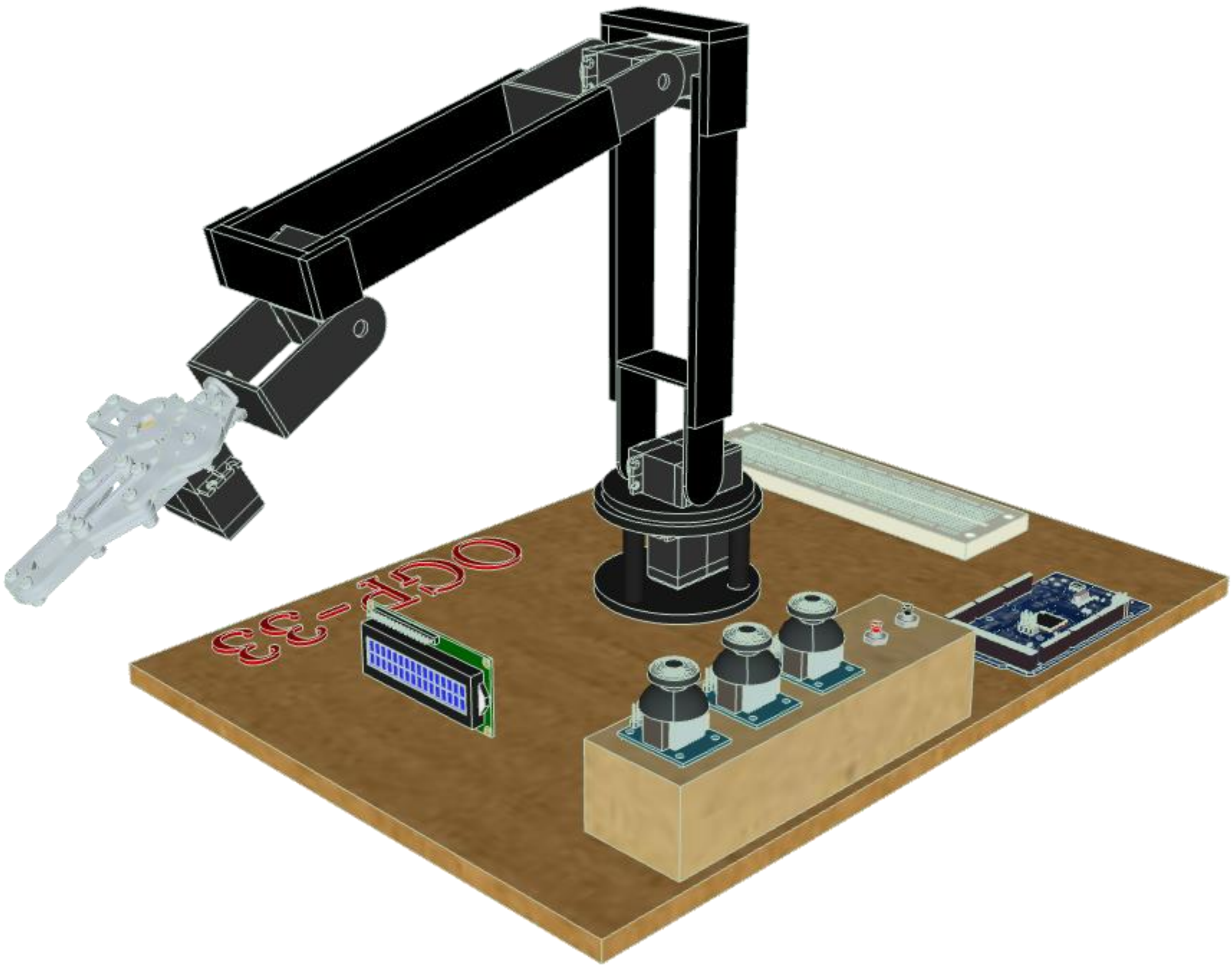
BRAZO ROBÓTICO (OGP-33)

OPERATIONAL GRIPPER POSITIONAL



PROYECTO REALIZADO POR:

ALBERT IZCO, MIGUEL ROCAFULL, PAU PASTOR y ÓSCAR GONZÁLEZ

TUTORIZADO POR:

ELENA GIL y CAROLINA CABALLERO

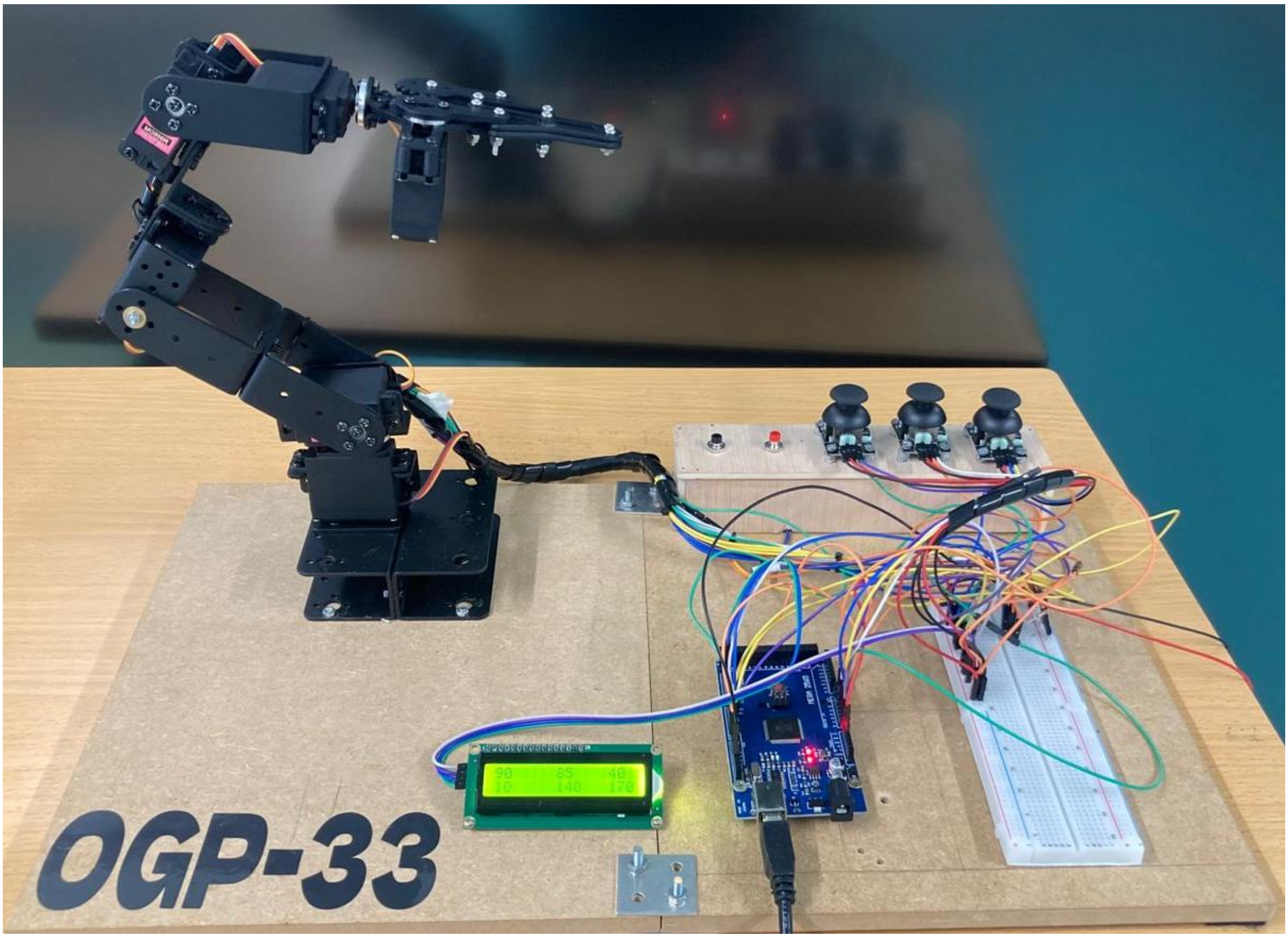
IES LA PATACONA (ALBORAYA-VALENCIA)

ÍNDICE

1. Resumen del proyecto

En este proyecto hemos creado un brazo robótico formado por varios servomotores que se controlan a través de sus correspondientes joysticks. Para programarlo hemos utilizado una placa Arduino Mega, que se adapta adecuadamente a los requisitos del proyecto.

El brazo robótico cuenta con una pinza funcional que es capaz de sostener objetos de unas determinadas dimensiones y transportarlos a otro lugar. Además, la posición de cada uno de los servos y otros datos relevantes del proyecto se verán reflejados en una pantalla LCD. Y para añadirle un punto de automatización tiene integrado una función de grabar y reproducir movimientos durante 20 segundos.



1. Foto final del brazo robótico

2. Introducción

Hoy en día, la robótica industrial desempeña un papel importante en las industrias para conseguir la máxima eficacia, seguridad y ventaja competitiva en el mercado actual, ya que las soluciones de fabricación automatizada son muy útiles.

Una de las soluciones que se han vuelto indispensable es el brazo robot industrial que, con su automatización ha logrado ejecutar diversas tareas sin necesidad de que el capital humano intervenga.

Un brazo robótico industrial es una máquina que está programada para ejecutar un trabajo específico de manera rápida, eficiente y extremadamente precisa.

Actualmente, el brazo robótico industrial se ha vuelto un autómata usual en las líneas de producción de diferentes sectores. Si bien fueron utilizados inicialmente para labores en la industria automotriz, los avances tecnológicos han permitido que otras ramas como la farmacéutica, la alimentaria e incluso la aeroespacial lo adopten ya que pueden automatizar tareas como soldadura, pintura, ensamblaje, entre otros.

El robot manipulador industrial es un gran apoyo para las industrias. Brinda múltiples beneficios como la ejecución de labores de forma automática por medio de su programación, al igual que la reducción de necesidad humana para maximizar la producción.

No obstante, otras de sus ventajas que sobresalen son:

- Seguridad mejorada. El brazo robot ayuda a proteger a los trabajadores al operar en ambientes peligrosos y al efectuar labores que pueden presentar un alto riesgo en lesiones para los mismos.
- Optimización en la eficiencia y productividad. Gracias a su ciclo repetitivo, el brazo robótico industrial puede trabajar 24/7 sin fatigarse, lo cual da en consecuencia; un mantenimiento de la producción, la ejecución de labores continuas y por supuesto, el incremento de ésta.
- Mayor precisión. Por su propia naturaleza, al igual que por los avances tecnológicos de los últimos años, los brazos robóticos se desempeñan de manera más consistente y precisa que los trabajadores en tareas que requieren extrema precisión o consistencia.

Cabe señalar que, el mayor beneficio que puede traer el uso de brazos robóticos industriales es que son reprogramables; es decir, pueden ser calibrados dependiendo del objetivo que se quiera llevar a cabo. Otra gran ventaja es la reducción de costos y su flexibilidad, pues posibilita cambiar su función a otras actividades y permite prescindir de personal para ejecutar otras labores, para ahorrar en tiempo y recursos.

En la actualidad, la clasificación de los brazos robots industriales sobresalen por la forma en que sus articulaciones están diseñadas y por el rango de movimiento y las funciones que pueden realizar.

Brazos robóticos cartesianos: A menudo denominados brazos robóticos rectilíneos o de pórtico, reciben el nombre del sistema de coordenadas cartesianas por René Descartes.

En relación con su movilidad, estos constan de 3 juntas articuladas que se mueven en los ejes X, Y y Z para efectuar su movimiento en 3 dimensiones. Asimismo, poseen una articulación de muñeca que les proporciona una funcionalidad de rotación adicional. Es importante señalar que, los brazos robóticos cartesianos utilizan varios motores y actuadores para colocar algún objeto y manipularlo por medio de una serie de movimientos lineales. [1]

Debido a las enormes ventajas que un brazo robótico aporta a la automatización industrial, hemos decidido diseñar, simular, construir y programar un brazo robot como proyecto final de 1º de bachillerato.

A lo largo de toda la etapa de la ESO hemos ido adquiriendo e incrementando nuestros conocimientos acerca de la programación por bloques haciendo, entre otros, proyectos como un robot saca-latas, un cubo de leds y un vúmetro. En todos los casos hemos diseñado nuestro proyecto con SketchUp y después lo hemos simulado con Tinkercad para, finalmente, programarlo con mBlock y subir el programa a la placa desde ArduinoIDE. Este curso (1º de bachillerato), sin embargo, hemos empezado a programar por código, lo que nos da más libertad y nos ofrece infinidad de posibilidades para crear y mejorar otros proyectos más complejos. En primer lugar, y antes de trabajar en el proyecto del curso, hemos aprendido a conectar y programar cada uno de los componentes por separado.

Nuestro proyecto final de 1º de bachillerato consistía en diseñar, construir y programar una casa domótica controlada mediante un mando de infrarrojos. Al ver que concluimos el proyecto con éxito pensamos que podíamos llegar más lejos y crear un proyecto que realmente tuviera una utilidad aplicable a nuestro día a día y que estuviera directamente relacionado con nuestra asignatura, Tecnología e Ingeniería I. Habíamos visto en el taller un prototipo que empezaron a construir los alumnos del curso pasado y que por falta de tiempo quedó inacabado, y nos llamó la atención, por lo que nos decantamos por la opción de diseñar, simular, construir y programar un brazo robótico con servomotores que pueda controlarse mediante joysticks.

Como hemos comentado antes, al inicio de este curso estudiamos los diferentes componentes y hemos trabajado con ellos haciendo simulaciones y programas simples para ver su funcionamiento de manera individual. Por este motivo, a la hora de empezar a trabajar en nuestro proyecto del brazo robot ya teníamos una idea de cómo comenzar a construirlo y programarlo sin dudar mucho.

Cuando hicimos una búsqueda para comprobar el diseño y funcionamiento de otros proyectos parecidos, vimos que en la mayoría de los casos se trataba de modelos más simples. Para añadir más complejidad al proyecto decidimos añadir a nuestro brazo robot una pantalla LCD que muestre los datos sobre el funcionamiento del brazo robótico. Además, decidimos cambiar nuestro diseño inicial y reducimos el número de joysticks de 5 a 3, pues nos dimos cuenta de que no resultaban del todo útiles y así limitábamos la cantidad de recursos que necesitábamos. Una vez superada esta primera ampliación del proyecto básico, la parte inicial del reto, quisimos añadir una función extra que nos permitirá grabar los movimientos del brazo presionando un pulsador (rojo) para después reproducirlos al pulsar otro pulsador (negro).

Las prácticas iniciales en las que trabajamos en clase, previamente a la realización de nuestro proyecto del brazo robot fueron las siguientes:

PRÁCTICAS CON PULSADORES

La resistencia Pull-Up:

- Pin con valor HIGH cuando el pulsador está abierto (Pull-Up significa tirar hacia arriba). Vemos que el pin digital va al valor alto.
- Pin con valor LOW cuando el pulsador está cerrado

Cuando está cerrado, el PIN se pone a LOW, ya que la corriente tiene dos caminos, uno fácil a tierra (0V) y el otro tiene una resistencia que limita el paso de la corriente. La corriente elige el camino más fácil, sin resistencia, es decir, circularía entre el pin digital y tierra.

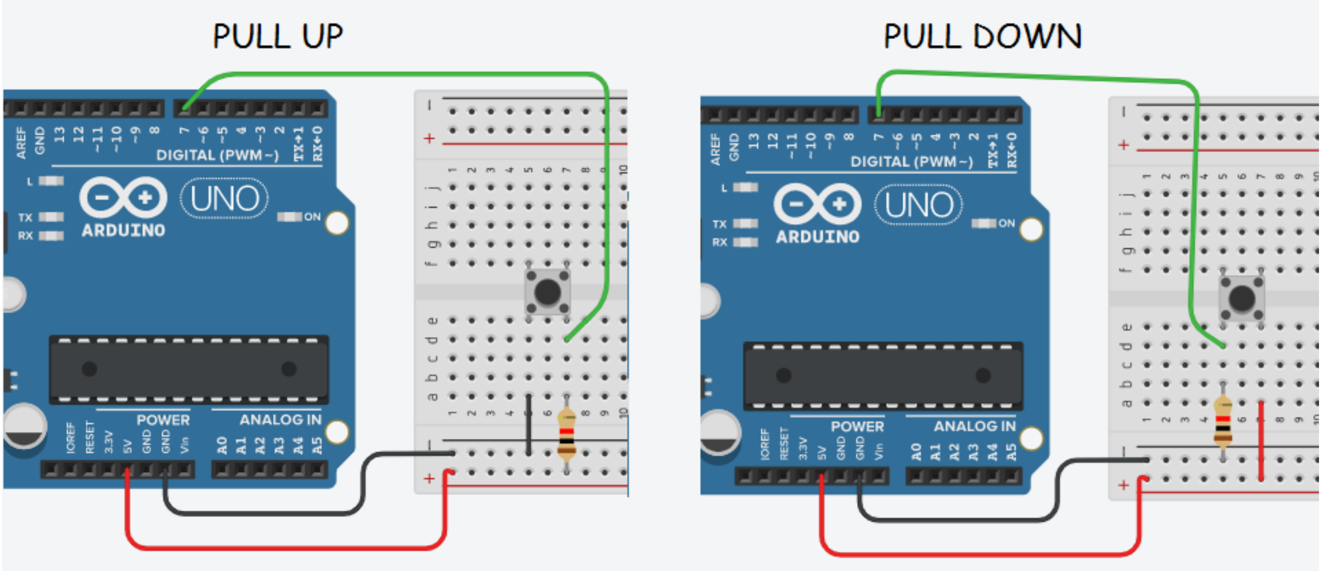
Esta es la configuración recomendada cuando queremos tener un valor LOW al presionar el pulsador y un valor HIGH al dejar de presionar el pulsador.

La resistencia Pull-Down:

- Pin con valor LOW cuando el pulsador está abierto (Pull-Down significa tirar hacia abajo).
- Pin con valor HIGH cuando el pulsador está cerrado.

Cuando está cerrado, el PIN se pone a HIGH, ya que la corriente elige el camino más fácil y entre el pin y tierra hay una resistencia. Entre el pin y 5 V la corriente circula libremente al no haber resistencia.

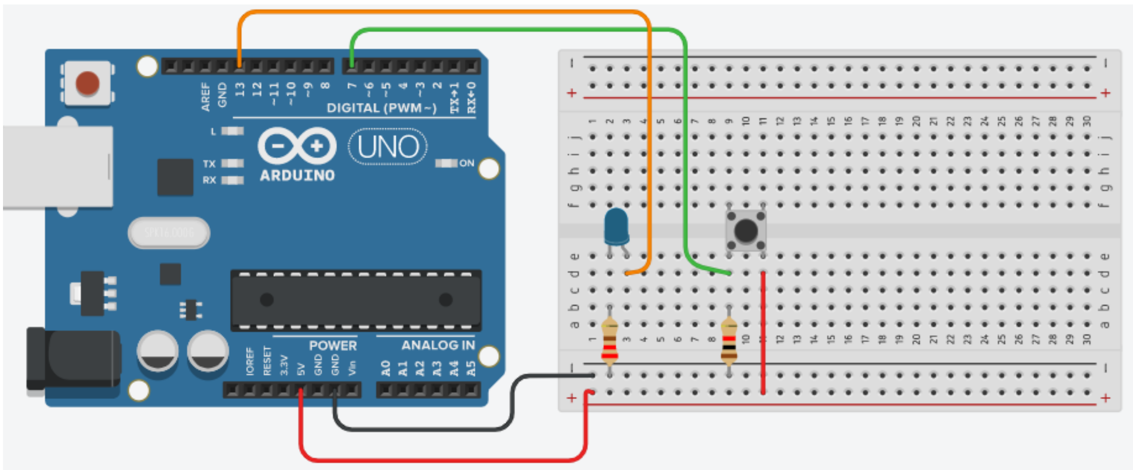
Esta es la configuración recomendada cuando queremos tener un valor HIGH al presionar el pulsador y un valor LOW al dejar de presionar el pulsador, que es el uso más común.



2. Configuración Pull-Up y Pull-Down en Tinkercad

PRÁCTICA: encender un led con pulsador Pull-Down

a) CIRCUITO: LED CON PULSADOR (PULL-DOWN): Rled: 220Ω; Rboton: 1kΩ



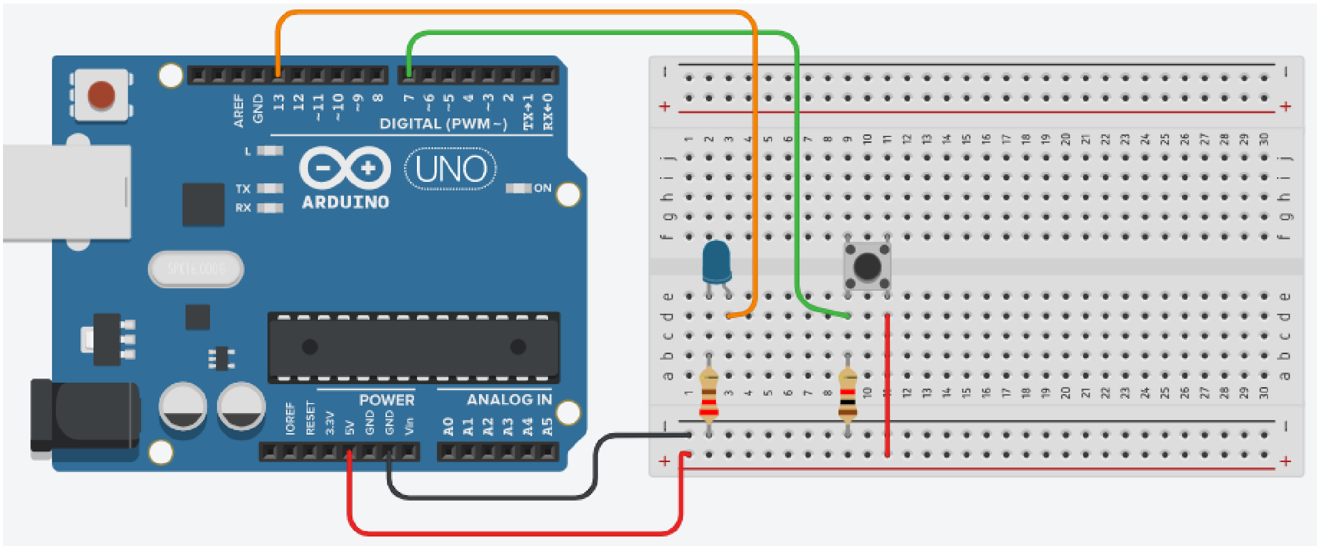
3. Controlar un LED con un pulsador Pull-Down en Tinkercad

b) PROGRAMA: Se enciende el led al pulsar y se apaga al dejar de pulsar.

```
   int LED =13;
   int BOTON = 7;
void setup() {
   pinMode(LED, OUTPUT);
   pinMode(BOTON, INPUT);
  }
void loop() {
   if (digitalRead(BOTON) == HIGH) {
     digitalWrite(LED, HIGH);
   }
   else {
     digitalWrite(LED, LOW);
   }
  }
```

PRÁCTICA: encender y apagar un led con el mismo pulsador

a) CIRCUITO:



4. Encender y apagar un led con el mismo pulsador en Tinkercad

b) PROGRAMA: Se enciende el led con un pulsador y se apaga con el mismo pulsador.

```
   int LED_pin = 13 ;
   int BOTON_pin = 7 ;
   bool estado_actual = 0 ; //Variable que guarda la situación del pulsador o botón
   bool estado_anterior = 0 ; //Situación anterior del pulsador o botón
void setup() {
    pinMode(BOTON_pin, INPUT);
    pinMode(LED_pin, OUTPUT);
    }
```

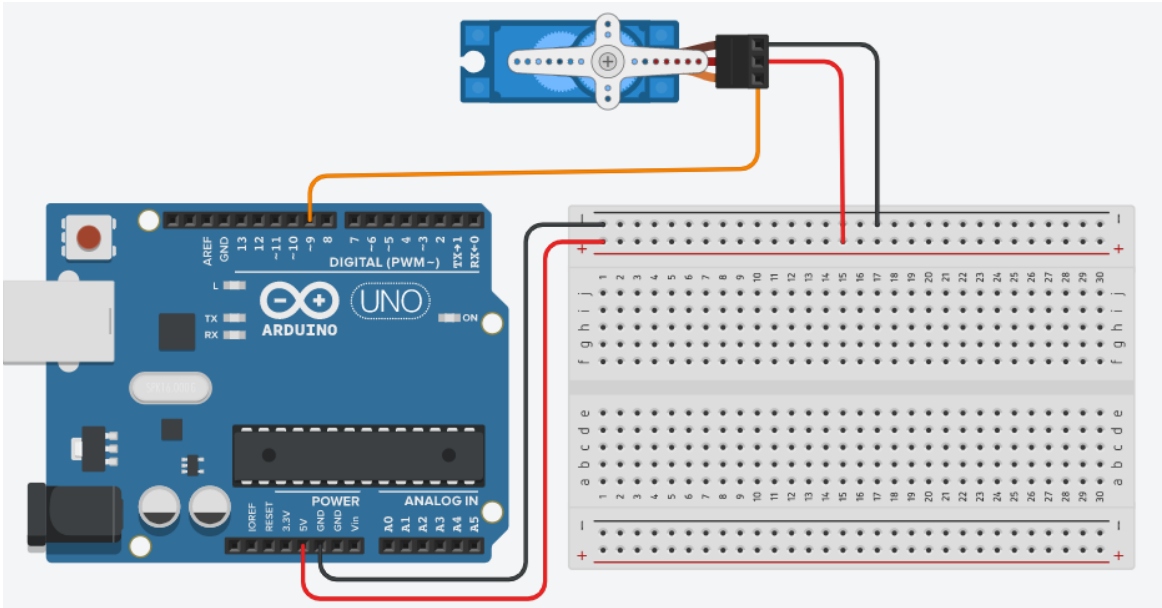
```
void loop() {
     estado_actual = digitalRead(BOTON_pin); //es 1 al pulsar el botón y 0 si no se pulsa
     if (estado_actual != estado_anterior)  //Para comprobar si dos valores son diferentes usamos !=
//Se cumple la condición en dos casos:
//a)Si estado_actual=1 (pulsado) y estado_anterior=0 (no pulsado), se cumple la condición (diferentes)
//b)Si estado_actual=0 (no pulsado) y estado_anterior=1 (pulsado), se cumple la condición (diferentes)
                    {
                      if (estado_actual == 1) {          //En el caso (a ) de arriba
                           digitalWrite(LED_pin, !digitalRead(LED_pin)); //cambia el estado en el led
                           delay (20);
                      }
                      estado_anterior = estado_actual ; // Guarda el estado_actual
                                                 // Si es 1, el estado_anterior cambia a 1
                                                 // Si es 0, el estado_anterior cambia a 0
                    }
              }
```

PRÁCTICAS CON SERVOMOTORES

En otro apartado de nuestro cuaderno de robótica estudiamos cómo programar servomotores, tanto continuos como posicionales. Internamente un servo está constituido por un motor de corriente continua, acoplado a un reductor para reducir la velocidad de giro, junto con la electrónica necesaria para controlar su posición (servos de 180º) o para regular su velocidad (servos de 360º o giro continuo). Los servos admiten una tensión de alimentación entre 4,8V a 7,2V, siendo el valor más adecuado el de 6V. Con tensiones inferiores el motor tiene menos fuerza y velocidad. Con tensiones superiores a 6,5V los servos empiezan a vibrar demasiado, lo cual los hace poco útiles.

PRÁCTICA: Control de un servo mediante diferentes instrucciones.

a) CIRCUITO:



5. Control de un servo en Tinkercad mediante diferentes instrucciones

Color de los cables:

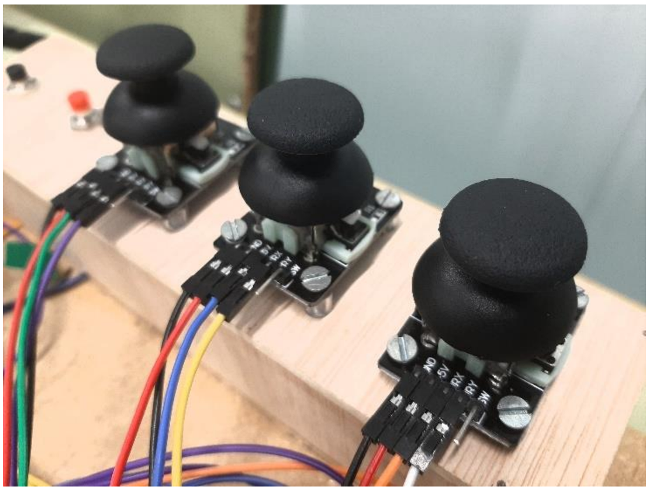
- Señal (a cualquier pin digital de Arduino): puede ser amarillo, naranja o blanco
- Vcc: es rojo
- GND: puede ser negro o marrón

b) PROGRAMA:

<table>
<tr><td>-Programa 1: Mueve el servo una única vez desde 0 a 180 º para volver enseguida a 0º. NOTA: debéis modificar int time</td><td rowspan="2">Responde:
a) El eje del servo necesitará un tiempo para llegar de 0 a 180º (delay).
El tiempo dado de 500 ms no será suficiente para ello. Debéis averiguar que valor en ms es el adecuado.
Tiempo: ☐ ms
b) Se ha puesto el servo en un pin PWM. Prueba a ponerlo en un pin digital (no PWM), cambia el pin en myservo.attach( ); y di si funciona: ☐</td></tr>
<tr><td><pre>
#include <Servo.h>
  Servo myservo; // crea el objeto servo
  int time = 500; //variable tiempo
void setup() {
  myservo.attach(9);
  myservo.write(0);
  delay(100); //le damos un tiempo pequeño para que vaya a 0º
  myservo.write(180); //va a la posición 180º
  delay(time); //tiempo para llegar a la posición 180º
  myservo.write(0); //vuelve a la posición 0º
  delay(time); //tiempo para llegar a la posición 0º
  }
void loop() { }
</pre></td></tr>
</table>

PRÁCTICAS CON JOYSTICK (Y SERVOS)

Un joystick es un componente analógico de entrada. Lleva un mando que se mueve en dos ejes o direcciones perpendiculares (X e Y) y que actúa sobre dos potenciómetros, uno para el eje X y otro para el eje Y. En la placa de Arduino se genera una señal proporcional a la posición de cada uno de los potenciómetros. Suele incluir un pulsador que se activa al presionar el mando hacia abajo.

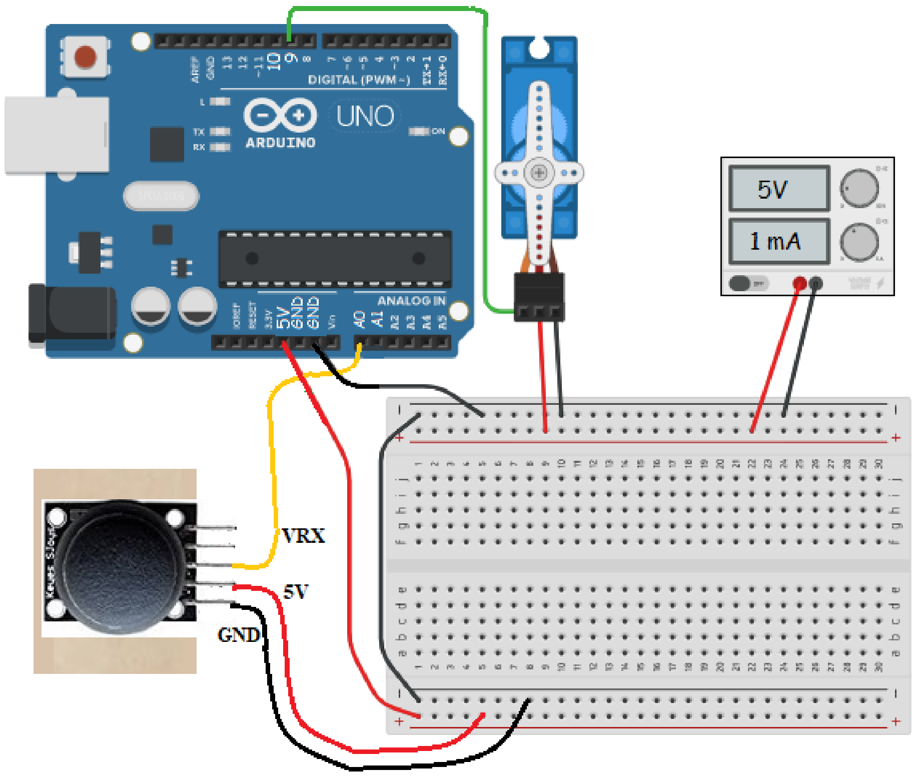


6. Joysticks del proyecto del brazo robótico

Así pues, suelen tener 5 pines: X, Y, botón y Vcc (5V) más GND

Los servos suelen funcionar con voltaje de 5 V aproximadamente, pero hace falta un amperaje de 500 mA por cada servo y la placa de Arduino solo saca 500 mA. Por ello, cuando manejamos más de un servo hay que utilizar una fuente de alimentación externa, como pilas o cargador de móvil que ofrezca suficiente amperaje. Hay que tener en cuenta que para que los servos funcionen será necesario conectar el negativo también a GND de la placa de Arduino.

Además, los servos deben ir alimentados con la fuente de alimentación externa y los otros componentes con el Vcc de la placa de Arduino (u otra alimentación diferente a los servos). Si no se hace así se producirán corrientes parásitas que harán girar a los servos a un lado y otro de forma incontrolada.



7. Conexión y alimentación de servos y joysticks en Tinkercad

Un servo para ser controlado necesita una salida PWM. Sin embargo, la librería servo nos permite utilizar cualquier pin de Arduino para controlar el servo, sin necesitar que sea PWM. Además, al utilizar la librería servo deberemos tener en cuenta, que ésta desactiva los pines 9 y 10 como salida PWM, en Arduino Uno, aunque se pueden seguir usando como pines digitales.

PROGRAMA:

```
#include <Servo.h> // Incluir la librería Servo
Servo pinza; // Crear un objeto tipo Servo
int joyX = A0; // Potenciómetro en eje X conectar al pin analógico 0
int valor_joyX=0; // Variable para leer el valor del pin analógico A0
```

//Continúa en pag. siguiente

```
void setup(){
    pinza.attach(9); // Conectar servo pinza al pin 9
    pinza.write(90); // Posición inicial del Servo
  }
void loop(){
    valor_joyX = analogRead(joyX); // lee valor de pot. eje X (entre 0 y 1023) y lo almacena
    valor_joyX = map(valor_joyX, 0, 1023, 0, 180); //escala el valor para uso en servo (entre 0 y 180)
    pinza.write(valor_joyX); // fija la posición del servo de acuerdo al valor escalado
    delay(40); // espera a que el servo se posicione
    }
```

PANTALLA LCD ARDUINO

Una pantalla LCD I2C tiene solo 4 pines que la conectan con el mundo exterior. Las conexiones son las siguientes:

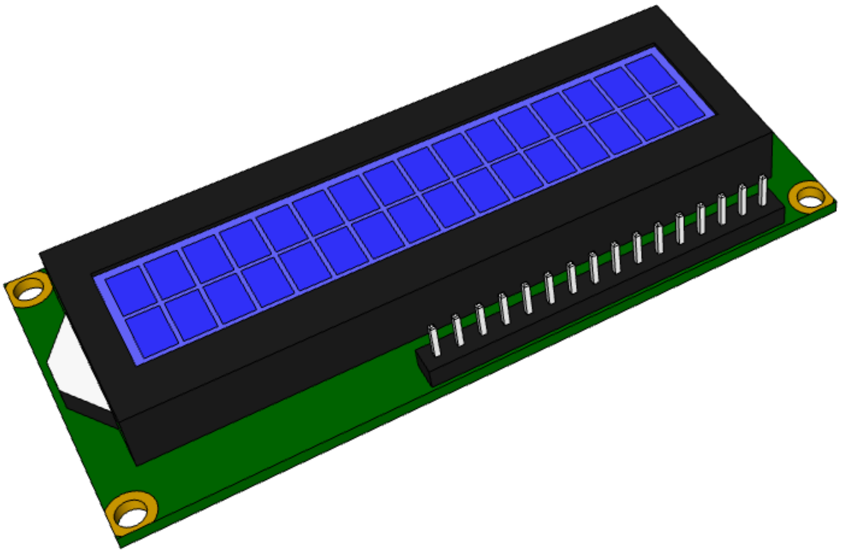
GND es un pin de tierra. Conéctalo a tierra del Arduino.

VCC suministra energía al módulo y al LCD. Conéctalo a la salida de 5V de Arduino o a una fuente de alimentación externa de 5V.

SDA es el pin de datos I2C. Conéctalo al pin de datos I2C de Arduino.

SCL es el pin del reloj I2C. Conéctalo al pin de reloj I2C de Arduino.

Conectar una pantalla LCD I2C es mucho más fácil que conectar una pantalla LCD estándar. Solo necesita conectar 4 pines en lugar de 12. Comienza conectando el pin VCC a la salida de 5V en Arduino y GND a tierra.



8. Pantalla LCD (fuente SketchUp)

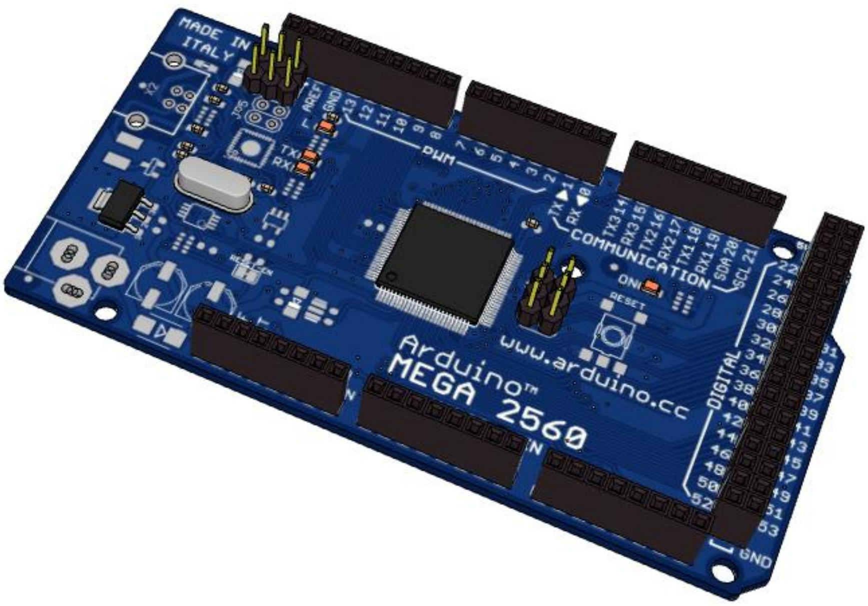
Ahora nos quedan los pines que se utilizan para la comunicación I2C. Ten en cuenta que cada placa Arduino tiene diferentes pines I2C que deben conectarse en consecuencia.

	SCL	ASD
arduino uno	A5	A4
arduino nano	A5	A4
arduino mega	21	20
Leonardo/Micro	3	2

Después de cablear la pantalla LCD, deberás ajustar el contraste de la pantalla. En el módulo I2C encontrarás un potenciómetro que puedes girar con un pequeño destornillador. Conecta el USB de Arduino para alimentar la pantalla LCD. Verás que se enciende la luz de fondo. Ahora, al girar la perilla del potenciómetro, comenzarás a ver la primera fila de rectángulos. Si eso sucede, ¡Felicidades! tu LCD está funcionando bien.

PLACA ARDUINO MEGA

Respecto a la placa que utilizamos para el proyecto usaremos una placa Arduino Mega 2560 tiene 54 pines de entrada/salida, de los que 14 pueden ser utilizados como salidas de PWM (Modulación por ancho de pulso), cuenta con 16 entradas analógicas. Tiene mayor memoria que la Arduino Uno y que la Leonardo, es por eso por lo que utilizaremos esta placa.



9. Placa Arduino Mega (fuente SketchUp)

3. Objetivos

A priori, el objetivo de este proyecto consistía en facilitarle las tareas cotidianas a personas que carecían de la extremidad del brazo a modo de prótesis. Sin embargo, a lo largo del desarrollo del proyecto le vimos algunos inconvenientes a este prototipo para lograr realizar todas las acciones de la extremidad con total normalidad. Es por ello por lo que pensamos que podría tener una función más concreta y útil a la hora de emplearse en la automatización de los procesos de fabricación. Este fin se ajusta a las características de los objetivos de desarrollo sostenible de la asamblea general de la ONU, con 17 objetivos previstos para 2030.

10. Objetivos de Desarrollo Sostenible (fuente www.un.org)

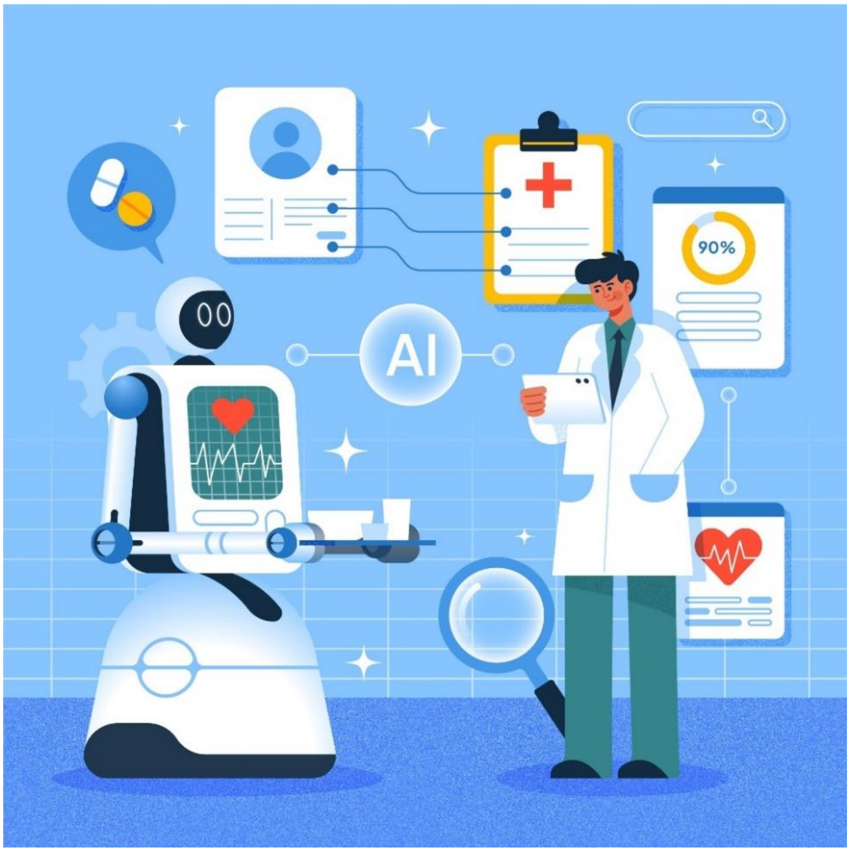
De los ODS adoptados por la Asamblea, en este proyecto nos centramos en cumplir los siguientes:

ODS 3: Salud y Bienestar:

Muchas ocupaciones implican movimientos repetitivos, que suponen para los trabajadores un mayor riesgo de lesiones por este tipo de movimientos. Entre las tareas que requieren movimientos repetitivos se cuentan clavar clavos, trabajar en una línea de ensamblado y usar un martillo neumático. Las lesiones por movimientos repetitivos son la tendinitis, la bursitis y la compresión de un nervio.

11. ODS3: Salud y Bienestar (fuente Freepik)

Por otro lado, en el ámbito médico, los brazos robóticos se utilizan en cirugías y rehabilitaciones, mejorando la precisión y reduciendo los riesgos asociados a los procedimientos médicos.



12. ODS3: Salud y Bienestar (fuente Freepik)

ODS 4: Educación de Calidad:

En la educación, tal y como ocurre en nuestro caso, los brazos robóticos pueden ser utilizados como herramientas didácticas para enseñar a estudiantes sobre robótica, ingeniería y programación.

13. ODS4: Educación de Calidad (fuente Freepik)

ODS 7: Energía Asequible y No Contaminante:

En la producción de energía renovable, los brazos robóticos pueden ayudar en la fabricación y mantenimiento de componentes de energía solar y eólica, promoviendo fuentes de energía más limpias. Un ejemplo sería las líneas de producción con brazos robóticos para ensamblar paneles solares.

14. ODS7: Energía Asequible y No Contaminante (fuente Freepik)

ODS 8: Trabajo Decente y Crecimiento Económico:

Un brazo robótico puede aumentar la productividad y eficiencia en la industria, contribuyendo al crecimiento económico y a la creación de empleo en sectores tecnológicos.

15. ODS 8: Trabajo Decente y Crecimiento Económico (fuente Freepik)

ODS 9: Industria, Innovación e Infraestructura:

Los brazos robóticos son fundamentales en la modernización de la infraestructura industrial y en la promoción de la innovación tecnológica.

16. ODS 9: Industria, Innovación e Infraestructura (fuente Freepik)

ODS 11: Ciudades y Comunidades Sostenibles:

Los brazos robóticos pueden desempeñar un papel en la construcción de infraestructuras más sostenibles y eficientes, contribuyendo al desarrollo de ciudades inteligentes.

17. ODS 11: Ciudades y Comunidades Sostenibles (fuente Freepik)

ODS 12: Producción y Consumo Responsables:

La automatización con brazos robóticos puede optimizar los procesos de producción, reduciendo el desperdicio y mejorando el uso eficiente de los recursos.

18. ODS 12: Producción y Consumo Responsables (fuente Freepik)

4. Materiales y metodología

Materiales

Los materiales que hemos utilizado han sido decididos tras varias pruebas, errores y mejoras que le hemos aplicado a nuestro brazo.

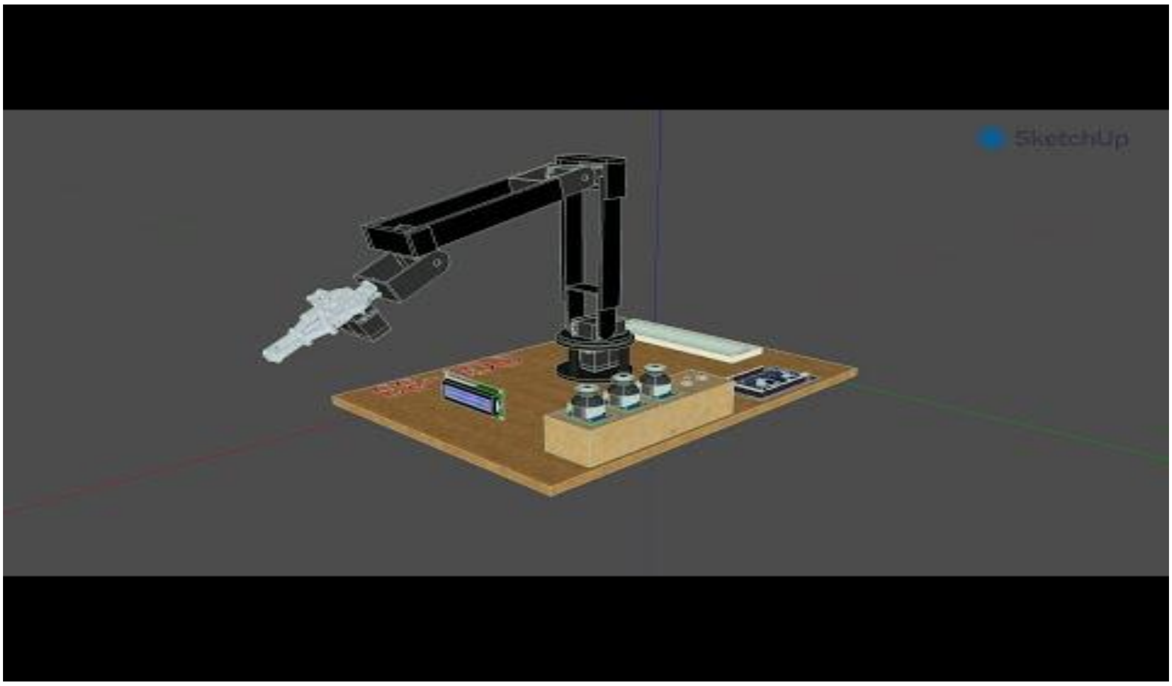
Material	Cantidad	Precio unitario	Precio total
Servomotores MG996R 180º	6		
Joysticks	3	2.75 €	8.25€
"Caja" con madera de contrachapado 20 x 5 x 4 cm	20x20x0.5 cm	9.80 €/m^2	0.39€
Arduino Mega	1	42 €	42 €
Pantalla LCD	1	12€	12€
Pulsador	2	1.75€	3.5€
Pinza y estructura del brazo robot (incluye los servomotores)	1	95€	95€
Placa de pruebas protoboard	1	8€	8€
Base de madera DM de dimensiones 30 x 24 x 1cm	2	3.04 €/m^2	0.96€
TOTAL		**170,1€**	

Tabla de presupuesto: elaboración propia

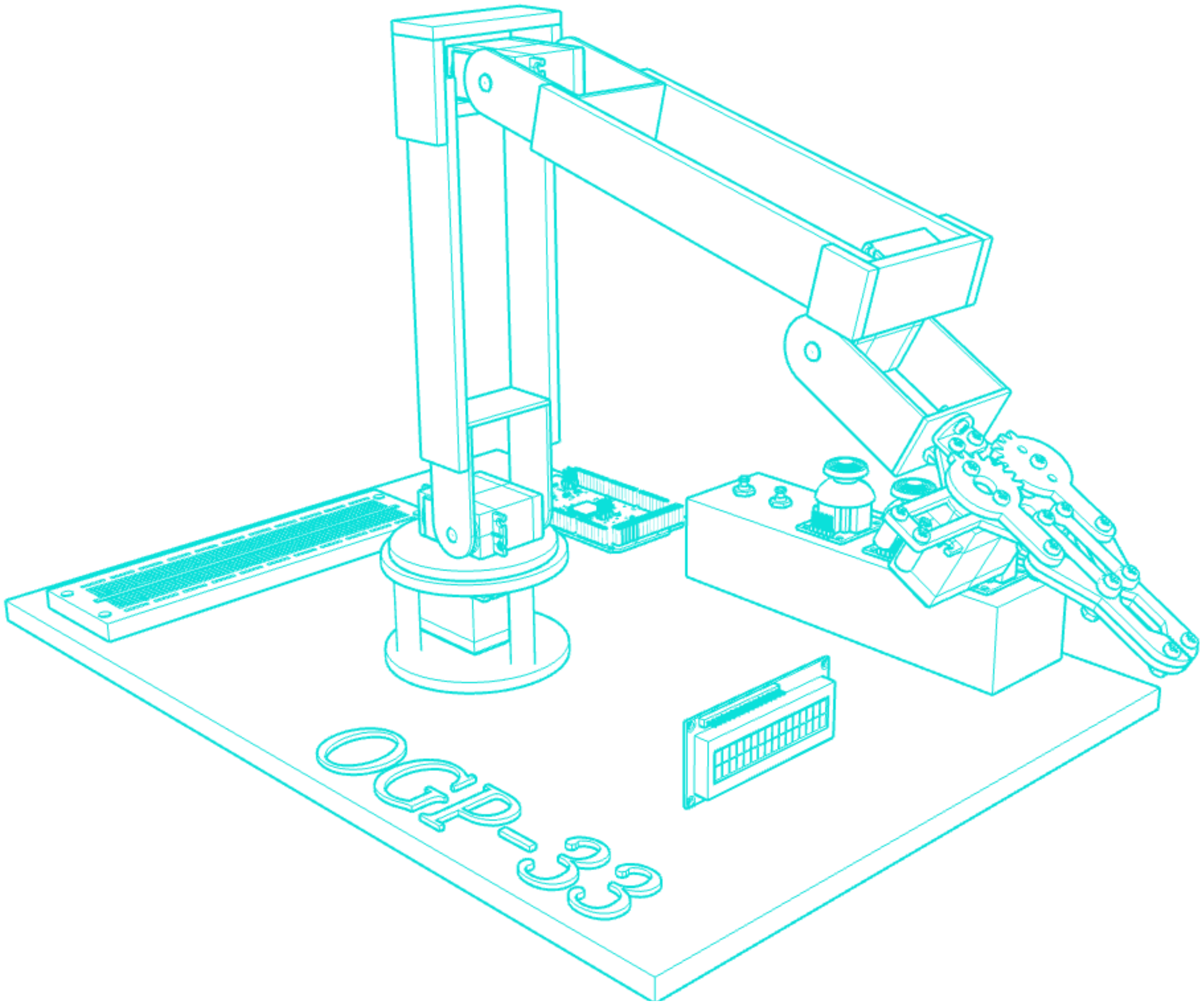
Diseño del proyecto con SketchUp

En primer lugar, diseñamos el proyecto con Sketch-Up para hacer un prototipo inicial del brazo robótico. Este primer diseño nos permitió empezar a construir el brazo y hacer una lista inicial de los materiales que necesitábamos para hacerlo. Decidimos que íbamos a necesitar 6 servomotores: 6 servos posicionales y 1 servo continuo que utilizaríamos para la base del brazo, tres joysticks, dos pulsadores, una placa Arduino Mega y una placa protoboard.

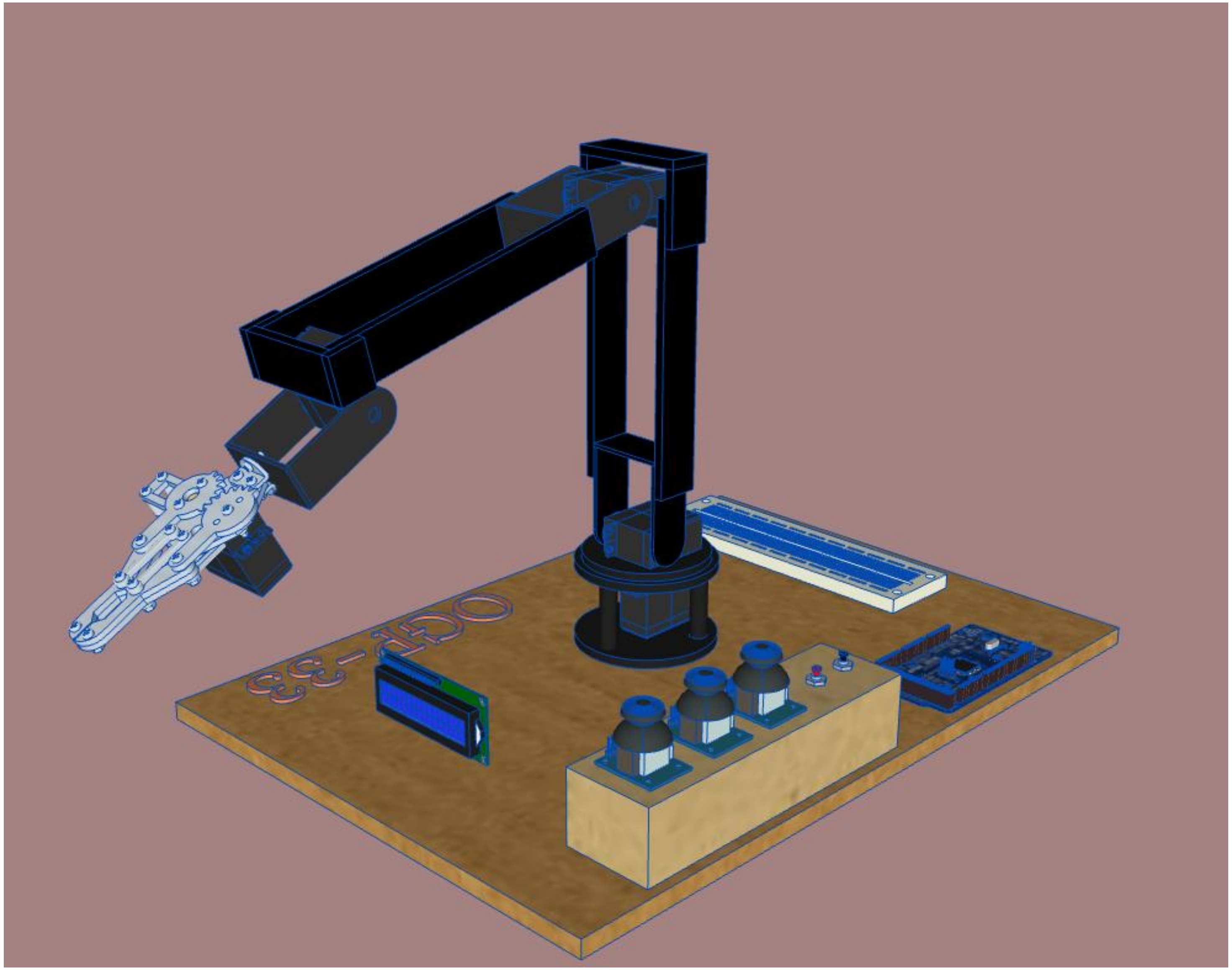
https://youtu.be/TkdRUkQB8Xw



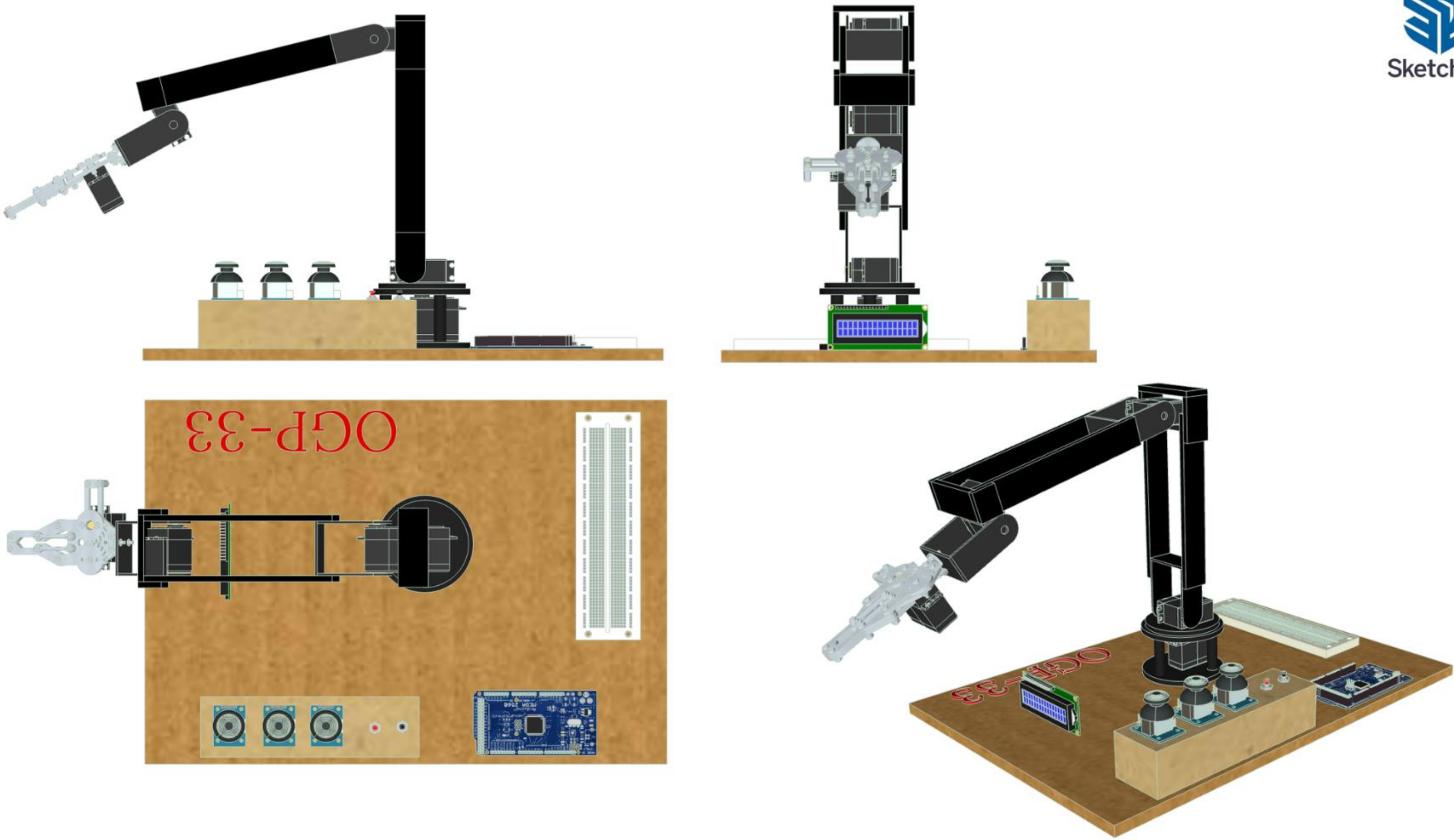
SketchUp



19. Diseño del brazo robótico con SketchUp-1



20. Diseño del brazo robótico con SketchUp-2

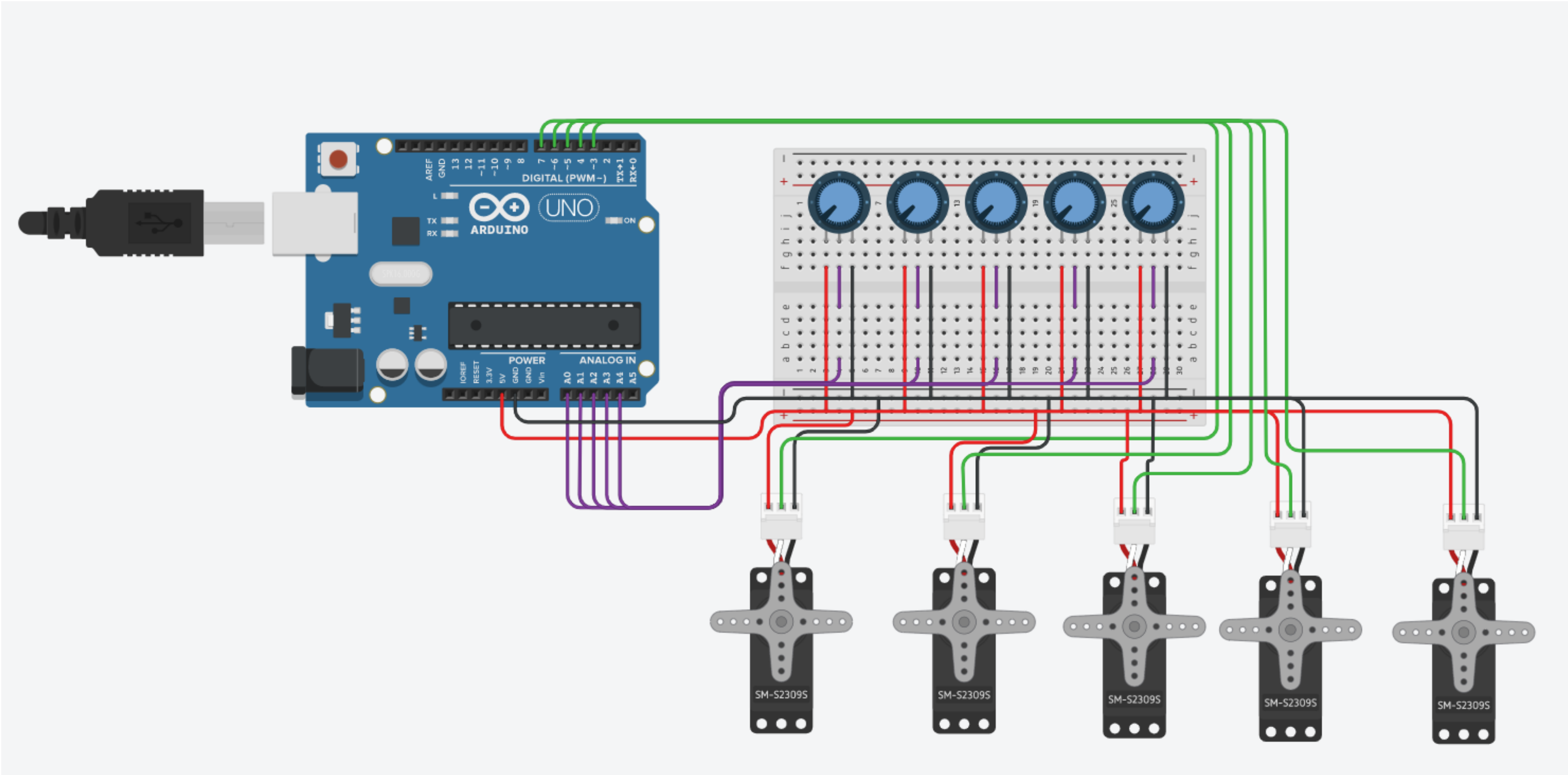


21. Vistas y perspectiva del brazo robótico en SketchUp

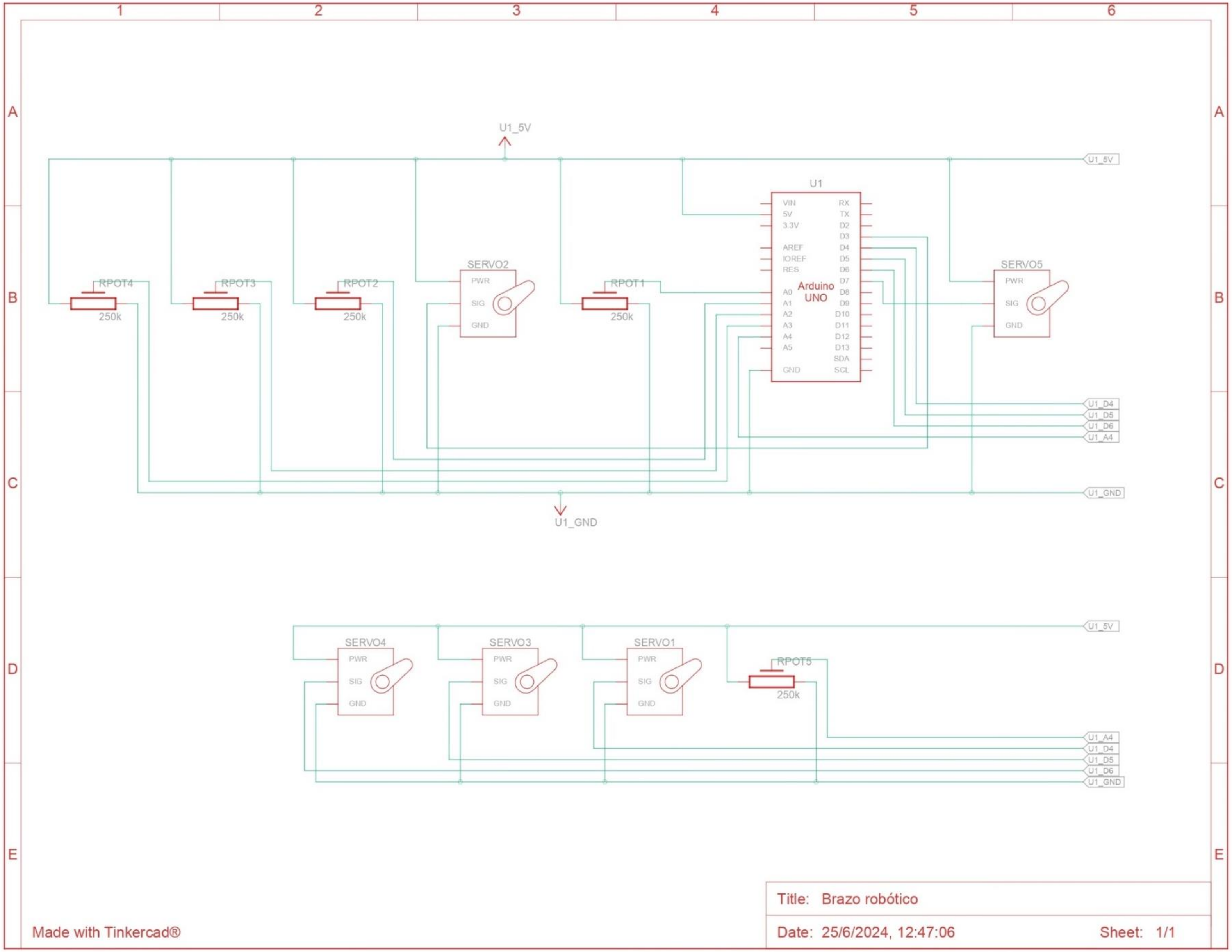
Simulación del programa

A continuación, para diseñar el circuito del brazo utilizamos la herramienta de TinkerCad, de modo que pudimos realizar simulaciones y probar el código inicial del proyecto.

La primera simulación que realizamos fue una básica, que permita controlar los servos mediante potenciómetros que hacen la función de los joysticks, ya que en el programa (Tinkercad) no cuenta con estos elementos.

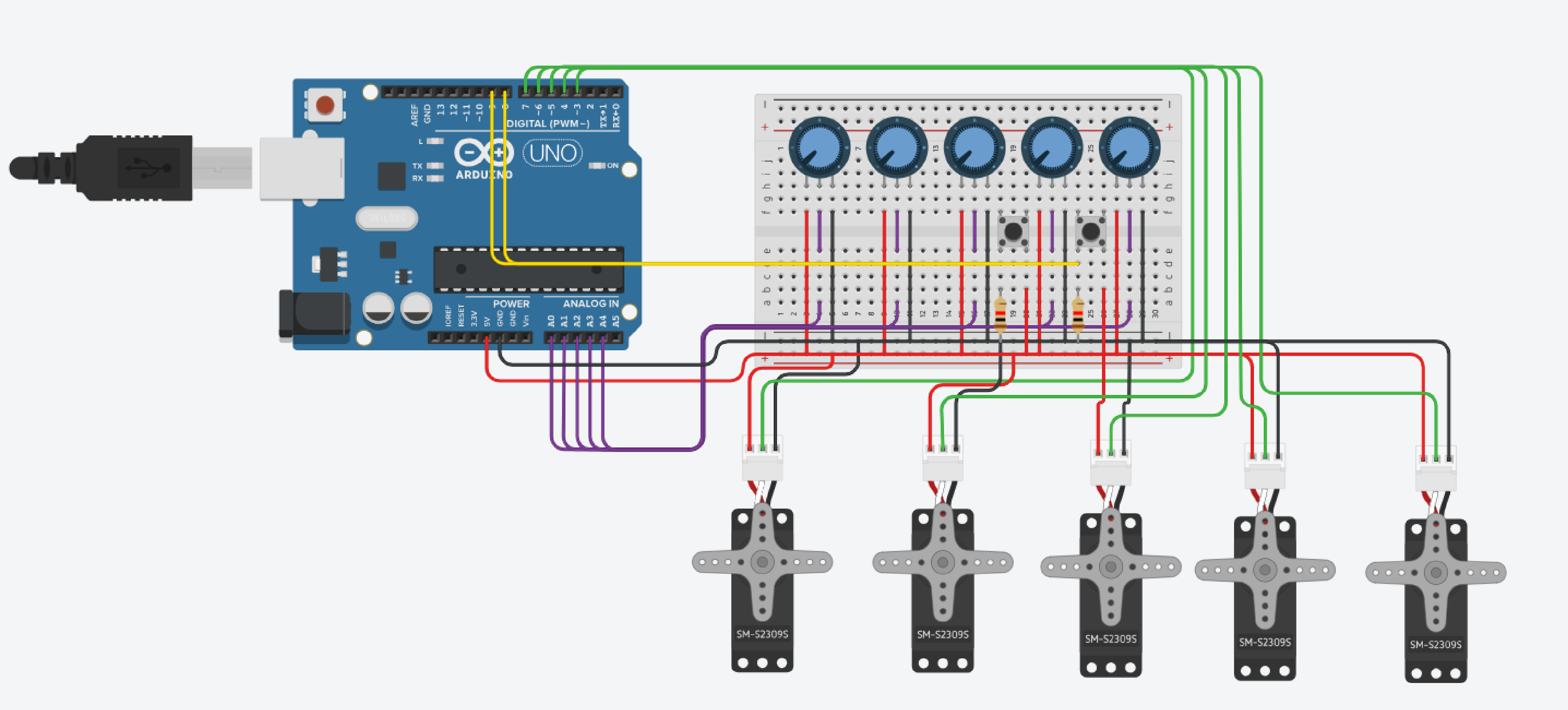


22. Control de servos mediante potenciómetros en Tinkercad

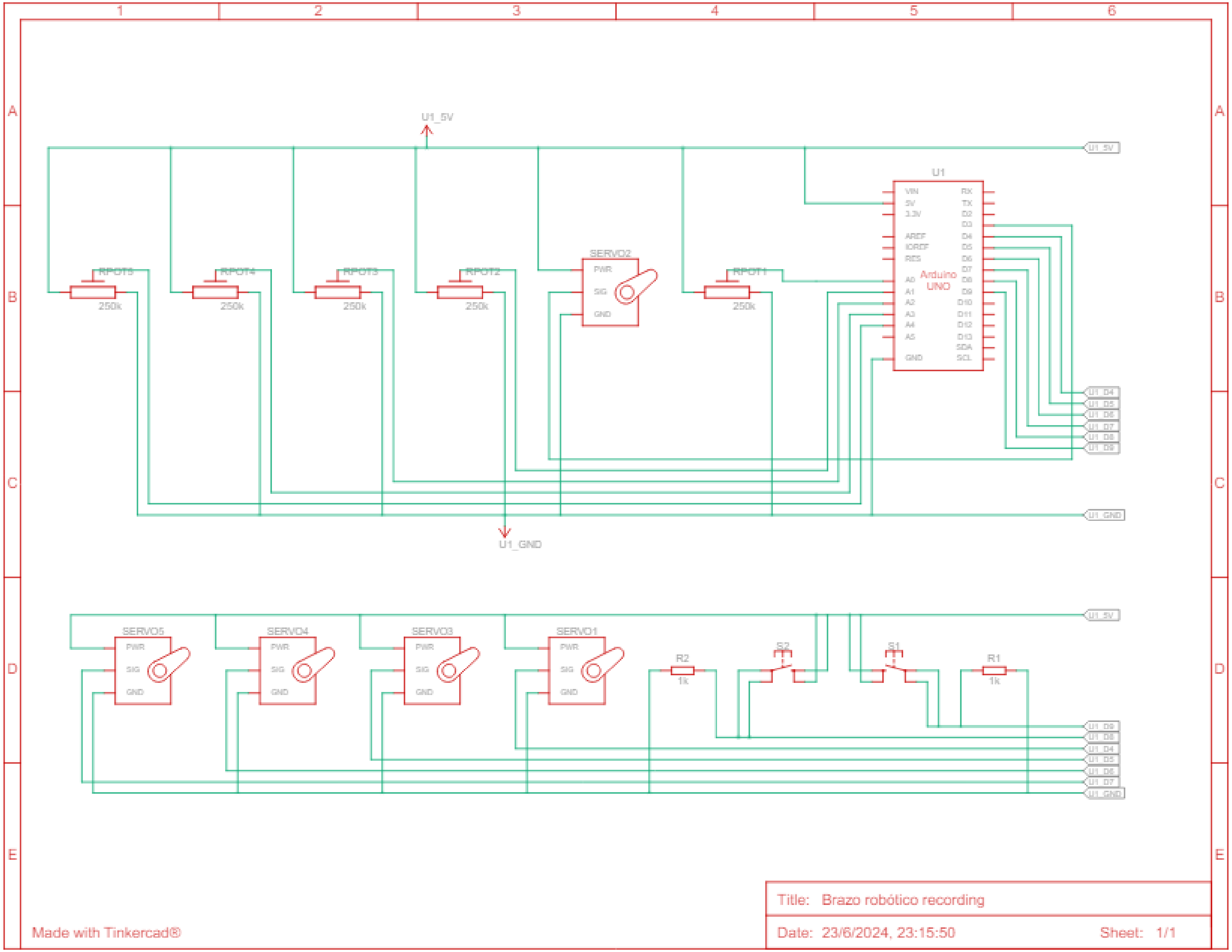


23. Esquema electrónico del control de servos mediante potenciómetros en Tinkercad

Después de tener un programa básico funcional, añadimos la función de grabar y reproducir movimientos a nuestro programa inicial. Este contiene dos pulsadores extras; uno tiene la función de grabar y el otro de reproducir los movimientos grabados.

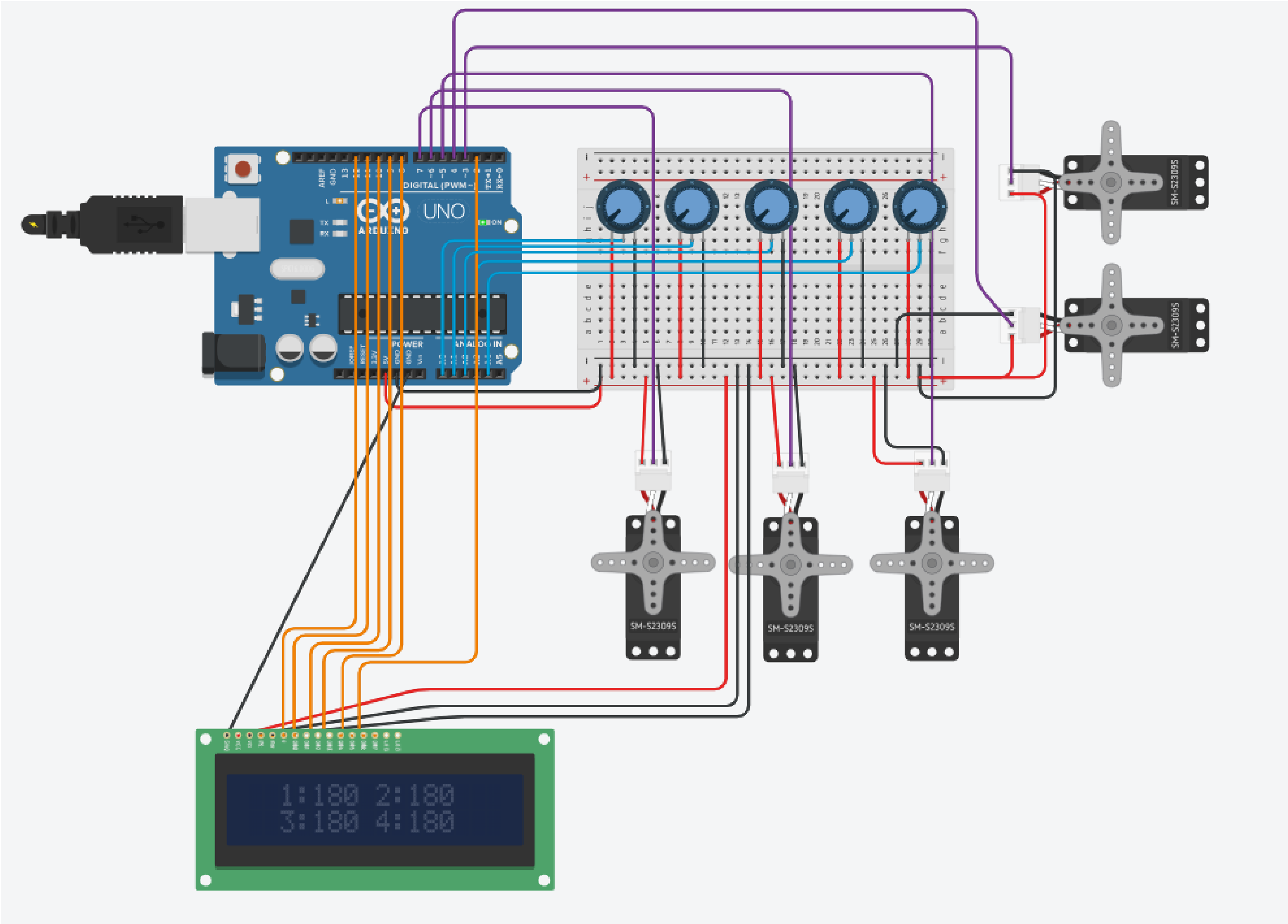


24. Grabar y reproducir movimientos en Tinkercad

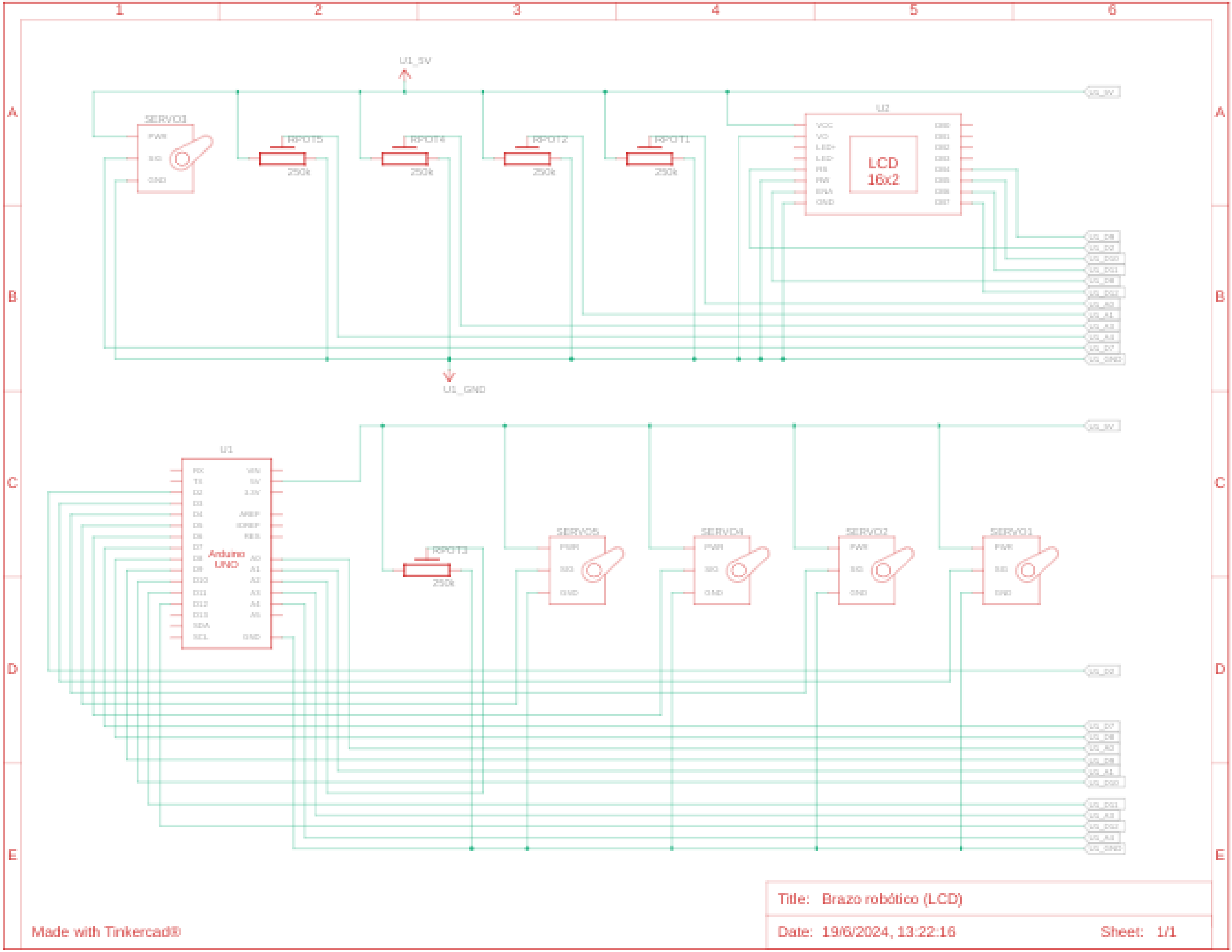


25. Esquema electrónico grabar y reproducir movimientos en Tinkercad

Más tarde incorporamos la pantalla LCD al circuito, ésta se encarga de indicar los grados de los servos posicionales y además nos indica si el programa está grabando o reproduciendo los movimientos grabados. Como se ha visto en esta secuencia de imágenes, cada vez íbamos incorporando nuevos elementos para aumentar la complejidad del proyecto y sus funcionalidades.

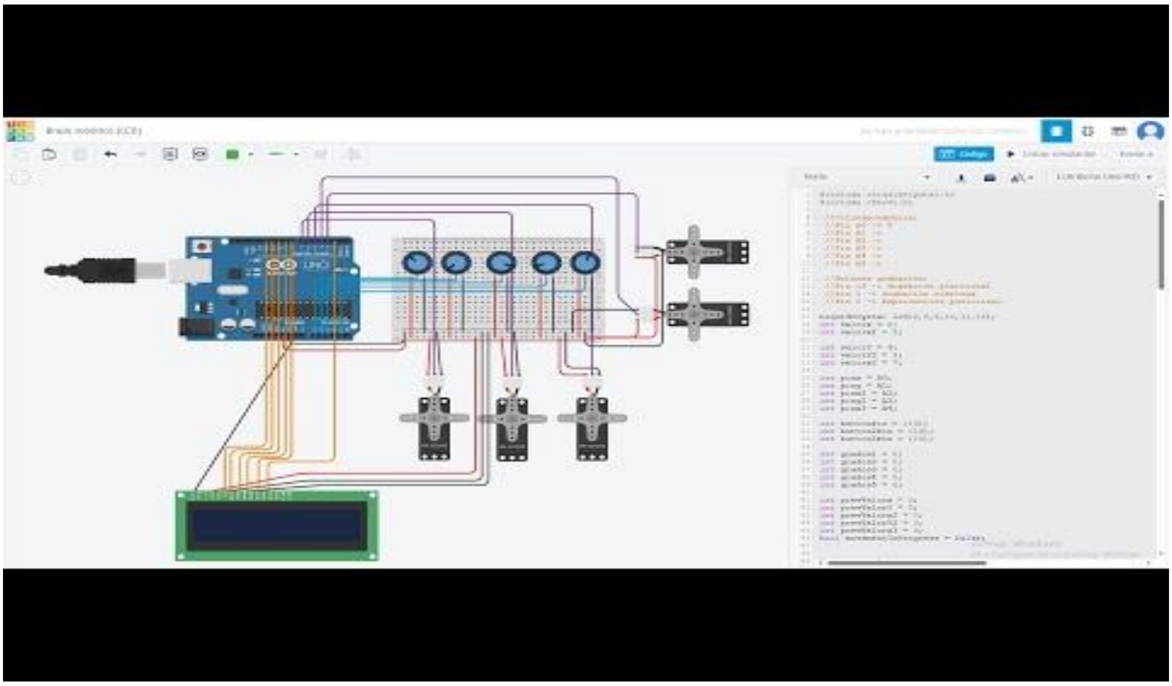


26. Pantalla LCD en el proyecto del brazo robótico en Tinkercad



27. *Esquema electrónico de la pantalla LCD del brazo robótico en Tinkercad*

https://youtu.be/fd-moGkacm8



Perfeccionamiento del programa

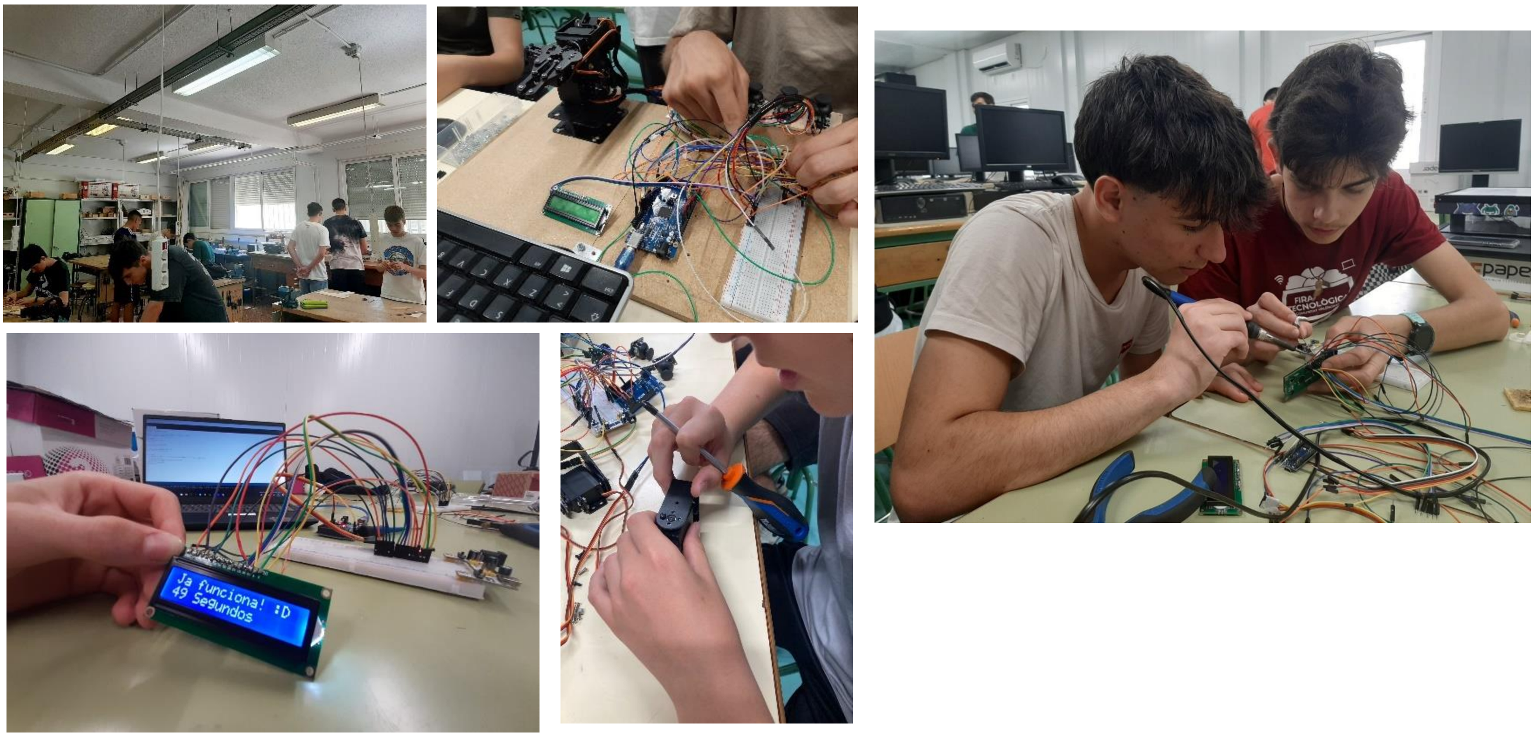
Teniendo ya el programa simulado y funcional en Tinkercad, solo nos faltaría subirlo a la placa Arduino Mega utilizando el programa Arduino IDE y Visual Studio Code.

Aquí tenéis parte del programa final del brazo robótico. Si queréis diseñar, construir y programar este proyecto tendréis que poneros a trabajar muy seriamente, tal y como han hecho nuestros alumnos:

```
int grados6 = 90;

//--Zona de identificr motores--

Servo motor1;
Servo motor2;
Servo motor3;
Servo motor4;
Servo pinza;
Servo base;

//--Zona de definición de los pines de los servos--

void setup() {
  lcd.init();
  lcd.backlight();
  lcdline("Iniciando...");
  Serial.begin(9600);
  randomSeed(analogRead(A7));
  motor1.attach(6);
  motor2.attach(5);
  motor3.attach(4);
  motor4.attach(7);
  pinza.attach(3);
  base.attach(2);
  pinMode(BotonGrabar, INPUT_PULLUP);
  pinMode(BotonReproducir, INPUT_PULLUP);
  Reset();
}

//--Zona de Grabar y Reproducir--

void loop() {

  Grabando = digitalRead(BotonGrabar) == LOW;
  Reproduciendo = digitalRead(BotonReproducir) == LOW;

  if(Grabando && Reproduciendo) Juego();

  else if(Grabando) {
     Serial.println("Grabando");
     Reset();
     lcdline("Grabando en:");
     lcdline2("3");
```

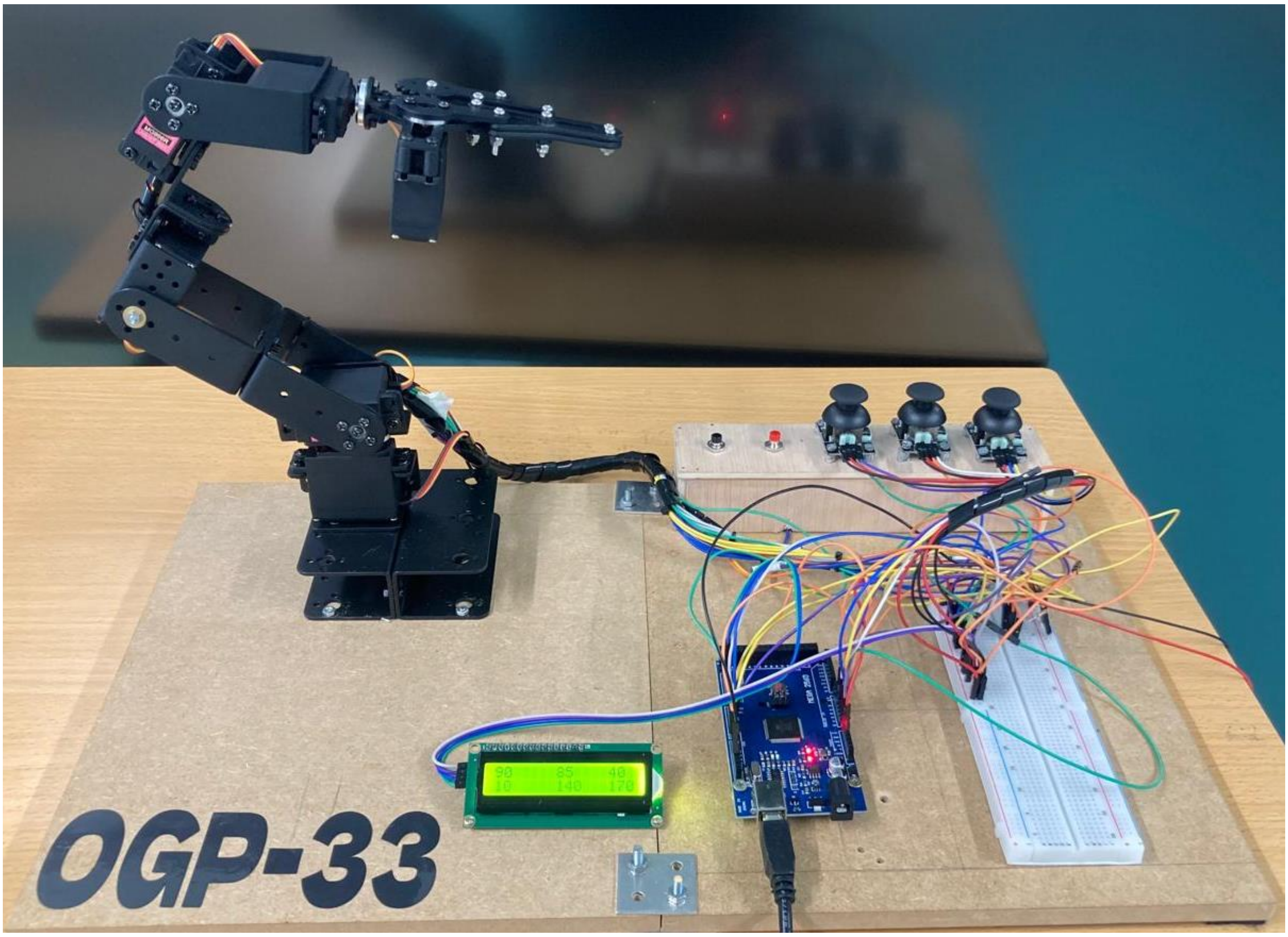
Construcción

Al mismo tiempo que programábamos el robot, empezamos a construir el brazo en el taller. Esta parte del proyecto ha sido en la que más cambios hemos hecho, pues según lo hacíamos iban surgiendo dificultades que necesitaban de cambios en el mismo.



28. Imágenes del proceso de construcción del brazo robótico en el taller

5. Resultado final



29. Resultado final del brazo robótico

6. Conclusiones

Durante el proceso de diseño, simulación, construcción y programación de nuestro brazo robot hemos aprendido y comprobado en primera persona lo que cuesta diseñar y crear un proyecto desde cero, enfrentándonos a una serie de contratiempos a lo largo de todas sus fases, como por ejemplo en el diseño de la estructura del brazo, que en un inicio nos dio problemas con los servomotores, pues tenían que soportar un trabajo excesivo. Por este motivo decidimos rediseñar el brazo y además sustituir los servomotores por otros más lentos pero que tienen más fuerza. Como consecuencia de estos cambios nos encontramos con problemas en la alimentación, debido al cambio de modelo de los servos, por lo que tuvimos que utilizar una fuente de alimentación externa que proporciona los 6,5 voltios que permiten el funcionamiento óptimo de los servos.

De la misma manera, a lo largo del proyecto, también nos hemos encontrado con problemas de funcionalidad en los joysticks, en la LCD y en las funciones de grabar y reproducir movimientos. Sin embargo, hemos ido solucionándolos uno a uno con paciencia y dedicación hasta el día de hoy, donde ya funciona todo como debería y nos sentimos muy satisfechos con el esfuerzo aportado y con el resultado obtenido.

7. Agradecimientos

Queremos agradecer a todas las personas que nos han ayudado en este proyecto, que han trabajado y aportado todo lo posible con esfuerzo y sudor para que finalmente haya salido hacia delante superando todo tipo de errores y obstáculos.

En especial queremos dedicar este proyecto a nuestros compañeros de clase y de otras clases por soportarnos cuando las cosas no salían como esperábamos cambiándonos el estado de ánimo.

GRACIAS ÓSCAR

(·) ¬ (·)

----

8. Bibliografía

- https://www.freepik.es
- https://app.sketchup.com/
- https://www.tinkercad.com/
- Arduino
- ¿Qué es un brazo robótico?
- Tutorial LCD con I2C, controla un LCD con solo dos pines
- Material propio del departamento de tecnología