}
```

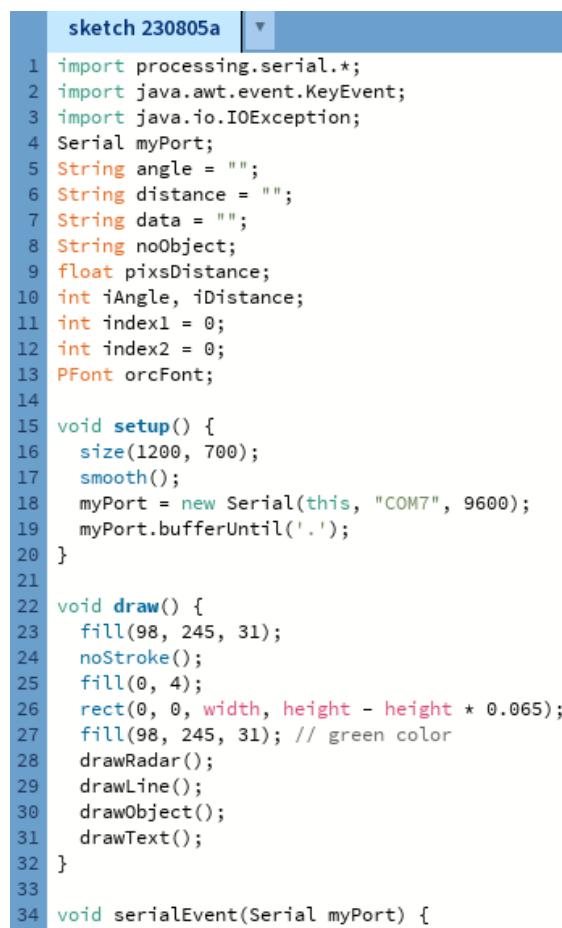
Por último, la función “calculateDistance()” sirve para medir la distancia utilizando el sensor de ultrasonido. Primero, se establece el pin “trigPin” en bajo durante 2 microsegundos y luego en alto durante 10 microsegundos. Esto envía un pulso al sensor para activar la medición. Luego, se mide la duración del pulso en el pin “echoPin” usando “pulseIn”, y esta duración se convierte en distancia en centímetros utilizando la fórmula de velocidad del sonido en el aire y dividiendo entre 2 para obtener la distancia de ida y vuelta.

```

int calculateDistance()
{
    digitalWrite(trigPin, LOW);
    delayMicroseconds(2);
    digitalWrite(trigPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPin, LOW);
    duration = pulseIn(echoPin, HIGH);
    distance = duration * 0.034 / 2;
    return distance;
}

```

Una vez hecha la programación en Arduino, debemos programar lo que sería la interfaz gráfica del radar en lo que es la aplicación “Processing”



```

sketch 230805a ▾
1 import processing.serial.*;
2 import java.awt.event.KeyEvent;
3 import java.io.IOException;
4 Serial myPort;
5 String angle = "";
6 String distance = "";
7 String data = "";
8 String noObject;
9 float pixsDistance;
10 int iAngle, iDistance;
11 int index1 = 0;
12 int index2 = 0;
13 PFont orcFont;
14
15 void setup() {
16     size(1200, 700);
17     smooth();
18     myPort = new Serial(this, "COM7", 9600);
19     myPort.bufferUntil('.');
20 }
21
22 void draw() {
23     fill(98, 245, 31);
24     noStroke();
25     fill(0, 4);
26     rect(0, 0, width, height - height * 0.065);
27     fill(98, 245, 31); // green color
28     drawRadar();
29     drawLine();
30     drawObject();
31     drawText();
32 }
33
34 void serialEvent(Serial myPort) {

```

```
35     data = myPort.readStringUntil('.');
36     data = data.substring(0, data.length() - 1);
37     index1 = data.indexOf(",");
38     angle = data.substring(0, index1);
39     distance = data.substring(index1 + 1, data.length());
40     iAngle = int(angle);
41     iDistance = int(distance);
42 }
43
44 void drawRadar() {
45     pushMatrix();
46     translate(width / 2, height - height * 0.074);
47     noFill();
48     strokeWeight(2);
49     stroke(98, 245, 31);
50     arc(0, 0, (width - width * 0.0625), (width - width * 0.0625), PI, TWO_PI);
51     arc(0, 0, (width - width * 0.27), (width - width * 0.27), PI, TWO_PI);
52     arc(0, 0, (width - width * 0.479), (width - width * 0.479), PI, TWO_PI);
53     arc(0, 0, (width - width * 0.687), (width - width * 0.687), PI, TWO_PI);
54     line(-width / 2, 0, width / 2, 0);
55     line(0, 0, (-width / 2) * cos(radians(30)), (-width / 2) * sin(radians(30)));
56     line(0, 0, (-width / 2) * cos(radians(60)), (-width / 2) * sin(radians(60)));
57     line(0, 0, (-width / 2) * cos(radians(90)), (-width / 2) * sin(radians(90)));
58     line(0, 0, (-width / 2) * cos(radians(120)), (-width / 2) * sin(radians(120)));
59     line(0, 0, (-width / 2) * cos(radians(150)), (-width / 2) * sin(radians(150)));
60     line((-width / 2) * cos(radians(30)), 0, width / 2, 0);
61     popMatrix();
62 }
63
64 void drawObject() {
65     pushMatrix();
66     translate(width / 2, height - height * 0.074);
67     strokeWeight(9);
68     stroke(255, 10, 10);
```

```

69     pixsDistance = iDistance * ((height - height * 0.1666) * 0.025);
70     if (iDistance < 40) {
71         line(pixsDistance * cos(radians(iAngle)), -pixsDistance * sin(radians(iAngle))), (width - width * 0.505) * cos(radians(iAngle)), -(width - width * 0.505) * sin(radians(iAngle)));
72     }
73     popMatrix();
74 }
75
76 void drawLine() {
77     pushMatrix();
78     strokeWeight(9);
79     stroke(30, 250, 60);
80     translate(width / 2, height - height * 0.074);
81     line(0, 0, (height - height * 0.12) * cos(radians(iAngle)), -(height - height * 0.12) * sin(radians(iAngle)));
82     popMatrix();
83 }
84
85 void drawText() {
86     pushMatrix();
87     if (iDistance > 40) {
88         noObject = "Out of Range";
89     } else {
90         noObject = "In Range";
91     }
92     fill(0, 0, 0);
93     noStroke();
94     rect(0, height - height * 0.0648, width, height);
95     fill(98, 245, 31);
96     textSize(25);
97     text("10cm", width - width * 0.3854, height - height * 0.0833);
98     text("120cm", width - width * 0.281, height - height * 0.0833);
99     text("130cm", width - width * 0.177, height - height * 0.0833);
100    text("140cm", width - width * 0.0729, height - height * 0.0833);
101    textSize(40);
102    text("Angle: " + iAngle + " °", width - width * 0.48, height - height * 0.0277);

```

```

102    text("Ángulo: " + iAngle + " °", width - width * 0.48, height - height * 0.0277);
103    text("Dist:", width - width * 0.26, height - height * 0.0277);
104    if (idistance < 40) {
105        text(
106            " " + idistance + " cm", width - width * 0.225, height - height * 0.0277);
107    }
108    textSize(25);
109    fill(98, 245, 60);
110    translate((width - width * 0.4994) + width / 2 * cos(radians(30)), (height - height * 0.0907) - width / 2 * sin(radians(30)));
111    rotate(-radians(-60));
112    text("30°", 0, 0);
113    resetMatrix();
114    translate((width - width * 0.503) + width / 2 * cos(radians(60)), (height - height * 0.0888) - width / 2 * sin(radians(60)));
115    rotate(-radians(-30));
116    text("60°", 0, 0);
117    resetMatrix();
118    translate((width - width * 0.507) + width / 2 * cos(radians(90)), (height - height * 0.0833) - width / 2 * sin(radians(90)));
119    rotate(radians(0));
120    text("90°", 0, 0);
121    resetMatrix();
122    translate(width - width * 0.513 + width / 2 * cos(radians(120)), (height - height * 0.07129) - width / 2 * sin(radians(120)));
123    text("120°", 0, 0);
124    resetMatrix();
125    translate((width - width * 0.5104) + width / 2 * cos(radians(150)), (height - height * 0.0574) - width / 2 * sin(radians(150)));
126    rotate(radians(-60));
127    text("150°", 0, 0);
128    popMatrix();
129}

```

Donde la explicación de que es lo que hace el código anterior, es la siguiente

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

Este bloque del código sirve para que se importen librerías para el procesamiento serial y el manejo de eventos.

```
4 Serial myPort;
5 String angle = "";
6 String distance = "";
7 String data = "";
8 String noObject;
9 float pixsDistance;
10 int iAngle, iDistance;
11 int index1 = 0;
12 int index2 = 0;
13 PFont orcFont;
14
```

Después se declaran lo que son variables globales, es decir, variables que serán utilizadas en todo el programa o resto del código, sirven para almacenar datos que se utilizaran para el funcionamiento del radar y su visualización.

```
15 void setup() {
16   size(1200, 700);
17   smooth();
18   myPort = new Serial(this, "COM7", 9600);
19   myPort.bufferUntil('.');
20 }
```

Luego en la función “setup()”, se establecen lo que son las configuraciones iniciales del programa. Se define el tamaño de la ventana de visualización, se suavizan las formas gráficas y se inicializa la comunicación serial a través del puerto "COM7" a una velocidad de 9600 baudios. Además, se llama a “bufferUntil(‘.’)” para indicar que los datos seriales se deben leer hasta que se encuentre un punto “.”, lo que delimita el final de una lectura.

```
22 void draw() {
23   fill(98, 245, 31);
24   noStroke();
25   fill(0, 4);
26   rect(0, 0, width, height - height * 0.065);
27   fill(98, 245, 31); // green color
28   drawRadar();
29   drawLine();
30   drawObject();
31   drawText();
32 }
```

Luego, en la función “draw()” se realiza la visualización del radar y sus elementos. Se dibuja un fondo verde y transparente y luego, se llama a diferentes funciones que dibujan el radar, la línea y el objeto detectado, así como el texto con la información.

```
34 void serialEvent(Serial myPort) {  
35     data = myPort.readStringUntil('.');  
36     data = data.substring(0, data.length() - 1);  
37     index1 = data.indexOf(",");  
38     angle = data.substring(0, index1);  
39     distance = data.substring(index1 + 1, data.length());  
40     iAngle = int(angle);  
41     iDistance = int(distance);  
42 }
```

En la función “serialEvent()”, se lee la información recibida por el puerto serial y se almacena en las variables “angle” y “distance”, después se convierten a enteros y se almacenan en las variables “iAngle” y “iDistance”, ya que dichos valores representan el ángulo y la distancia del objeto detectado por el radar.

64 void drawObject()

76 void drawLine()

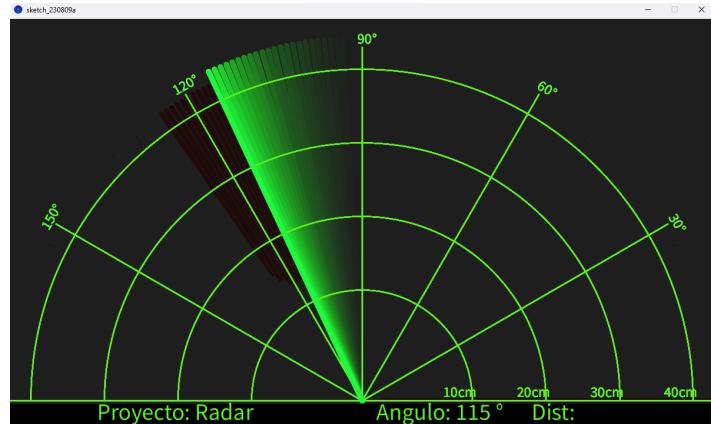
85 void drawText()

Por último, en las funciones de “void drawRadar()”, “void drawObject()”, “void drawLine()” y “void drawText()”, sirven para dibujar diferentes elementos en la visualización del radar. Cada función se encarga de pintar un componente específico del radar, como el fondo del radar, el objeto detectado, la línea que indica la distancia, y el texto que muestra la información del ángulo y la distancia del objeto.

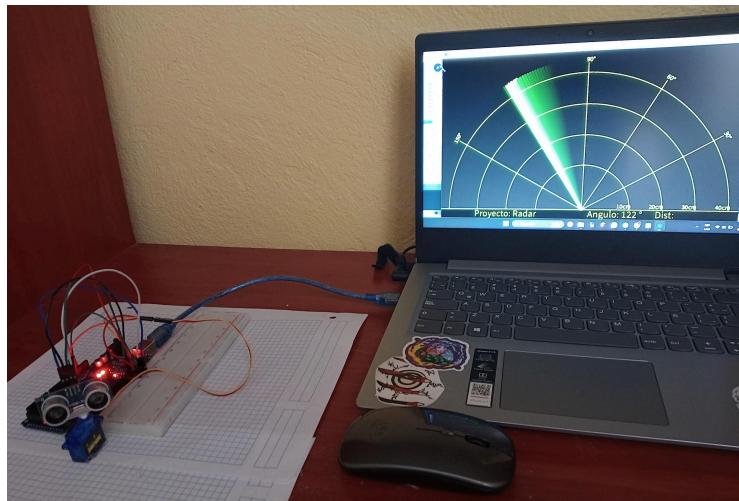
Finalmente, subimos el código de Arduino, de forma que para asegurarnos que el código está cargado en la tarjeta “Arduino” nos salga un mensaje similar al siguiente:

```
Subido  
El Sketch usa 5612 bytes (2%) del espacio de almacenamiento de programa. El máximo es 253952 bytes.  
Las variables Globales usan 342 bytes (4%) de la memoria dinámica, dejando 7850 bytes para las varia
```

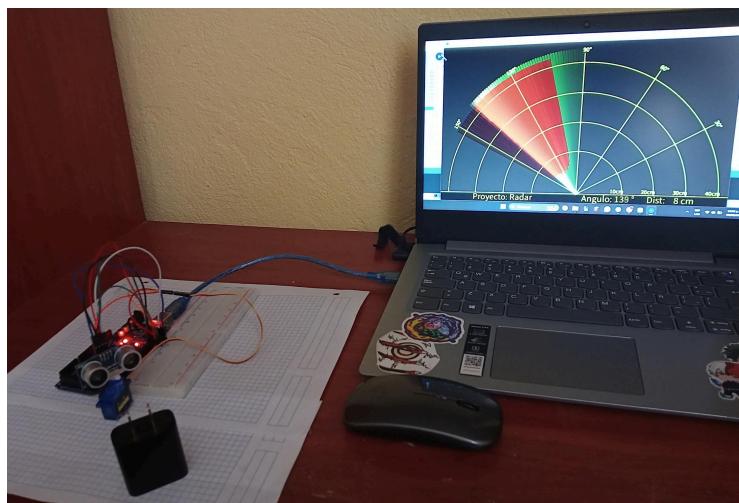
En la aplicación de “Processing” simplemente le daremos ejecutar para poder ver la interfaz gráfica, a continuación se muestran unas imágenes para ejemplificar cómo sería el resultado final.



La siguiente imagen, muestra el hardware y software funcionando, sin ningún objeto enfrente:



Ahora, la siguiente imagen muestra el hardware y software trabajando en conjunto con un objeto en frente (Un cuadro de un cargador de celular), como se ve en la pantalla de la computadora, lo detecta y muestra en pantalla que dicho objeto se encuentra a 8 CM de distancia y a un ángulo de 139°



Conclusión

Este proyecto ha demostrado que la combinación de estos conocimientos ha permitido desarrollar un dispositivo capaz de detectar y medir la presencia, dirección y distancia de objetos distantes mediante el uso de ondas de ultrasonido.

Creo que la programación juega un papel muy crucial para el diseño del algoritmo que procesa las señales de retorno del radar y calcula la distancia de los objetos detectados o para dar cualquier tipo de instrucciones que deseemos realizar, pero pienso que esta tiene un impacto muy grande si no va de la mano con otras áreas como la mecánica, matemáticas, electrónica, etc. Por ejemplo, la electrónica ha sido una pieza fundamental en la construcción física del radar. El diseño y montaje de los circuitos electrónicos necesarios para la generación y recepción de las ondas han sido determinantes en la funcionalidad y el rendimiento del dispositivo. Además, la elección y calibración adecuada de los componentes electrónicos han asegurado la estabilidad y precisión del radar.

La utilización de la plataforma Arduino ha demostrado ser una excelente elección para este proyecto y en mi opinión una excelente herramienta con la cual el aprendizaje y entendimiento de lo que es la mecatrónica se vuelve algo muy sencillo y hasta cierto punto divertido, pero sobre todo, interesante, ya que gracias a este tipo de proyectos o trabajos, uno puede entender o siquiera hacerse una idea de cómo es que funcionan las cosas que nos rodean dia a dia y aplicamos, en donde muchos casos, la gente no se pregunta ni le interesa cómo es que dichas herramientas funcionan, pero creo que es importante al menos tener la noción de todo esto y más cuando vivimos en una era donde la tecnología se vuelve algo tan importante al punto en el que un ser humano moderno, depende de dicha herramienta.

Arduino proporciona una interfaz amigable y versátil para la programación y el control de los componentes electrónicos, lo que ha facilitado la integración de las diferentes áreas y agilizado el proceso de desarrollo.

El trabajo en conjunto de estas áreas ha llevado a la creación de un radar detector de objetos que ofrece grandes posibilidades en diversas aplicaciones, desde la navegación y control de tráfico hasta la seguridad y vigilancia. Además, la combinación de programación, electrónica y Arduino ha permitido que el proyecto sea accesible para aquellos interesados en la tecnología, brindando una oportunidad de aprendizaje en campos interdisciplinarios.

Este proyecto representa un ejemplo valioso de cómo la convergencia de distintos conocimientos puede dar lugar a soluciones innovadoras y efectivas. Con este

enfoque interdisciplinario, se abre la puerta a nuevas investigaciones y desarrollos tecnológicos que pueden tener un impacto significativo en diversas industrias y aplicaciones prácticas, pero sobre todo en el aprendizaje y formación de un estudiante relacionado a esta disciplina.

Referencias

Max. (2023). Cómo estudiar Ingeniería Mecatrónica en la UNAM | Plan de estudios, convocatoria y más.

Unibetas curso examen de admisión online.

[https://unibetas.com/ingenieria-mecatronica-unam#:~:text=La%20principal%20aplicaci%C3%B3n%20de%20la%20salud.](https://unibetas.com/ingenieria-mecatronica-unam#:~:text=La%20principal%20aplicaci%C3%B3n%20de%20la%20salud)

Ingeniería mecatrónica, lo que debes saber. (s. f.). UAG Media Hub.

<https://www.uag.mx/es/mediahub/ingenieria-mecatronica-lo-que-debes-saber#:~:text=Un%20Ingeniero%20en%20Mecatr%C3%B3nica%20es,computadoras%2C%20dise%C3%B3n%20mec%C3%A1nico%20y%20simulaciones.>

Fundación Aquae. (2020, 30 octubre). ¿Sabes qué es un arduino y para qué sirve? - Fundación AQUAE.

<https://www.fundacionaqua.org/wiki/sabes-arduino-sirve/#:~:text=1.,2.>

Aula. (2023, 8 mayo). Arduino: guía completa para principiantes y expertos | Aprende ya. aula21 |

Formación para la Industria.

<https://www.cursosaula21.com/arduino-todo-lo-que-necesitas-saber/>

Software. (s. f.). Arduino. https://www.arduino.cc/en/software

Tech, L. (2019, 8 marzo). Qué es arduino Lozury Tech.

<https://www.lozurytech.com/single-post/qu%C3%A9-es-arduino>