# Laberinto de pelotas

Integrantes: Alma Gutiérrez, Gustavo Creczuk, Santiago Pont

Año: 4to 3ra

Profesor:  Gonzalo Consorti

## Introducción

El laberinto de pelotas controlado por Arduino es una iniciativa tecnológica y educativa que combina ingeniería, programación y diseño creativo. Este proyecto busca aprovechar el potencial del microcontrolador Arduino para desarrollar un sistema automatizado capaz de manipular y guiar pelotas a través de un laberinto interactivo.

El objetivo principal es diseñar un laberinto en el que las pelotas puedan desplazarse mediante la intervención de componentes electrónicos controlados por Arduino, como servomotores y sensores . Este enfoque permite no solo construir un sistema físico atractivo, sino también incorporar lógica de programación y elementos de automatización que aumentan la funcionalidad y la experiencia del usuario.

## Carpeta de Campo

Cuando elegimos nuestro proyecto, empezamos a investigar todo lo que necesitábamos para hacerlo. Decidimos crear un laberinto para pelotas controlado con Arduino. Este proyecto mezcla partes mecánicas, electrónicas y programación.

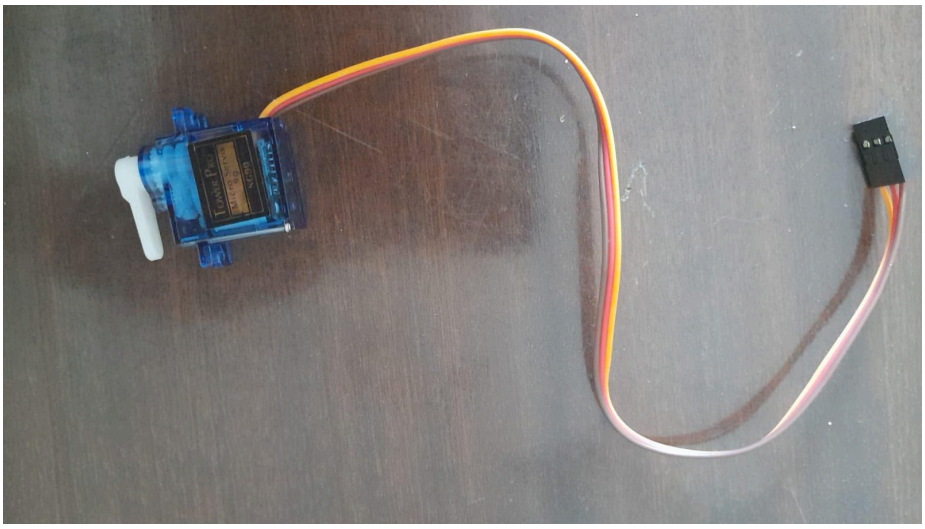
## Materiales

- Arduino Uno (o cualquier otra placa Arduino compatible).
- Servomotores (x2): Para controlar la inclinación del laberinto en los ejes X e Y.
- Joystick : Para controlar manualmente la inclinación.
- Cables de conexión .
- Protoboard: Para conectar componentes fácilmente.

- Fuente de alimentación (puedes usar el cable USB del Arduino o una batería externa).

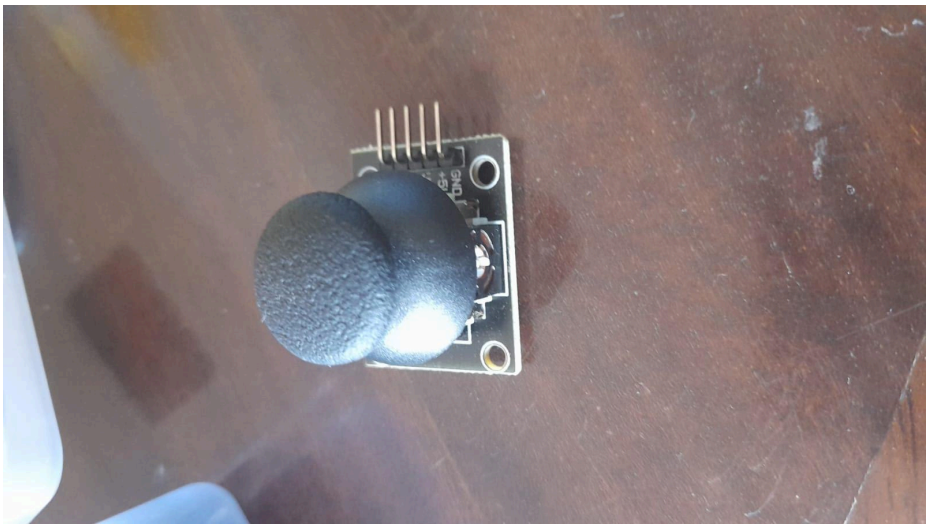
Después de ver qué materiales necesitábamos, nos dimos cuenta de que había cosas que no sabíamos usar, como el servomotor y el joystick. Por eso, investigamos cómo conectarlos y utilizarlos correctamente.

Primero, aprendimos que el servomotor necesita tres conexiones principales: una para la alimentación (positivo y negativo) y otra para la señal, que va conectada a la placa Arduino. Descubrimos que la señal permite controlar el ángulo de giro del servomotor a través de programación. Miramos varios tutoriales y probamos ejemplos simples para asegurarnos de que funcionara bien.

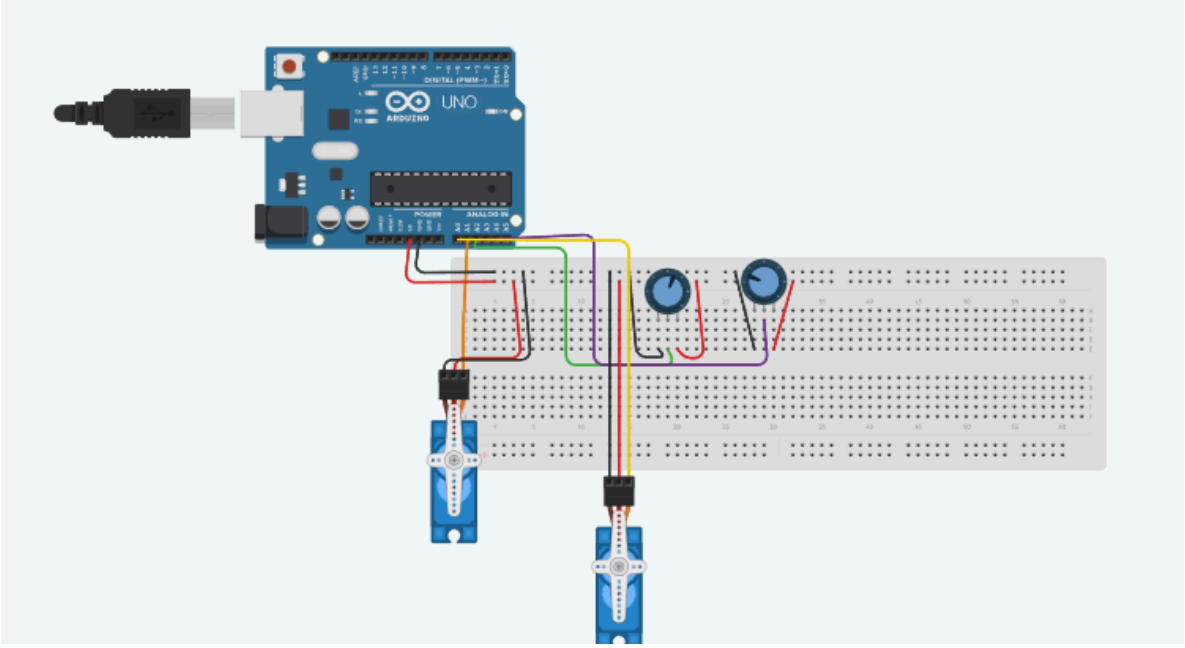


Luego, investigamos el joystick. Aprendimos que tiene dos ejes principales, el X y el Y, que envían señales al Arduino para mover el laberinto en distintas direcciones. También descubrimos que el joystick tiene un botón que se puede programar para funciones adicionales pero no lo usaremos en este proyecto. Nos aseguramos de conectar correctamente cada pin del

joystick a los puertos del Arduino y verificamos que las lecturas fueran precisas.



Luego, empezamos haciendo el código en tinkercad para que todo funcionara



```
#include <Servo.h>
#define x A2
#define y A3


Servo servoBasex;//Asigno un nombre especifico
Servo servoBaseY;
void setup() {
   servoBasex.attach(A0);//Pin a utilizar para servo
   servoBasex.write(0);  //asigno 0 al servo motor
   servoBaseY.attach(A1);//Pin a utilizar para servo
   servoBaseY.write(0);  //asigno 0 al servo motor
   pinMode(y,INPUT);
   pinMode(x,INPUT);
   Serial.begin(9600);
}

void loop() {
    int valor = analogRead( A2 );
  int grados = map(valor,  0, 1023, 0, 255);
  servoBasex.write(grados);

int valory = analogRead( A3 );
  int gradosy = map(valory,  0, 1023, 0, 255);
  servoBaseY.write(gradosy);


  Serial.println( grados);

}
```

El código utiliza la biblioteca Servo.h para controlar dos servomotores asignando los nombres servoBaseX y servoBaseY define las constantes X como A2 e Y como A3 en la función setup se configura el pin A0 para el servomotor servoBaseX y el pin A1 para el servomotor servoBaseY ambos se inicializan con un valor de cero se configuran los pines X e Y como entradas y se inicia la comunicación serial a 9600 baudios en la función loop primero se lee un valor analógico desde el pin A2 correspondiente a la constante X este valor se mapea de su rango original de 0 a 1023 al rango de 0 a 255 y se asigna al servomotor servoBaseX luego se repite el proceso para el pin A3 correspondiente a la constante Y asignando el valor al servomotor servoBaseY finalmente el valor de grados mapeado del primer servomotor se imprime en el monitor serial pero este codigo no funciono al intentar mover el eje y el servomotor del eje y no funcionaba entonces cambiamos el codigo

```
#include <Servo.h>

Servo Xservo;
Servo Yservo;

int VRXpin=A0;
int VRYpin=A1;

int XServoPin=9;
int YServoPin=10;

int WVx;
int WVy;

int Xval;
int Yval;

int dt=100;

int Xrange=10;
int Yrange=20;

int adjustX=0;
int adjustY=0;
```

El código incluye la librería `Servo.h` para controlar los servomotores. Se crean dos objetos, `Xservo` y `Yservo`, para controlar los movimientos en los ejes X e Y del laberinto. Se definen los pines del joystick (A0 para X y A1 para Y) y los pines de los servos (9 para X y 10 para Y).

Se usan variables para leer los valores del joystick (`Xval` y `Yval`) y para procesar esos valores en `WVx` y `WVy`. También se establece un intervalo de 100 milisegundos (`dt`) y se definen los rangos de movimiento (`Xrange` y `Yrange`). Las variables `adjustX` y `adjustY` permiten calibrar los movimientos.

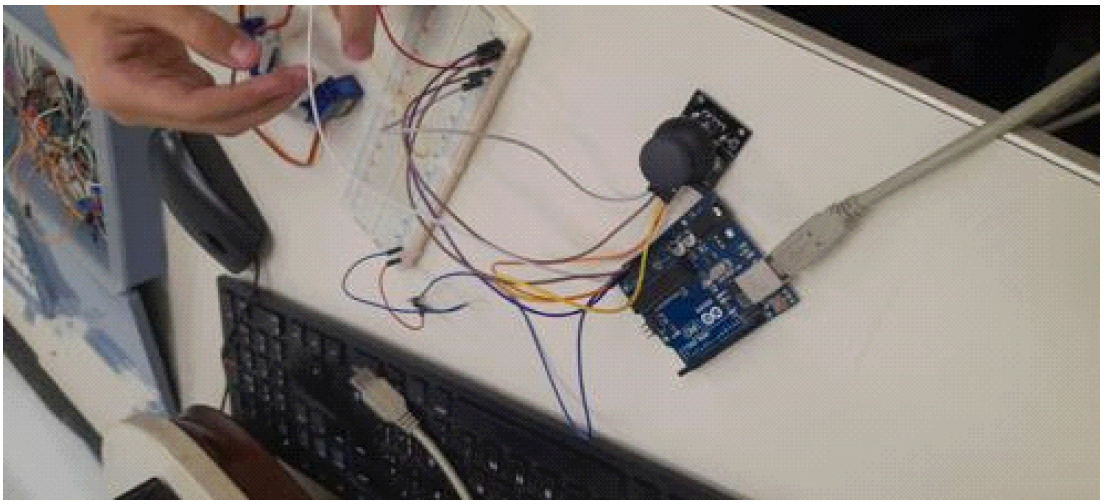
```
void setup() {

pinMode(VRXpin, INPUT);
pinMode(VRYpin, INPUT);

pinMode(XServoPin, OUTPUT);
pinMode(YServoPin, OUTPUT);

Xservo.attach(XServoPin);
Yservo.attach(YServoPin);
}
```

El código es parte de un programa de Arduino que configura pines para manejar un joystick y dos servomotores. La primera sección configura los pines VRXpin y VRYpin como entradas para leer los valores del joystick. La segunda sección configura los pines XServoPin y YServoPin como salidas para controlar los servomotores. Finalmente, los objetos Xservo y Yservo se vinculan a los pines de salida usando la función attach de la biblioteca Servo para

permitir el control de los servos. El objetivo es controlar los servos en función de los movimientos del joystick.

```
pinMode(VRXpin, INPUT);
pinMode(VRYpin, INPUT);

pinMode(XServoPin, OUTPUT);
pinMode(YServoPin, OUTPUT);

Xservo.attach(XServoPin);
Yservo.attach(YServoPin);
}
```

La función `servo.attach` sirve para vincular un objeto de tipo servo a un pin específico de Arduino. Esto establece el control del servomotor permitiendo enviarle señales desde el programa.

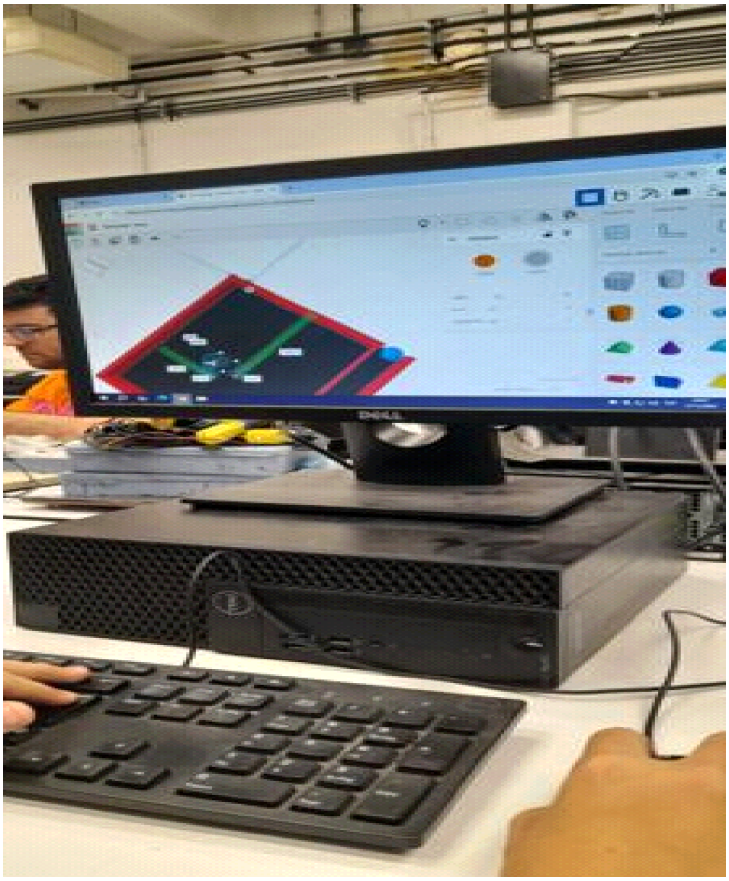


Cuando conectamos los dos servomotores y los ejes X y Y, probamos el programa inicialmente en Tinkercad utilizando potenciómetros, y todo funcionó correctamente. Sin embargo, al intentar realizar la misma prueba con entradas analógicas reales en lugar de los potenciómetros, el sistema no respondió de la misma manera.

La posible causa de esta diferencia puede estar relacionada con cómo Tinkercad procesa las señales analógicas en comparación con un entorno físico.

```
#include <Servo.h>
#define x A2
#define y A3


Servo servoBasex;//Asigno un nombre específico
Servo servoBaseY;
void setup() {
   servoBasex.attach(A0);//Pin a utilizar para servo
   servoBasex.write(0);  //asigno 0 al servo motor
   servoBaseY.attach(A1);//Pin a utilizar para servo
   servoBaseY.write(0);  //asigno 0 al servo motor
   pinMode(y,INPUT);
   pinMode(x,INPUT);
   Serial.begin(9600);
}

void loop() {
    int valor = analogRead( A2 );
  int grados = map(valor,  0, 1023, 0, 255);
  servoBasex.write(grados);

int valory = analogRead( A3 );
  int gradosy = map(valory,  0, 1023, 0, 255);
  servoBaseY.write(gradosy);


  Serial.println( grados);
```

Este es el código que funcionaba correctamente cuando utilizamos dos potenciómetros para controlar los servomotores, pero no funcionó de la misma manera cuando intentamos realizar la prueba utilizando un joystick en lugar de los potenciómetros. Pero el segundo código funciona correctamente.



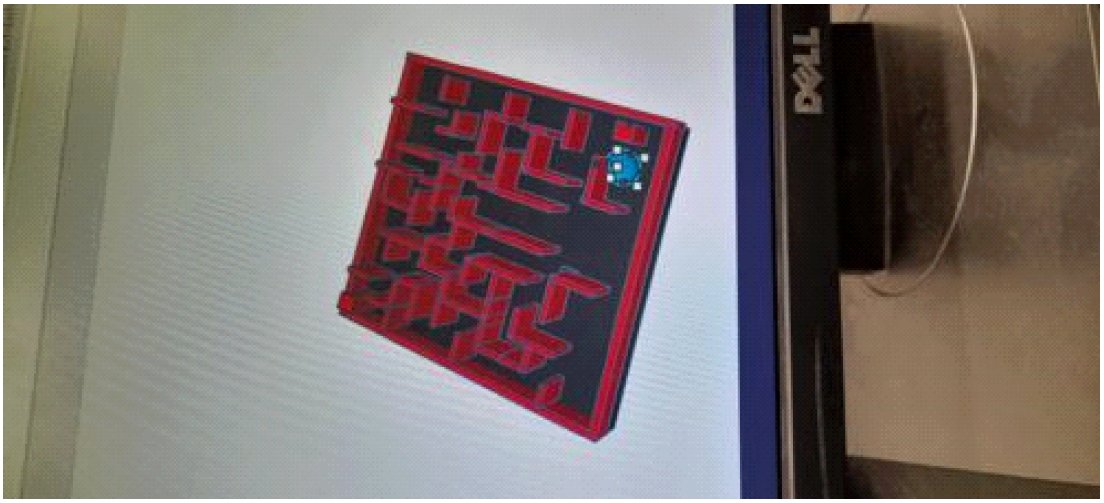
Nos encontrábamos en una de las fases clave del proyecto, evaluando la mejor forma de diseñar la entrada del laberinto. Ajustamos los ángulos y las dimensiones para garantizar no solo un acceso fluido para la pelota, sino también un diseño visualmente atractivo que reflejara la atención al detalle.

También nos concentramos en la disposición de los muros internos. Probamos diferentes configuraciones para asegurarnos de que fueran lo suficientemente firmes como para guiar el movimiento de la pelota a lo largo del recorrido, sin convertirse en un obstáculo que afectara su desplazamiento natural. El reto consistía en encontrar el equilibrio ideal entre la estabilidad de los muros y el espacio necesario para que la pelota se moviera libremente.

Simultáneamente, avanzábamos en la construcción de la base del laberinto. Diseñamos una estructura sólida y resistente que pudiera soportar el peso del montaje completo, sin comprometer la precisión de los movimientos controlados por los servomotores. Esto implicó analizar materiales, probar diferentes tipos de uniones y asegurarnos de que los servomotores tuvieran el rango de movimiento necesario para inclinar la plataforma con precisión, fluidez y rapidez.

Luego, usted nos mencionó que existía una página web que podía generar laberintos de manera automática, lo que podría haber simplificado parte del proceso de diseño inicial. Sin embargo, ese día nos encontramos con que la página no estaba funcionando correctamente, lo que nos llevó a improvisar y buscar alternativas para avanzar con el proyecto.

Decidimos retomar el diseño manual, basándonos en nuestras ideas originales y utilizando herramientas más convencionales. Esto implicó recurrir a bocetos en papel y programas de diseño para conceptualizar el laberinto. Aunque la ausencia de la herramienta fue un contratiempo, también resultó en una oportunidad para personalizar el diseño de una manera más creativa y acorde a las necesidades específicas del proyecto.



Trabajamos en el diseño del laberinto, afinando los detalles para que el recorrido de la pelota fuera tanto fluido como desafiante. Nos aseguramos de que las paredes tuvieran la altura adecuada para evitar que la pelota se saliera, mientras creábamos un trazado dinámico que pusiera a prueba el control mediante los servomotores. Aunque el diseño era sencillo, debía cumplir con los requisitos funcionales básicos y garantizar un desempeño óptimo durante las pruebas del sistema.

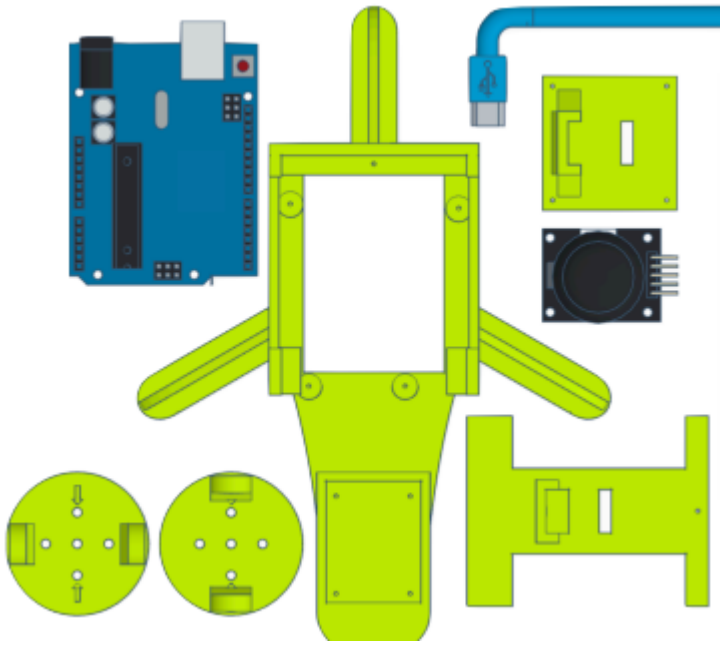
A continuación, comenzamos con la búsqueda de los soportes necesarios tanto para los servos como para la estructura del laberinto. Este paso fue crucial, ya que estos componentes debían garantizar la estabilidad del sistema y permitir un funcionamiento preciso de los servomotores, que serían responsables de inclinar la plataforma y dirigir el movimiento de la pelota.

Evaluamos varias opciones de materiales y diseños, buscando soportes que fueran robustos, fáciles de ensamblar y compatibles con el tamaño y el peso del laberinto. La elección debía considerar no solo la resistencia estructural, sino también la posibilidad de realizar ajustes durante las pruebas, ya que cualquier desalineación podría afectar la eficacia del sistema.

Además, revisamos diferentes métodos de fijación para asegurarnos de que los soportes pudieran mantenerse firmes bajo las cargas dinámicas generadas por los movimientos del laberinto. Esto implicó probar configuraciones con tornillos, adhesivos y piezas impresas en 3D, evaluando cuál ofrecía el mejor balance entre rigidez y flexibilidad.

La búsqueda también incluyó analizar cómo los soportes se integrarían visual y funcionalmente al diseño general, asegurándonos de que no interfirieran con los movimientos del laberinto ni con la estética del proyecto. Una vez definidos los componentes, nos preparamos para la etapa de montaje y pruebas, donde ajustaríamos cada pieza hasta alcanzar la configuración óptima.

Finalmente, encontramos un soporte que se ajustaba perfectamente a nuestras necesidades. Este soporte ofrecía la combinación ideal de estabilidad y facilidad de montaje, además de ser compatible con las dimensiones y el peso del laberinto



Para ensamblar este proyecto, se requiere una lista específica de componentes que aseguren el correcto funcionamiento del sistema. Estos elementos incluyen tanto piezas impresas en 3D. A continuación, se detallan los materiales necesarios:

- **Piezas impresas en 3D:** Estas partes son fundamentales para construir la estructura del laberinto y los soportes de los servos. Su diseño se adapta a las dimensiones requeridas y permite un ensamblaje preciso.
- **Placa UNO R3 (equivalente a Arduino Uno R3):** Este microcontrolador es el cerebro del sistema, encargado de interpretar las señales del joystick y controlar los servos para mover la plataforma del laberinto.

- **2 servos SG90 con brazos y tornillos:** Los micro servos SG90 proporcionan el movimiento necesario para inclinar la plataforma. Su ligereza y precisión los hacen ideales para este proyecto.
- **Joystick KY023:** Este componente actúa como la interfaz de control, permitiendo al usuario inclinar el laberinto en distintas direcciones de manera intuitiva.
- **13 tornillos autoperforantes M2 Phillips:** Estos tornillos se utilizan para ensamblar y fijar las distintas partes impresas en 3D, garantizando una estructura firme y estable.
- **4 cables jumper macho a macho:** Utilizados para conectar los componentes electrónicos dentro del circuito.
- **4 cables jumper macho a hembra:** Sirven para establecer conexiones entre la placa Arduino y otros componentes.
- **Pelota de acero de 1/4”:** Este elemento es el protagonista del laberinto, y su tamaño y peso están diseñados para interactuar adecuadamente con la inclinación de la plataforma.
- **2 remaches de 1/8” x 5/16” (solo las cabezas):** Estas piezas pequeñas son esenciales para crear puntos de pivote en los soportes.
- **2 tornillos para máquina 4-40 x 1/2”:** Utilizados para fijar los servos a la estructura de manera segura.
- **2 tuercas para tornillos 4-40:** Complementan los tornillos para proporcionar una fijación segura y estable.