CURSO BÁSICO DE ROBÓTICA APLICADA

5 horas

Pablo Del Otero Sevillano

Pablo2profesor

https://www.youtube.com/channel/UCz5IjDKT6-hJjenq19SeSpw

http://pablo2profesor.blogspot.com/

1. INTRODUCCIÓN 3
2. CONTEXTO 4
3. PROYECTO DE ROBÓTICA: ROBOT MIEDOSO LUMINOSO 6
3.1. FASES GENERALES DEL PROYECTO 6
3.2. FASE 1: IDEA 7
3.3. FASE 2: NECESIDADES MATERIALES HARD/SOFT 7
3.4. FASE 3: DISEÑO Y PROGRAMACIÓN 9
3.4.1. CREAR UN PROYETO 9
3.4.2. HARDWARE – CIRCUITOS 11
3.4.3. PROGRAMACIÓN Y SIMULACIÓN 13
3.4.4. MONTAJE FÍSICO SOBRE ENTRENADOR 34
3.4.5. PROGRAMACIÓN, COMPILACIÓN Y CARGA DEL PROGRAMA EN EL MICROCONTROLADOR ARDUINO 37
3.4.6. CARGA DE PROGRAMA Y PRUEBAS 42
3.4.7. MONTAJE SOBRE PIEZAS EN 3D: ROBOT 45
4. CONCLUSIONES 46
4. MODULO DE APLICACIÓN 47
4.1. MATERIALES (TINKERCAD) 47
4.2. FUNCIONAMIENTO (SECUENCIA) 47
4.3. Pistas y ayudas 48
4.4. Entregas 49

1. INTRODUCCIÓN

Esta pequeña introducción pretende ser una justificación de los contenidos que vamos a presentar en esta publicación, centrada en su objetivo de ayudar a docentes y alumnos a emplear técnicas robóticas en las aulas, proponiendo diseños, proyectos, soluciones y propuestas de trabajo y estableciendo los resultados que podemos conseguir con ellos.

Comencemos por el principio…el *origen*, como diría *Dan Brown*[1]. Se entiende la robótica como "*Técnica que aplica la informática al diseño y empleo de aparatos que, en sustitución de personas, realizan operaciones o trabajos, por lo general en instalaciones industriales*"[2].

La robótica siempre ha ido de la mano de la automatización ya que, su cometido final, es realizar un proceso o parte de el de forma autónoma. Desde el siglo pasado, han ido apareciendo diferentes soluciones robóticas para "liberar" a los humanos, principalmente, de trabajos pesados y tediosos.

Con los avances de la informática, la electrónica digital y, sobre todo, la imaginación humana, las capacidades de la robótica se han disparado, multiplicando sus posibilidades de uso y ampliándolo, de su tradicional industria, a incluso nuestros hogares.

La reflexión sobre hacia donde apunta esta tecnología si introducimos los algoritmos de inteligencia artificial, que tan de moda están ahora, pero que ya en los 90´s desarrollaban y usaban en Japón (lógica borrosa, redes neuronales), nos llevaría a una conversación de varios cafés inacabable. Pero lo que si se ha terminado hace dos décadas, es el pensamiento de que los robots son solo para las fábricas y para los dibujos animados.

Para los amantes de la ciencia y la tecnología, como un servidor, es apasionante el buscar soluciones robóticas todos los días. Diseñar, fabricar, montar y programar nuestros propios "juguetes", teniendo así la oportunidad de aportar soluciones reales a problemas que nuestro hogar o la sociedad nos plantea día a día.

Si, lo sé, es abrumador ver noticias sobre robótica, avances y tecnología, y esto es, porque nos queremos *comer el bocadillo de un solo bocado*. Y es aquí, donde radica la justificación de este curso. Como todo en esta vida, tenemos que empezar por el principio, ir avanzando poco a poco, porque cualquier gran proyecto, es la suma de pequeñas y sencillas configuraciones que todos podemos hacer.

Es ahí, donde la electrónica nos ha tendido una mano amiga. El desarrollo de la placa electrónica o microcontrolador ARDUINO (y simulares) ha puesto en nuestras manos, una herramienta barata, muy potente y, sobre todo, totalmente segura, con la que poder enseñar a nuestros alumnos a pensar, razonar, tocar y entender la base del funcionamiento de la tecnología. Me gusta decir que podemos "jugar a ser ingenieras e ingenieros" y, por qué no, aprendiendo.

Así pues, vamos a empezar por el principio, todos juntos.

1. https://www.planetadelibros.com/libro-origen/254690
2. https://dle.rae.es/rob%C3%B3tica

2. CONTEXTO

En este punto vamos a atacar esa pregunta que los docentes podemos llegar a plantearnos a la hora de escuchar la palabra robótica. Nuestra mente nos dice: "sí, me llama la atención, me gusta, pero como empiezo", o, por otro lado, "y esto, que les va a aportar a nuestros alumnas y alumnos".

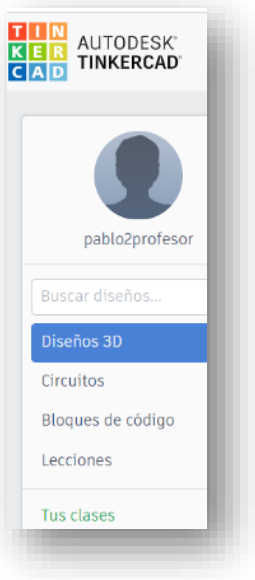
En este curso, pretendemos dar respuesta a estas dos preguntas de forma rápida y sencilla, que no tiene por qué ser la mejor solución del mercado, de hecho, no lo es, pero si una forma muy fácil de empezar y que alumnos de todos los niveles pueden aplicar.

La respuesta a la primera pregunta es TINKERCAD[3]:

Tinkercad es una plataforma online desarrollada por autodesk (desarrolladores de autocad, Revit…e infinidad de software de diseño que, además, posee licencias de estudiantes gratuitas. Pero este no es el caso, este software es online, totalmente gratuito y fácilmente accesible con nuestra cuenta Google.

Y cuáles son las principales ventajas que nos proporciona Tinkercad:

- Herramienta online totalmente gratuita.
- Al estar alojada en un servidor siempre está guardando lo que estamos haciendo.
- Accesible desde cualquier dispositivo.
- Modo profesor, con el que podemos intercambiar lecciones con los alumnos.
- Infinidad de ayudas, videos y repositorios de otros programadores.
- Facilidad, entorno amigable y adaptable a diferentes niveles.
- Modo simulación para testar nuestros diseños antes de montarlos físicamente.
- Infinidad más, seguro…



En lo que concierne a este curso, nos propone dos aplicaciones que nos permiten trabajar nuestros objetivos:

- Circuitos: Circuitos programables con Arduino.
- Diseños 3D

Gracias a estos dos "pestañas" podremos diseñar nuestros sistemas robóticos, imprimirlos en 3D, programarlos y, por supuesto, ponerlos en funcionamiento.

3. https://www.tinkercad.com/

Y es ahí donde entra *la respuesta a la segunda pregunta* ***¿Qué les aporta a nuestros alumnos?***.

Ciertamente, no soy pedagogo ni puedo establecer verdades de fe sobre lo que va a afectar en el medio y largo plazo a los alumnos, la programación y la robótica en su educación, pero sí que podemos resumir conceptos sencillos que manejaremos con ellos:

- Trabajo por proyectos. Ya conocemos las ventajas del trabajo por proyectos en cuanto al aprendizaje, educativo y social, y en cuanto a la motivación. El desarrollar roles dentro del grupo, llevar un proyecto adelante en equipo, es una de las mejores sensaciones que existen a nivel educativo para nuestro alumnado y esto…"engancha".
- Pensamiento de diseño y pensamiento crítico. Por medio de estas aplicaciones, el alumno desarrolla su visión espacial, cálculos matemáticos, lógica de programación, trabajo en equipo, orden y estructura de proyecto, confianza en sí mismo. Fomenta la creatividad y la capacidad de superación.

- Motivación, actitud, capacidades…

En resumen, les adentra en el mundo de la ciencia, tecnología y las matemáticas. Me gusta decir que estamos sembrando ingenieras e ingenieros para los nuevos tiempos.

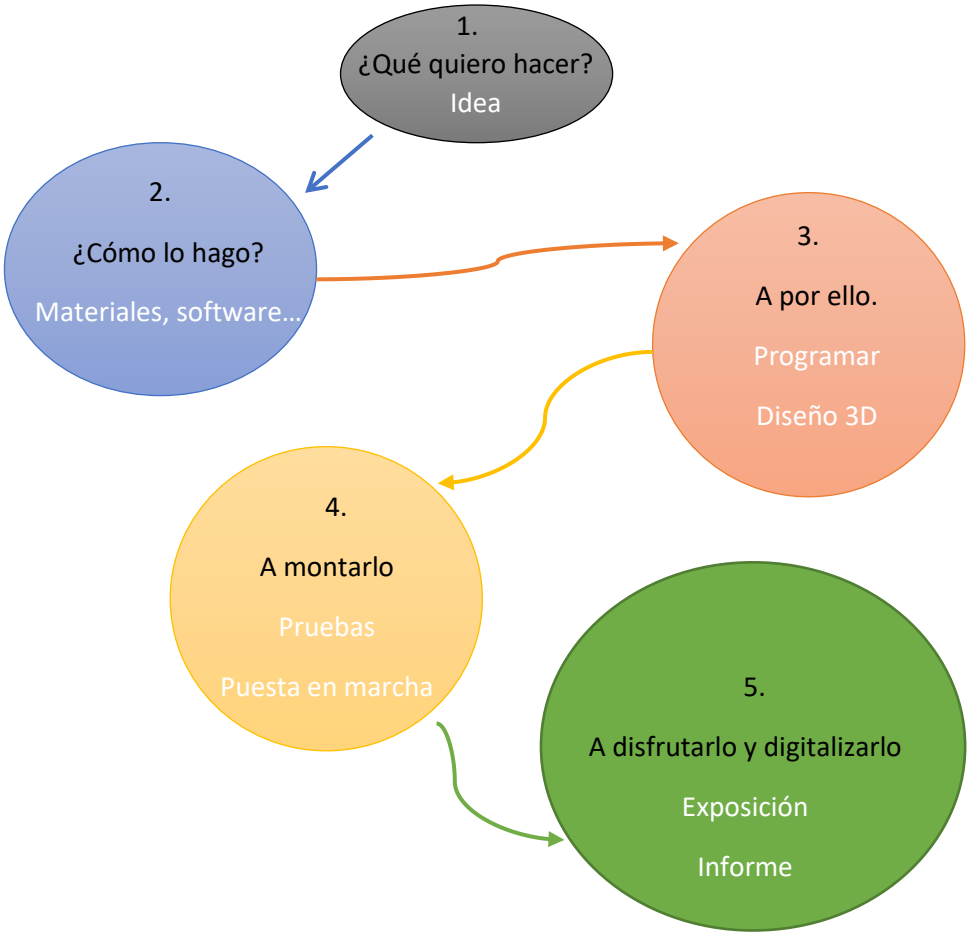
Así pues, si todo es un camino de rosas, pongámonos a ello. Lo sé, no es tan fácil, lleva tiempo, lleva preparación, pero os puedo asegurar, que es gustoso ver el resultado, ver la cara de nuestros alumnos cuando consiguen por sí mismos desde lo más mínimo, hasta lo más complejo.

Ahora sí, comencemos.

3. PROYECTO DE ROBÓTICA: ROBOT MIEDOSO LUMINOSO

3.1. FASES GENERALES DEL PROYECTO

Comencemos nuestro proyecto de robótica viendo los puntos en los que nos basamos a la hora de desarrollar cualquier proyecto:



Una vez vistas las fases del proyecto, vamos a centrarnos en los pasos a seguir para conseguir nuestro proyecto, que se denomina: **ROBOT MIEDOSO LUMINOSO**.

Aquí es donde la imaginación de cada docente y el conocimiento de las inquietudes de su alumnado debe orientarlos hacia la implementación de proyectos que les/nos llamen la atención, y, si pueden tener aplicaciones reales, pues mejor.

En este caso, hemos elegido este proyecto, aprovechando la estética de robots del cine de animación, aunque propondremos otros proyectos prácticos para aplicar a otro tipo de campos.

SOFTWARE NECESARIO

- TINKERCAD (online, no necesita instalación)
  - CIRCUITOS: Diseño y programación simulada de la solución robótica.
  - DISEÑO 3D y generación de archivos STL para impresión.

- ARDUINO (gratuito en https://www.arduino.cc/en/software)
  - Software para cargarle de forma real el programa a la placa.

Dentro de los anexos, incluiremos una guía básica de manejo de estos softwares aplicados a nuestro proyecto.

Empezamos a trabajar y vamos a seguir las fases del proyecto:

3.2. FASE 1: IDEA

El objetivo será montar este robot, al que hemos bautizado como: *ROBOT MIEDOSO LUMINOSO* debido a sus características:

a) El robot posee la capacidad de seguir la luz, de orientarse al sol, porque, como a los lagartos y las personas en las playas, le gusta tomar el sol.
b) Pero si alguien se acerca demasiado, le entra la vergüenza, enciende sus mofletes y se da la vuelta.

Una vez definido el objetivo del proyecto, pasamos a la siguiente fase.

3.3. FASE 2: NECESIDADES MATERIALES HARD/SOFT

En este punto debemos definir las necesidades de nuestro robot, sobre todo materiales.

Así pues, necesitaremos:

- Un cerebro que aplique la lógica. Arduino.
- Sensor de nivel de iluminación, para poder orientar la cara al sol. Fotorresistencia.
- Sensor de distancia, que mida cuando algo se acerca demasiado. Sensor de ultrasonidos.
- Luz de peligro (mofletes). Luces de Led de color rojo y resistencias de protección.
- Motor que gire la cabeza, tanto para orientar al sol, como para darse la vuelta. Usaremos un servomotor de Arduino.
- Diseño 3D o maqueta para montaje del robot (podría ser de cartón u otros materiales). Material PLA para impresión 3D (e impresora 3D, por supuesto).
- Cables, pegamento, tronillos…

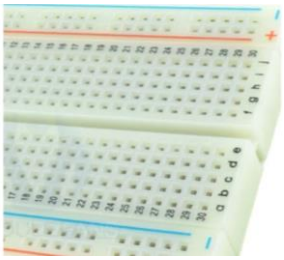
ELEGOO Arduino MEGA 2560

Diodos Led

Sensor de distancia ultrasónico SR04

Fotoresistencia

Servomotor 5Vcc



Placa de conexiones

Una vez recopilados los materiales, nos ponemos manos a la obra.

3.4. FASE 3: DISEÑO Y PROGRAMACIÓN

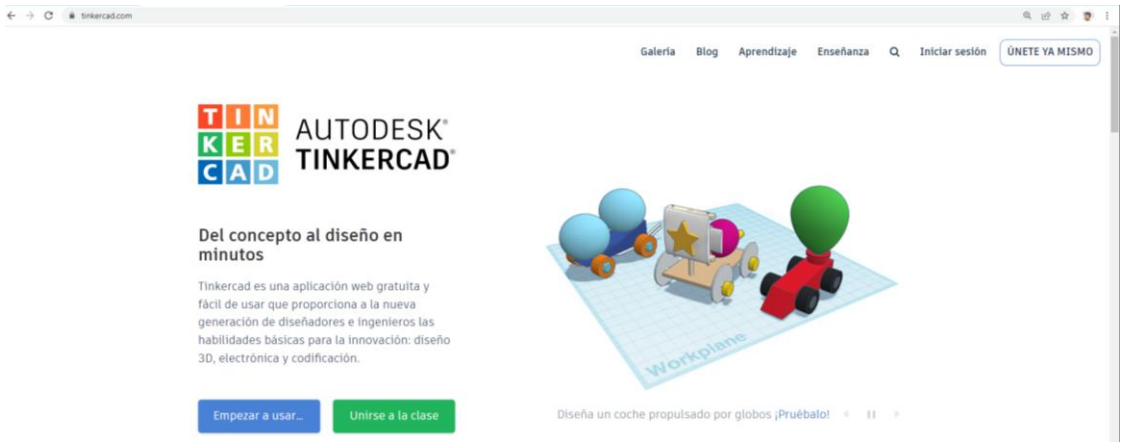
La verdad es que estos dos puntos son perfectamente compatibles y pueden ser simultáneos en el tiempo. En este caso comenzaremos por la programación (simulación + carga) para posteriormente pasar al diseño 3D e impresión de piezas.

3.4.1. CREAR UN PROYETO

La programación, como hemos disco con anterioridad, la realizaremos en Tinkercad. De esta manera, podemos probar el funcionamiento e idoneidad de nuestro proyecto de forma simulada (sin deteriorar tarjetas reales) para después, cargarle el programa a la placa de Arduino.

Para ello seguiremos los siguientes pasos:

1. **Abriremos Tinkercad en nuestro navegador: https://www.tinkercad.com/**



2. **Pulsamos unirse, logándonos con nuestra cuenta de Google.**

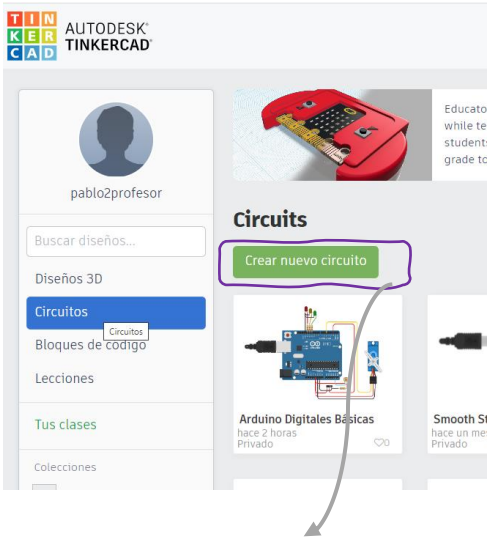
Podemos crear una cuenta:

Como educador (docente).
Como estudiante (alumnos).
Como personal (sin ser educativa)

O iniciar sesión directamente

Todas estas opciones nos llevan al menú de acceso por cuenta de Google (Apple-Otros).

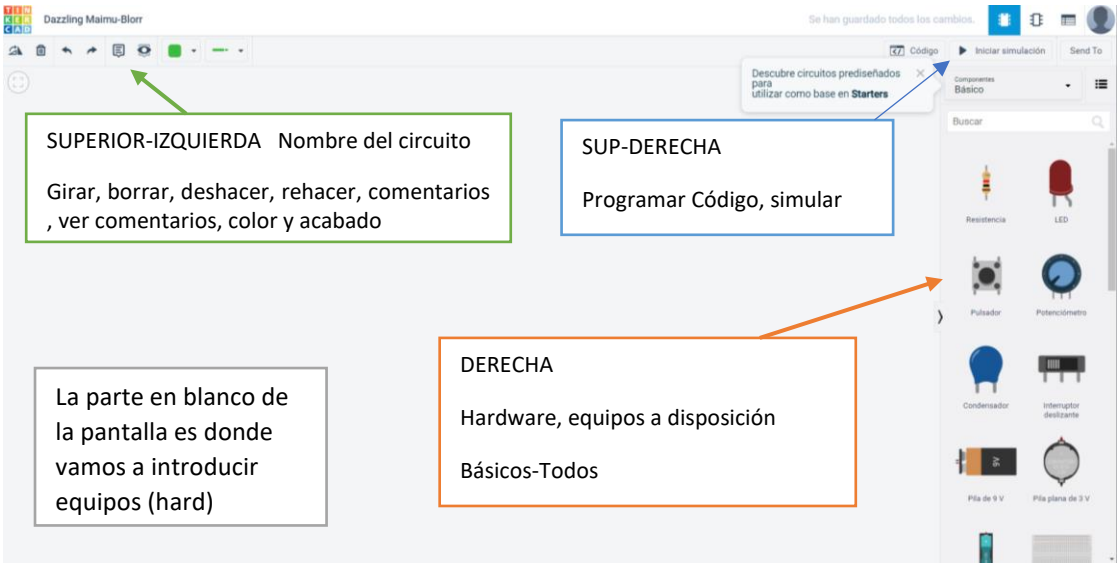
3. Dentro del escritorio de Tinkercad, escogemos CIRCUITS



En CIRCUITS, nos aparecerán todos los circuitos que hemos realizado a lo largo del tiempo, en su última versión.

Debajo de TUS CLASES te aparecerán las clases que hayas creado con alumnos para trabajar y compartir diseños.

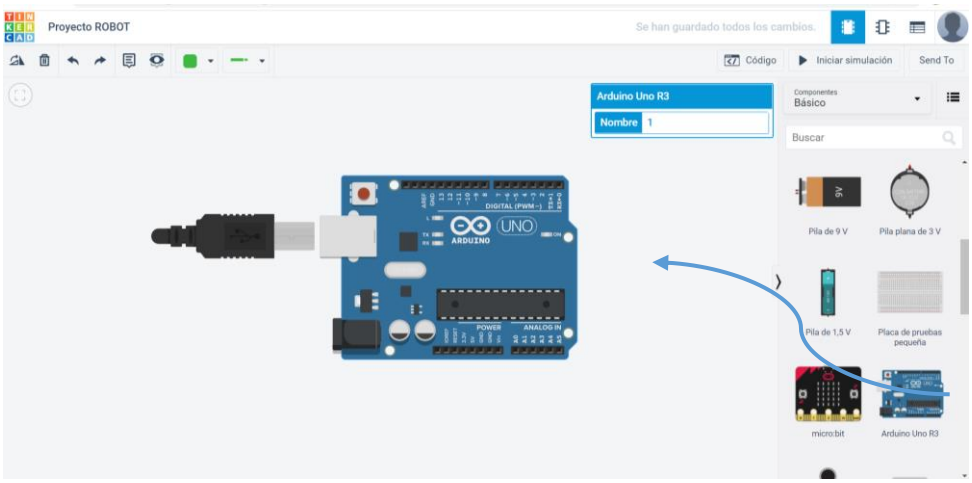
4. Crear circuito nuevo en el botón.



3.4.2. HARDWARE – CIRCUITOS

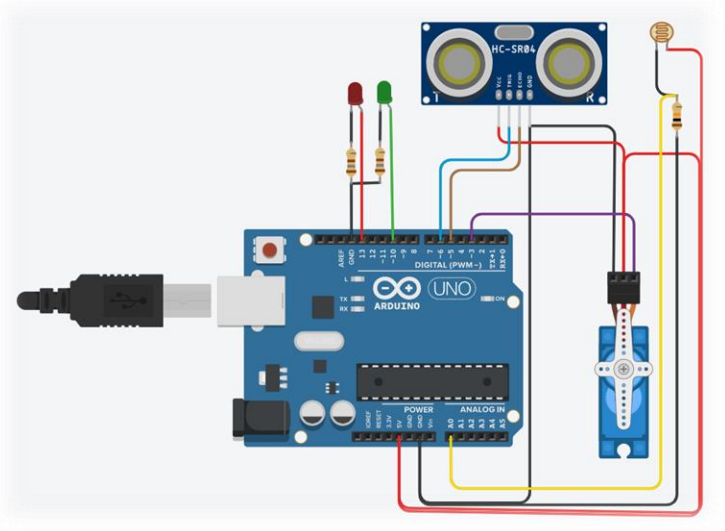
Tras crear el proyecto nuevo y darle un nombre, vamos a ir introduciendo y programando los contenidos.

1. **<u>Introducimos la placa de ARDUINO desde la parte izquierda con solo arrastrar la placa.</u>**



2. **<u>Introducimos los elementos que nos va a hacer falta.</u>**

   - Diodo led rojo, para indicarnos que estamos muy cerca. Necesita resistencia de 180 ohmios.
   - Diodo led verde, para indicarnos que está todo bien. Necesita resistencia de 180 ohmios.
   - Sensor ultrasonidos, que permite medir la cercanía de objetos al robot.
   - Servomotor, que produce el giro de la cabeza del robot.
   - Fotorresistencia (sensor de luz). Necesita resistencia de 10K ohmios.



3. Consideraciones al montaje

- *Los diodos se alimentan a 3.3V y las salidas de Arduino son a 5Vcc (ánodo-polo positivo-patilla retorcida) , por ello, tendremos que colocar una resistencia de 180ohm para que no se averíen, lo haremos en su polo negativo de vuelta a la placa (GND) (cátodo-polo negativo-patilla recta).*

- *El sensor ultrasónico necesita el emisor de señal (trigger) y el receptor de la señal (eco), como los murciélagos (mide el tiempo entre la ida y la vuelta y calcula la distancia).*

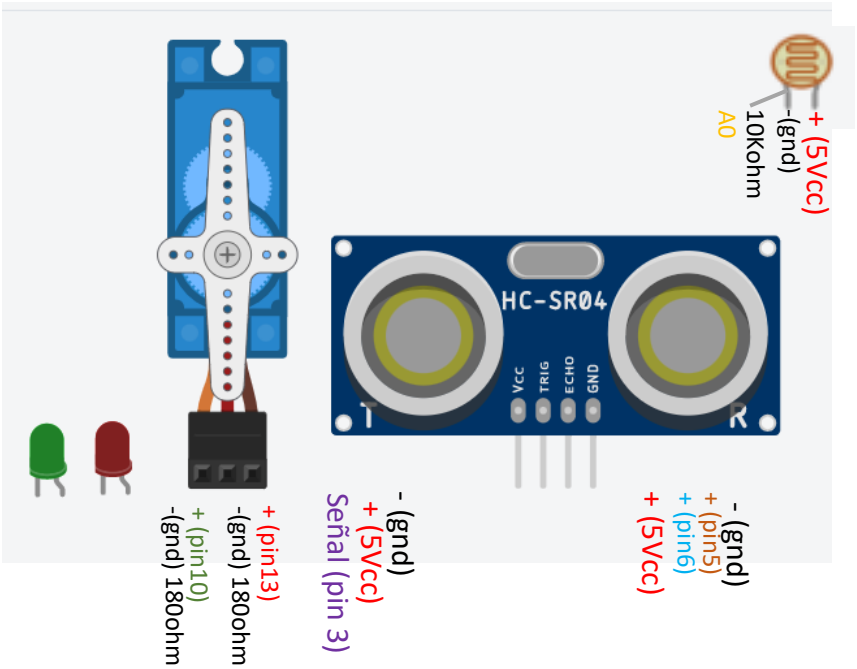
- *El servomotor recibe por el cable de señal pulsos para ir a los grados exactos.*

- *La foto resistencia, necesita un puente en H (bajar la tensión en su entrada derivándola) para leer de forma correcta en las entradas analógicas, Necesitaremos una resistencia de 10khm y conectar según la figura.*

- *El valor de las resistencias se introduce de forma manual haciendo clic sobre ellas.*

4. Pines de conexión

- *Led Rojo (patilla 13)*
- *Led verde (patilla 10)*
- *Ultrasónico (Triger – Eco)*
- *Servo (patilla 6)*
- *Fotorresistencia (A0)*



5. CONSIDERACIONES DE PROGRAMACIÓN DE SISTEMAS AUTOMÁTICOS:

- Debemos tener en cuenta cómo funciona un microprocesador o controlador.
  Lee las entradas → Realiza el programa → Ofrece las soluciones en las salidas.
  Este proceso se repite cada milisegundo.

  Así pues, debemos tener claro que son entradas (me aportan información) y que son salidas (actuó sobre ellas). Analizando nuestro sistema, tenemos:

  - **Led**: Salidas. Soy yo el que las mando encenderse y apagarse.
  - **Servomotor**: Salida, igualmente, pero de posición mediante señal pulsada. El programa nos ofrece como entrada, la lectura de la última posición (de la señal actual de la salida).
  - **Fotorresistencia**. Entrada. Recibo la información.
  - **Sensor de distancia**: Doble función. El disparador será salida (yo decido cuando quiero medir) y el ECO será entrada (recibo la señal de vuelta).

3.4.3. PROGRAMACIÓN Y SIMULACIÓN

Empezaremos paso a paso por lo más sencillo y lógico, que es…analizar lo que queremos que haga el robot.

Le hemos denominado ROBOT MIEDOSO LUMINOSO porque:
- Debe notar que algo se acerca demasiado
- Debe buscar, si nadie está cerca, el punto de mejor orientación a la luz.
  Esto lo hará reorientándose de manera autónoma.

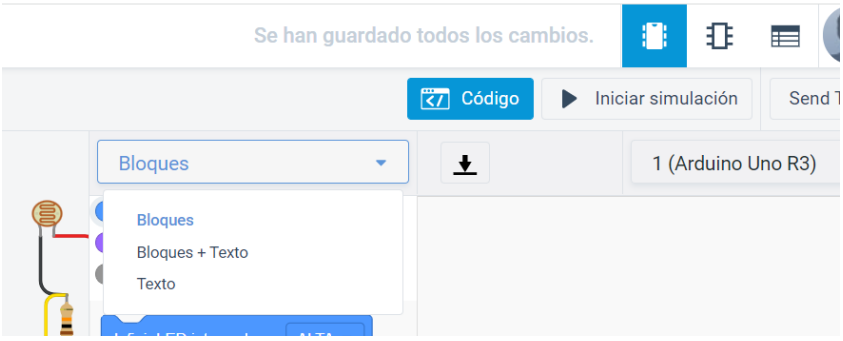
a) PRIMERA FASE: Robot Miedoso

A nivel de equipos, trasladamos esa información a nuestras necesidades:

- Medir la distancia con el sensor ultrasónico. Si algo supera un umbral, el servo girará 180º y se encenderá la luz roja.
- Si no se supera el umbral de cercanía, que la luz esté verde.

1. Traslademos nuestra lógica a nuestro software TINKERCAD:

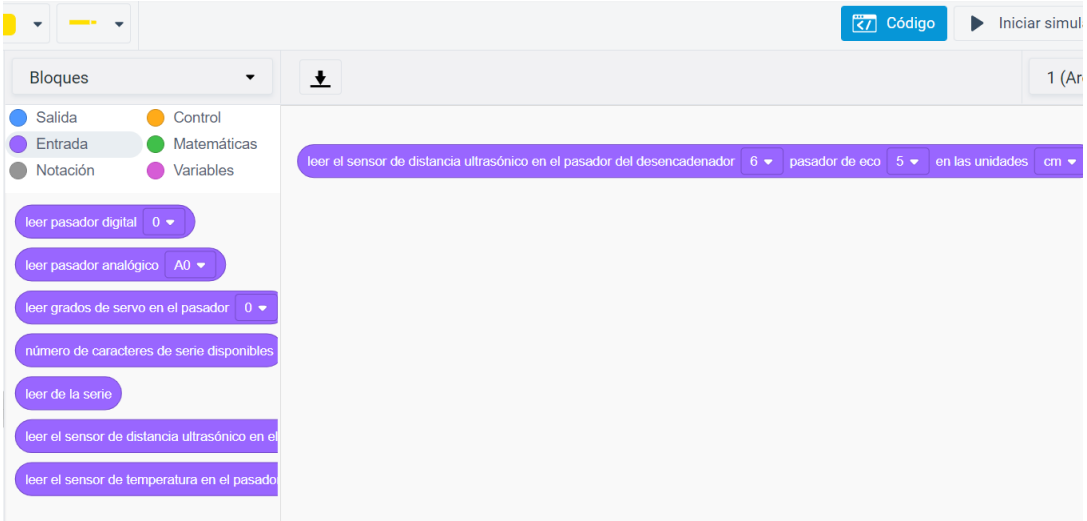
Pulsaremos sobre el icono **CODIGO** y después seleccionaremos **BLOQUES**[4].



4 Si queremos ver la programación en lenguaje de instrucciones, tipo C, Python…podemos seleccionar Bloques + Texto, pero no podremos programar en texto, dado que solo veremos la programación y copiarla para pasarla a la tarjeta física (lo haremos dentro de poco)

2. Lectura del SENSOR ULTRASÓNICO.

Ahora vamos a ver cómo podemos leer el sensor ultrasónico. Para ello, dentro del menú **ENTRADA**, nos llevamos el bloque **"Leer el sensor de distancia ultrasónico…."** al espacio de programación (derecha). Configuramos el desencadenador en 6 (pin 6) y el Eco en el 5 (pin 5) y la salida nos la dará en centímetros (cm).

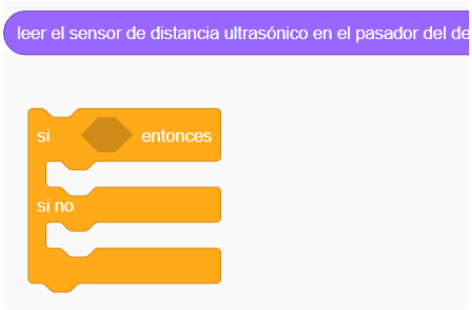


3. Comparación de lectura de distancia con la distancia de consigna

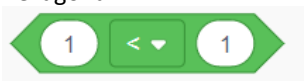
Ahora que ya hemos leído la distancia, queremos compararla con el valor mínimo que consideramos "demasiado cerca". Pero primero, vamos a ver los límites del sensor ,teóricamente entre 2cm y 450cm. Vamos a tomar 20cm como el límite de acercamiento.

Así pues, nuestra lógica sería:
" Si estas a menos de 20cm, enciende luz roja y gira la cabeza. Si no, luz verde y cabeza adelante".

Para ello usaremos el bloque **Si** (if en inglés) que encontramos en el menú **CONTROL**.
SI-entonces, sino…



Este bloque "si" nos exige una variable de comparación, que para nosotros es "cm"<20. Esto es una operación matemática que encontramos en su menú.
Vemos como el hueco hexagonal del "si" ya nos ayuda a elegir el bloque de comparación hexagonal.



Ahora configuramos la **comparación**, por un lado, los centímetros (que arrastramos la señal de lectura al primer hueco circular, donde pone "1" por defecto) y por el otro, los 20cm a configurar. Elegimos el signo de la comparación, en este caso menor.

4. Asignación de luces y posición de servo, en función de la comparación.

Una vez introducida la condición que nos va a permitir seleccionar las dos posibilidades, "si" y "si no", vamos a programar cada uno de los casos.

SI: Luz Roja, servo a 180º.
NO: Luz Verde, servo a 0º.

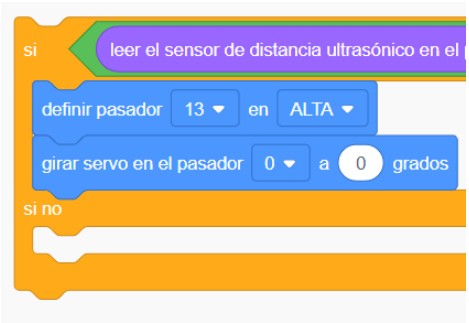
Como dijimos anteriormente, los LED y el SERVO son salidas. En el menú Salida, encontraremos como usarlos.

LED Rojo:

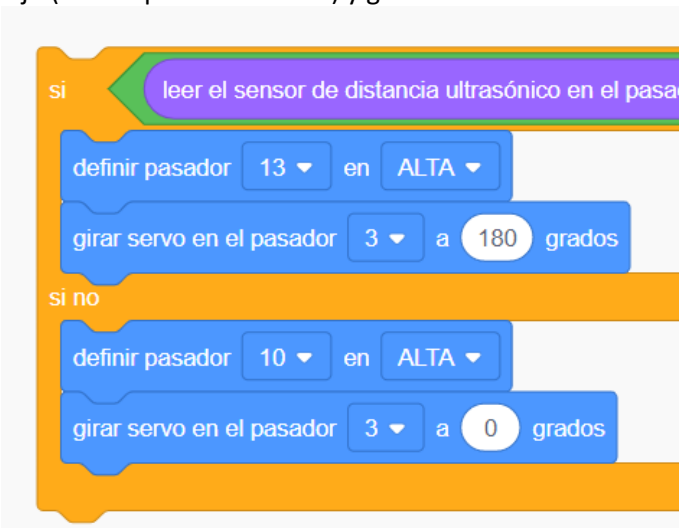
Tomo el bloque de **"definir pasador…"** (pin13) y lo arrastro. Configuro la entrada 13 y además, decido sus dos únicos posibles estados: Encendido- Nivel "ALTA" – Apagado-Nivel "BAJA".

SERVO:

Arrastramos el bloque **"girar servo en el pasador…".** En el pasado configuramos el pin de la salida de señal (pin3), y en grados…pues los gradoss deseados, en este caso 180º.

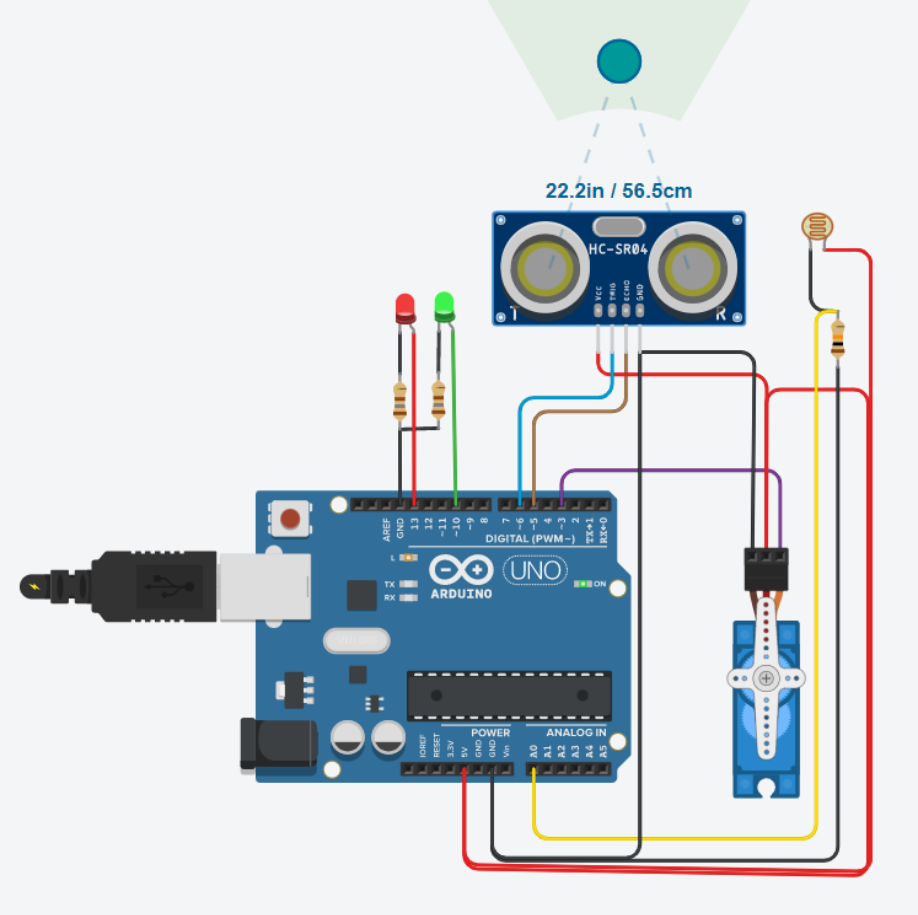


Y ahora vamos a ver que se hace en la segunda condición "Si no".
Encendemos led Rojo (definir pasador 10 Alta) y giramos el servo a 0º.



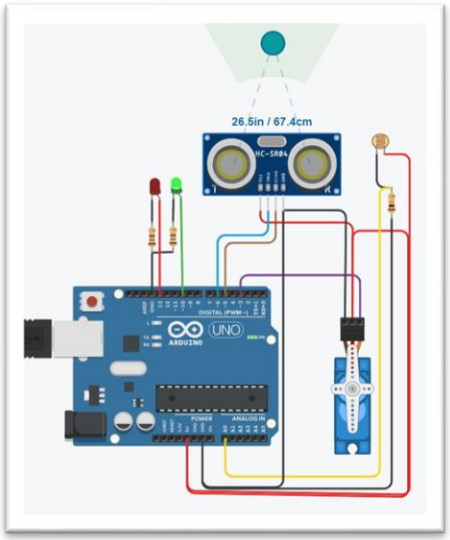
5. Modo simulación

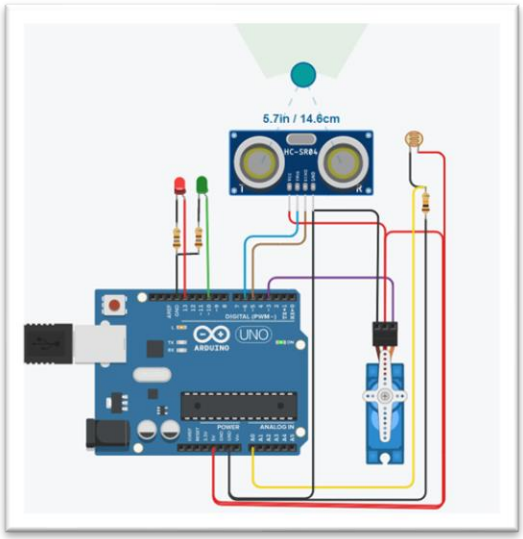
Ha llegado el momento de probarlo…¿cómo? Pulsando el botón de iniciar simulación.
Una vez arrancada la simulación, si pulsamos sobre el sensor ultrasónico, podremos ver la medida y con solo desplazar la bolita hacia delante y hacia atrás, podemos ver la distancia medida.

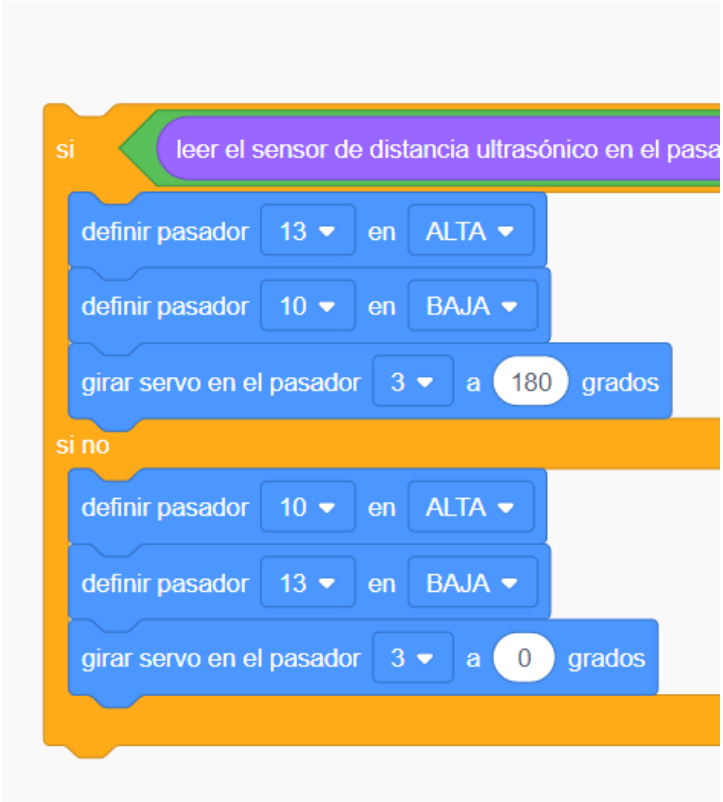


Funciona!!!!! Bueno, no del todo. Estamos viendo que si estamos por debajo de 20cm se enciende la roja y se mueve el servo, pero cuando volvemos a más de 20cm, la verde se enciende, pero la roja no se apaga. Eso es, porque debemos apagar las luces, porque no se apagan solas 😊. Para ello, debemos introducir en el condicionante de cada encendido, el apagado de la otra luz.

Detenemos simulación y activamos apagados. Ahora si!


26.5in / 67.4cm
HC-SR04
UNO
ARDUINO


5.7in / 14.6cm
HC-SR04
UNO
ARDUINO


si
leer el sensor de distancia ultrasónico en el pasa
definir pasador 13 en ALTA
definir pasador 10 en BAJA
girar servo en el pasador 3 a 180 grados
si no
definir pasador 10 en ALTA
definir pasador 13 en BAJA
girar servo en el pasador 3 a 0 grados

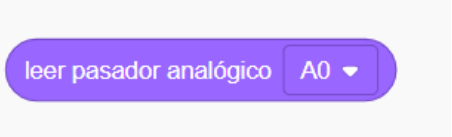
b) SEGUNDA FASE: Seguidor de Luz

El robot debe, siempre que no le entre el miedo, orientarse hacia el sol, así pues, vamos a configurar nuestro sensor de luz.

1. Configuración de entradas analógicas – sensor de luz

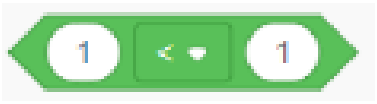
Para ello primero, analicemos como funciona la fotorresistencia. Apartir de la corriente que le atraviesa y de la intensidad de una señal luminosa, nos proporciona una señal de entrada, en el pin analógico 0 (tiene valores contínuos, no solo ALTA-BAJA), un valor entero, con el que podremos trabajar.

Para leer este valor, debemos ir al menú de **ENTRADAS** y escoger la instrucción **"leer pasador analógico**…" configurando la señal de entrada del pin correspondiente, A0

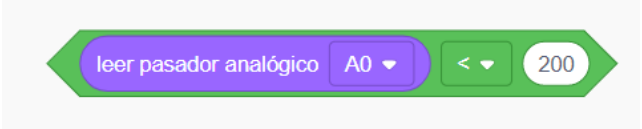


2. Comparación de lectura con valor de consigna

Como ese valor, lo queremos comparar con un valor que estableceremos como máximo (valor de prueba en este caso), debemos ir al menú **MATEMATICAS** y seleccionar una comparación (hexagonal)



Esta comparación la completaremos seleccionando en la entrada (primer orificio obalado) la lectura de la variavble analógica, y en el segundo, el valor que establecemos como máximo para la prueba (que es el punto de máxima luz, orientado al sol, 200 en nuestro ejemplo).

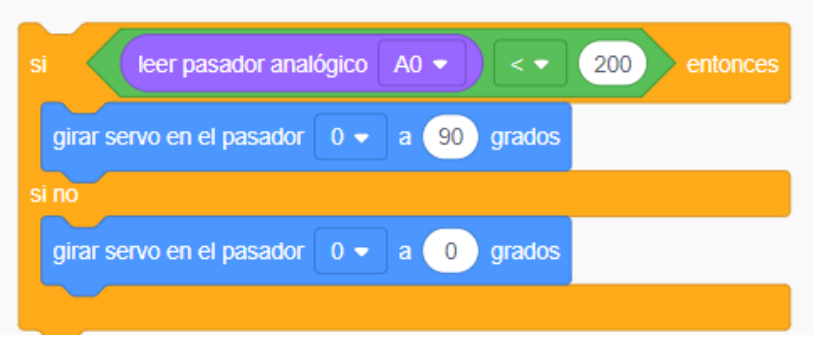


3. Lógica condicional a la lectura

Ahora debemos establecer el condicionante, es decir, cómo se va a mover, nuestra lógica será:

- Si el valor de luz es menor de 200 → Orientate hasta que sea mayor.

  Como estamos realizando una prueba, simplemente mandaremos al servo a un Ángulo conocido, por ejemplo 90º. Y si no, 0º.



4. Orientación en funcion de los valores leidos

En este punto nos aparecen conceptos un poco más complejos que debemos abordar:

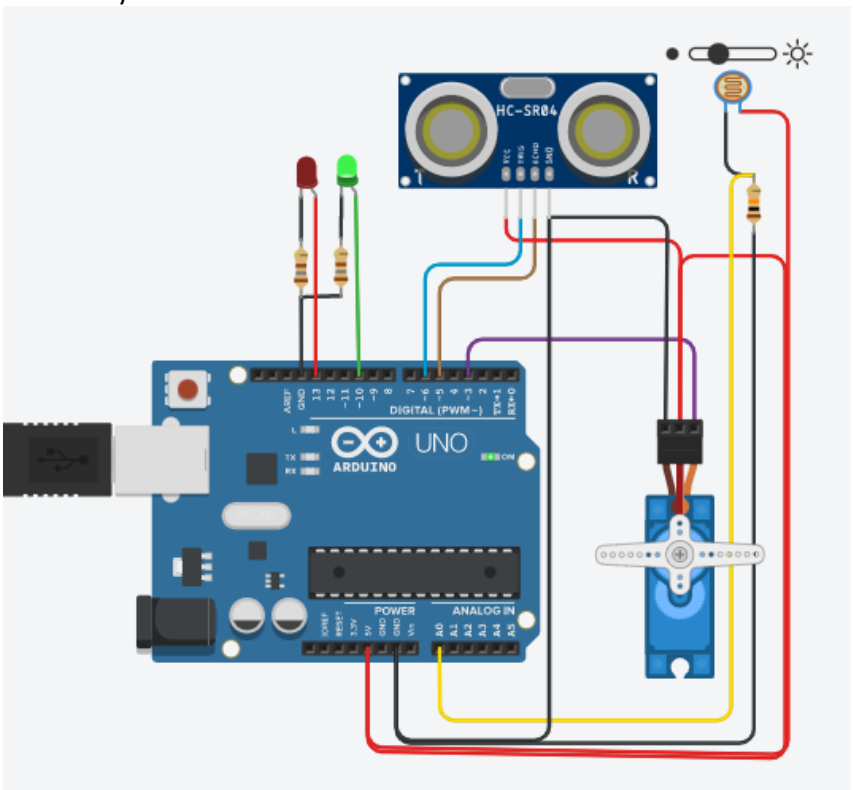
1. Esta orientación a la luz, solo puede funcionar cuando tengamos permiso, es decir, cuando el robot esté en VERDE.
   Así pues, hay que anidar este bucle "si no" dentro del "si no" principal que teníamos hecho.
   No podemos hacerlo de forma independiente, puesto que estamos trabajando con el mismo servo, a la hora de la función "miedosa" y a la hora de la "luminosa" y, como decíamos anteriomente, un procesador lee entradas, procesa y envia salidas. Si tenemos 2 instrucciones que actuan sobre el servo, el procesador mandará la señal del último valor, es decir, la última instrucción servo, por lo que las demás quedan invalidadas.
   Ejemplo: si en la 3ª linea le indicamos que el servo vaya a 180º, en la 5ª que vaya a 270º y en la 12ª que vaya a 50º, entonces el servo irá a 50º (última instrucción ejecutada).

2. Para poder incrementar o decrementar el valor del ángulo a indicarle al servo, debemos incluirlo en una variable con la que podamos operar e ir reduciendo u aumenando el ángulo a nuestro gusto de forma automática.

Resolvamos el primer punto. Debemos introducir un bucle "si no" nuevo, dentro del "si no" actual. En la parte de "si no" vamos a arrastrar lo que teníamos programado, y en la parte de "si" nuestra nueva programación.

Como podemos observar, hemos introducido los 3 bloques programados dentro del "si no" nuevo, y hemos introducido lo mismo en la parte de "si", pero en este caso le vamos a poner ángulo 90.

Esto lo hacemos para probar el sensor de luz, si la resistencia nota mucha luz, el servo se pone a 90º, si nota menos de 200, el servo va a 0º. Podemos simular el nivel de luz pulsando sobre el sensor y moviendo la barra de menos a más luz.



Eureka!!! Nos hace caso. Somos unos hackers de la electrónica.

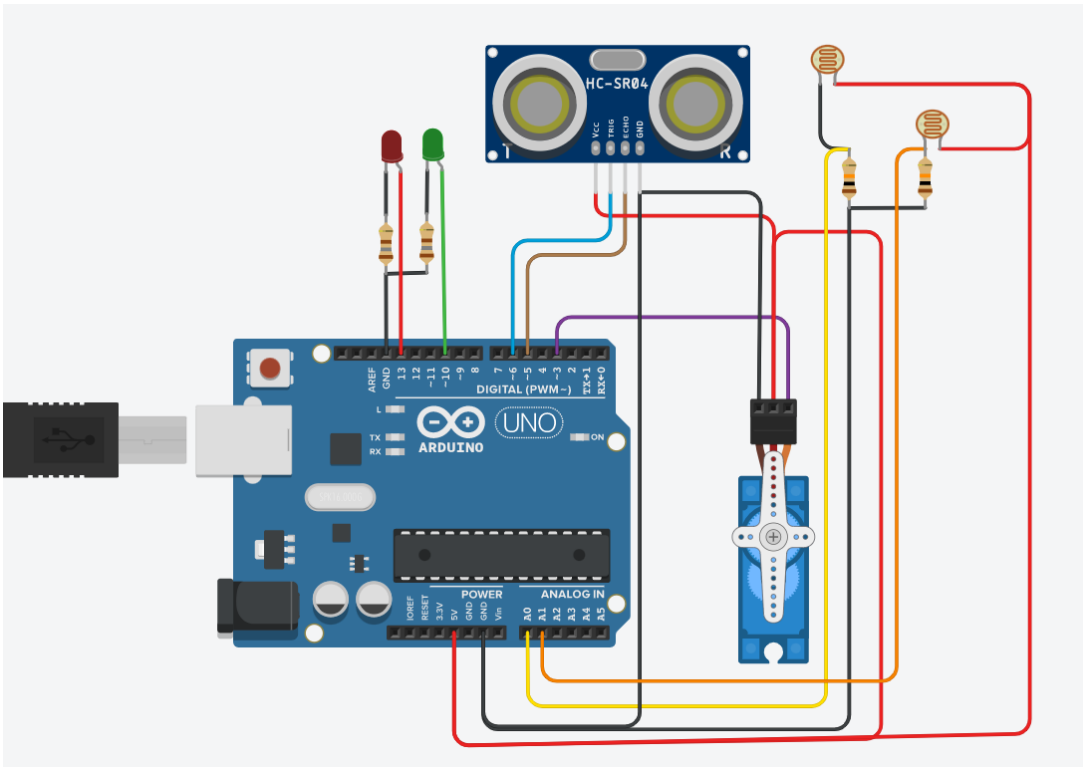
5. Orientación según la acometida de la luz, izquierda – derecha.

Ahora bien, si reflexionamos sobre nuestro diseño. ¿Podemos saber si la luz nos llega de la derecha o de la izquierda con un solo sensor? Pues no, claro. Si que podemos calcularlo mediante algoritmos de movimiento y comparaciones, pero, si tenemos dos sensores en nuestro pack…vamos a usarlos.

Si tenemos 2 sensores, la cosa cambia, porque la filosofía es distinta. El sol estará al frente, cuando nos llegue un nivel de luminosidad similar en los dos sensores al mismo tiempo y, por ende, si recibimos más por la derecha, debemos incrementar el ángulo, y si es al contratrario, por la izquierda, incrementarlo, hasta que se igualen.

Así que, lo primero que debemos hacer, es introducir otro sensor en nuestro montaje (y una resistencia de 10K también) en el terminal A1 (naranja).

6. Esquema modificado a 2 sensores de luz



7. Nueva lógica de funcionamiento

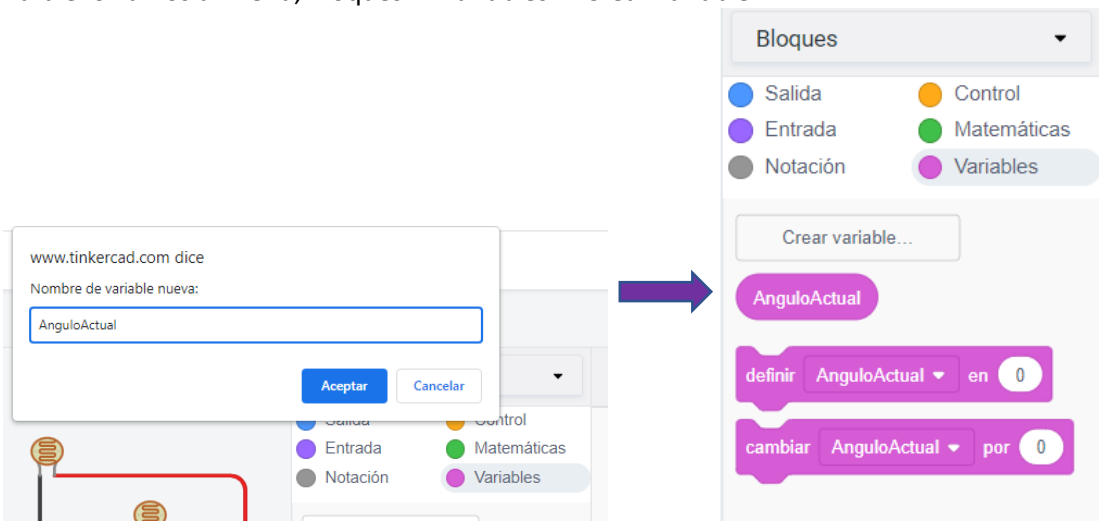
En este punto, nos replanteamos la lógica del funcionamiento en pseudocódigo para nuestro control de posición:

- Si LuzA0=LuzA1 Manten posición (=)
- Si luzA0<LuzA1 incrementa ángulo (+1)
- Si luzA0>LuzA1 decrementa ángulo (-1)

8. Creación de variables de cálculo

Ahora podemos ver que para, poder incrementar y decrementar el ángulo, vamos a necesitar guardar el valor del ángulo en una variable con la que poder operar.

Así pues, creamos la variable "AnguloActual", en la que guardaremos el valor del ángulo actual en el que se encuentra el servo.
Para ello vamos al menú, Bloques→ Variables→ Crear variable…



Tras crearla, nos aparecerán unos nuevos bloques que se corresponden con el manejo de esta variable. Redondeado, como valor de la variable o en bucle, para asignar valores a las variables.

- Angulo constante: Primer bloque
- Angulo incremento: Cambiar AnguloActual por Angulo actual + 1
- Angulo incremento: Cambiar AnguloActual por Angulo actual – 1

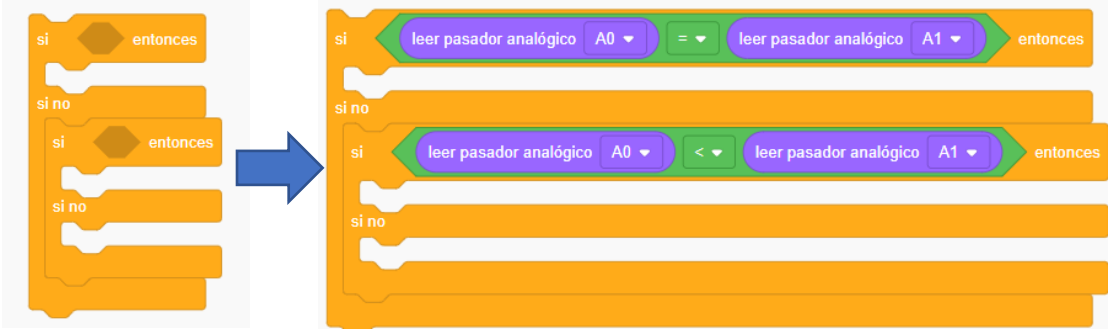
9. Programación de nueva lógica de incrementos y decrementos

Sin más que actuar con operaciones matemáticas, recuperamos el pseudocódigo lógico aplicando un bucle anidado (un si dentro de otro si-sino):

- Si LuzA0=LuzA1 Manten posición (=)
- Si no:
  - Si luzA0<LuzA1 incrementa ángulo (+1)
  - Si no luzA0>LuzA1 decrementa ángulo (-1)

Vamos a verlo en tinkercad. Primero el solo y luego, dentro de lo ya programado con anterioridad.

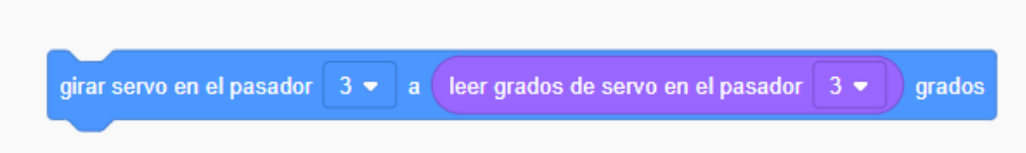
En primer lugar introducimos los 2 bucles si – sino, anidados entre ellos. Luego introducimos la comparación (hexagonal verde) y dentro de ellos, las lecturas de las entradas analógicas A0 y A1.



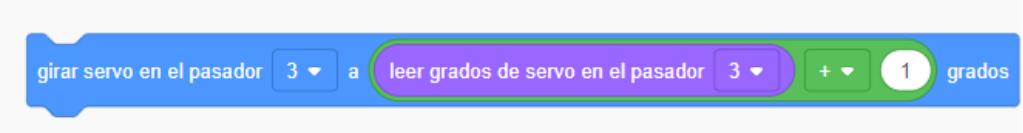
Ahora, mantener el servo en el posición actual, significa, que el valor leido de posición sea igual al actual. Por ejemplo, podemos no decirle nada para que no se mueva, o decirle, vete a la posición actual. La posición actual nos la ofrece tinkercad en las entradas como “lee grados del servo en el pasador…”.

10. Ordenes de posición al servo en tiempo real

Así pues, ordenamos al servo ir a la posición leida del servo.



Si lo que queremos es que el servo incremente la posición 1 grado, tendremos que sumar 1 a los grados actuales y pasárselos al servo en la salida. Para ello necesitaremos operaciones matemáticas.

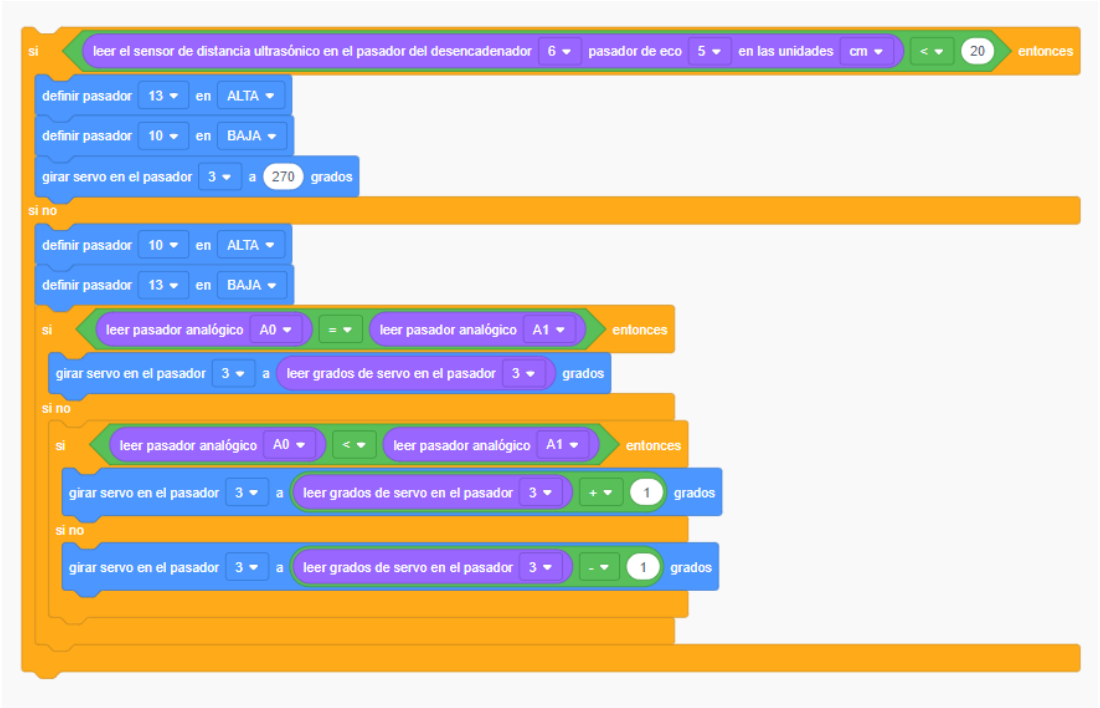


Y lo mismo para decrementar los grados. Quedándonos la programación de la orientación de la siguiente manera:

11. Bucle de funcionamiento del servo en función la luz

Ahora que ya lo hemos programado, debemos integrarlo en nuestro programa principal, dentro del bucle if general que habíamos diseñado. De esa manera, integramos la función de girarse y rojo, cuando alguien se acerque mucho, y, cuando estñe en verde, que permita orientarse al sol.

Así pues, este ultimo bucle de orientación, debemos introducirlo dentro del "si no" del anteriomente programado, eliminando el "si-sino" anterior de prueba, sin olvidarnos de introducir el encendido de la luz verde y apagado de la roja.



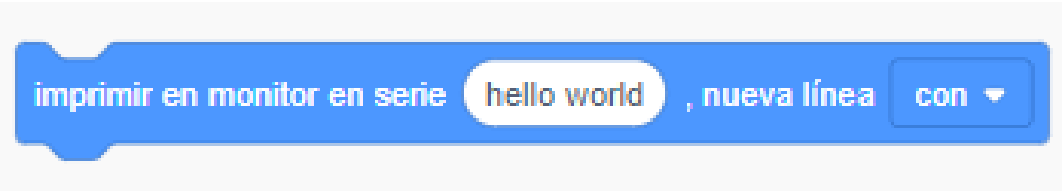
De esta forma, vemos como el sistema responde a nuestra programación, pero se produce un error en el funcionamiento, y es que no se detiene el servo. En realidad no es un error, es que, en la realidad, cuando se esté reorientando el panel, el sensor notará un cambio de luminosidad que nosotros no estámos simulando (deberíamos ir moviendo la barra hasta igualar los dos).

c) *Introducción al MONITOR SERIE*

En este punto, vamos a introducir un concepto que nos ayuda mucho dentro de la programación con arduino y el MONITOR SERIE. Gracias a él, podemos saber el valor de variables o imprimir mensajes que nos permitan revisar o interactuar con el programa.

Para utilizarlo, solo tenemos que decirle lo que queremos que nos muestre al simular, en este caso, los valores de los 2 sensores de luminosidad (A0 y A1).

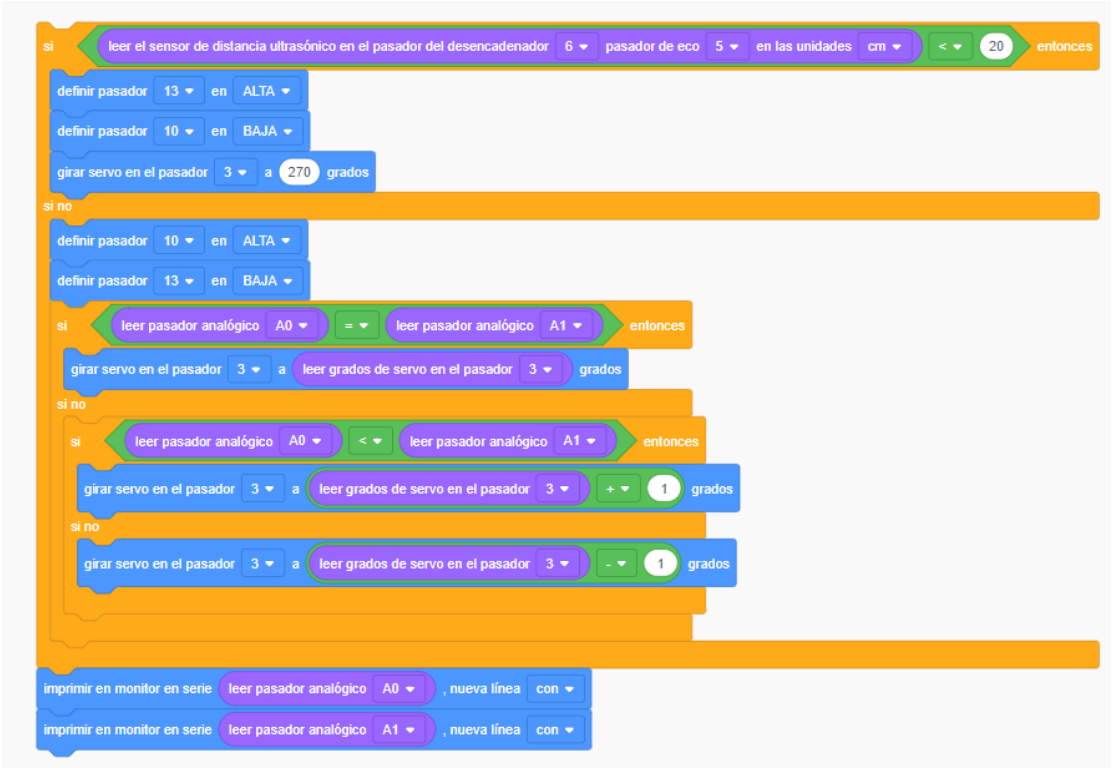
Para ello vamos al menú SALIDA y seleccionamos el bloque "imprimir en monitor serie…".
Y seleccionamos nuestras variables.



Dentro de la elipse "hello world" podemos:

- Escribir el texto que deseeamos
- Colocar la variable que queramos aparezca el valor
- Introducir cualquier funcioón circular de lectura de entradas, salidas…

En nuestro caso, introducimos dos bloques, uno para cada variable, y lo anidamos al final de nuestra programación para poder ver los valores de lectura de nuestros sensores de luz.



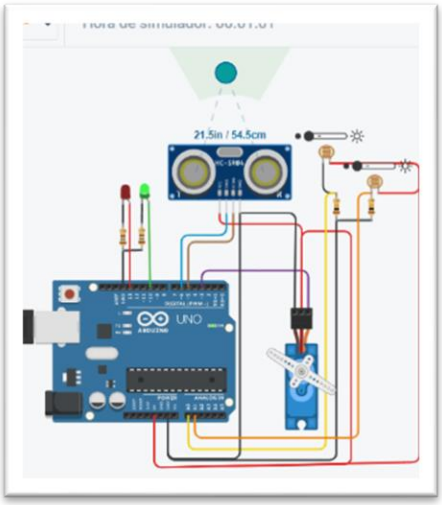
Al pulsar “iniciar simulación” nos aparece abajo “monitor serie” pulsamos, y nos aparecerán los valores leidos. Girando la barra, podemos ver que serán entre 54 y 974 en modo simulación.

Ahora, que conocemos al detalle este sistema, si podemos simular el funcionamiento completo.

d) *Pruebas finales de simulación*

PASO 1: Arrancamos simulación

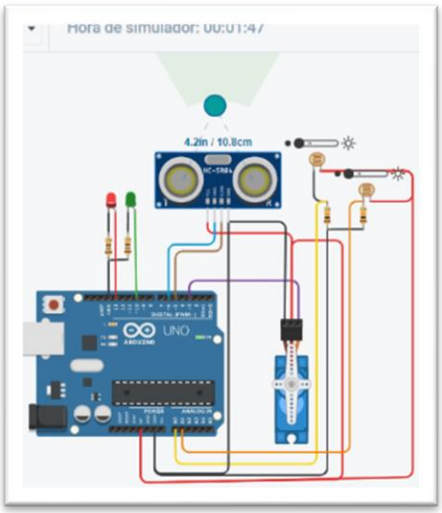
Sensor de presencia a 40cm, por ejemplo Sensores de luz al mínimo, 54.
El servo arranca en la última posición conocida.
Luz verde.



PASO2: Bajamos la distancia a 10cm

Sensor distancia 10cm
Luz roja
Servo a 270º
Sensores de luz al mínimo

Podemos comprobar que si actuamos sobre los sensores de luz, nada ocurre. Está en modo "vergüenza" protegiendose de un acercamiento indebido.

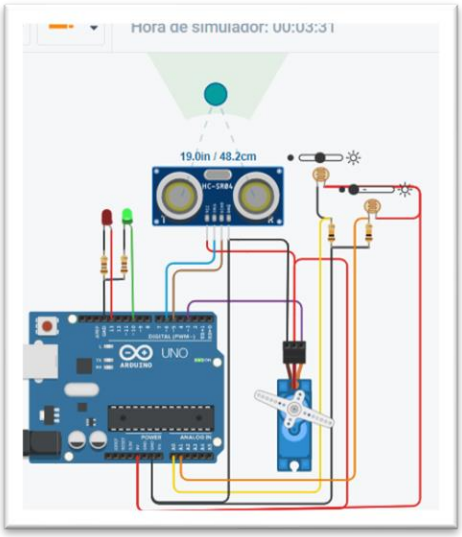


PASO 3: Recuperación

Volvemos a 50cm y todo sigue igual, salvo la luz verde (roja off).
El servo no se mueve, ya que, al leer la luminosidad igual en A0 que en A1, mantiene la posición.

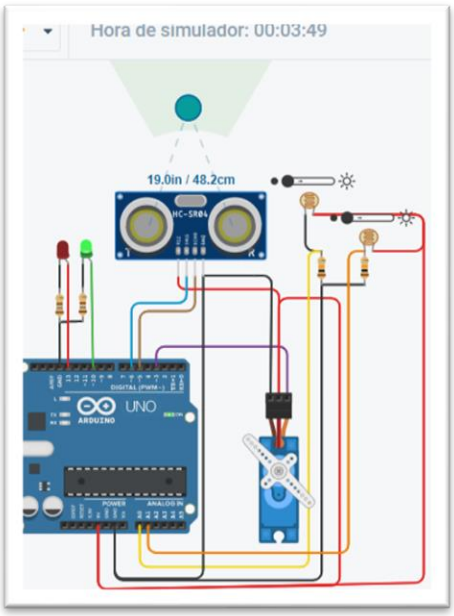
PASO 4: Luz izquierda

Subimos el sensor A0.
En el momento que lo subimos, el angulo empieza a crecer, el servo se orienta.



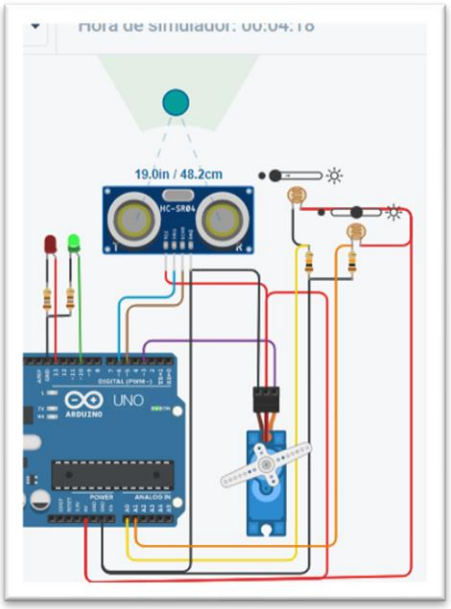
PASO 5: Orientado

Bajamos el sensor A0, para simular que se han igualdo los valores de A0 y A1.
El servo se para.



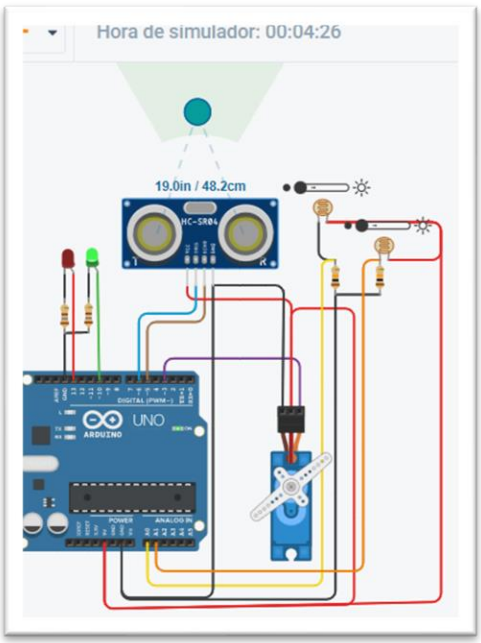
PASO 6: Luz derecha

Subimos el sensor A1.
En el momento que lo subimos, el angulo empieza a decrecer, el servo se orienta.



PASO 5: Orientado

Bajamos el sensor A1, para simular que se han igualdo los valores de A0 y A1.
El servo se para.



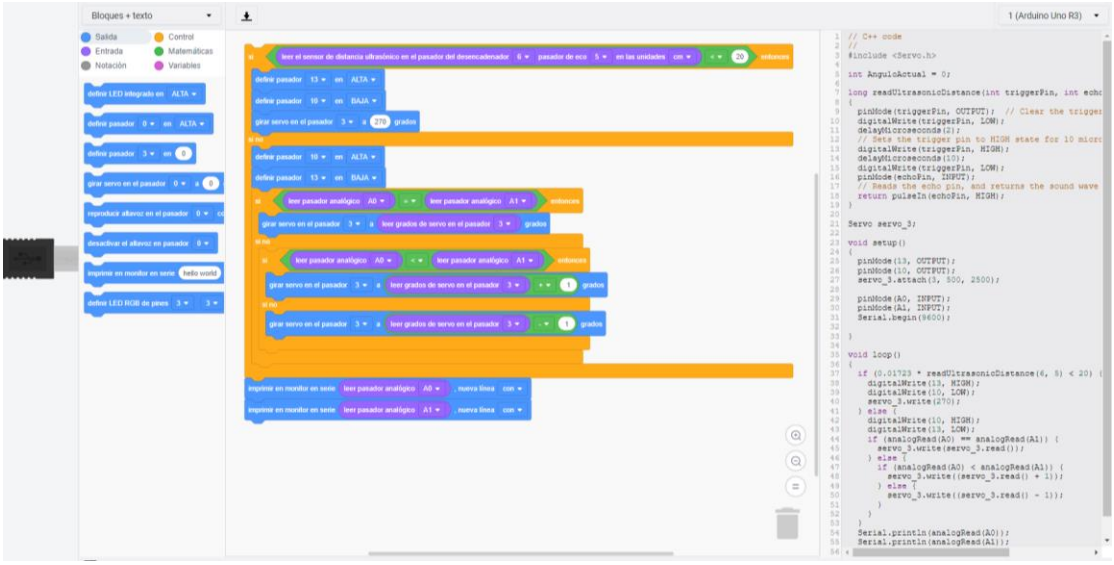
Bueno, pues lo hemos conseguido. Hemos programado y simulado nuestro robot miedoso, ahora nos quedará montarlo.

LO HICIMOS…SIMULADO.

e) Apuntes a la programación

Aunque lo veremos más adelante, tinkercad no nos permite comunicarnos con el ARDUINO directamente, ya que es un simulador online. Para pasar el programa a una tarjeta real, usaremos el software de arduino UNO oficial.

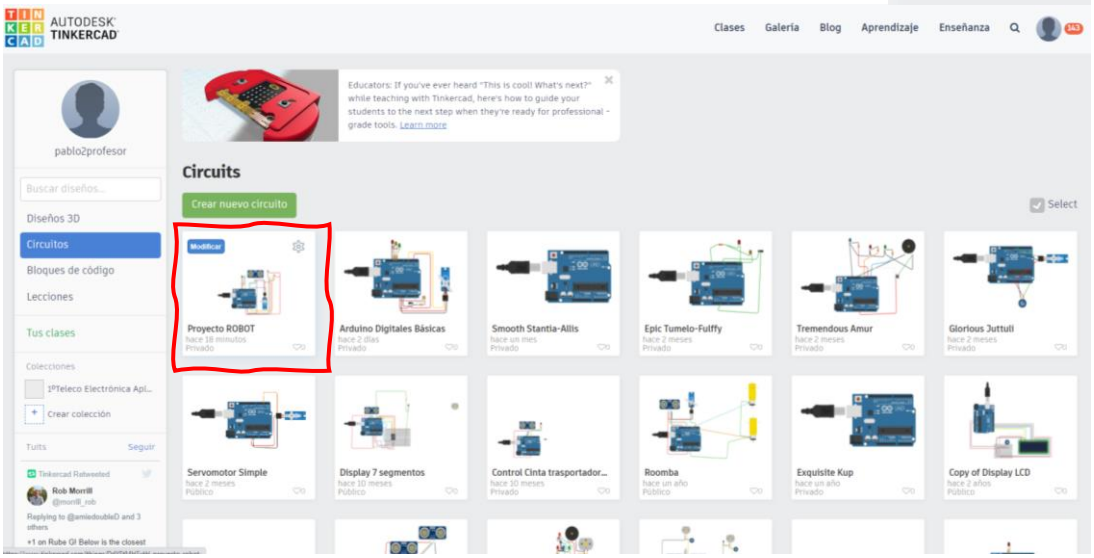
Como algunos de vosotros ya sabréis, arduino se programa en lenguaje de alto nivel (C++). Así pues, tendremos que copiar el código de tinkercad y pegarlo en la aplicación de Arduino. Aquí podemos ver las 57 lineas de nuestro programa (Vista de Bloques +Texto)



IMPORTANTE: Si se selecciona el modo SOLO TEXTO, desactiva bloques y no se puede volver atrás.

f) Apreciaciones sobre el funcionamiento de Tinkercad

Lo primero siempre, **guardar el diseño**. Tinkercad lo hace directamente porque estamos en servidor, pero recordar también, lo que borremos, al volver a abrirlo de nuevo, no se podrá deshacer. Si borramos un diseño de nuestro escritorio tinkercad, se fue para siempre.
Si pulsamo sobre el icono de TINKERCAD (colores) , nos llevará a nuestro escritorio de programación donde podremos ver los diseños colocados por fecha de acceso (de más reciente a más longevo)



Que nadie se asuste si ve nombres “raros”, si no le ponemos un nombre al proyecto, Tinkercad nos propone un nombre por defecto bastante curioso cada vez.

3.4.4. MONTAJE FÍSICO SOBRE ENTRENADOR

a) Kit de montaje

Para ellos contamos con los KIT de Arduino Elegoo MEGA 2560, que se componen de:

Del KIT ELEGOO MEGA2560:

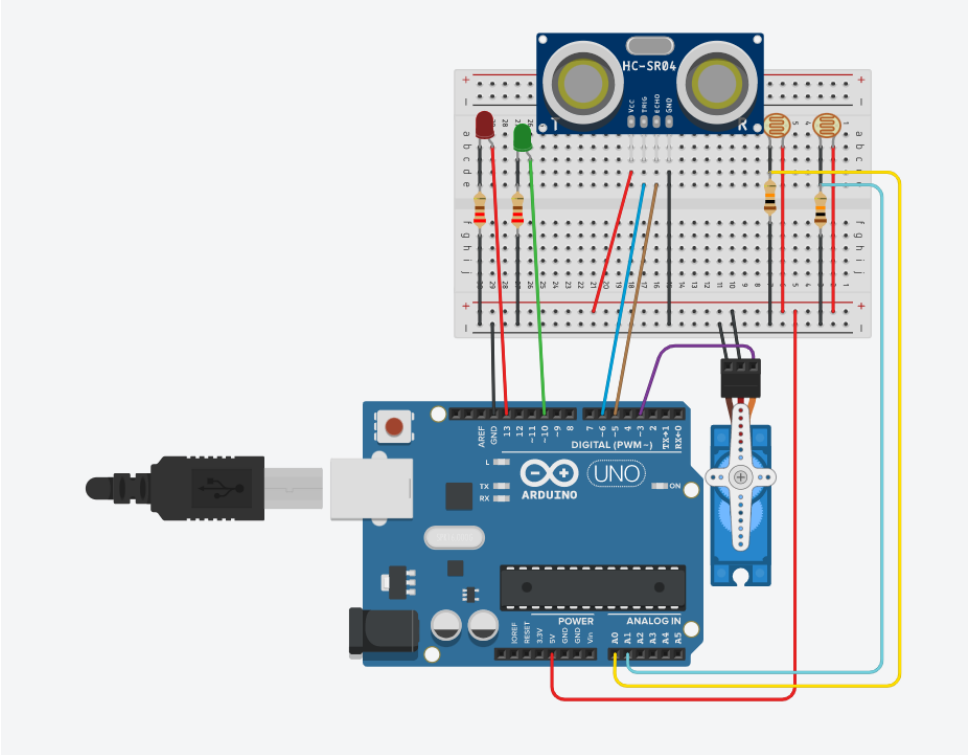
- 1 Ud. Placa MEGA: 1pcs Mega 2560 Controller Board
- 1 Ud. Cable USB
- 1 Ud. Servomotor
- 2 Ud. Fotorresistencia
- 2 Ud. Resistencia 10K ohm.
- 1 Ud. Led Rojo
- 1 Ud. Led Verde
- 2 Uds. Resistencias 220 ohm
- 1 Ud. Sensor ultrasónico SR06
- 1 Ud. Placa de conexiones
- 1 Ud. Pila
- Cables y conectores.

https://www.elegoo.com/products/elegoo-mega-2560-the-most-complete-starter-kit

Ahora tomaremos lo que nos hace falta del kit, comprobando que nuestro diseño es correcto y que tenemos todos los componentes.
Poco a poco y fijándonos bien, vamos a realizar un primer montaje directo entre la placa de Arduino y la protoboard, con los elementos que hemos definido.

b) Montaje

1. **Conexión placa ARDUINO y Protoboard**

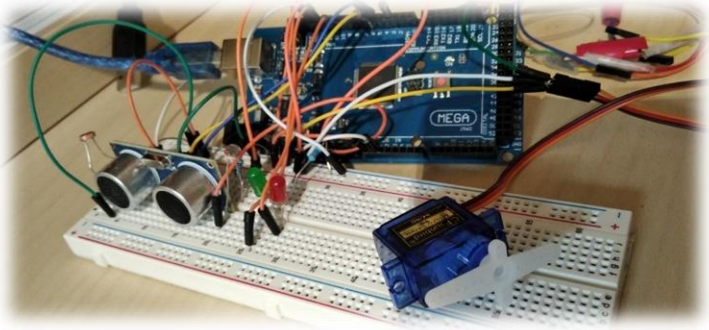


Vamos a facilitar las cosas fijándonos en como lo hemos cableado en Tinkercad, pero implementando una protoboard.

Esto nos ayuda a cablear en puntos (nudos) donde tenemos más de 2 conectores. En este ejemplo, tenemos:

- Conexión común para 5Vcc
- Conexión común para GND.
- Salida de las fotorresistencias
- Resistencias, led…

Ya lo sé, en Tinkercad queda muy limpio, pero en la realidad…

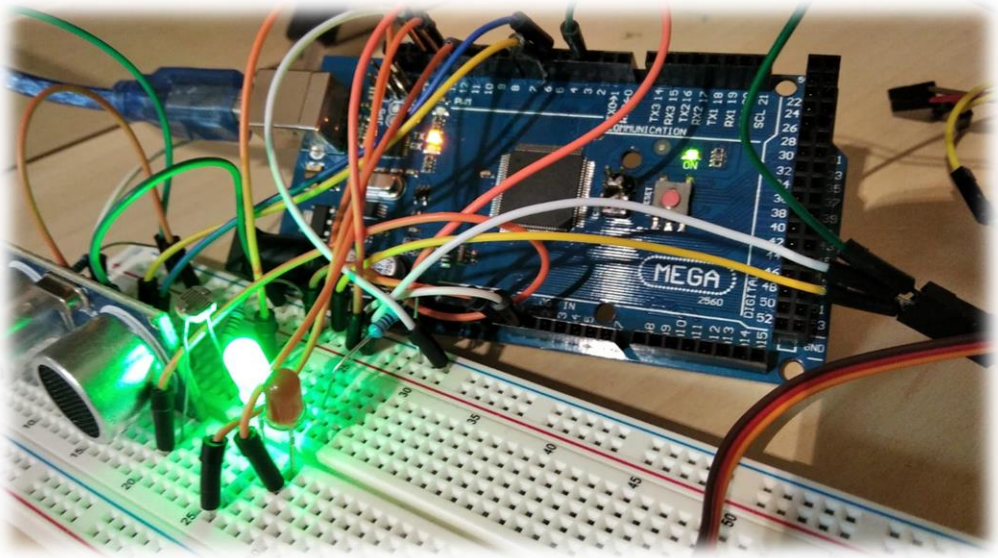


2. Conexión placa ARDUINO al PC

Simplemente, enchufamos el USB a nuestro PC y veremos que la placa se ilumina. También veremos que quizás se nos enciendan los LED u otras consideraciones de arranque que tenga que hacer la placa por defecto, ya que no tiene programa o tendrá un programa anterior.

Principalmente veremos:

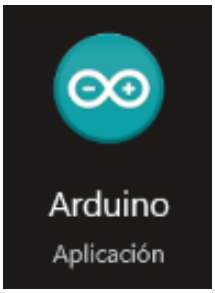
- ON (alimentación correcta y preparada para trabajar)
- TX-RX (pines de comunicaciones envío-recepción con el PC)



Ahora, que ya lo hemos montado y cableado, pasamos a pasarle el programa por el USB.

3.4.5. PROGRAMACIÓN, COMPILACIÓN Y CARGA DEL PROGRAMA EN EL MICROCONTROLADOR ARDUINO

Como hemos mencionado con anterioridad, Tinkercad no nos permite comunicarnos directamente con nuestra placa, así que, debemos utilizar un software compilador que pueda cargar el programa físicamente a la placa.
El programa que usaremos será el oficial de ARDUINO, abriendo la aplicación en nuestro PC.
https://www.arduino.cc/en/software



1. **Al abrir el programa**

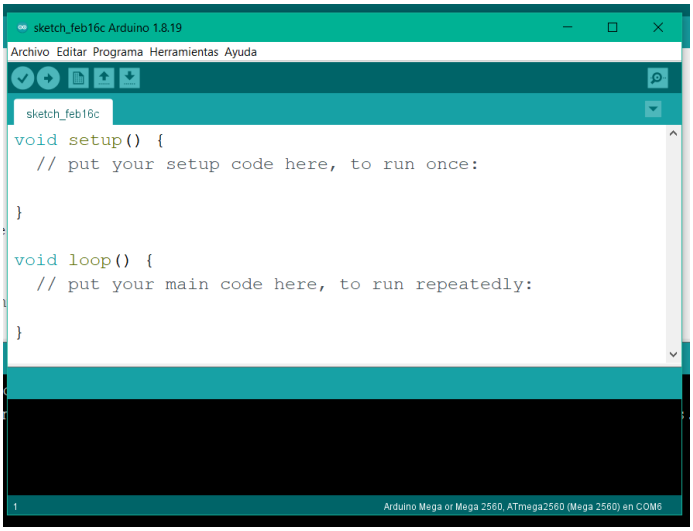
   el software nos abre el último archivo abierto o, en su defecto un programa nuevo.

   Como podemos ver, los procesadores entienden de lenguaje de instrucciones, así que, no tenemos la opción de cargarle bloques.

   Por resumir sin entrar a fondo, el programa se divide en tres zonas:

   - Zona superior: Definición de variables, llamada a funciones…
   - Zona intermedia: VOID Setup, inicialización de variables, puerto serie…
   - Zona media: VOID Loop, el programa, todas las instrucciones que el microprocesador va a realizar de forma cíclica y continua.
   - Zona baja: Funciones a llamar programadas en el código del programa.

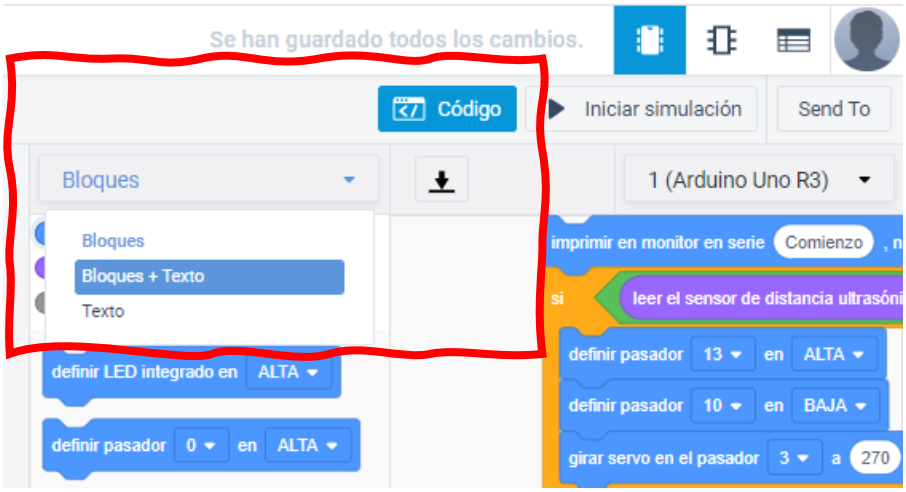
Esta es la pinta que nos ofrece un programa nuevo.



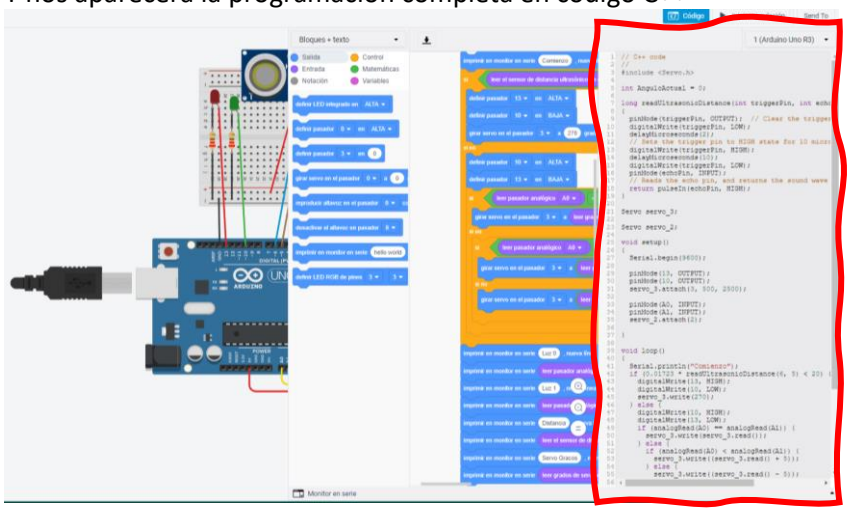
2. Abrir Tinkercad en modo programa en C++

Ahora, es cuando necesitamos nuestro programa de Tinkercad en texto. Así pues:

- Abrimos la aplicación Tinkercad Online
- Abrimos nuestro diseño-circuito
- Pulsamos CODIGO
- Seleccionamos BLOQUES+TEXTO



Y nos aparecerá la programación completa en código C++



Este código, debemos copiarlo de inicio a fin, directamente de la pantalla de Tinkercad y pegarlo en la pantalla de Arduino (vacía).De esta manera, tenemos el programa listo para enviárselo a la placa de Arduino.

Analicemos un poco su contenido. En este caso, me he permitido de la licencia de comentarlo en castellano al completo, para que se puedan entender todas las líneas de trabajo.

3. Copiar en TINKERCAD – pegar en ARDUINO

ZONA SUPERIOR: Antes de "**void setup ()**"

Proyecto_ROBOT_miedoso_Luminoso §

```
// C++ code Comentado por Pablo2profesor

//Incluimos una función ya programada por desarrolladores que nos permiten mover el servo y saber su posición.
#include <Servo.h>

// Declaramos la variable AnguloActual como un valor entero, y además lo ponemos a 0.
int AnguloActual = 0;

// Esta función también esta preprogramada (estándar) para calcular la distancia en función del tiempo que tarda
// la señal desde que se envía (trigger) hasta que se recibe (eco).
long readUltrasonicDistance(int triggerPin, int echoPin)
{  pinMode(triggerPin, OUTPUT);  //Declara el valor triggerPin (el terminal por el que lo definamos al llamar a la función) como salida
  digitalWrite(triggerPin, LOW);// Apaga el disparador, para aseguranos que está a cero al inicio.
  delayMicroseconds(2);         // Espera 2 microsegundos para asegurarnos que se apaga.
  digitalWrite(triggerPin, HIGH);  // Enciende el disparador.
  delayMicroseconds(10);          // Espera 10 microsegundos, tiempo que necesita para lanzar la señal.
  digitalWrite(triggerPin, LOW);  // Apágalo de nuevo, el disparador.
  pinMode(echoPin, INPUT);        //Declara el pin de entrada del eco.
  return pulseIn(echoPin, HIGH);  // Lee el pin del eco, y devuelve el tiempo que a tardado la onda de sonido en volver en microseg
}
//Define la función SERVO que va a utilizar con el pin 3 de salida
Servo servo_3;
```

ZONA INTERMEDIA: "**void setup ()**" Inicialización de variables.

```
void setup()
{
  Serial.begin(9600);   //Arrancamos el monitor SERIE para ver comentarios o valores
  pinMode(13, OUTPUT);  //Definimos el pin 13 como salida (Led Rojo)
  pinMode(10, OUTPUT);  //Definimos el pin 10 como salida (Led Verde)
  servo_3.attach(3, 500, 2500); //Definimos el servo en el pin 3, ademas pulso mínimo para 0°(500) y máximo 180° (2500)
  pinMode(A0, INPUT);   //Definimos el pin analógico A0 como entrada (fotorresistencia 1)
  pinMode(A1, INPUT);   //Definimos el pin analógico A1 como entrada (fotorresistencia 2)
}
```

ZONA MEDIA: "**void loop ()**" Bucle de programa

```
void loop()
{
  Serial.println("Comienzo"); //Escribo en el puerto serie la palabra Comienzo y salto de línea
  if (0.01723 * readUltrasonicDistance(6, 5) < 20) {  //Realiza la función de CM con los pines 6 y 5 y, si es <20cm entonces
                                                      //0.01723 es elcoeficiente de cálculo de tiempo a distancia
      digitalWrite(13, HIGH); // Enciende luz Roja
      digitalWrite(10, LOW);  // Apaga luz Verde
      servo_3.write(270);     // Lleva a 270° al servo del pin 3
   } else {    // Sino, es decir, si cm>20
       digitalWrite(10, HIGH);  // enciende luz verde
       digitalWrite(13, LOW);   // apaga luz roja
          if (analogRead(A0) == analogRead(A1)) {  //Si se cumple, además que la lectura de la luz de A0 es igual a la de A1
              servo_3.write(servo_3.read());       // Deja el servo en la misma posición actual
            } else {
               if (analogRead(A0) < analogRead(A1)) {  // Si Lectura A0<A1
               servo_3.write((servo_3.read() + 5));    // Incrementa la posición actual del servo en +5°
            } else {                                   // Si no, es decir, Lectura A0<A1
               servo_3.write((servo_3.read() - 5));    // Decrementa la posición actual del servo en -5°
      }
    }
  }

  Serial.print("Luz 0");  //Escribe en el monitor serie la palabra Luz 0
  Serial.println(analogRead(A0)); //Escribe en el monitor serie el valor de A0 y pasa a la siguiente línea
  Serial.print("Luz 1");  //Escribe  palabra Luz 1
  Serial.println(analogRead(A1));  //Escribe el valor de A1 y pasa a la siguiente línea
  Serial.print("Distancia"); //Escribe  palabra distancia
  Serial.println(0.01723 * readUltrasonicDistance(6, 5)); //Escribe el valor de la distancia calculada por la función
  Serial.print("Servo Grados"); //Escribe  palabra Servo Grados
  Serial.println(servo_3.read());  //Escribe el valor del servo
  delay(100); // Pequeño retraso para poder ver la simulación bien
}
```

Este es el programa completo que nos encontramos, comentado línea a línea.

En nuestro caso, tomamos el programa directamente del código en Tinkercad (trae algún comentario en inglés) y lo pegamos en nuestro Arduino.

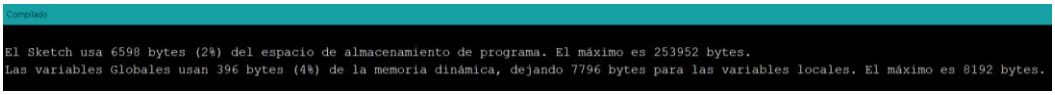
4. COMPILAR programa.

Por último, nos queda ***COMPILAR.***

Es la primera comprobación para ver que lo hemos hecho bien, sin dejarnos nada atrás. Pulsaremos el botón de  compilar y esperamos a que termine su misión.

Si aparece algún error (barra de abajo roja), nos indicará que tipo de error ha encontrado, y nos dará la opción de copiarlo para buscar en internet.

Si todo sale bien, nos dará su visto bueno y nos indica la memoria que ocupa en nuestro procesador.



Hasta aquí todo bien.

3.4.6. CARGA DE PROGRAMA Y PRUEBAS

Ahora es el momento de cargar el programa (por fin) a nuestra placa, y para eso, debemos seguir los siguientes pasos:

PASO 1: Abrir ARDUINO (aplicación)

PASO 2: Copiar Código (texto) de Tinkercad

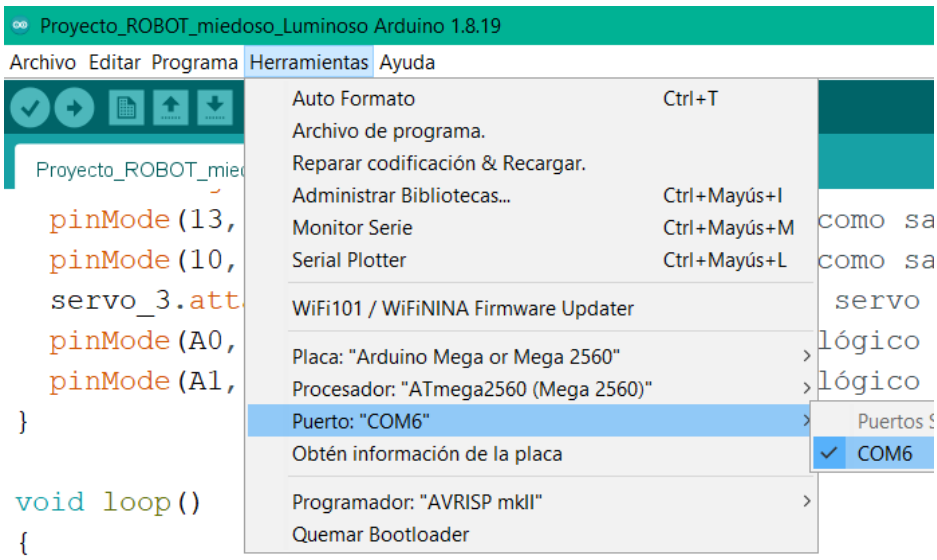
PASO 3: Pegar código copiado en ARDUINO

PASO 4: Compilar programa

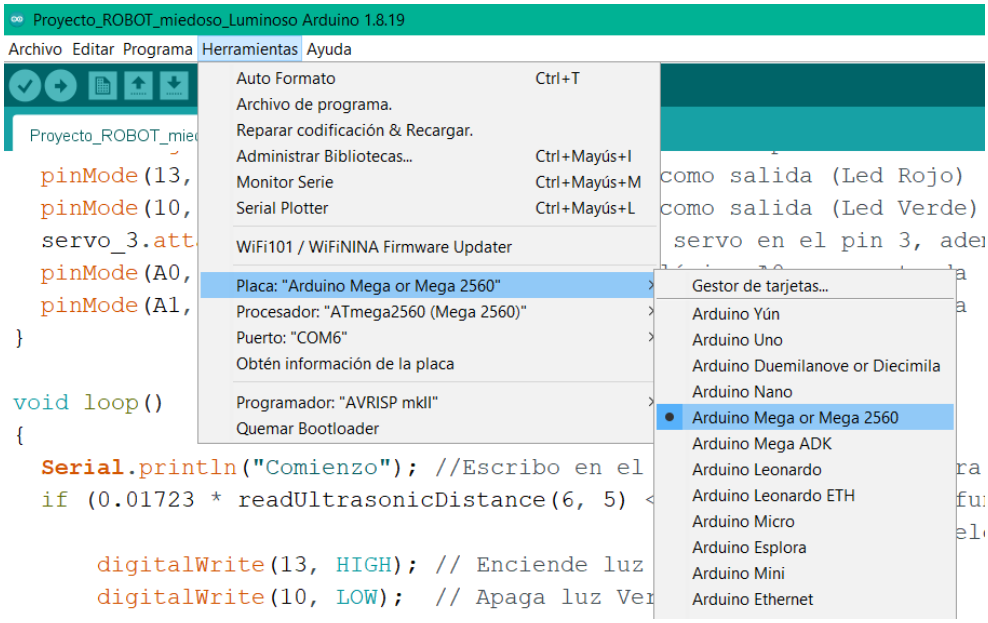
PASO 5: Conectar USB de la placa ELEGOO Mega al PC

PASO 6: Comprobar la conexión del puerto serie.

A cada placa, cuando se conecte por primera vez, se le asigna un puerto serie numerado correlativamente. Seleccionemos el puerto que nos haya aparecido nuevo. En este caso COM 6.



PASO 7: Seleccionar la placa de Arduino con la que vamos a trabajar. MEGA 2560



El resto, se nos configura por defecto.

PASO 8: CARGAR

Ahora ya, podemos cargar el programa, sin más que pulsar la fecha 

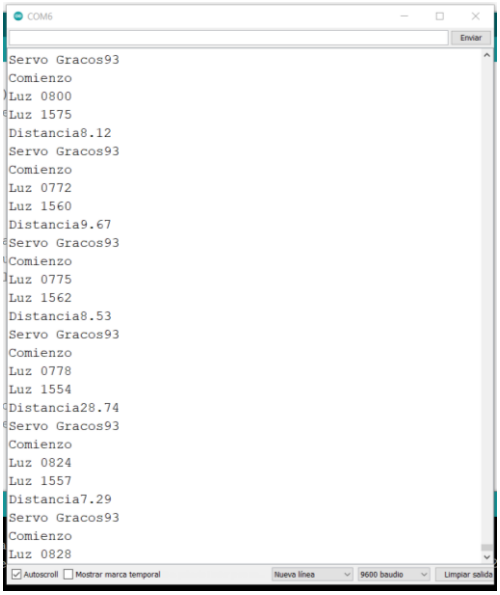
Ya tenemos cargado nuestro **programa cargado y funcionando en nuestra placa ARDUINO real.** Ahora toca probar si todo funciona como queríamos.

PASO 9: MONITOR SERIE

Pero primero, vamos a ver como abrir el **MONITOR SERIE** para poder ver las lecturas y mensajes de la placa.

- Opción 1: Ir directamente al botón de la  lupa.
- Opción 2: Menú HERRAMIENTAS → MONITOR SERIE

Entonces nos aparecerán los mensajes en tiempo real definidos en el programa:



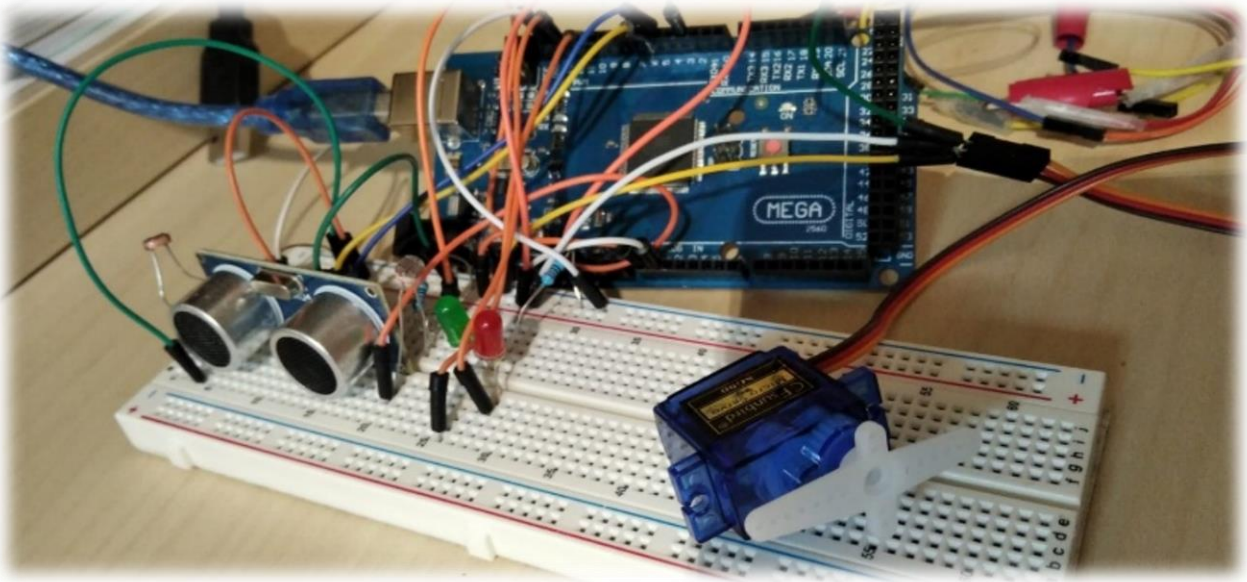
Tenemos opción de:

- Enviar: Datos a nuestra placa.
- Limpiar salida: borrar todo en ese instante.
- 9600 baudio: cambiar la comunicación serie.
- Configuraciones de línea

Los servos Grados son como los grados del servo, pero escrito rápido 😊.

Nuestro programa se encuentra totalmente cargado y probado. Ahora es el momento de ajustar parámetros:

- De distancia (distancia a la que se asusta el robot)
- De luz (Cambiar umbrales de luz de orientación)
- De grados (por ejemplo, este servo azul, sólo va a 180º, así pues, los 270º que le indicamos, nunca pasarán de 180º. En otros servos, sí)

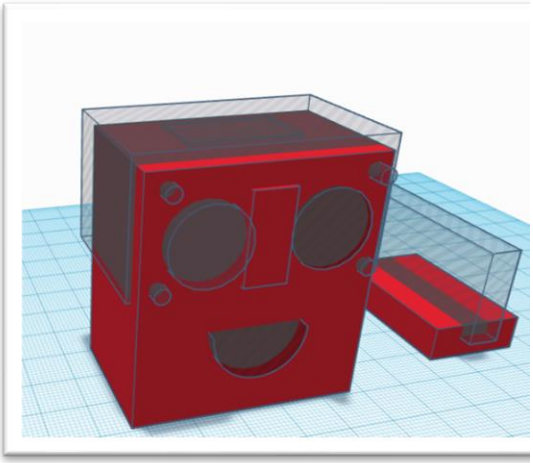


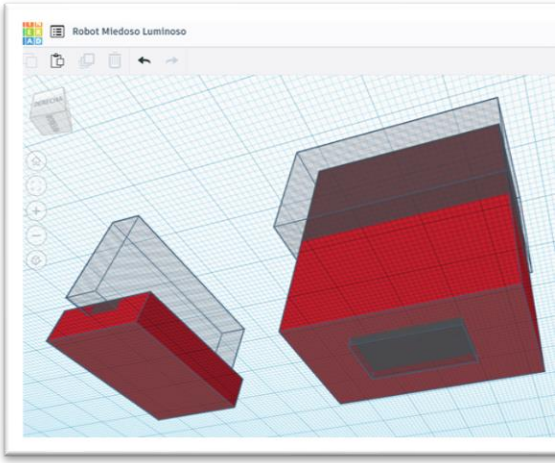
3.4.7. MONTAJE SOBRE PIEZAS EN 3D: ROBOT

Ahora es cuando toca que los alumnos puedan ver que es un robot de verdad, es decir, toca vestirlo.

Para este proyecto hemos planteado un diseño sencillo, basado en Tinkercad (diseño 3d) para, de forma rápida y fácil, poder montarlo.

Será tarea del profesor, adaptarlo al nivel de sus alumnos. Desde montarlo sobre un cartón doblado y silicona caliente, hasta diseños profesionales en programas de diseño tipo Solid Edge e impresos en 3d.





Montemos ahora nuestra electrónica, ya programada, en nuestras piezas, y probemos a nuestro pequeño:

- Ojos: sensor de distancia ultrasónico.
- Sobre los ojos, fotorresistencias.
- Bajo los ojos, led, verde y rojo.
- Interior pieza grande, servomotor..
- Soporte del eje del servo, sobre la piecita 2.

Planteamos este diseño como sencillo en tinkercad, francamente mejorable, combinable y cambiable…en cada curso, durante el 3d, imaginaremos nuestras piezas.

4. CONCLUSIONES

Plantemos esta tarea de forma proporcional y gradual, pretendiendo que sea sencilla y apta para todos los niveles.

Tinkercad, como hemos visto, es un software que nos sirve para poder:

1. **Hacer una simulación y programación de un montaje.**
   Pudiendo ser este el objetivo final de la práctica. Orientado a la realidad de la programación sin tomar riesgos de equipos con una curva de aprendizaje interesante. De esta manera, no tenemos porque tener equipos físicos y los alumnos pueden trabajar desde casa con multitud de posibilidades.
   Aquí solo hemos presentado 4 pinceladas, hay un universo Arduino por delante que descubrir.

2. **Probar antes de cargar**
   Podemos hacer nuestro montaje y probarlos antes de pasarlos a las placas reales, evitando así fallos de concepto e iniciales que puedan deteriorar nuestro equipo.

3. **Diseños sencillos en 3d**
   Podemos rematar nuestros diseños de circuitos electrónicos robóticos y automatizados, diseñando nuestros propios soportes y estructuras para imprimir y montar.

   En resumen, Tinkercad nos ofrece la posibilidad de montar nuestros propios juguetes, nuestras propias soluciones y con una comunidad creativa alrededor apoyándonos.

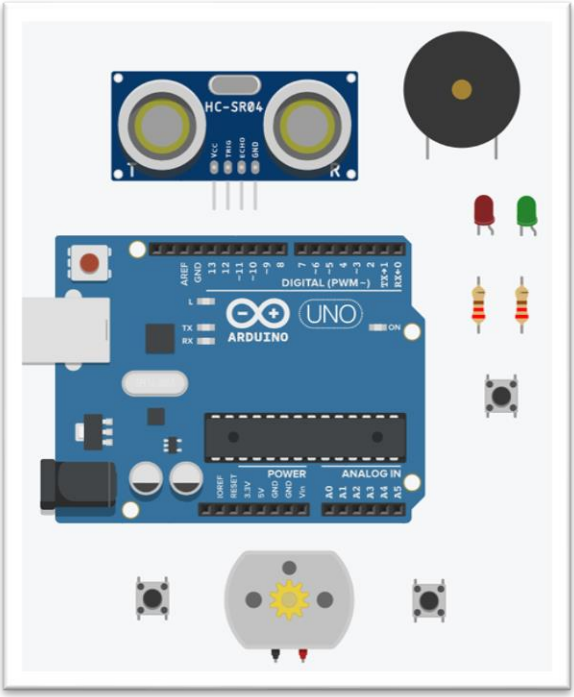
4. MODULO DE APLICACIÓN

Para este curso planteamos un módulo de aplicación que nos permita practicar con lo que hemos aprendido.

El modulo de aplicación se basará en la SIMULACIÓN DE PUERTA DE GARAGE:

4.1. MATERIALES (TINKERCAD)

- Pulsador de apertura de puerta
- Motor de CC
- Detector de distancia
- Led Rojo
- Led Verde
- Sirena (piezoeléctrico)
- Pulsador Final de carrera arriba
- Pulsador Final de carrera abajo
- Resistencias



4.2. FUNCIONAMIENTO (SECUENCIA)

1. Al iniciar la simulación:
   - El motor debe estar parado
   - Luz Roja encendida (prohibido el paso)
   - Luz verde apagada.

2. Pulsamos el pulsador de apertura (pulsador 1)
   - El motor comienza a girar
   - La luz roja sigue encendida y la verde apagada
   - Se enciende la bocina de peligro puerta en movimiento

3. Cuando pulsamos el FC superior (pulsador 2), la puerta ha llegado arriba.

- La luz verde se enciende y se apaga la roja.
- El motor se detiene
- La bocina acústica se detiene
- Se activa un temporizador (retraso) de 5 segundos.

4. Pasados los 5 segundos (tiempo para que un coche pase)
- Se desactiva la luz verde
- Se activa la luz roja
- Se activa la bocina
- Se activa el motor en sentido contrario (puerta bajando)

5. Pulsando el final de carrera abajo (pulsador 3)
- La puerta se detiene
- Se apaga la bocina
- Se queda la luz roja
- Vuelve al estado inicial

6. La puerta posee un sistema de seguridad para proteger ante un golpeo de la puerta al bajar, este funcionamiento consiste en (solo en modo puerta bajando)

- Si el sensor de distancia detecta un objeto (menos de 50cm) → La puerta vuelve al modo abrir.

- Cuando llegue al sensor superior de nuevo, se parará y, después de los 5 segundos, volverá a bajar normal.

4.3. Pistas y ayudas

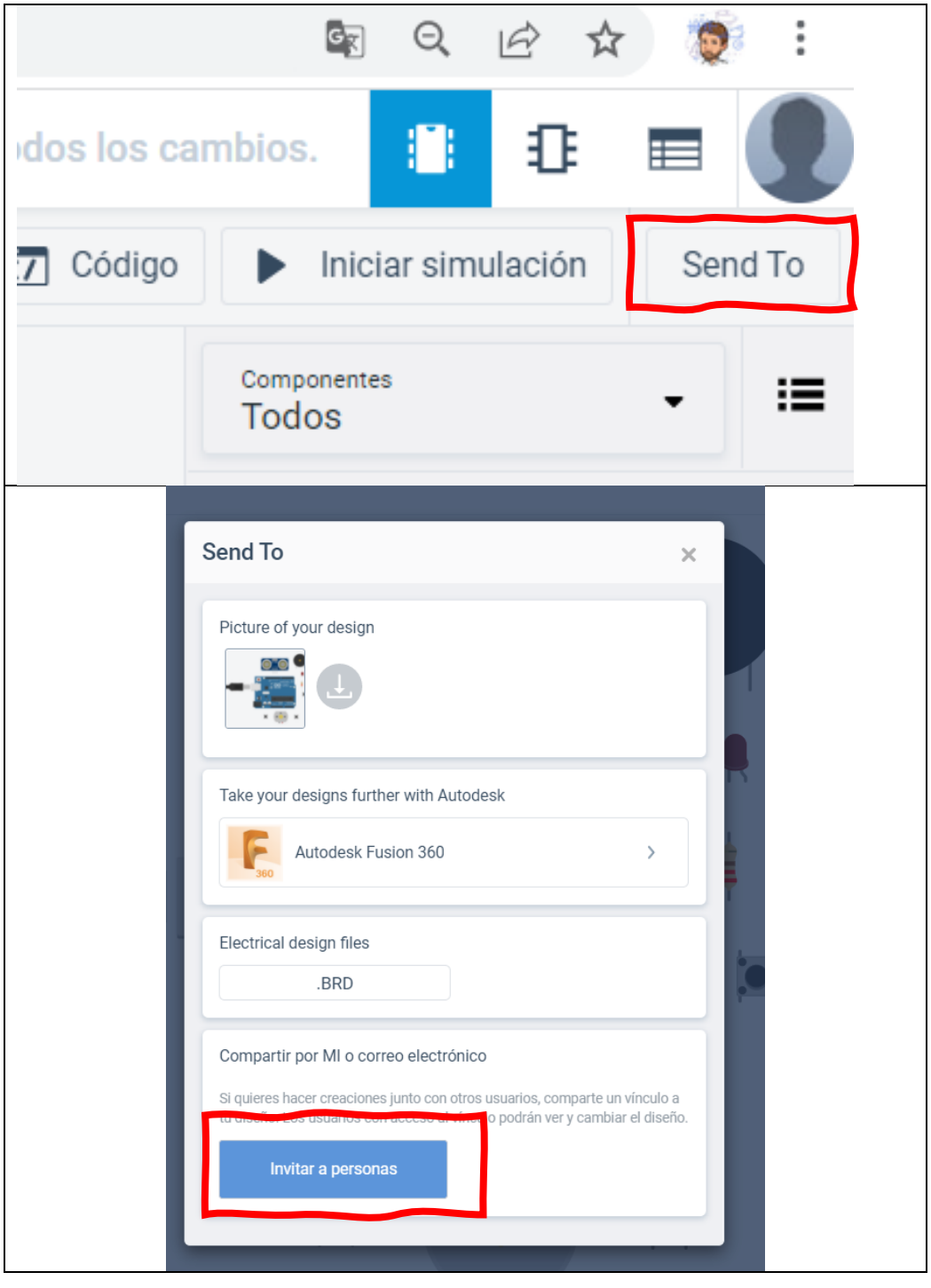
a. Siempre que algo en Arduino lo dudamos, podemos preguntar a Google como se conecta. En este caso, debéis buscar como se conecta el pulsador.
b. La bocina piezoeléctrica es digital todo-nada, así que, su conexión será similar a la del LED.
c. Para cambiar el sentido de giro de un motor, debemos invertir la polaridad de sus terminales (+ por – o – por +).
d. Me tenéis a vuestra disposición en el foro.

4.4. Entregas

Las entregas se deben realizar compartiendo el diseño realizado con el profesor a través del mail del curso.

COMPARTIR

En Tinkercad: **SEND TO** → **Invitar a personas** → **Copiar enlace**

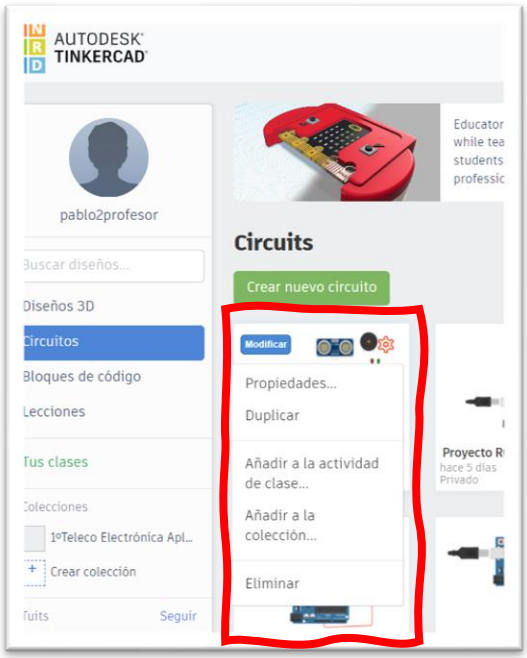


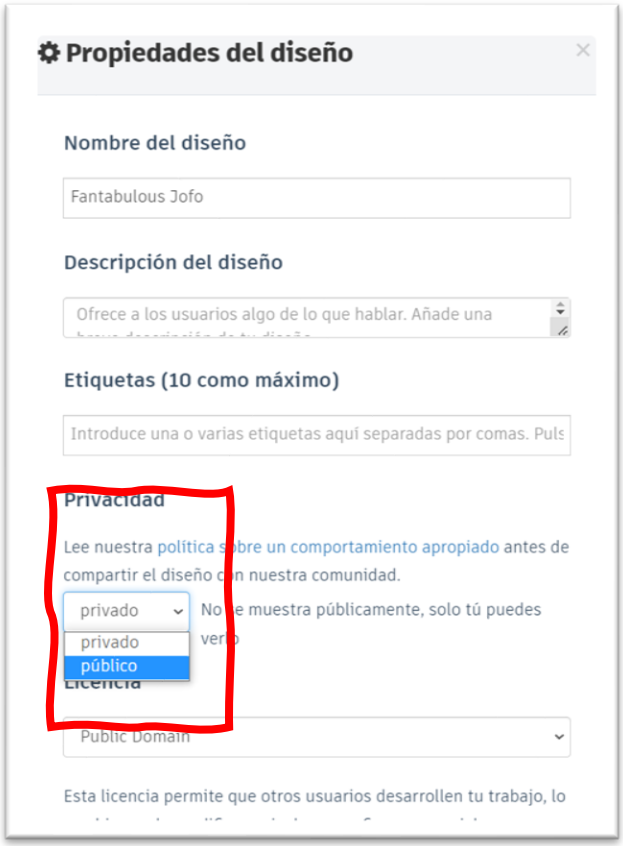
Posteriormente, pegarlo en el cuerpo del correo y enviar al profesor en la tarea, des esta manera, podré entrar en vuestro diseño y probarlo.

PUBLICO

IMPORTANTE. Para poder compartir un diseño, debemos hacerlo público (por defecto no lo es).

Nos vamos a la pantalla de entrada de CIRCUITS de Tinkercad → Pulsamos sobre las propiedades de nuestro diseño → y en privacidad, seleccionamos PUBLICO → Guardar cambios





Muchas gracias por vuestro tiempo y a disfrutar de la robótica…