

# Proyecto Seguidor de Luz

*Alejandro Castaños, Gabriel García, Lucía Morales y Jorge Ramírez*

<b>Planificación Temporal</b>	<b>1</b>
<b>Definición del proyecto</b>	<b>2</b>
<b>Información</b>	<b>2</b>
<b>Diseño</b>	<b>3</b>
<b>Proceso de Construcción</b>	<b>5</b>
<b>Evaluación</b>	<b>8</b>

## Planificación Temporal

Planificación Temporal																		
	Días																	
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Diseño	■	■	■	■	■	■	■	■	■	■	■	■						
Programación								■	■	■	■	■	■	■	■			
Arduino		■	■	■	■	■	■	■										
Construcción													■	■	■	■	■	■

# Definición del proyecto

Este ha sido el proyecto más grande y complejo que hemos tenido este año, ha sido un trabajo muy duro y en donde hemos tenido que utilizar muchas herramientas y hemos aprendido muchísimas cosas nuevas. A primera vista parece un proyecto sencillo en donde tenemos que crear un seguidor de luz funcional, y que gire en 2 direcciones diferentes, arriba-abajo y derecha-izquierda. Pero en verdad, es mucho más complejo que eso. Engloba todo el pensamiento y el trabajo del código y los bocetos con la construcción.

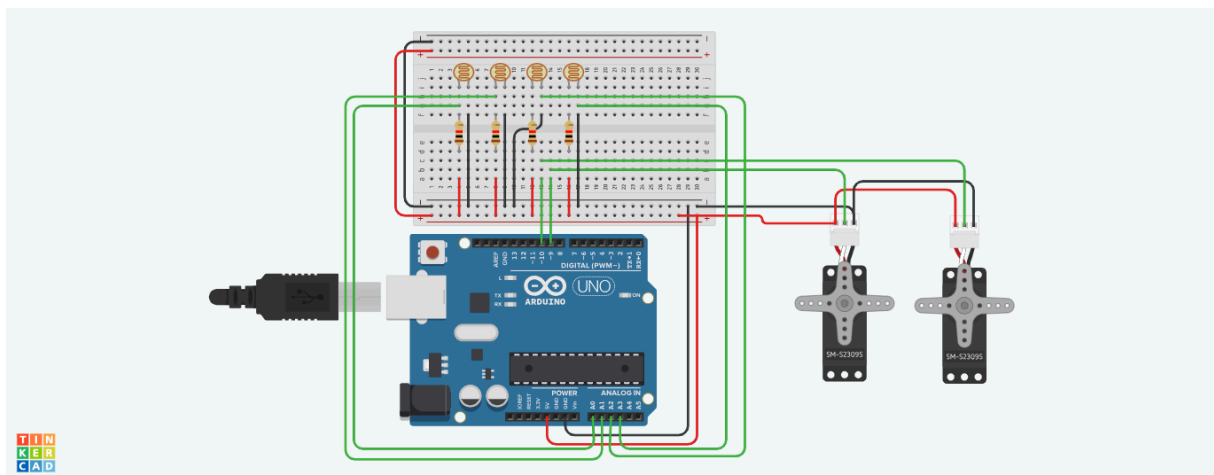
Las condiciones iniciales fueron efímeras, es decir, teníamos solo lo esencial. No sabíamos cómo alcanzar el objetivo final de este proyecto, para ello buscamos información y diferentes trabajos en los que inspirarnos. Se nos dijeron que materiales teníamos que utilizar y luego nosotros nos adaptamos a estos: 4 LDR, 2 servomotores, 1 placa arduino, 1 protoboard y 20 cables macho y 8 cables hembra.

La mayor dificultad del trabajo era entender el mismo, al principio interpretamos que sólo tenías que moverse izquierda y derecha y abajo y arriba, y sin graduación intermedia, pero luego acabamos aprendiendo que no era así y que la graduación era muy importante a la hora de el movimiento de los servomotores.

## Información

Para realizar un trabajo ejemplar nos hemos visto en la obligación de buscar información de aquello que teníamos que hacer. Por eso la ayuda de algunas páginas web era esencial, hemos usado estas (*About Arduino, 2021*) (*Programar En Arduino: Lenguajes, Programas Y Primeros Pasos, 2024*), con la ayuda de estas páginas web, el código fue adelante mejorando de manera muy drástica, usando funciones novedosas como el “**abs**”, que sirve para hacer un valor absoluto de los respectivos valores.

Aparte de toda esta útil información recabada, tuvimos la necesidad de ver trabajos ya hechos como inspiración para crear el nuestro. Para el diseño en físico miramos nuestros trabajos de años anteriores y de este mismo año en donde utilizamos por separado los LDR y los servomotores. Aunque por precaución buscamos algún trabajo con el mismo hardware para revisar simplemente, este fue el que encontramos, que fue creado y diseñado por “Jorge Luis Cornejo Plata”



Para crear el código ya fue más complicado buscar trabajos en los que inspirarse, ya que ninguno tenía exactamente lo que necesitábamos. Por lo que decidimos buscar comandos que nos ayudarían para un mejor resultado, encontramos esta página (*Arduino - Servomotor Con Arduino Tutorial De Programación Paso a Paso, n.d.*) que nos ayuda a entender el funcionamiento del propio servomotor y además nos dice trucos para programarlo de mejor manera.

Para los demás componentes del trabajo no buscamos nada de información ya que sabíamos con certeza cómo se usaba cada uno al haberlo hecho años pasados. Aunque donde sí indagamos un poco fue en el uso de los 4 LDR al mismo tiempo, ya que habíamos utilizado uno pero no más a la vez, por lo que esto podría llevar a un problema mayor más adelante.

En relación a la realización del diseño, como teníamos una vaga idea de lo que queríamos que fuera el resultado final, nos informamos un poco acerca de ello. Primero, investigamos un poco sobre el concepto de seguidor de luz, ya que nos era un poco confuso al no estar seguros de su funcionamiento. Para ello, buscamos una versión completa de esto que nos pudiera servir de inspiración general, (*Building an Automatic Solar Tracker With Arduino Nano V2, n.d.*).

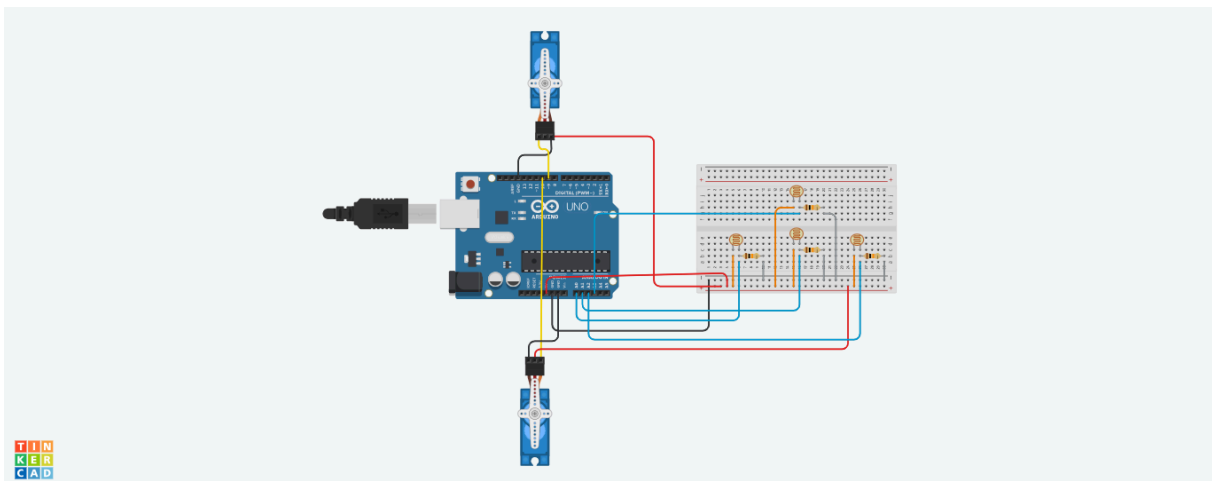


Después de contar con los conocimientos suficientes para entender el concepto de lo que tendríamos que diseñar, comenzamos a pensar en la estética que queríamos que tuviera. Tras un tiempo de reflexión, decidimos que queríamos que se asemeje a un girasol, por su característica forma de seguir el sol. Además de aportar un añadido estético, le da un toque funcional al diseño.

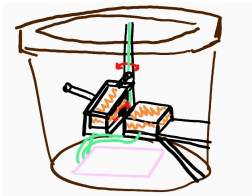
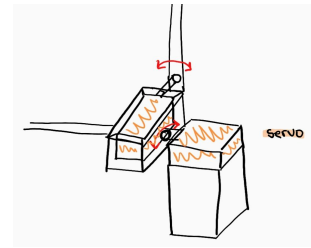
## Diseño

El diseño del proyecto consistía en dos partes, el diseño del hardware y el del software. El equipo se dividió en 2, una pareja haría el software y la otra parte del hardware. Lo primero por lo que teníamos que empezar era el diseño de dónde íbamos a poner las cosas y aunque nos olíamos que íbamos a tener que cambiarlo debido a que la protoboard no va a poder estar dentro de la pieza, lo hicimos igualmente para asegurar que sabíamos como funcionaba y que el futuro código funcionará igualmente

Utilizamos la aplicación de Tinkercad, que ya habíamos utilizado años anteriores y por eso no nos costó nada hacer todo el trabajo, y nuestro primer y único boceto para la parte del hardware es este: <https://www.tinkercad.com/things/gbEUMceJj9Z-copy-of-seguidor-de-luz/editel?returnTo=https%3A%2F%2Fwww.tinkercad.com%2Fdashboar>

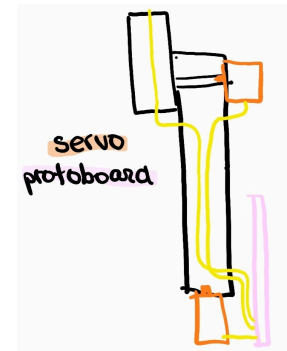


A la hora de crear el diseño físico del seguidor, como ya comentamos, optamos por la estética de una flor, concretamente, un girasol. Antes de ponernos a crear bocetos, aclaramos la forma, sobre qué se sostendría y el funcionamiento de la misma. Primeramente, pensamos en crear una base que tuviera forma de maceta. Esta base guardaría toda la parte funcional, la protoboard, la placa arduino, los cables... Además, la forma de movimiento que tendría sería, como podemos ver en la imagen, un servo sosteniendo al otro servo en el aire, y este segundo servo movería un eje que giraría la flor.



A continuación, adjuntamos un boceto más completo de la idea inicial que tuvimos, mostrando también la base que pensamos que podría sostener todo.

Más tarde, nos dimos cuenta de que de esta forma, el movimiento realizado por la flor no sería creíble, pues esta rotaría izquierda y derecha, pero a la hora de moverse hacia adelante y hacia atrás simplemente se inclinaría, de tal forma que parecería que el tallo se está inclinando. Es por esto que decidimos cambiar de perspectiva, optando por la siguiente idea: habría un servo fijado a la base, este servo, rotaría un eje, llegado a cierta altura, este eje tendría una articulación, donde estaría el segundo servo, que permitiría a la flor (no el tallo) inclinarse hacia adelante y hacia atrás.



Tras esto, comenzamos a diseñar, sin embargo, había algo en la idea pensada que no nos terminaba de convencer. La forma de distribución de los servomotores no nos parecía la ideal. Por eso, investigamos un poquito más a ver si había alguna idea parecida a la nuestra en internet. Tras un poco de búsqueda, dimos con un vídeo de una persona que realizaba un proyecto similar, así que tomamos de inspiración su funcionamiento ya que nos parecía perfecto (CIIDEPT Centro de Innovación, n.d.).

Después de tener claro finalmente la versión final de nuestra idea, pasamos a la práctica y nos pusimos a diseñar en el programa *Fusion360*.

Diseño: Opciones planteadas, discusión y diseño final. Añadir cuantos bocetos, imágenes y diagramas sean necesarios. Debe quedar completamente definido lo que se va a construir.

# Proceso de Construcción

Para la construcción nos dividimos en dos grupos, aquellos que se encargaban del software y del circuito. Para ello nos apoyamos en la aplicación de Tinkercad. Una aplicación de simulación de códigos y de circuitos para hacer todas las pruebas necesarias para llegar a la excelencia.



Con lo mencionado, iniciamos a crear el circuito, en tinkercad para no tener fallos una vez lo recreáramos en la vida real. Con el circuito ya enlazado, empezamos con el programa, usando algunas páginas ya mencionadas. Tras prueba y error dimos con el código correcto. Y empezamos a representarlo en la vida real.

---

```
#include <Servo.h> // Librería para controlar servos

Servo servo1; // Servo para movimiento horizontal
Servo servo2; // Servo para movimiento vertical

// Rango de movimiento de los servos
const int MIN_ANGLE = 0; // Ángulo mínimo para los servos
const int MAX_ANGLE = 180; // Ángulo máximo para los servos

// Valores base para LDR (resistencias dependientes de la luz)
const int LIGHT_THRESHOLD = 50; // Umbral mínimo de diferencia para mover el servo
const int SENSOR_MIN = 0; // Límite mínimo de lectura del LDR
const int SENSOR_MAX = 1023; // Límite máximo de lectura del LDR
const int STEP_SIZE = 4; // Tamaño del paso para movimiento angular

// Posiciones actuales de los servos
int currentAngle1 = 90; // Ángulo inicial del servo horizontal
int currentAngle2 = 90; // Ángulo inicial del servo vertical

void setup() {
  Serial.begin(9600); // Inicializa comunicación serial
  servo1.attach(9); // Conecta el primer servo al pin 9
  servo2.attach(10); // Conecta el segundo servo al pin 10

  pinMode(A0, INPUT); // LDR izquierda (conectado al pin A0)
  pinMode(A1, INPUT); // LDR abajo (conectado al pin A1)
  pinMode(A2, INPUT); // LDR derecha (conectado al pin A2)
  pinMode(A3, INPUT); // LDR arriba (conectado al pin A3)

  // Inicializamos los servos en posición central
  servo1.write(currentAngle1); // Ajusta servo1 (horizontal) al ángulo inicial
  servo2.write(currentAngle2); // Ajusta servo2 (vertical) al ángulo inicial
}

void loop() {
  // Lectura de valores de los LDR
  int izquierda = analogRead(A0); // Lee el valor del LDR izquierdo
  int abajo = analogRead(A1); // Lee el valor del LDR inferior
  int derecha = analogRead(A2); // Lee el valor del LDR derecho
  int arriba = analogRead(A3); // Lee el valor del LDR superior
```

```

// Calcular diferencias de luz entre los LDR
int diferenciaHorizontal = izquierda - derecha; // Diferencia de luz horizontal
int diferenciaVertical = arriba - abajo; // Diferencia de luz vertical

// Movimiento horizontal (servo1)
if (abs(diferenciaHorizontal) > LIGHT_THRESHOLD) { // Si la diferencia de luz es mayor que el umbral
  if (diferenciaHorizontal > 0) { // Si
    currentAngle1 = constrain(currentAngle1 - STEP_SIZE, MIN_ANGLE, MAX_ANGLE); // Mueve a la izquierda
  } else { // Si la luz es mayor a la derecha
    currentAngle1 = constrain(currentAngle1 + STEP_SIZE, MIN_ANGLE, MAX_ANGLE); // Mueve a la derecha
  }
  servo1.write(currentAngle1); // Actualiza la posición del servo1 (horizontal)
}

// Movimiento vertical (servo2)
if (abs(diferenciaVertical) > LIGHT_THRESHOLD) { // Si la diferencia de luz es mayor que el umbral
  if (diferenciaVertical > 0) { // Si la luz es mayor arriba
    currentAngle2 = constrain(currentAngle2 - STEP_SIZE, MIN_ANGLE, MAX_ANGLE); // Mueve hacia arriba
  } else { // Si la luz es mayor abajo
    currentAngle2 = constrain(currentAngle2 + STEP_SIZE, MIN_ANGLE, MAX_ANGLE); // Mueve hacia abajo
  }
  servo2.write(currentAngle2); // Actualiza la posición del servo2 (vertical)
}

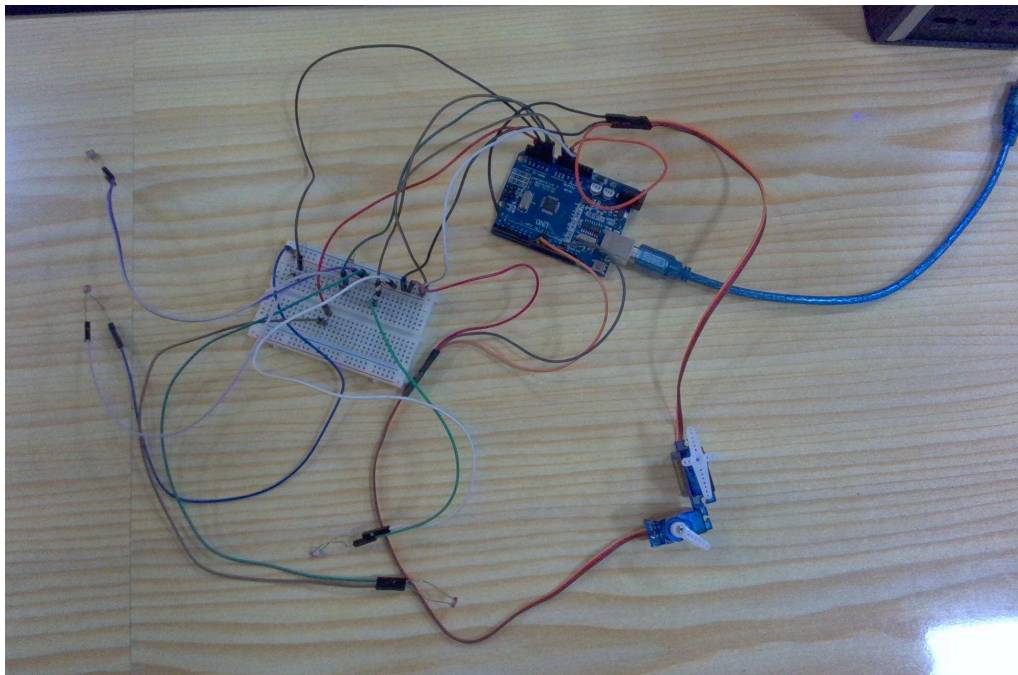
// Mostrar datos por Serial
Serial.print("Izquierda: "); Serial.print(izquierda); // Muestra valor LDR izquierda
Serial.print(" | Derecha: "); Serial.print(derecha); // Muestra valor LDR derecha
Serial.print(" | Arriba: "); Serial.print(arriba); // Muestra valor LDR arriba
Serial.print(" | Abajo: "); Serial.println(abajo); // Muestra valor LDR abajo

// Pausa para estabilizar
delay(50); // Espera 50 ms antes de la siguiente iteración
}

```

---

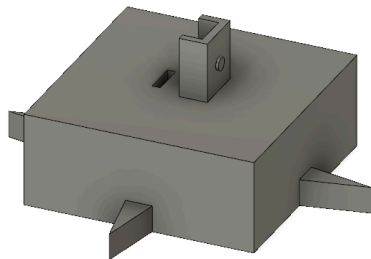
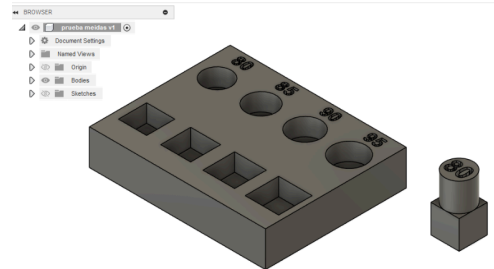
Más tarde, llegó la hora de representar lo que habíamos hecho en el Tinkercad a la realidad. Para ello, disponíamos de una cantidad limitada de materiales, constando de 4 LDR, 2 servomotores, 1 placa arduino, 1 protoboard y 20 cables macho y 8 cables hembra, en el caso de los cables no eran limitados.



Con ya todo el circuito en funcionamiento y sin errores solo quedaba esperar por el hardware, es decir, por las piezas creadas a través del Fusion360.

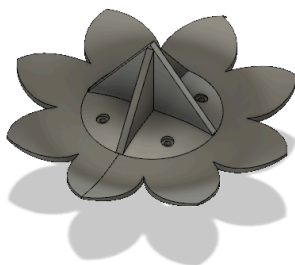
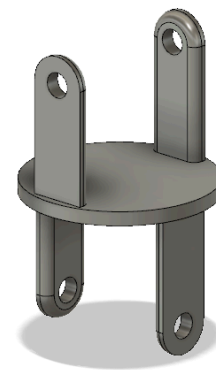
Por otro lado, la otra pareja se encarga del Fusion360, creando un soporte para cumplir el objetivo final, además de hacerlo de manera estética. Para lograr este objetivo se estuvo debatiendo durante un tiempo, llegando a la conclusión final de que la mejor idea para el proyecto sería crear una base de la que saliese un tallo, y al final de este un girasol.

Antes de pasar los bocetos realizados a *Fusion360*, decidimos necesario realizar una prueba para comprobar con certeza la separación que deberíamos dejar entre piezas dependiendo de si queríamos que encajaran o que se deslizaran. Para esto, simplemente creamos una base con huecos de diferentes medidas y una sola pieza de una longitud determinada, usamos esa misma pieza en todos los agujeros para comprobar su comportamiento.



Después de imprimir la pieza anterior y decidir qué medidas usaríamos entre piezas dependiendo de su unión, procedimos a diseñar la base de la pieza. La función de esta estructura sería la de sostener el primero de los servos y servir de apoyo para que el movimiento de la flor no hiciera que se tumbara y cayera, aportando así equilibrio al conjunto. Para ello, creamos una base rectangular con una pata triangular en cada lado para aportar mayor estabilidad. En la parte superior, la pieza tiene una hendidura que sujeta al servo. También, tiene un saliente donde se encaja la segunda pieza, que hará de unión entre la base (el primer servo) y la flor (el segundo servo).

La siguiente pieza, como ya hemos comentado anteriormente, sirve de conexión entre las dos piezas principales, y por tanto, une el movimiento de ambos servos. Esta pieza, aunque simple, es de las más fundamentales, consiste en un círculo del que parten, en uno de los lados, dos patillas con un agujero circular en el extremo cada una (donde va acoplado el servo). Por el otro lado, lo mismo pero en lugar de que las patas estén posicionadas “izquierda-derecha”, están “arriba-abajo”. La separación entre las patillas que van conectadas a la base es ligeramente mayor a la separación de las otras dos ya que a la longitud del servo se le suma el grosor del saliente realizado en la base.

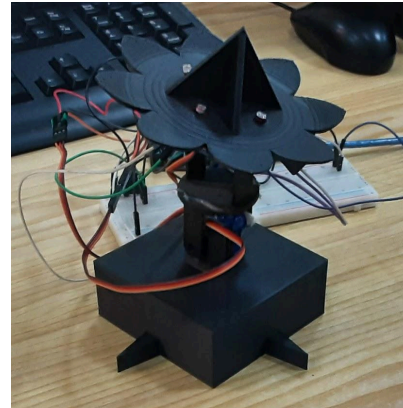


Ya después de esto, realizamos la pieza más compleja, que ya sería la flor. Para ello, lo primero que hicimos fue realizar mediante una extrusión cilíndrica, dando como resultado la forma de la flor pero sin los pétalos hechos, sino que era una circunferencia completa. Después, hicimos la forma de uno de los pétalos y la duplicamos mediante la herramienta *mirror*. Una vez hecho esto, decidimos hacerle una cruz piramidal en la parte del medio para que la luz diera a cada LDR en función de la posición de la luz, de esta forma evitaríamos que la luz incidiera en todos los LDR a la vez. Por último, hicimos cuatro agujeros donde irían insertados los cuatro LDR. Lo último que realizamos fue, en la parte inferior de la pieza, creamos un hueco donde pegaríamos el segundo servomotor.

Después de diseñar todas las piezas necesarias, las fuimos mandando a imprimir. Para ello, empleamos el programa *UltimakerCura*. Gracias a esto, pudimos diseccionar cada pieza en rodajas (“*slice*”) y prepararlas para la impresión.

Una vez impresas las piezas ensamblamos el conjunto y fusionamos ya el circuito con el soporte estético. Como era de esperar, al principio la pieza no funcionaba como esperábamos, no lográbamos colocar bien los LDR así que siempre había alguno de ellos que no recibía señal. Por eso, este fue el primer error que tratamos de corregir, para ello simplemente pusimos cinta aislante en las patas de los LDR para que no se tocaran las patas entre sí.

Una vez logramos que esto no fuera un problema, encontramos otro inconveniente en la pieza que hace de unión entre la base y la flor. De tanto probar el funcionamiento y por tanto insertar y quitar los servos, el agujero donde iba encajado el servo inferior se fue haciendo más grande hasta el punto de que ya no encajaba sino que resbalaba. Por esto es que tuvimos que volver a imprimir esta pieza.



Finalmente, tras varias clases de realizar pruebas, algunos ajustes al código, y modificar algunas partes de las piezas, conseguimos que la pieza se moviera exitosamente. Cabe aclarar que realmente la pieza siempre funcionó de forma correcta, el problema realmente fue que nunca conseguíamos disponerla de manera adecuada, pues como ya comentamos, el principal inconveniente era la disposición de los LDR o que no conseguíamos calibrar bien alguno de los servos, por lo que aunque se moviera perfectamente, llegado a un límite no completaba el movimiento completo.

## Evaluación

Antes de realizar una evaluación completa de todo el trabajo, pondremos a continuación el link para ver el vídeo final donde se muestra la evolución del funcionamiento.

 Girasol.mp4

Gracias a este vídeo vemos que tanto el código como la estructura funcionan, fundiéndose en una combinación perfecta que dan como resultado lo que vemos anteriormente. A pesar de haber encontrado algún que otro impedimento a lo largo del proyecto, hemos observado que hemos trabajado excepcionalmente combinando las habilidades de todos los miembros del equipo. Además, podemos decir que el resultado ha sido más que satisfactorio.

## References

*About Arduino*. (2021, September 15). Arduino. Retrieved February 12, 2025, from

<https://www.arduino.cc/en/about>

*Arduino - Servomotor con Arduino tutorial de programación paso a paso*. (n.d.). Programarfácil.

Retrieved February 13, 2025, from

<https://programarfácil.com/blog/arduino-blog/servomotor-con-arduino/>

*Building an Automatic Solar Tracker With Arduino Nano V2.* (n.d.). Instructables. Retrieved February 18, 2025, from

<https://www.instructables.com/Building-an-Automatic-Solar-Tracker-With-Arduino-N/>

CIIDEPT Centro de Innovación. (n.d.). *TEMA LIBRE CIIDEPT | Lab Robótica y Tec | "GIRASOL ROBÓTICO"*. YouTube. <https://www.youtube.com/watch?v=iinnAbMIW3o>

*Programar en Arduino: lenguajes, programas y primeros pasos.* (2024, June 7). SoftZone. Retrieved February 13, 2025, from <https://www.softzone.es/programas/lenguajes/programar-arduino/>