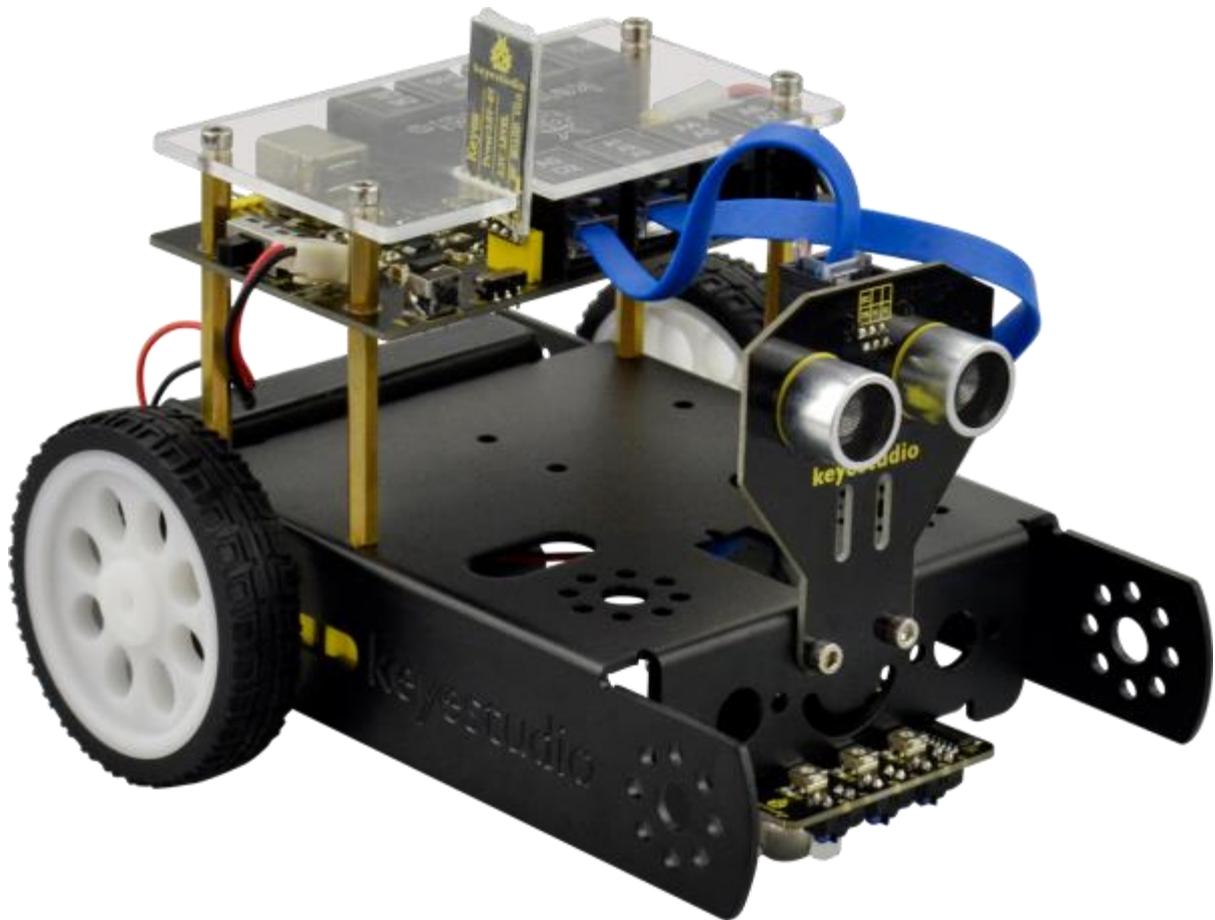


# Manual de Actividades con ArduinoBlocks y el robot KEYBOT (KS0353)



# Índice

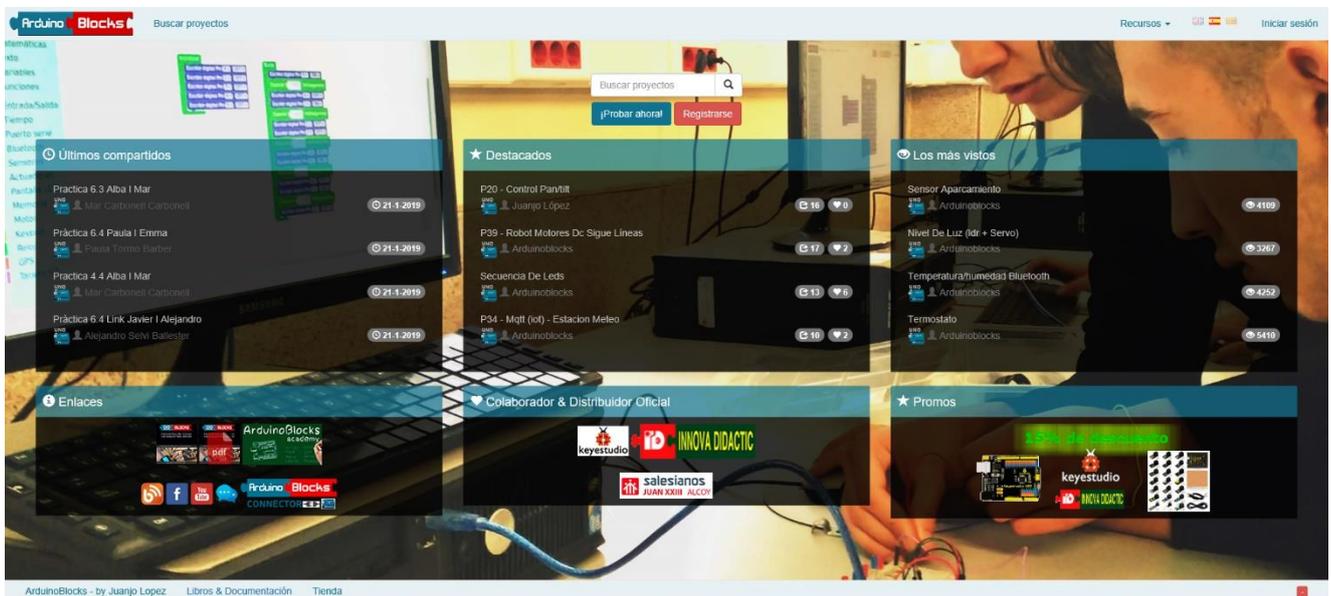
Introducción .....	3
¿Cómo funciona un robot? .....	4
Robot KEYBOT .....	5
Listado del material del Robot KEYBOT .....	6
Guía de montaje del Robot KEYBOT .....	10
Guía de Conexión: .....	20
Placa de control .....	22
Material opcional.....	23
ArduinoBlocks.....	24
ArduinoBlocks y el Robot KEYBOT.....	25
Preparativos: instalación de los drivers y programas .....	25
Programación Arduino con ArduinoBlocks .....	27
Actividades con el Robot KEYBOT .....	31
A01. – Encender/Apagar un LED.....	31
A02. – Control de brillo de un LED usando el PWM. ....	33
A03. – Generar notas con el Buzzer o Zumbador.....	36
A04. – Sensor siguelíneas I .....	39
A05. – Sensor siguelíneas II .....	43
A06. – Sensor de Ultrasonidos I.....	44
A07. – Sensor de Ultrasonidos II.....	46
A08. – Sensor de Ultrasonidos III.....	50
A09. – Funciones .....	53
A10. – Movimientos con KeyBot .....	55
A11. – Movimientos con KeyBot II .....	59
A13. – Módulo Bluetooth .....	61
A14. – Matriz de LED 8 x 8 .....	63
A15. – Pantalla LCD 2x16 .....	66
A16. – Controlar un servomotor .....	68
A17. – Sensor de Color.....	70
Proyectos con el KEYBOT completo.....	72
P1. – KeyBot seguidor de líneas .....	72

P2. – KeyBot explorador autónomo .....	79
P3. – KeyBot Marcha Imperial “Star Wars” .....	81
P4. – KeyBot el músico de los colores .....	83
P5. – KeyBot controlado por bluetooth.....	85

# Introducción

El presente manual pretende ser una herramienta base para el montaje y la programación del [Robot KEYBOT](#). Encontraremos una serie de actividades guiadas para aprender a programar de una manera entretenida y divertida mientras aprendemos conceptos relacionados con las **S.T.E.A.M.**

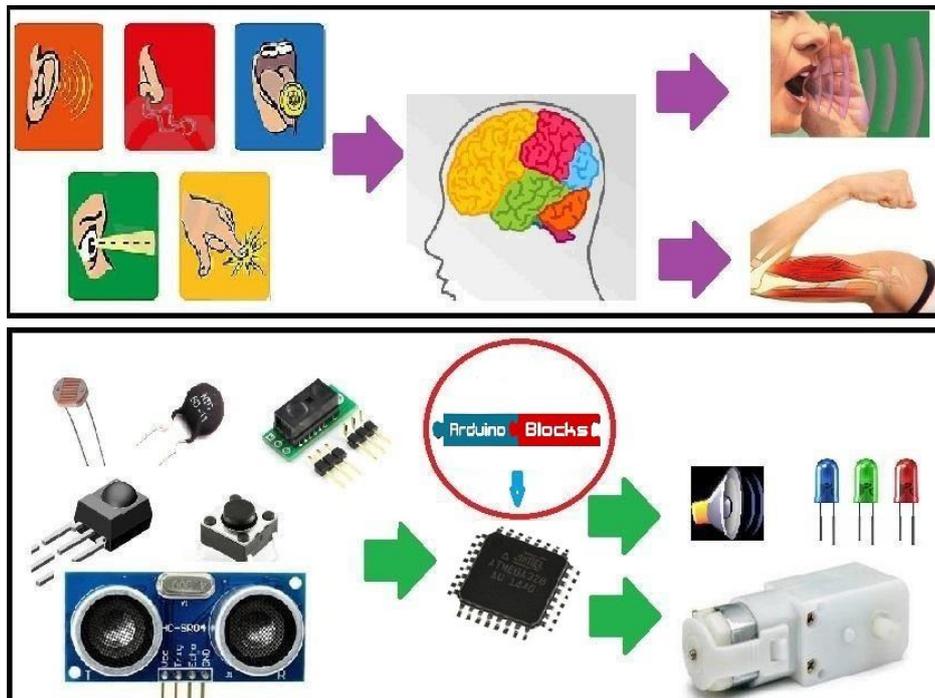
Para realizar la programación de nuestro KEY BOT utilizaremos un lenguaje de programación visual basado en bloques llamado [ARDUINOBLOCKS](#). Hay quien podría pensar que un lenguaje de este estilo es muy básico y limitado, pero ya veremos a lo largo del manual la gran potencialidad y versatilidad de este programa.



El portal es [www.arduinoblocks.com](http://www.arduinoblocks.com).

## ¿Cómo funciona un robot?

Los robots funcionan de forma similar a nosotros. Cuando nuestro cerebro recibe información de los sentidos; oído, olfato, gusto, vista y tacto; analiza esta información, la procesa y da estímulos a las cuerdas vocales para emitir sonido u órdenes a los músculos para que se muevan. Los 5 sentidos equivalen a entradas de información y, la voz y los músculos serían las salidas sonoras y motrices.



En el caso de un robot, un chip hace la función de cerebro. Este chip se llama microcontrolador y tiene unas entradas de información donde se conectan los sensores de luz (LDR), temperatura (NTC), sonido... y también tiene salidas, donde se conectan los motores, LEDs...

La diferencia principal es que, así como nuestro cerebro ha ido aprendiendo lo que tiene que hacer a lo largo de nuestra vida a base de estímulos positivos y negativos, el robot tiene su memoria vacía, es decir, no sabe lo que debe hacer. Entonces nosotros tenemos que decirle como tiene que actuar en función de las señales que recibe de los sensores. *¡A esta acción se le llama programar!*

## Robot KEYBOT

El **robot KEYBOT** se basa en una plataforma Arduino de código abierto, flexible y fácil de usar. La placa de control **KEYBOT** viene con conectores RJ11 para conectar los sensores y actuadores, esto hace que las conexiones sean tremendamente sencillas, cerrando la posibilidad de realizarlas mal, eliminando todo tipo de riesgo de dañar el equipo por errores en la conexión.

El robot está diseñado en estructura metálica, sólida y duradera. El ensamblaje es realmente sencillo estimando el tiempo de montaje en 30 min.

### Parámetros KEYBOT:

- Rango de fuente de alimentación externa: 7-12 V

- Rango de corriente: mínimo 800mA

- Velocidad del motor: 6.0V 100rpm / min

- El control del motor es controlado por TB6612

- Tres grupos de módulos de seguimiento de línea, para detectar líneas en blanco y negro con mayor precisión y también se pueden utilizar para el control anticaídas.

- El módulo ultrasonidos se utiliza para detectar la distancia del obstáculo, evitando el obstáculo delantero cuando la distancia detectada es inferior a un cierto valor.

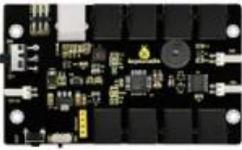
- El módulo inalámbrico Bluetooth se puede emparejar con un dispositivo Bluetooth en el teléfono móvil para controlar de forma remota el KEYBOT.

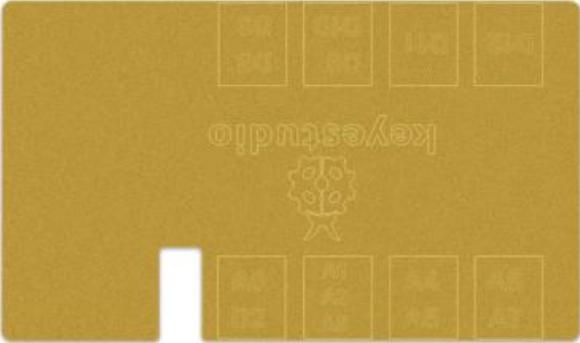
- La shield dispone de entradas y salidas digitales y analógicas con conector RJ11 y de dos conexiones para servomotores.

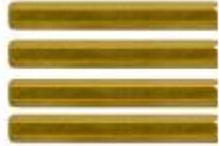


## Listado del material del Robot KEYBOT

En la caja del KEYBOT nos encontramos con las siguientes piezas y tornillos para realizar el montaje de nuestro robot.

				
Control Board	Keystudio Sensor	Keystudio Sensor	White Piranha LED module	Battery Case with lead
				
6-cell AA Battery Case	USB cable	Car body stents	Wheel	HC-06
				
Universal wheel	Acrylic board	Telephone Cable	Telephone Cable	Motor with lead
				
Motor with lead	Screwdriver	Inner Hex wrench	Hex Copper Pillar	Hex Copper Pillar
				
Screw	Screw	Inner Hex Screw	Inner Hex Screw	Nut

No.	Componente	Cantidad	Imagen
1	Tarjeta de control	1	
2	Panel acrílico superior protección de la tarjeta de control	1	
3	KEYBOY Sensor de Ultrasonidos	1	
4	KEYBOY Sensor de Línea	1	
5	Módulo Bluetooth (HC-06)	1	
6	Rueda universal de acero W420	1	

7	Motorreductor de eje simple con cable KF2510-2P de 2,54 zócalos, <b>cable rojo-negro 200 mm derecha</b>	1	
8	Motorreductor de eje simple con cable KF2510-2P de 2,54 zócalos, <b>cable rojo-negro de 140 mm a la izquierda</b>	1	
9	18650 caja de batería de 2 celdas	1	
10	Caja de batería AA de 6 celdas	1	
11	Rueda de robot 6515 negro-blanco	2	
12	Pilar hexagonal de cobre M3 * 40MM de doble paso	4	
13	Pilar hexagonal de cobre M3 * 15 + 6MM de una pasada	4	

14	Tornillo M3 * 30MM cabeza redonda	4	
15	Tornillo de cabeza plana M3 * 8MM	4	
16	Tornillo hexagonal interno de acero inoxidable M3 * 8	10	
17	Tornillo hexagonal interno de acero inoxidable M3 * 10MM	10	
18	Tuerca M3 niquelado	14	
19	KEYBOT cuerpo soporte negro	1	
20	Mango amarillo- negro Destornillador Phillips 3 * 40MM	1	
21	FÁCIL enchufe módulo piraña LED blanco	1	
22	Cable 6P6C RJ11 10CM azul y ecológico	1	
23	Cable 6P6C RJ11 20CM azul y ecológico	2	

24	Llave Allen M2.5 tipo L niquelada	1	
25	USB CABLE	1	

## Guia de montaje del Robot KEYBOT

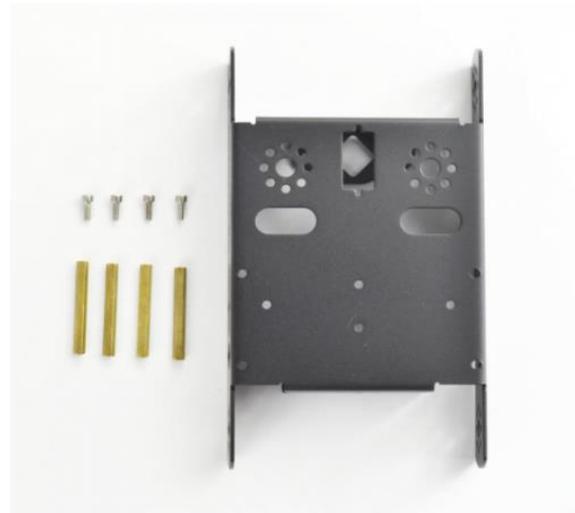
A continuación siga estos pasos para montar su Robot KEYBOT.

Paso 1; Comience con la parte del cuerpo KEYBOT. En primer lugar, debe preparar los componentes de la siguiente manera:

- 1 ud. Soporte para el cuerpo Keystudio KEYBOT

- 4 ud. Tornillo hexagonal interno de acero inoxidable M3 \* 8

- 4 ud. Pilar de cobre de doble paso M3 \* 40 mm



Después, fije los cuatro tornillos M3 \* 8 y los cuatro pilares de cobre M3 \* 40 mm en el soporte del cuerpo KEYBOT.



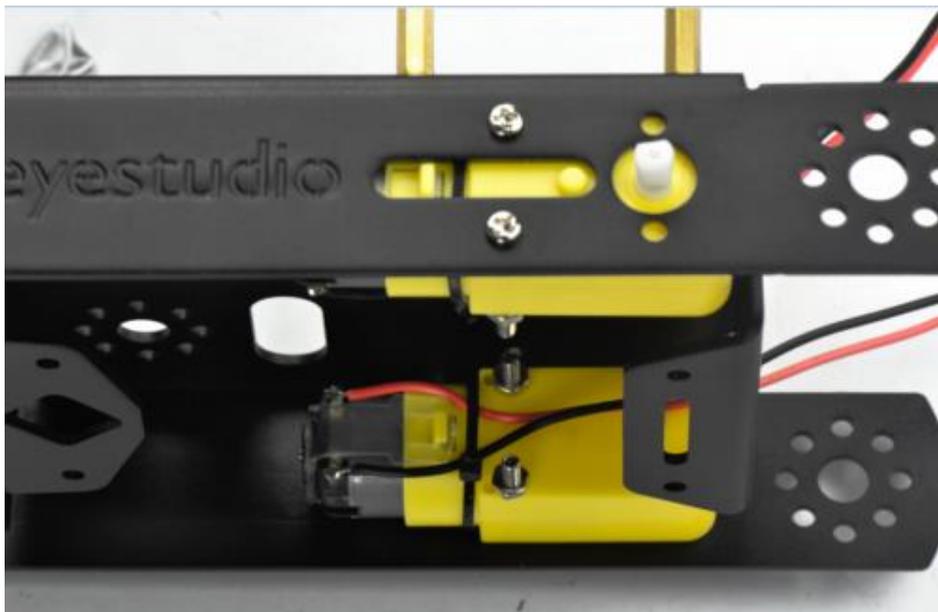
👉 Paso 2; Instale los motores para el robot y prepare los componentes de la siguiente manera:

- 2 ud. Motorreductor
- 4 ud. Tornillo de cabeza redonda M3 \* 30MM
- 4 ud. Tuerca níquelada M3

En primer lugar, coloque el soporte del cuerpo KEYBOT como se muestra a continuación.



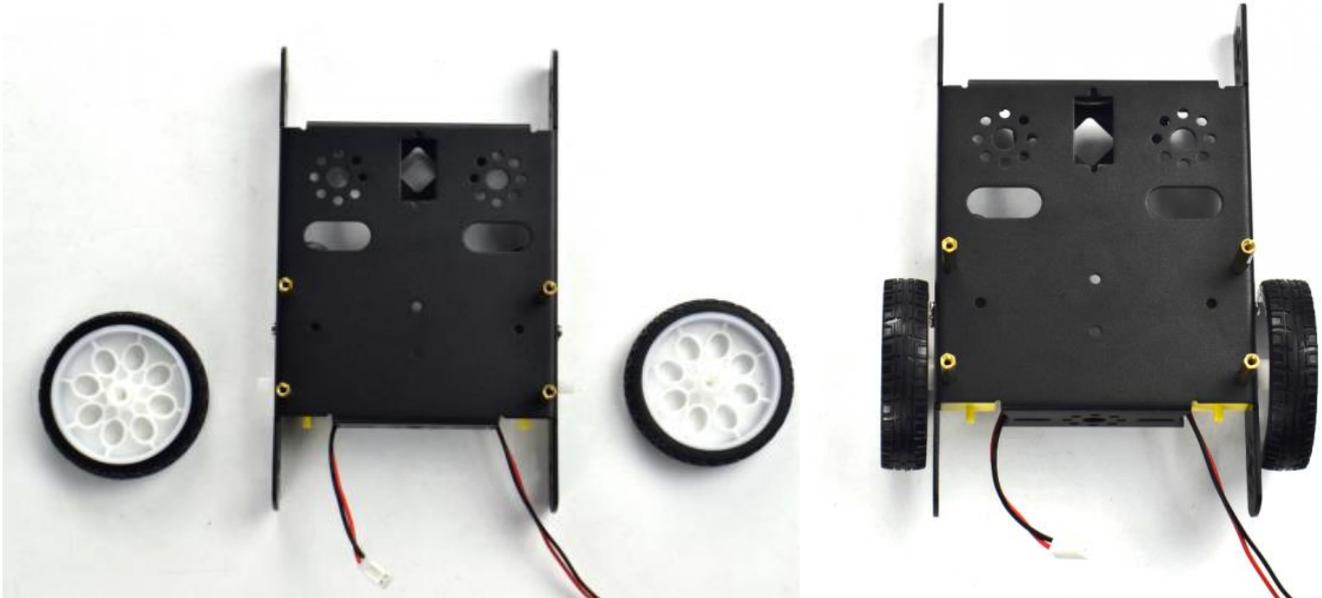
Monte el motorreductor con cable corto a la izquierda del soporte y monte otro motor con cable más largo a la derecha del soporte.



👉 Paso 3; Instalemos las ruedas para el KEYBOT.

.- 2 ud. 6515 rueda.

Monte las dos ruedas 6515 en los dos motorreductores.

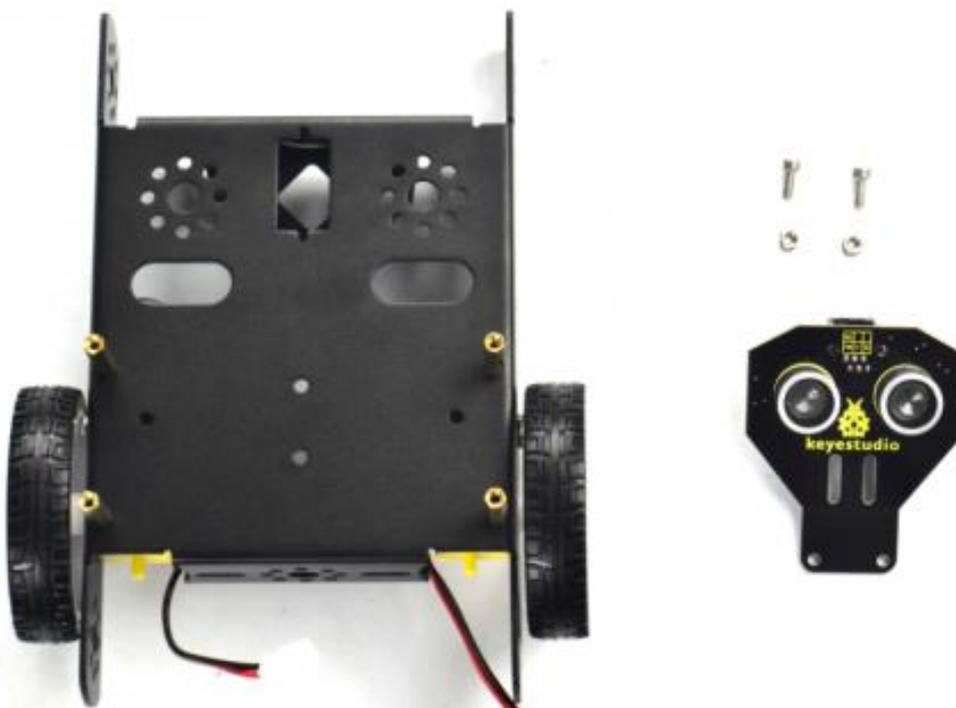


👉 Paso 4; Ahora vamos a instalar el “ojo” para el robot, es decir, el módulo de ultrasonidos. Debe preparar los componentes de la siguiente manera:

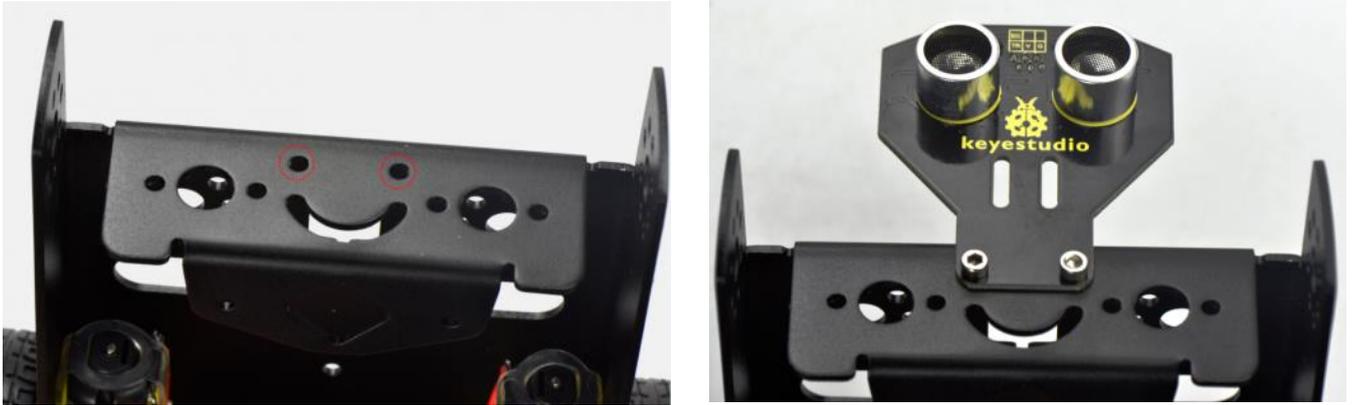
.- 2 ud. Tornillo hexagonal de acero inoxidable M3 \* 8

.- 2 ud. Tuerca níquelada M3

.- 1 ud. Sensor ultrasonidos.

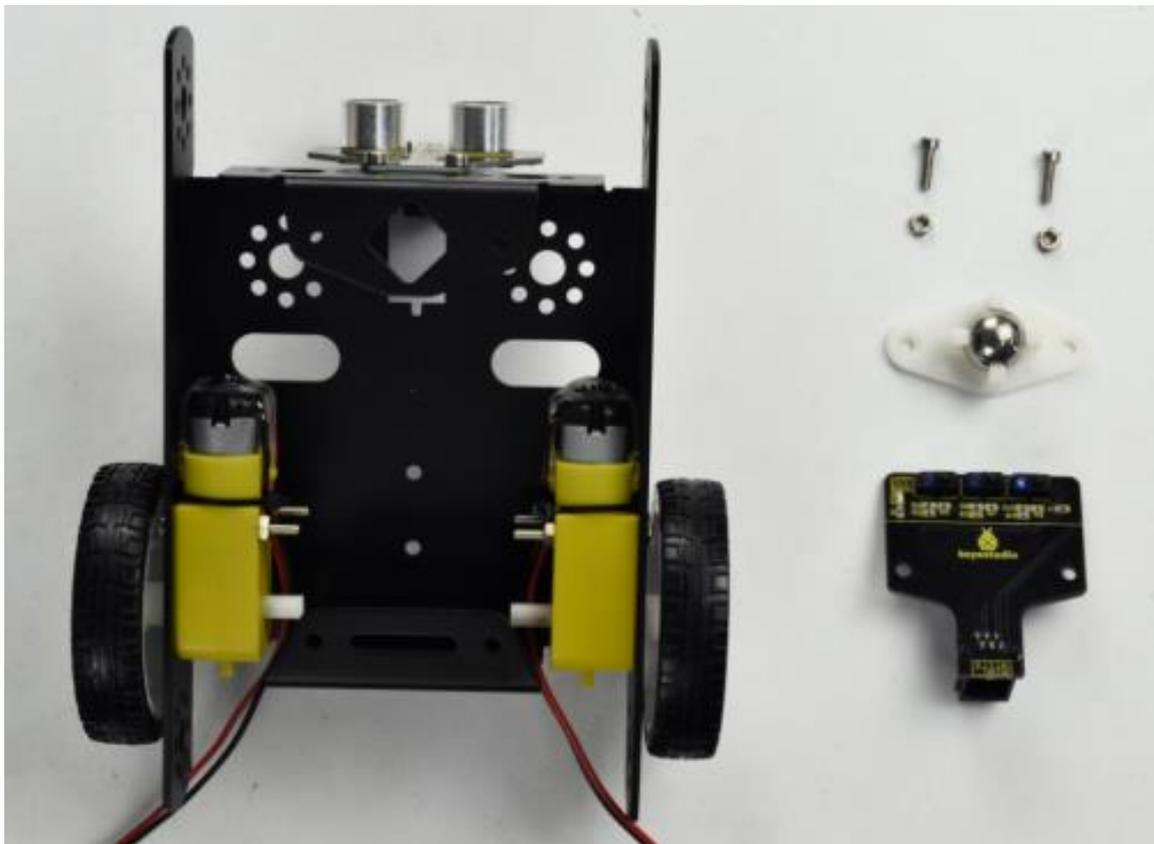


Monta el sensor de ultrasonidos en el soporte del cuerpo KEYBOT con dos tornillos M3 \* 8 y dos tuercas M3.



🔧 Paso 5; En el siguiente paso, ensambla el sensor seguidor de línea y la rueda de bolas de acero W420. Necesitarás;

- 2 ud. Tornillo hexagonal de acero inoxidable M3 \* 10MM.
- 2 ud. Tuerca niquelada M3.
- 1 ud. Módulo siguelíneas.
- 1 ud. Rueda universal de acero W420.



En primer lugar, monta el sensor de seguimiento de línea en la parte inferior del soporte del cuerpo KEYBOT con dos tornillos M3 \* 10.



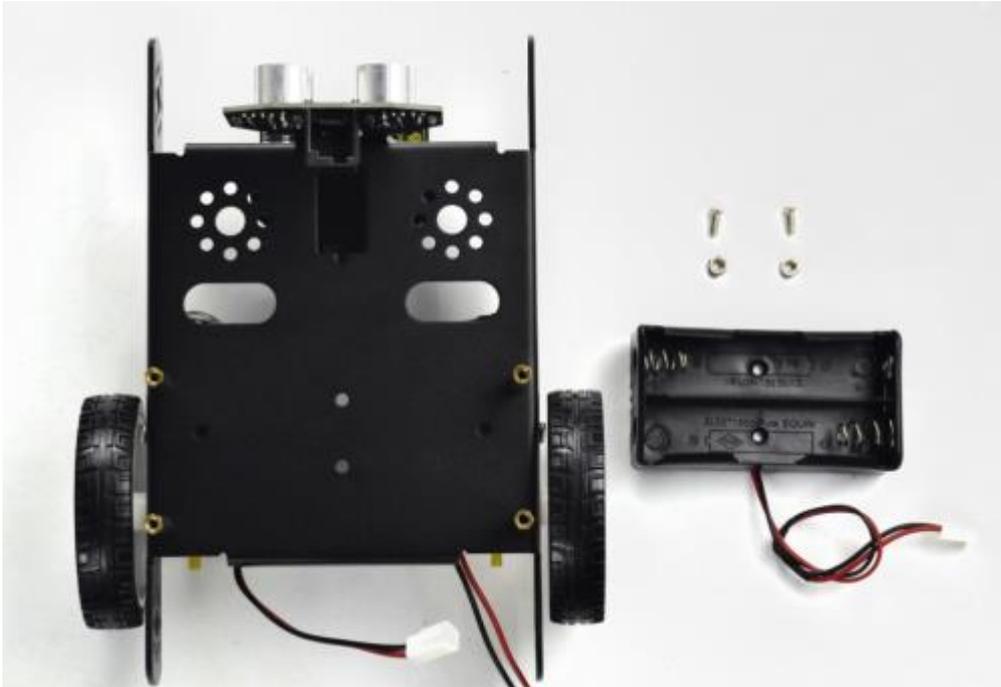
Luego fija la rueda W420 al sensor de seguimiento de línea con dos tuercas M3 tal y como se muestra en la imagen.



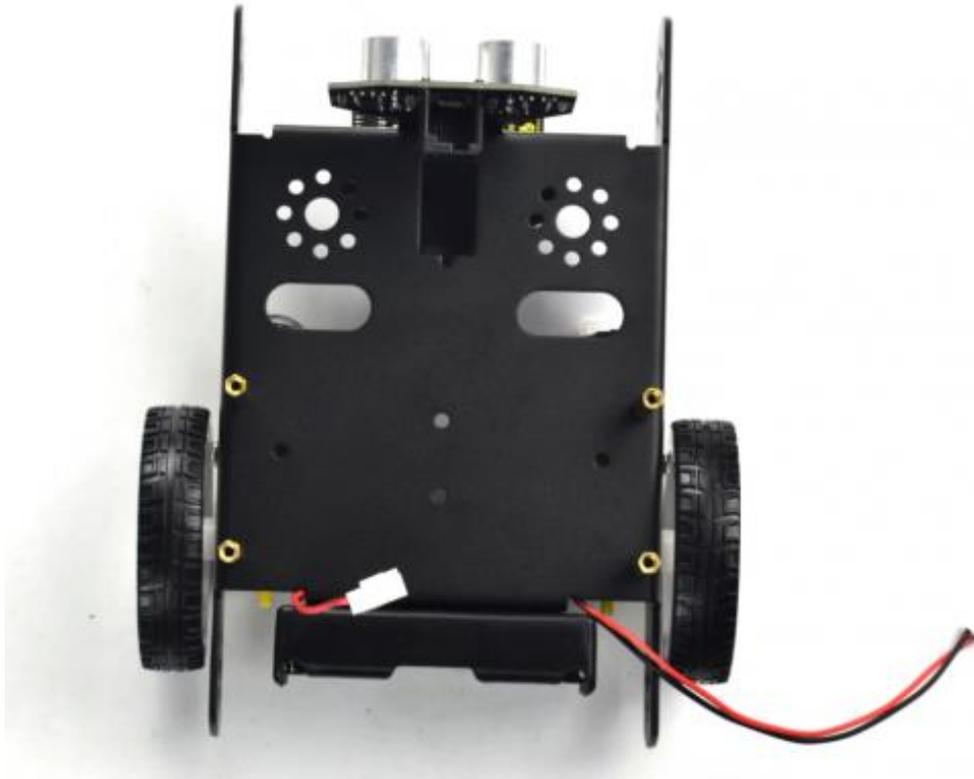
Paso 6; Fijar la caja de la batería en el soporte del cuerpo KEYBOT. Podemos elegir la caja de batería 18650 de 2 celdas o la caja de batería AA de 6 celdas. La forma de montaje para la caja de batería de 2 celdas 18650 es el siguiente;

.- 2 ud. Tornillo de cabeza plana M3 \* 8MM

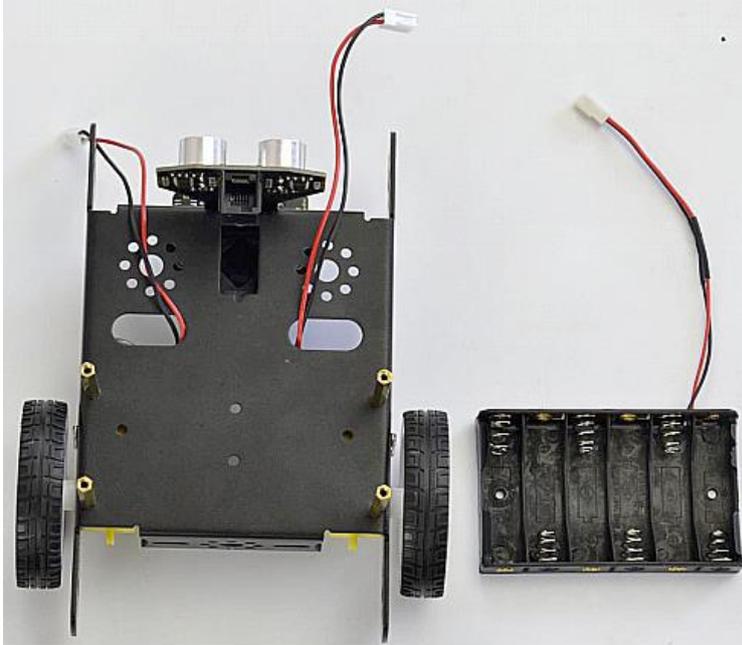
- 2 ud. Tuerca niquelada M3.
- 1 ud. 18650 caja de batería de 2 celdas.



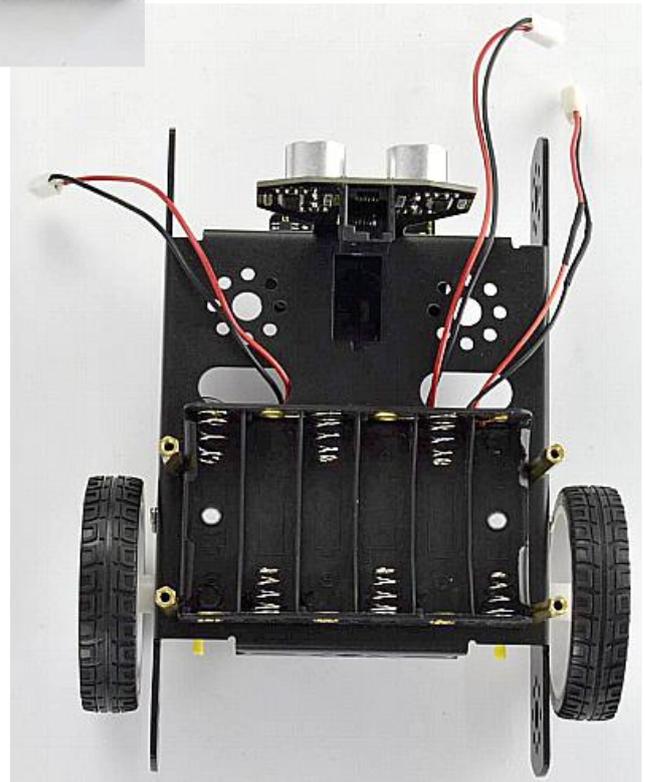
Monte la caja de la batería de 2 celdas en la parte posterior del soporte del cuerpo KEYBOT con dos tornillos de cabeza plana M3 \* 8MM y dos tuercas M3.



Si quieres instalar la caja de la batería AA de 6 celdas, toma la Caja de batería AA de 6 celdas y fíjate en la imagen.

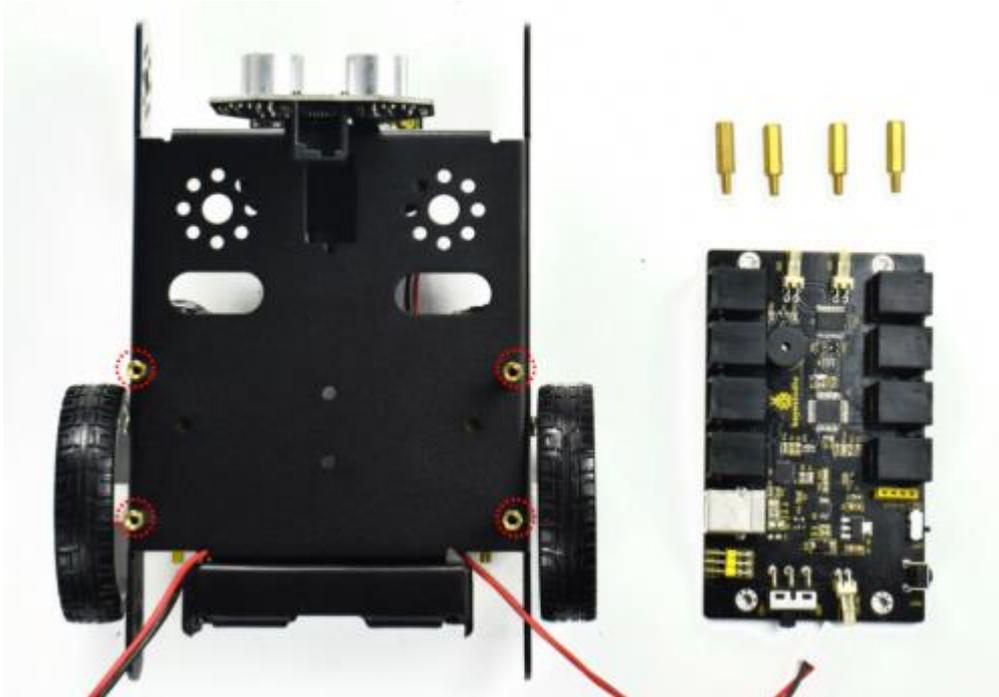


Así es como quedaría el portapilas de 6 AA en tu KEYBOT. (Necesitas 6 pilas AA de 1,5 v.).

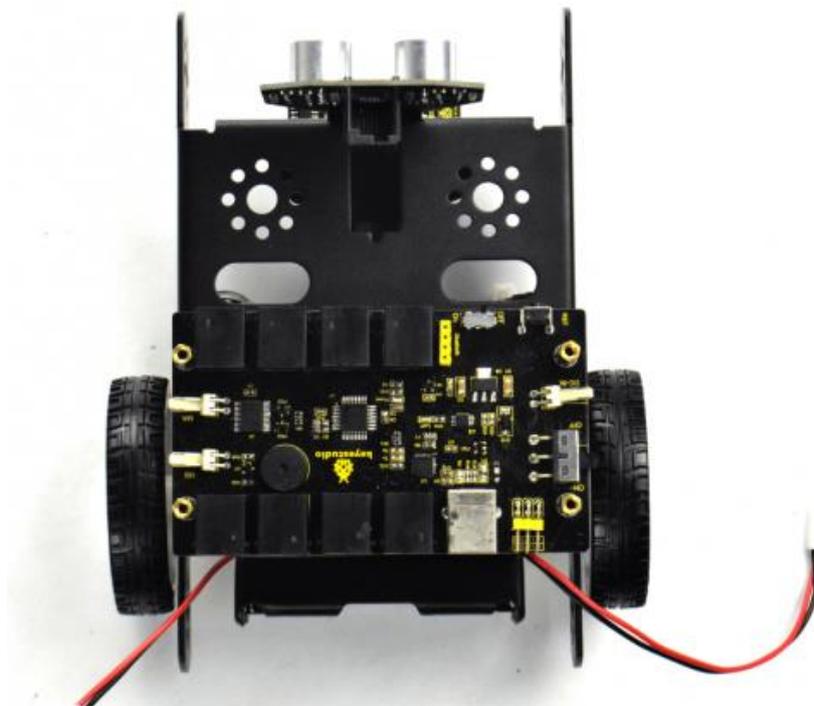


🔗 Paso 7; Vamos a fijar la placa de control del KEYBOT en el soporte del cuerpo del robot. Necesitamos;

- 4 ud. Pilar de cobre de paso único M3 \* 15 + 6MM.
- 1 ud Placa de control KEYBOT.



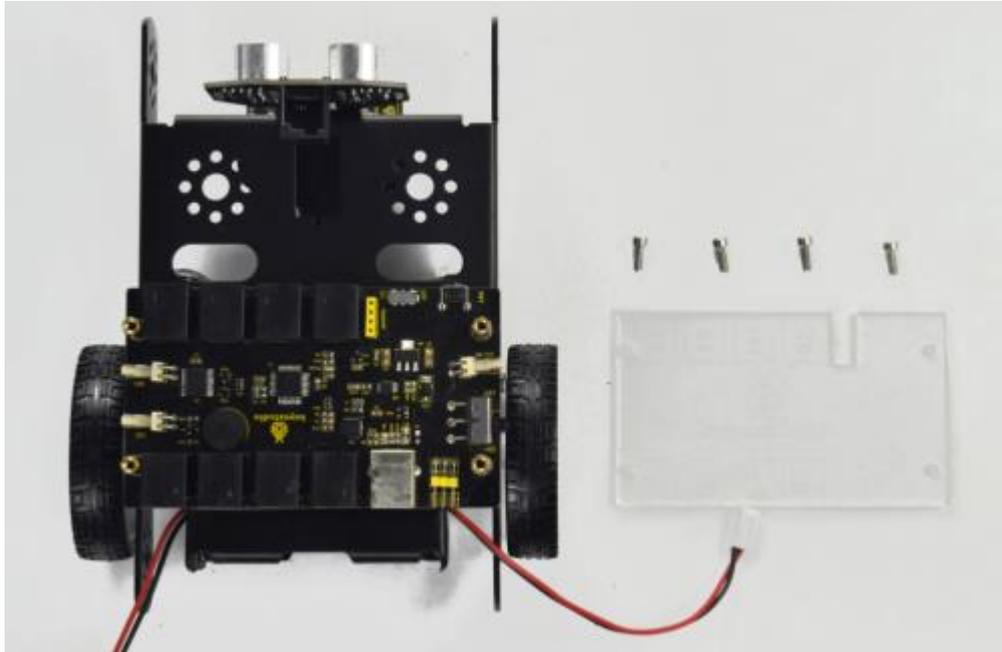
Monta la placa de control KEYBOT en la parte superior del soporte del cuerpo KEYBOT con cuatro pilares de cobre de un solo paso M3 \* 25 + 5MM.



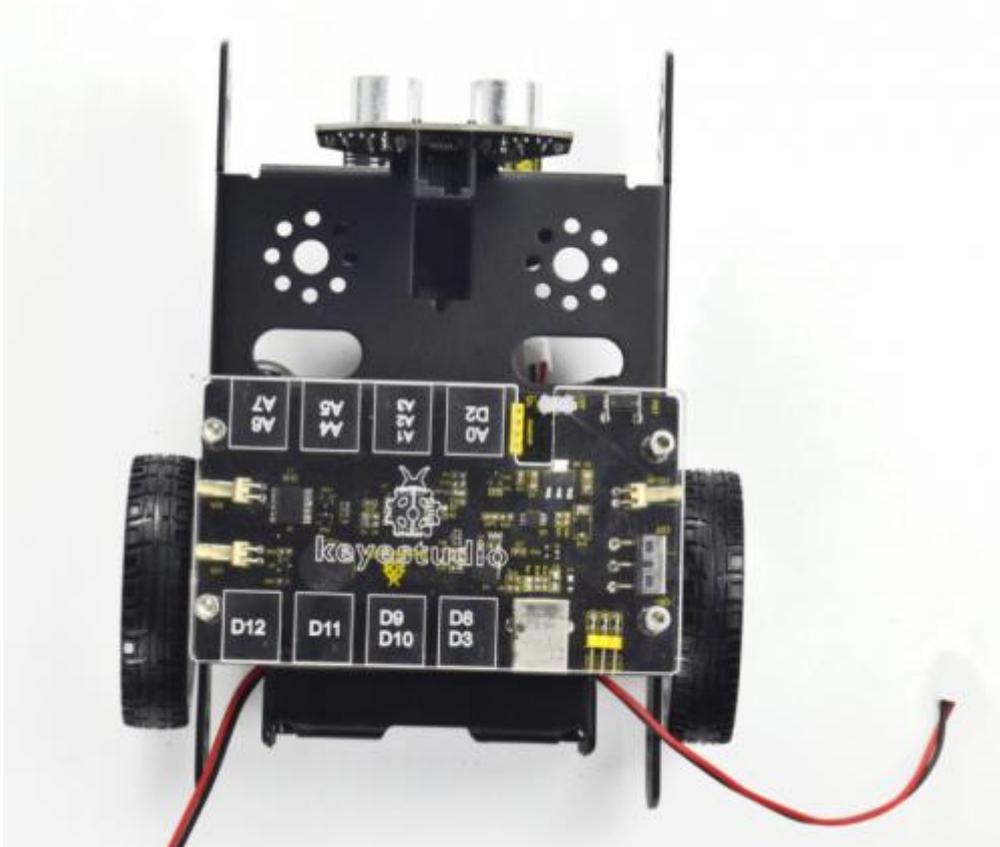
👉 Paso 8; El siguiente paso es instalar el panel superior protector de acrílico en la placa de control.

- 4 ud. Tornillo hexagonal de acero inoxidable M3 \* 10MM.

- 1 ud. Panel superior de acrílico.



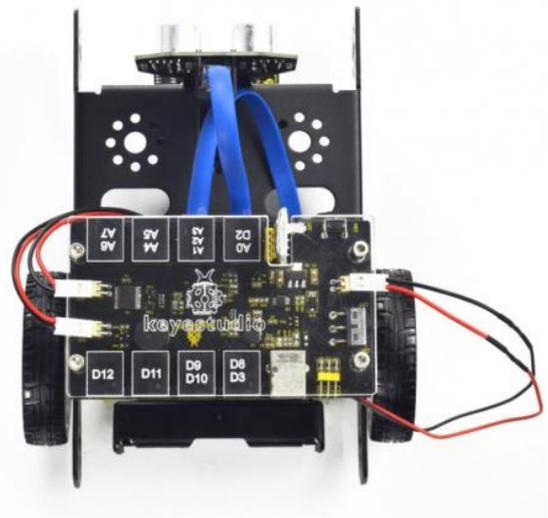
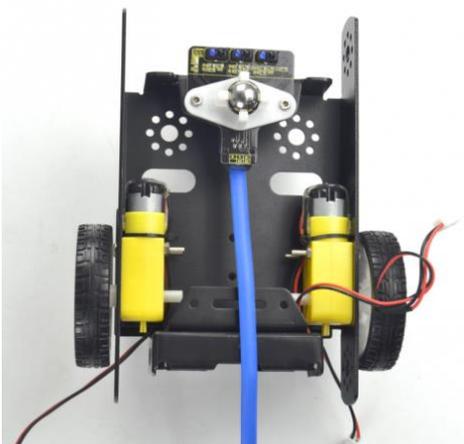
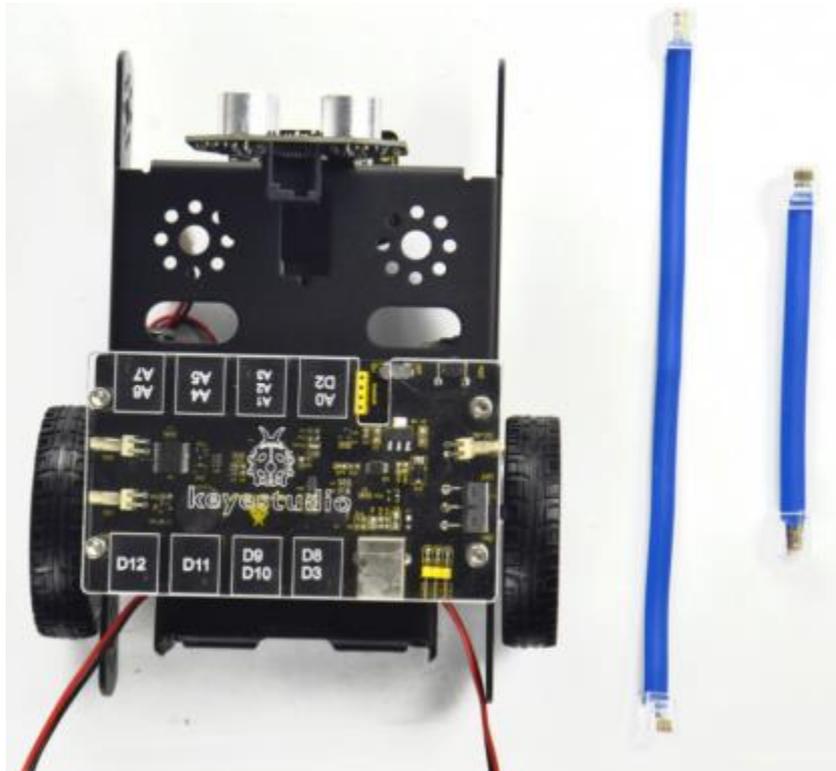
Coloca el panel superior de acrílico en el tablero de control con cuatro tornillos M3 \* 10MM.



👉 Paso 9; Ahora vamos a proceder a realizar las conexiones de los sensores de ultrasonidos y seguidor de líneas, de los dos motorreductores, del módulo bluetooth y de la alimentación (portapilas) con las placa de control. Para ello utilizaremos los cables que ya tienen los motores, portapilas y el bluetooth y para los sensores utilizaremos los siguientes cables;

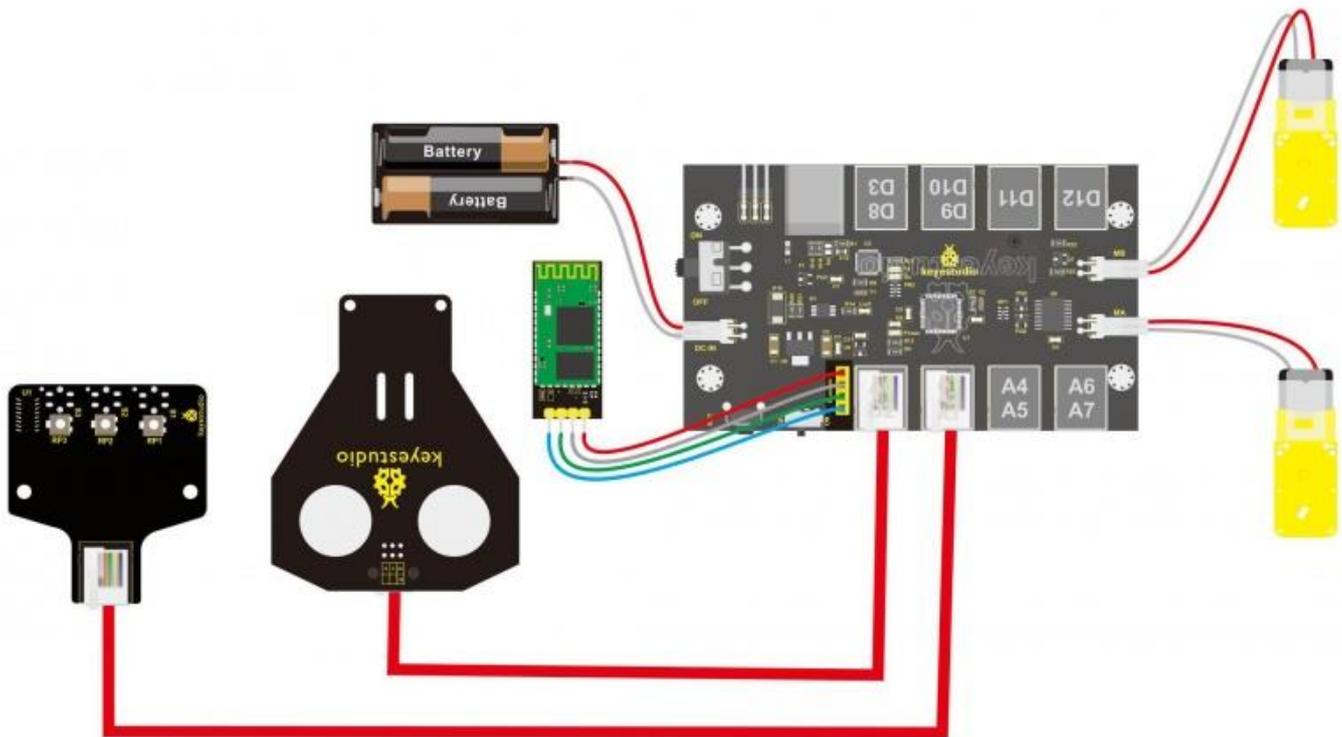
.- 1 ud. Cable 6P6C RJ11 10CM.

.- 1 ud. Cable 6P6C RJ11 20CM.



## Guía de Conexión:

- Conecta el sensor de ultrasonidos y el sensor de seguimiento de línea a la placa de control KETBOT.
- Conecta el sensor de ultrasonidos al conector A0-D2 con el cable RJ11 de 10 cm.
- Conecta el sensor de seguimiento de línea al conector A1-A2-A3 con el cable RJ11 de 20 cm.
- Conecta el motorreductor con cable corto a MA y conecte otro motorreductor con cable más largo a MB.
- El portabaterías se debe conectar al conector DC-IN de la placa de control.



Por último, conecta el módulo Bluetooth HC-06 a la placa de control, pero recuerda:

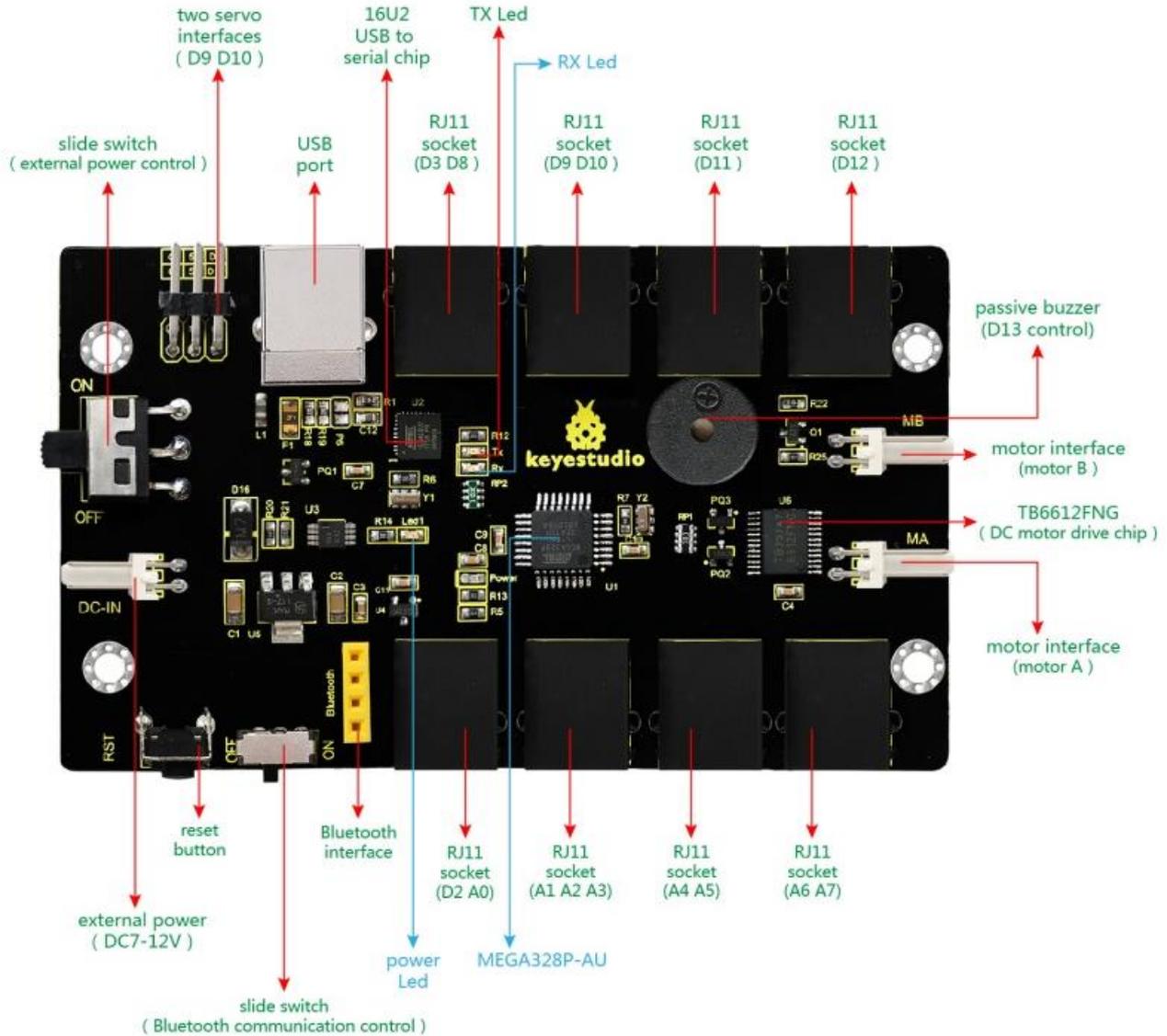
*IMPORTANTE: Antes de conectar el módulo bluetooth deberás cargar el programa previamente. Cada vez que quieras cargar un programa a la placa deberás desconectar el módulo bluetooth y después volver a conectarlo.*



# Placa de control

Esta es la placa de control de nuestro KEYBOT con las entradas y salidas definidas.

Esta placa está basada en una ARDUINO UNO. Dispone de un driver de motores lo que facilita su montaje y funcionamiento. La placa también lleva integrada un Buzzer o zumbador pasivo (Pin D13) y dos conexiones para servomotores (Pin D9 y D10).



Recuerda que la placa dispone de entradas **Analógicas (A)** y de entradas/salidas **Digitales (D)** todas ellas con conexiones RJ11 que evitan errores en conexiones. A lo largo del manual iremos viendo su funcionamiento.

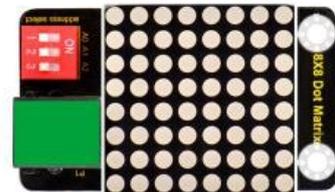
# Material opcional

Pregunta a [Innova Didactic](#) para asesorarte y adquirir el material opcional.

.- 1 Keyestudio EASY Plug Módulo LCD 1602 I2C (Ref. [KS0137](#))



.- 1 Matriz de EASY Plug LED 8x8 I2C (Ref. [KS0139](#)).



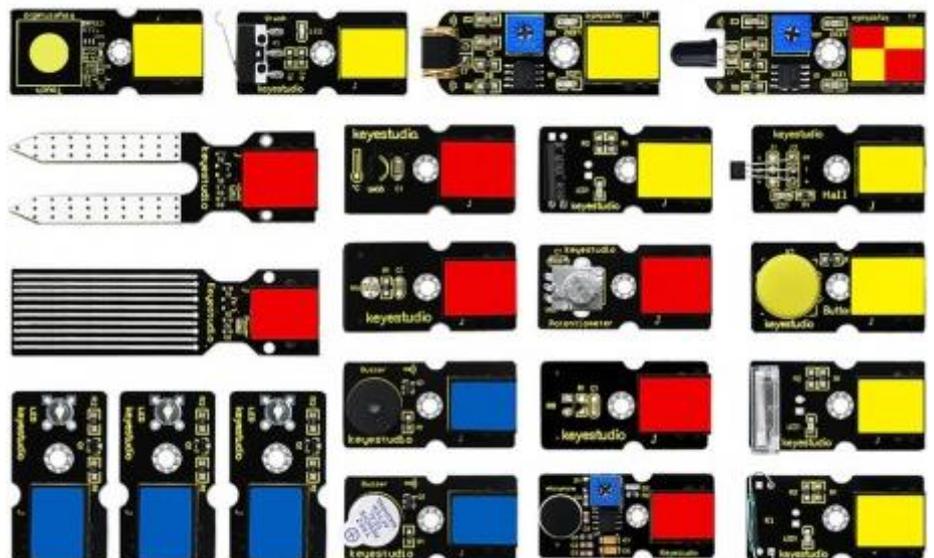
.- 1 Servomotor (Ref. [KS0194](#) o [KS0209](#)).



.- 1 Sensor de color RGB TCS34725 (Ref. [K0407](#)).



.... Y decenas de referencias para hacer volar la imaginación y crear cualquier idea!!!



# ArduinoBlocks

[ARDUINOBLOCKS](#) es un lenguaje de programación gráfico por “Bloques” creado por el profesor Juanjo López. Está pensado para que niños y niñas aprendan a programar con placas Arduino a partir de los 8 años.

Los distintos bloques sirven para leer y escribir las distintas entradas y salidas de la placa, así como programar funciones lógicas, de control, etc.

The screenshot shows the Arduino Blocks interface with the following blocks in the workspace:

- Leer digital** Pin 2
- Escribir digital** Pin 2 ON
- Leer analógica** Pin A0
- Escribir analógica (PWM)** Pin 3 Valor 0
- Leer digital** Pin 2
- Escribir digital** Pin 2 false
- Leer analógica** Pin A 0
- Escribir analógica (PWM)** Pin 3 Valor 0 (range 0 - 255)
- Leer pulso** Pin 2 ON Tiempo de espera 1000
- Interrupción** Pin 2 RISING

Control blocks on the right include **Inicializar** and **Bucle**.

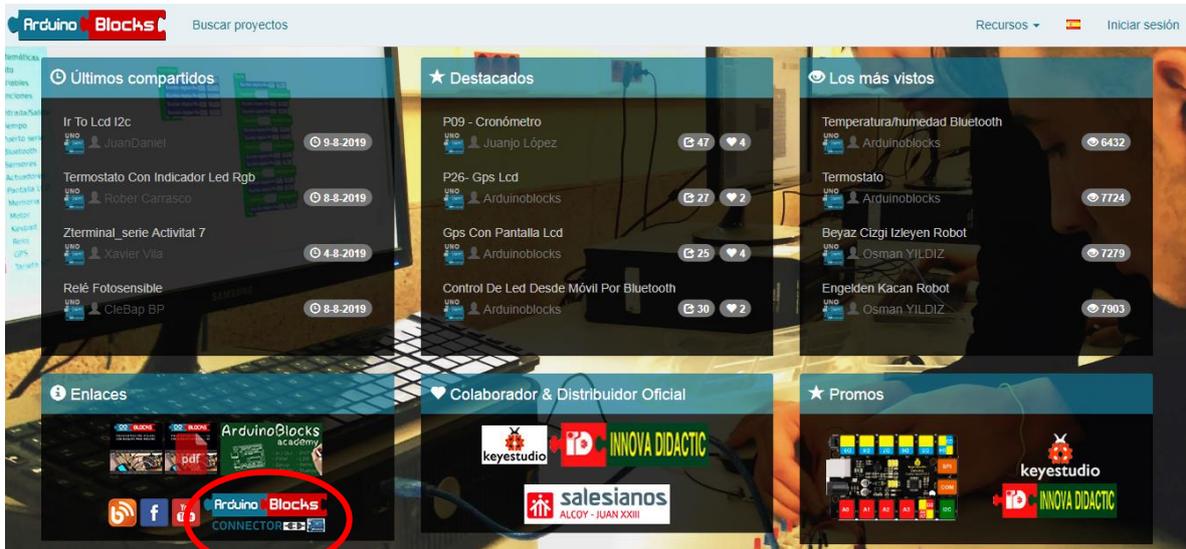
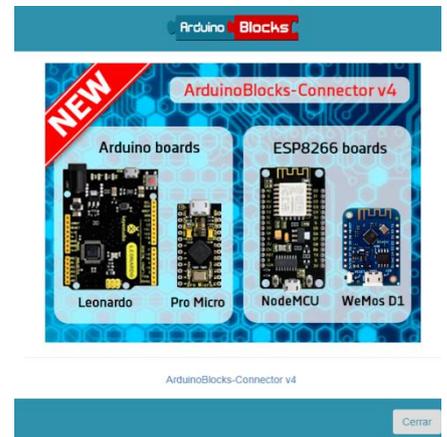
# ArduinoBlocks y el Robot KEYBOT

En este manual usaremos un menú dedicado sólo al [Robot KEYBOT](#), con estos bloques podremos programar las entradas y salidas de la placa Arduino Uno de nuestro robot para que realice las tareas que queramos.

Antes de comenzar necesitaremos instalar unos drivers y programas en nuestro ordenador.

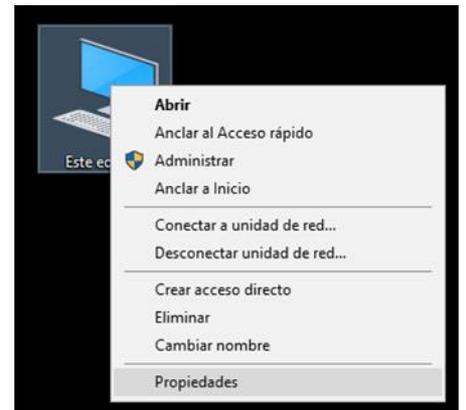
## Preparativos: instalación de los drivers y programas

Para programar la nuestro Robot KEYBOT con el programa online [ArduinoBlocks](#), necesitaremos instalar un pequeño programa disponible en la sección de “Enlaces” en la página principal. Se trata de [ArduinoBlocks Connector v.4](#)



Una vez clicado el link nos aparecerá esta ventana en la que podremos elegir nuestro sistema operativo; Windows, Ubuntu, MacOS o incluso RaspberryPi.

Una vez instalado [ArduinoBlocks Connector v.4](#), conectaremos nuestro KEYBOT utilizando el cable USB de nuestra placa a nuestro ordenador. Esperamos un breve tiempo y nos aseguraremos que se ha reconocido la placa. Para ello, con el botón derecho nos dirigiremos sobre el icono “Este equipo” y pulsaremos en “Propiedades”. Nos dirigiremos a “Administrador de dispositivos” y observaremos si en “Puertos (COM y LPT)” aparece nuestra placa Arduino UNO con un (COMXX) disponible.



*\*En caso de que no nos reconociera la placa, como paso opcional podemos descargar e instalar el programa oficial [Arduino IDE](#) para que se instalen los drivers necesarios, ni tan si quiera es necesario ejecutarlo.*



Una vez logueados nos aparece una ventana como esta, en la que tendremos guardados todos los proyectos que vayamos haciendo para poder acceder a ellos desde cualquier lugar.

ArduinoBlocks permite crear proyectos personales, pero también permite crear proyectos como profesor para compartirlo con los alum@s y poder supervisarlos y corregirlos. Otra aspecto muy destacable es que puedes hacer públicos tus proyectos por lo que hay una gran comunidad que comparte sus proyectos y cualquier usuario puede verlos y editarlos.

Nombre	Fecha creación	Fecha modificación ↓↑	Tipo de p
uiio	2019-07-24 08:52:45	2019-07-24 09:16:46	Keyestudi
Teclado con leonardo	2019-07-24 07:44:33	2019-07-24 08:51:52	Arduino L

Pues vamos a empezar a programar...., para ellos seleccionaremos “Proyectos”, “Nuevo proyecto” e “Iniciar un proyecto personal”.



### Nuevo proyecto

En el menú de “Tipo de proyecto” ArduinoBlocks permite programar varios tipos de placas Arduino y diferentes Robots, en nuestro caso seleccionaremos la opción “Keyestudio KEYBOT”.

Esta opción nos presentará los menús necesarios para poder programar nuestro robot de forma fácil y sencilla con las funciones preparadas expresamente para el KEYBOT.

Una práctica muy recomendable es ir documentando los proyectos que se van haciendo, para ello la plataforma de ArduinoBlocks dispone de un menú de cada proyecto en el que se pueden anotar el **Nombre del proyecto**, una **Descripción**, los **Componentes del proyecto** y **Comentarios**.

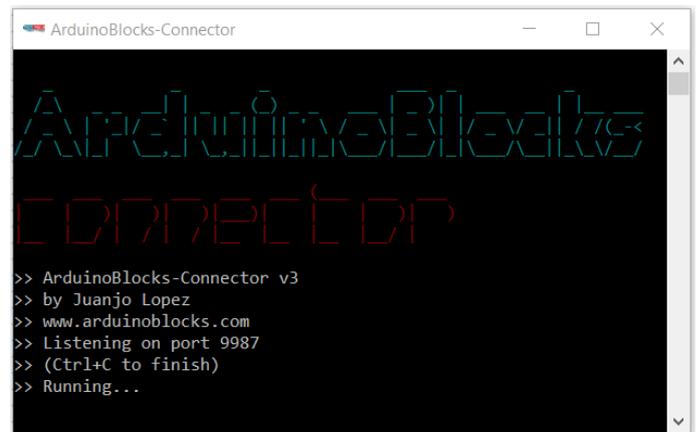
Una vez rellenados todos los campos daremos al botón de “**NUEVO PROYECTO**”.

La siguiente imagen nos muestra la interface de programación con los bloques específicos para poder programar con mucha facilidad nuestro **KEYBOT**.



Por último, y una vez tengamos finalizado el programa, la forma de transferirlo a la placa siempre será la misma.

Primero tenemos que asegurarnos que **ArduinoBlocks Connector** está en marcha. (Haremos click en su icono y lo dejaremos abierto en un segundo plano).



Después, comprobaremos que nuestra placa está detectada en el puerto correspondiente. En el caso de la imagen en el COM11 (1). Si no fuera así daríamos al botón de refrescar (2).

Por último, con el botón “Subir” (3) transferimos el programa del ordenador a la placa.

# Actividades con el Robot KEYBOT

A continuación, os proponemos una serie de actividades y prácticas para aprender a programar vuestro KEYBOT paso a paso para que realice infinidad de tareas.

## A01. – Encender/Apagar un LED

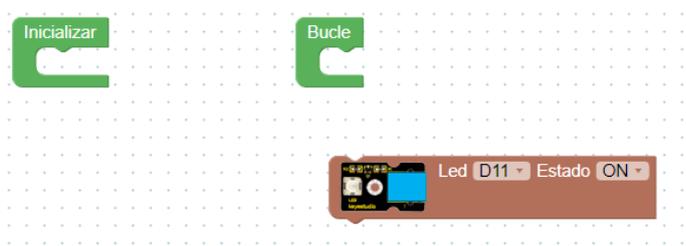
Vamos a empezar con nuestro primer programa que será encender y apagar un LED variando los tiempos de encendido y apagado.

Para ello vamos a necesitar nuestro LED y un cable RJ11. Conectaremos el LED al pin D3 de la placa de control.



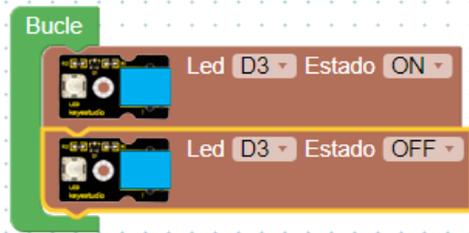
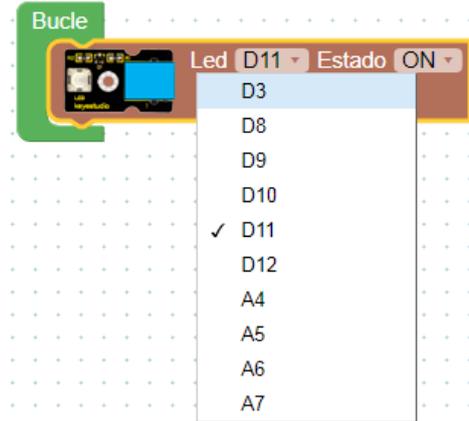
En programación el primer programa que se realiza se le suele llamar “*Hola Mundo*”, así que este será nuestro “*Hola Mundo*” personal. Una vez seleccionado en ArduinoBlocks en el *tipo de proyecto* el *Keystudio KEYBOT* y puesto el nombre (*Hola Mundo*), picharemos en el bloque de **ACTUADORES**.

Clicamos sobre el bloque 1 y lo arrastramos sobre el área de programación. Fíjate que en el área de programación hay dos bloques verdes (*Inicializar* y *Bucle*) y que podríamos meter nuestro bloque del LED en cualquiera de ellos (*Los bloques cuando son compatibles encajan como un puzzle*). Pues bien, todo lo que se meta dentro del bloque de **Inicializar** sólo se ejecutará la primera vez que se inicie el programa, mientras que si se colocan dentro del **Bucle** se ejecutarán una y otra vez hasta que apaguemos la placa.



Vamos a meter nuestro bloque de LED en el *Bucle* y cambiaremos el Pin D11 por el Pin D3. El LED puede tener dos estados; ON/OFF, que podemos cambiar en el menú desplegable.

Si sólo dejamos este bloque con el LED en estado ON, este quedaría encendido para siempre, para que se apague deberemos meter el estado OFF.



Pero este programa no es correcto del todo ya que no hay tiempos que indiquen cuanto tiempo tiene que estar encendido o apagado el LED. Necesitamos ir al bloque de **TIEMPO** y seleccionar *Esperar XXXX milisegundos* (recuerda; 1.000 milisegundos son 1 seg).



Ahora tenemos el LED encendido durante 1 seg. y apagado otro. Prueba a cambiar los valores del tiempo.

## A02. – Control de brillo de un LED usando el PWM.

Continuando con la práctica anterior, ahora vamos a controlar el brillo de un LED utilizando el PWM. Pero lo primero de todo, ¿Qué significa PWM? Bien, PWM es la abreviatura de **pulse-width modulation** (modulación de ancho de pulso).

Las salidas de voltaje de Arduino sólo tienen dos estados ALTO/BAJO, ON/OFF, ENCENDIDO/APAGADO,... es decir, corresponden a una salida de 5 V (ON) y de 0 V (OFF). Con esto sólo podemos hacer actividades como la anterior de encender y apagar un LED, no podríamos controlar su brillo de menos a más o viceversa. Sin embargo el PWM permite hacer un rango de valores de 0 a 255 entre el 0 y 5 V. De esta manera podemos controlar el brillo del LED y muchas cosas más.

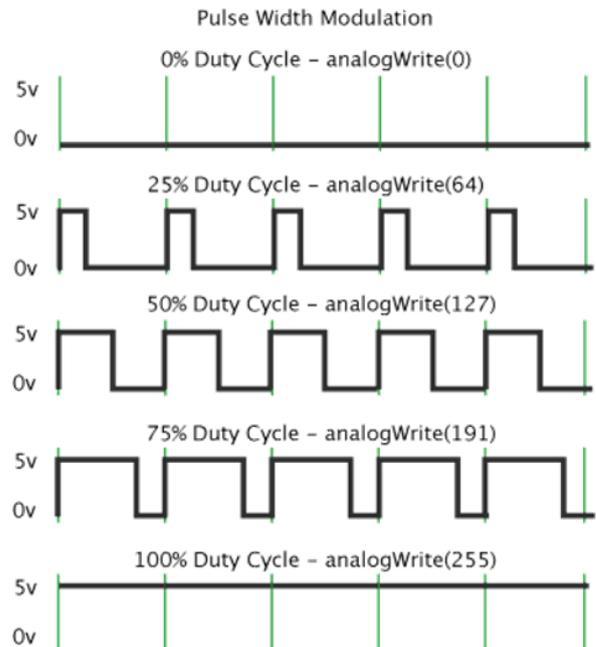
La modulación de ancho de pulso, o PWM, es una técnica para obtener resultados analógicos con medios digitales.

El control digital se utiliza para crear una onda cuadrada de ciclo de trabajo diferente, una señal conmutada entre encendido y apagado. Este patrón de encendido y apagado puede simular voltajes entre encendido total (5 voltios) y apagado (0 voltios) al cambiar la parte del tiempo que la señal pasa en comparación con el tiempo que la señal pasa.

El PWM se utiliza mucho para controlar lámparas, velocidades de motores, producción de sonidos,...

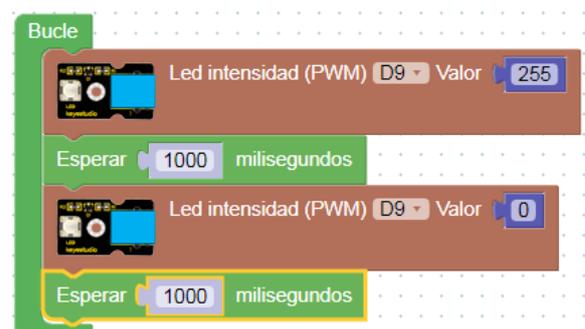
El Arduino UNO tiene un total de 6 salidas PWM, que son digitales 3, 5, 6, 9, 10 y 11.

[ArduinoBlocks](#) tiene un bloque específico para controlar un LED por PWM y sólo permite utilizar el Pin D9.



El rango de VALOR lo podemos variar desde 0 hasta 255, de tal modo que si el valor es 0 el LED estará totalmente apagado y si el valor es 255 el LED brillará al máximo.

Podríamos hacer la actividad anterior con este programa:



Ahora vamos a hacer que el brillo del LED varíe.

Empezaremos introduciendo un valor de 0 en el PWM para ir aumentándolo de 50 en 50 unidades hasta llegar a 255. Haremos una espera de 100 milisegundos entre un aumento de valor y otro. El programa quedaría como este:

Imagínate que en lugar de incrementar los valores de 50 en 50 lo quisieras hacer de 1 en 1, tendrías que tener 255 bloques más otros tantos de esperas... Como verás esta forma de programar no es muy efectiva, es muy repetitiva y tendría muchos bloques. Existe un bloque con el que podríamos hacer esto de una manera mucho más sencilla. Es el siguiente bloque:



Este bloque lo tenemos dentro de **CONTROL**.



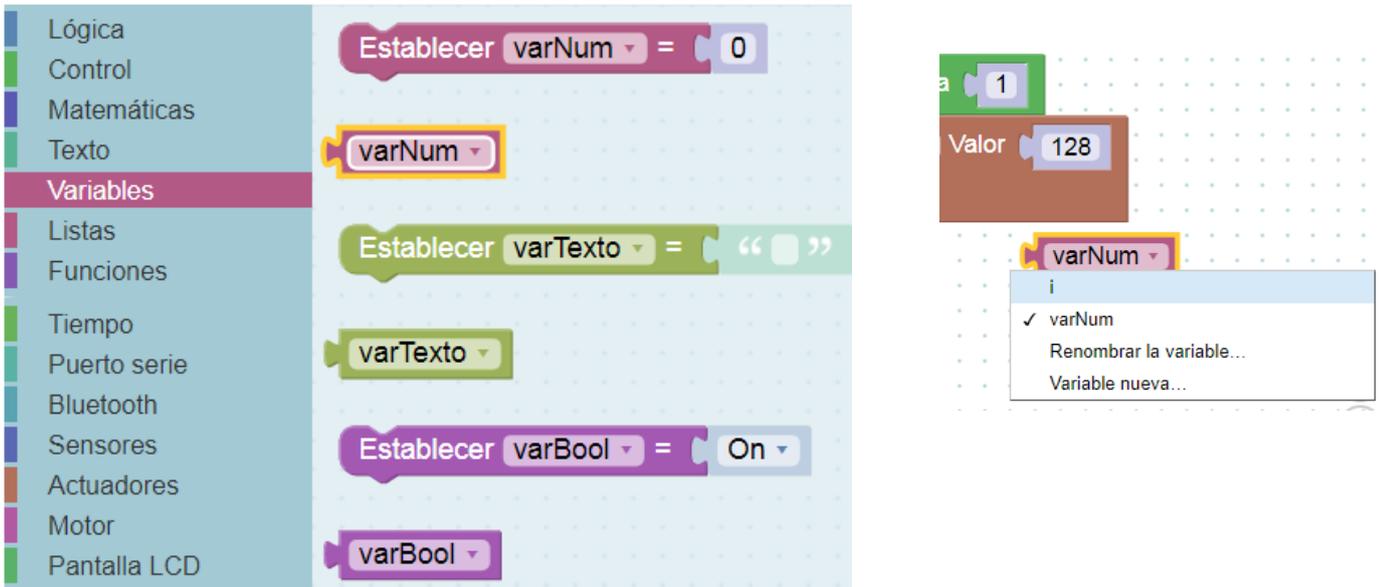
Este bloque tiene un concepto muy utilizado en programación que son las **VARIABLES**. En este bloque la variable se llama *i* (nombre que se puede cambiar). A grandes rasgos, una variable es una "cajita" en la cual se van introduciendo diferentes valores que el programa utilizará según necesite.

Este sería el programa de ir aumentando el brillo de un LED de 1 en 1 desde 0 hasta 255 con una espera de 20 milisegundos.



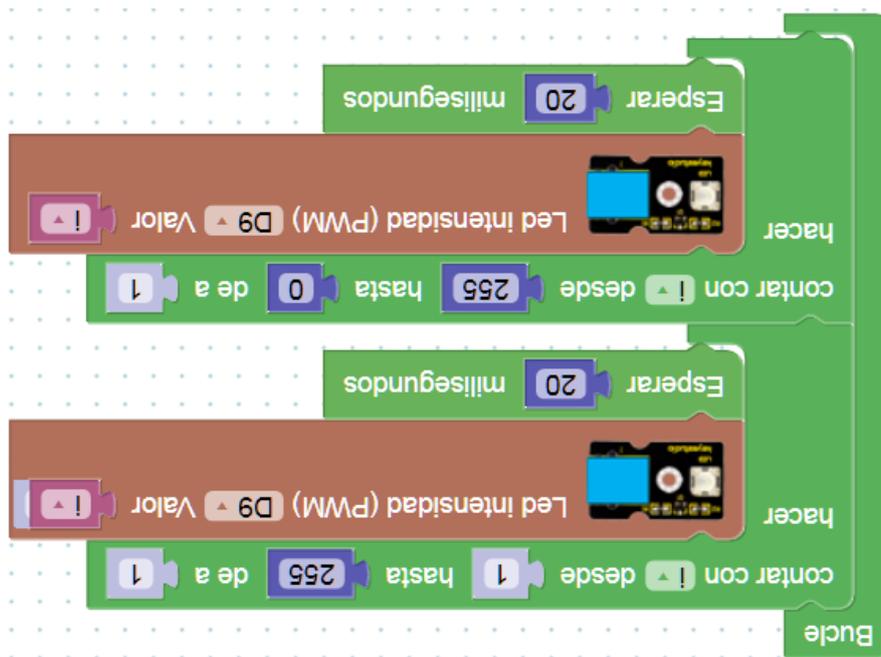
En el programa inicialmente el valor de la **VARIABLE** *i* vale 1, cada vez que repite el bucle su valor va aumentado de 1 en 1.

Para introducir el Valor del PWM, del bloque del LED, la variable *i*, debemos ir a **Variables**, seleccionar el segundo bloque y en el menú desplegable seleccionar *i*.



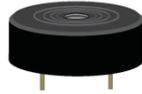
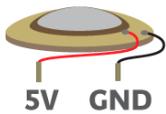
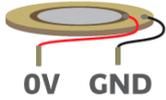
Intenta completar el programa haciendo que el brillo del LED ascienda de 0 a 255 y que descienda de 255 a 0.

*Solución:*

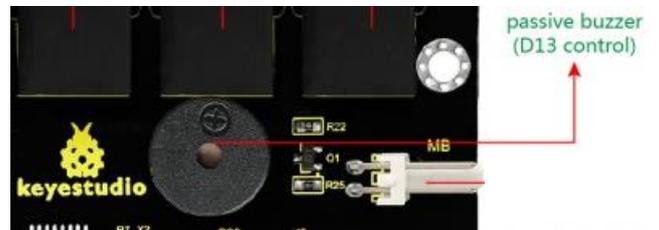


## A03. – Generar notas con el Buzzer o Zumbador.

Zumbador, buzzer en inglés, es un transductor electroacústico que produce un sonido o zumbido continuo o intermitente. En función de si se trata de un buzzer Activo o Pasivo, este zumbido será del mismo tono o le podremos variar. Sirve como mecanismo de señalización o aviso y se utiliza en múltiples sistemas, como en automóviles o en electrodomésticos, incluidos los despertadores.



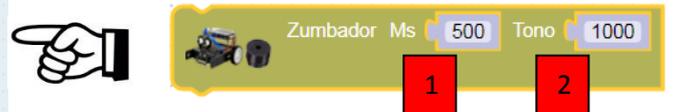
Recordad que nuestro Robot KEYBOT tiene un Buzzer o Zumbador pasivo integrado en su placa de control y que está conectado internamente al Pin D13.



ArduinoBlocks tiene un bloque específico para el Zumbador del KEYBOT está en el menú del KeyBot.

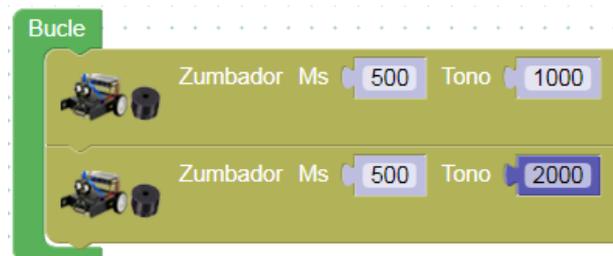


En el bloque Zumbador podemos variar dos parámetros; Ms (Milisegundos) (1) es el tiempo que dura cada sonido y Tono (2), que es la frecuencia a la que vibra la membrana para emitir el sonido.



Prueba con este sencillo programa para ver cómo suena.

Cambia ligeramente el programa introduciendo unas esperas entre un sonido y otro. ¿Ves la diferencia?



En la tabla de la derecha tienes las frecuencias de las notas musicales.

Vamos a hacer una escala de DO a DO utilizando estos valores.

Nota	Frecuencia (Hz)
do (control)	261.6
do#	277.2
re#	293.7
mi	329.6
fa	349.2
fa#	370
sol	392
sol#	415.3
la	440
la#	466.2
si	493.2
do	523.3

**Bucle**

¿Te atreves a programar esta melodía?

### HIMNO DE LA ALEGRÍA (PARTE 1)

| MI MI FA SOL | SOL FA MI RE | DO DO RE MI | MI RE RE |  
 (1) (2) (1) (2)

| MI MI FA SOL | SOL FA MI RE | DO DO RE MI | RE DO DO |  
 (1) (2) (1) (2)

En el menú de actuadores existe el bloque de Tono (Hz) y sus valores están en cifrado anglosajón.



En esta tabla comparativa podemos ver las equivalencias con el sistema latino.

C	D	E	F	G	A	B	C
Do	Re	Mi	Fa	Sol	La	Si	Do

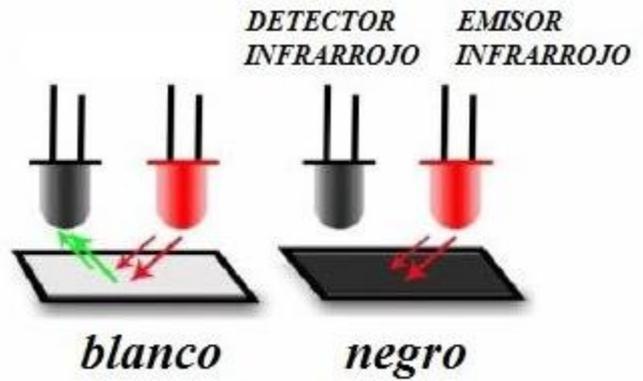
De esa forma podremos programar el zumbado sin necesidad de usar las frecuencias.



## A04. – Sensor siguelíneas I

El sensor siguelíneas del **KEYBOT** está formado por tres sensores infrarrojos TCRT5000. Su principio de funcionamiento es utilizar la diferente reflectividad de la luz infrarroja para el color y convertir la intensidad de la señal reflejada en una señal de corriente.

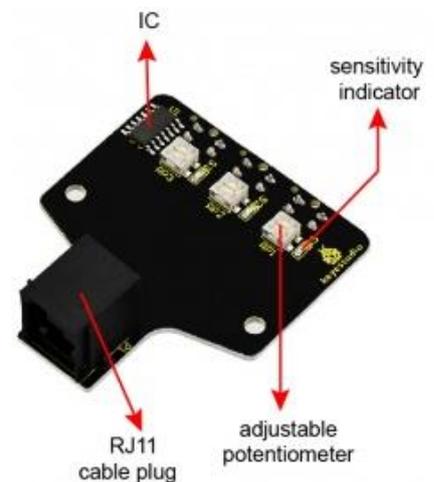
El sensor TCRT5000 tiene un emisor de infrarrojos y un fototransistor que recibe la luz reflejada. En el caso de que el color sea negro, este absorbe todo el espectro de luz por lo que no refleja nada y el fototransistor no recibirá nada. En caso del color blanco ocurre lo contrario, sí refleja la luz por lo que el fototransistor la recibirá. Durante el proceso de detección, el negro está activo en el nivel ALTO, pero el blanco está activo en el nivel BAJO. Y la altura de detección es de 0,5 a 3 cm.



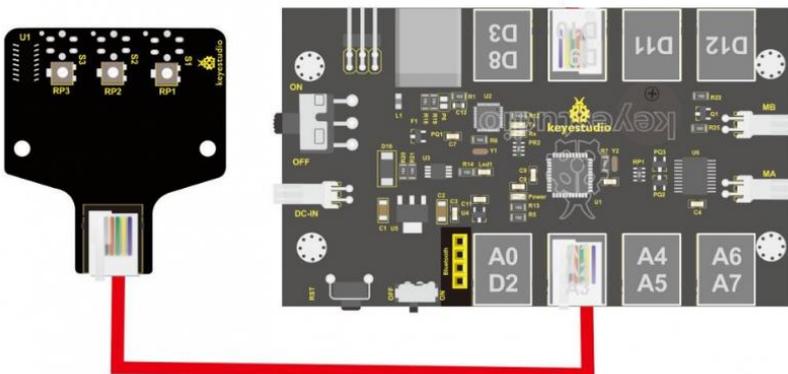
La siguiente imagen es el módulo de seguimiento de línea KEYBOT de 3 canales. Dispone de 3 sensores infrarrojos TCRT5000 en una sola placa.



En la parte superior de la placa existen 3 potenciómetros que nos permiten ajustar la sensibilidad de detección de los sensores (S1, S2 y S3).

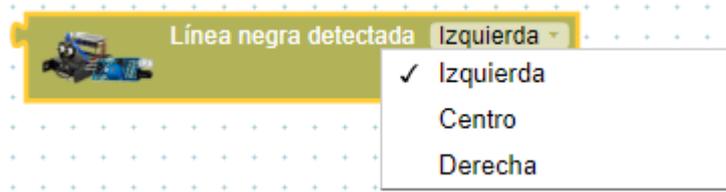


Junto a cada potenciómetro existe un pequeño led rojo que se enciende cuando el sensor detecta color blanco y se apaga con el color negro.



Recordamos que conectamos el sensor siguelíneas a la placa de control con el cable RJ11 al Pin (A1, A2, A3) y eso es porque el sensor S1 está conectado con la entrada analógica A1, el sensor S2 con la A2 y el S3 con A3.

ArduinoBlocks tiene un bloque específico dentro del menú KeyBot para utilizar el sensor siguelíneas.



En esta actividad vamos a encender el LED conectado en el Pin D9 cuando el sensor S3 (del sensor siguelíneas) detecte blanco y cuando detecte negro que se apague.

Vamos a empezar a utilizar funciones del menú “Lógica” con las funciones de condición.



Condiciones “Si...hacer”.



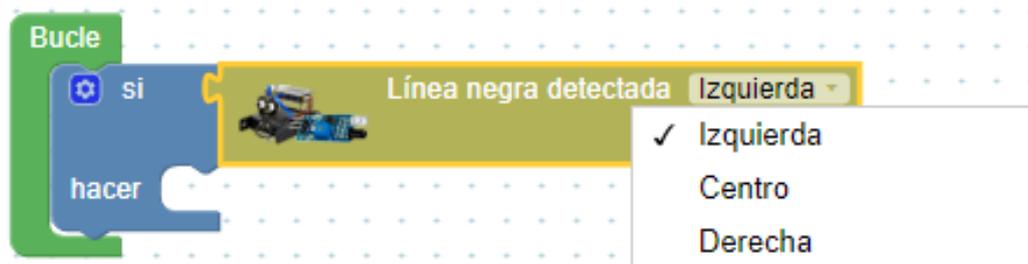
Se trata del famoso bucle *Si* (*if* en inglés) que es uno de los pilares de la programación, ya que permite evaluar estados y tomar decisiones en consecuencia.

Funciona como una oración condicional en español, si se cumple la condición incluida en su primer apartado, entonces se realiza la acción incluida en su segundo apartado. En caso contrario, no se hace nada.



En el apartado de condiciones se pueden introducir multitud de factores: estado de sensores, (analógicos o digitales), comparaciones, igualdades, operaciones matemáticas, etc.

En esta actividad vamos a utilizar el bloque de **línea negra detectada** eligiendo la opción de **IZQUIERDA**.

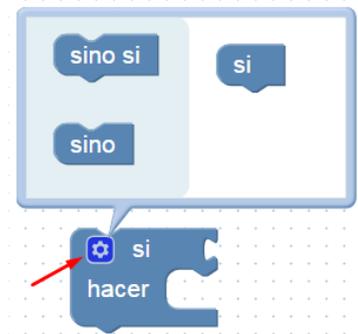


Con ese bloque crearemos la condición. Como acción a ejecutar, si se cumple la condición, vamos a encender un led. Para ello volveremos a utilizar la función “LED”, cuyo bloque localizamos en la misma sección que el pulsador.

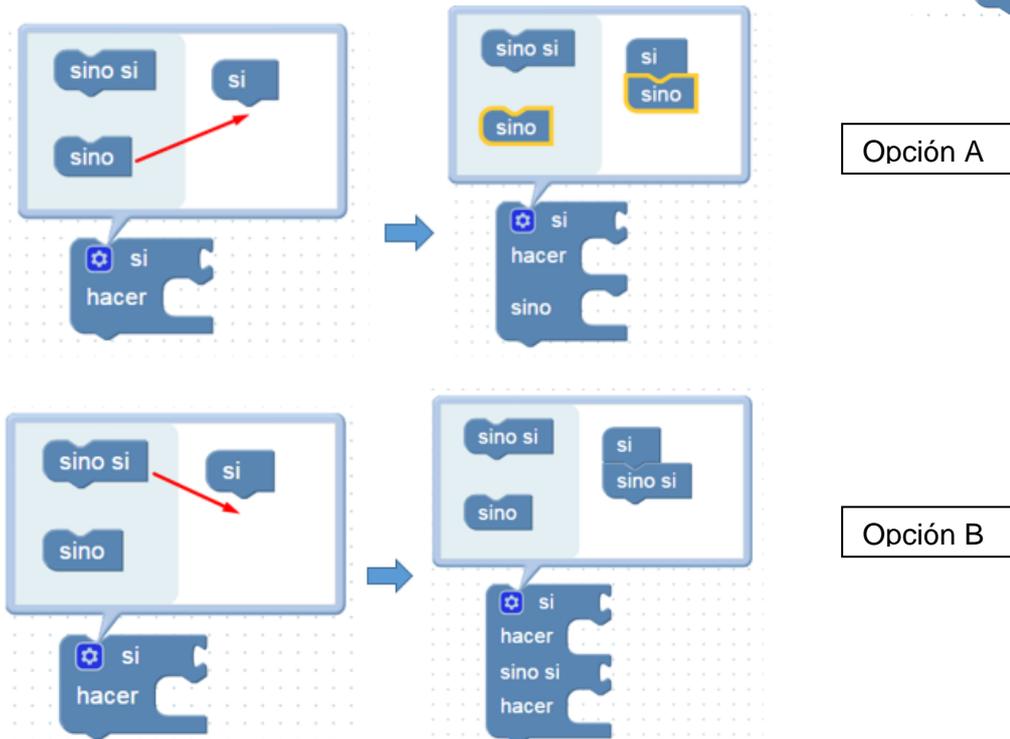


Si dejáramos el programa así tendríamos un problema, ya que la primera vez que se cumpliera la condición, el LED se encendería pero nunca se apagaría. Debemos poner una nueva condición; un **SINO** para en ella cambiar el estado del LED a OFF. Para ello debemos ampliar el condicional de la siguiente manera;

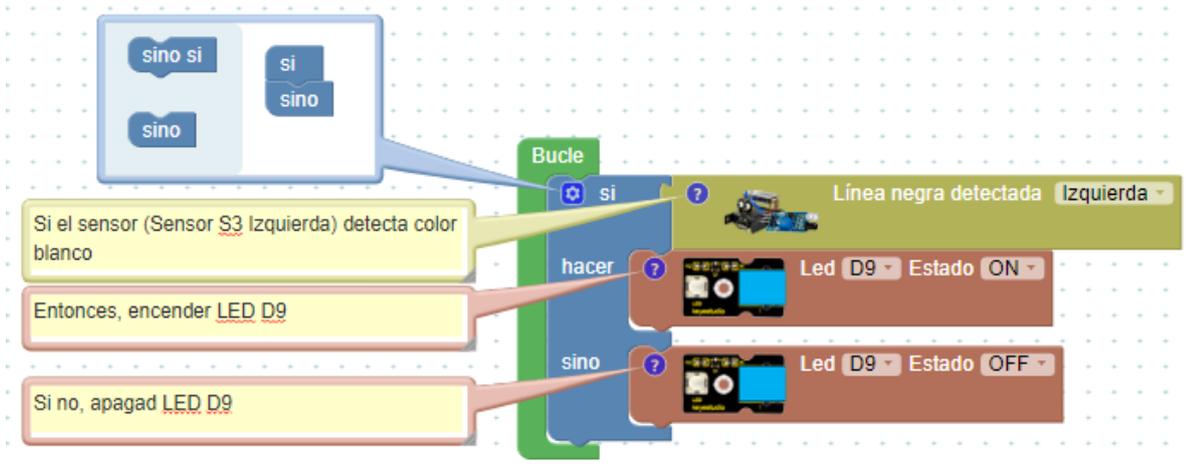
Haciendo clic sobre el símbolo del engranaje señalado con la flecha roja en la imagen, nos aparece un cuadro con funciones con las que podemos ampliar el condicional “Si”.



Hay dos opciones que se consiguen arrastrando los bloques como se aprecia en las siguientes imágenes:



Veremos ejemplos del uso de estas variantes a lo largo de diferentes programas en este documento. Pero continuando con esta actividad, realizaremos un programa que al detectar el sensor de línea izquierdo (S3) el color blanco se encenderá el LED y si no, se apagará.



Comprueba su funcionamiento.

Vamos a dar un paso más y hacer que se cumpla una doble condición.

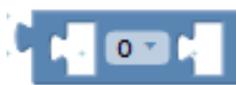
Vamos a modificar el programa anterior para que sólo se encienda el LED cuando dos sensores (S3 y S2) detecten blanco a la vez.

Para ello en el menú **LÓGICA** seleccionamos el siguiente bloque, que devuelve verdadero si ambas entradas con verdadero.



El programa quedaría así;



Prueba a cambiar el  por  ¿Qué ocurre?.

## A05. – Sensor siguelíneas II

En la actividad anterior hemos visto como funciona el sensor siguelíneas, ¿crees que podrías construir un instrumento musical con él? Vamos a intentarlo.

Basémonos en el funcionamiento del sensor, cuando detecta negro no recibe señal y tampoco recibe señal cuando la distancia a la superficie es muy lejana (> de 5 cm) por lo que su respuesta será igual que detectar negro.

Bien, pues vamos a utilizar el zumbador para emitir notas en función de los estados de los sensores siguelíneas.

```

Bucle
  si [Línea negra detectada] Izquierda
    hacer [Zumbador Ms 250 Tono 277.2]
  sino si [Línea negra detectada] Centro
    hacer [Zumbador Ms 250 Tono 293.7]
  sino si [Línea negra detectada] Derecha
    hacer [Zumbador Ms 250 Tono 329.6]
  
```



¿Crees que hay alguna manera de poder tener más notas sólo con tres sensores?

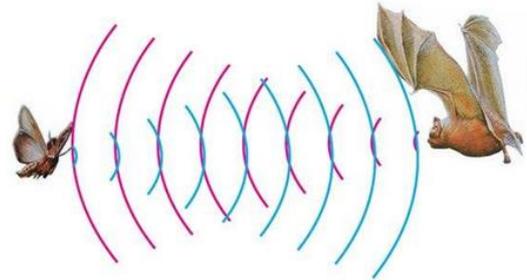
¿Y si utilizas el bloque  ?

Y ahora piensa en diferentes maneras con las que podrías accionar los sensores del siguelíneas para construir tu instrumento musical con tu KEYBOT.



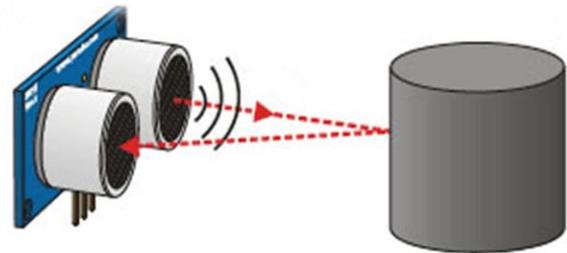
# A06. – Sensor de Ultrasonidos I

Los “ojos” del KeyBot en realidad son un sensor de ultrasonidos. Al igual que los murciélagos tienen muy poca vista y se guían por ultrasonidos, nuestro robot puede hacer lo mismo utilizando este sensor.



El modo de funcionamiento de este sensor es muy sencillo, uno de los dos “ojos” emite ondas ultrasónicas (no perceptibles para el oído humano) después de la señal de disparo. Cuando estas ondas ultrasónicas encuentran un objeto chocan y se reflejan como eco. Al volver son recibidas por el otro “ojo”, por lo que se puede determinar la distancia del objeto a partir de la diferencia de tiempo entre la señal de disparo y la señal de eco.

Conociendo la velocidad del sonido, que es de 340 m/s, y sabiendo el tiempo transcurrido entre la emisión del sonido y su recepción, calcularemos muy fácilmente la distancia del objeto.

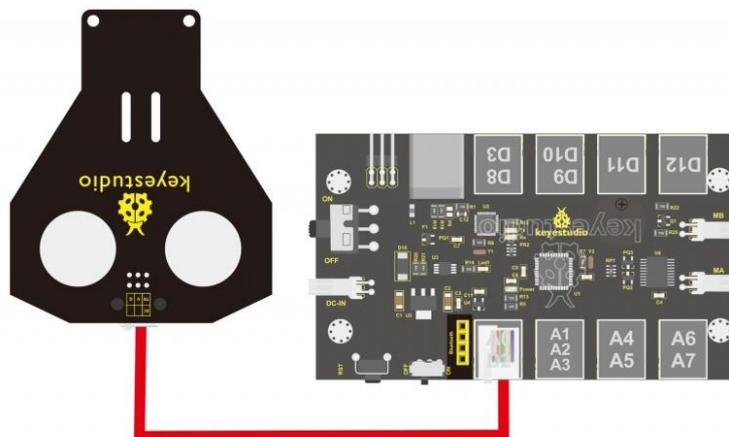


$$\text{Tiempo} = 2 * (\text{Distancia} / \text{Velocidad})$$

$$\text{Distancia} = \text{Tiempo} \cdot \text{Velocidad} / 2$$

Como todo en la vida, este sensor tiene sus limitaciones, no podremos localizar objetos a distancias mayores de 3 m. ni a distancias menores de 2,5 cm. Aun así, el rango de valores es más que suficiente para nuestros propósitos.

Nuestro sensor de ultrasonidos lo tenemos conectado al Pin A0-D2 como muestra la imagen.



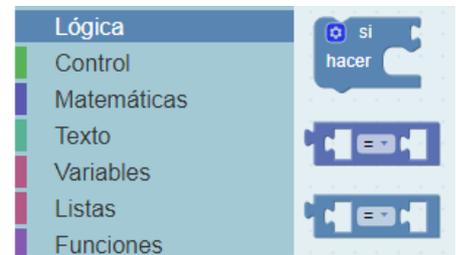
En la siguiente actividad vamos a utilizar el sensor de ultrasonidos para que cuando detecte un objeto a una distancia menor de 25 cm se encienda un LED. (Conectaremos el LED en el Pin D9).



ArduinoBlocks tiene un bloque específico para el sensor de ultrasonidos dentro del menú **KeyBot**.



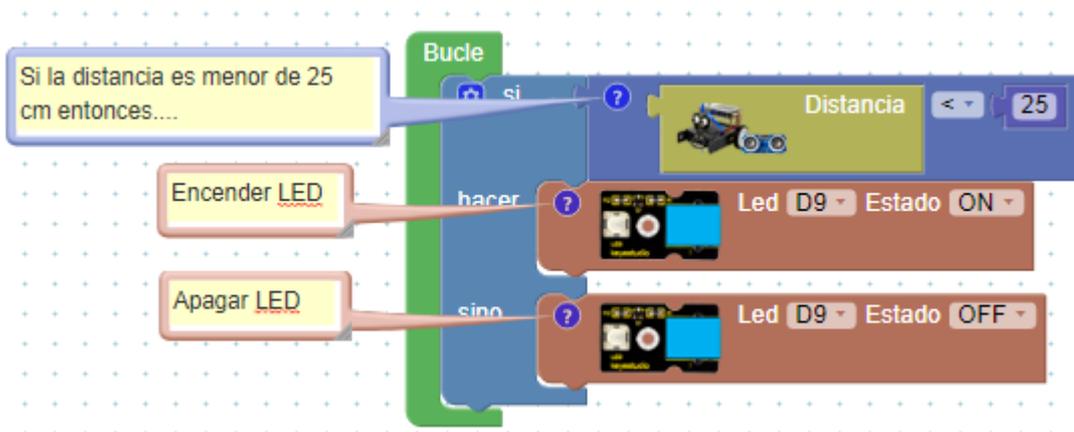
Utilizaremos nuevamente el bloque condicional para realizar esta actividad. En esta ocasión usaremos el “*Menor que*”



Dentro del menú de matemáticas necesitaremos usar el bloque “número” para dar el valor de los 25 cm.



Bien, con todo esto el programa quedaría de la siguiente manera:



# A07. – Sensor de Ultrasonidos II

En esta nueva actividad vamos descubrir dos cosas nuevas, la primera es el **“Puerto Serie”** para poder visualizar por la pantalla del ordenador los datos que nos envía el **KeyBot** y la segunda un nuevo bloque del menú **“Tiempo”** que es la función **“Ejecutar cada”**

Vamos a aprovechar la función **“Ejecutar cada”** junto con la función **“Enviar”** para ver los datos del sensor de ultrasonidos en la pantalla del ordenador.

Pero antes de hacer el ejercicio vamos a ver las distintas funciones poco a poco.

- Gestión de tiempos con el bloque **“Ejecutar cada”**.



Encontramos el bloque en el menú **“Tiempo”**. Este bloque ejecuta una vez cada X tiempo las órdenes que estén dentro de él. La gran diferencia con bloque **“Espera”** es que **NO** detiene el programa en el estado anterior a él, como si hace el bloque **“Espera”**.

Dentro de menú **“Puerto serie”** tenemos el bloque **“Enviar”**, que es la función que utilizaremos para imprimir datos enviados desde la placa de control el **KeyBot** a el ordenador.



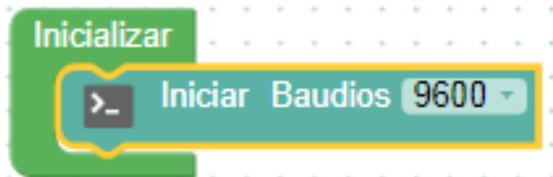
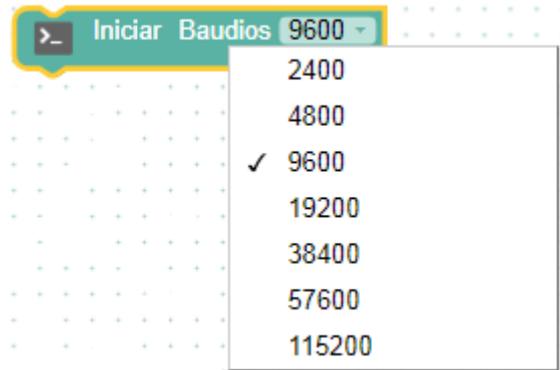
Podemos enviar cualquier orden o dato que queramos solo con introducirlo en el espacio en blanco entre comillas del bloque.



Lo primero que debemos hacer es inicializar la comunicación entre la placa Arduino y el ordenador. Utilizaremos el siguiente bloque:

Por defecto lo dejaremos a la velocidad de 9.600 Baudios.\*

*\*El baudio (en inglés baud) es una unidad de medida utilizada en telecomunicaciones, que representa el número de símbolos por segundo en un medio de transmisión digital.1 Cada símbolo puede comprender 1 o más bits, dependiendo del esquema de modulación.*



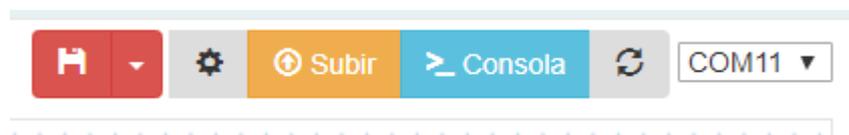
Este bloque lo colocaremos dentro de la función “**INICIALIZAR**”.

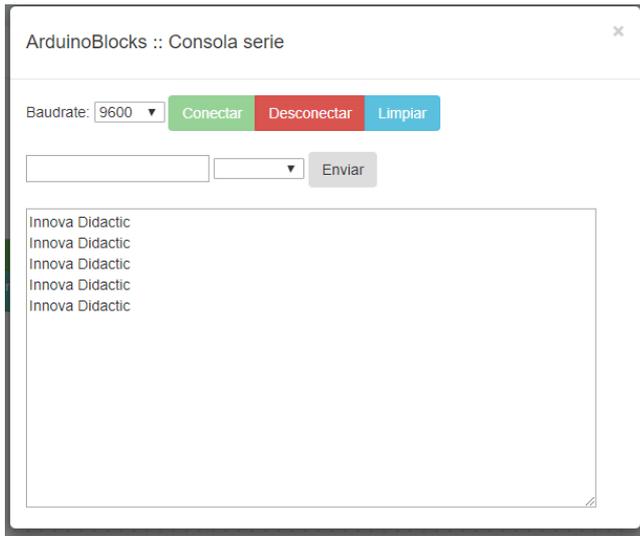
Bien, con todo lo visto por el momento, realiza el siguiente programa de ejemplo;



Con este programa lo que hacemos es enviar a la pantalla del ordenador la frase “Innova Didactic” una vez por segundo (1000 ms).

Tras “Subir” y cargar el programa, pulsa el botón “**Consola**”, se abrirá una nueva ventana en la que clicaremos sobre el botón “**Conectar**”, veremos en la pantalla del ordenador los valores enviados.





Hay que destacar que, si no usamos el bloque “Ejecutar cada”, enviamos al ordenador un dato cientos de veces por segundo, lo que satura la comunicación y bloquea el sistema. De ahí la importancia de este bloque.

También podríamos usar el bloque “Esperar” para enviar datos cada segundo, pero entonces, en los tiempos de espera, el robot no podría escuchar otras órdenes ni realizar otras acciones. Tranquilos, veremos todo esto más adelante con calma... ¡y con más ejemplos!

Ahora vamos a aplicar estos dos conceptos para leer los valores del sensor de ultrasonidos.

Para hacerlo vamos a utilizar el bloque del sensor de ultrasonido como dato para enviar a la pantalla del ordenador.



El programa quedará de la siguiente manera;



Introducimos el siguiente programa y abrimos la consola:

ArduinoBlocks :: Consola serie

Baudrate: 9600 Conectar Desconectar Limpiar

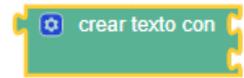
Enviar

```
14.66
15.03
12.83
12.86
15.76
18.78
19.05
18.17
14.88
18.38
20.53
7.86
```

Vemos como al colocar la mano delante del sensor e ir cambiando su distancia, el valor que aparece en la pantalla va variando. Los valores que aparecen son directamente en cm.

Fíjate en la velocidad del **Baudrate**, por defecto es 9.600, es importante que sea la misma que tenemos en el programa al Inicializar.

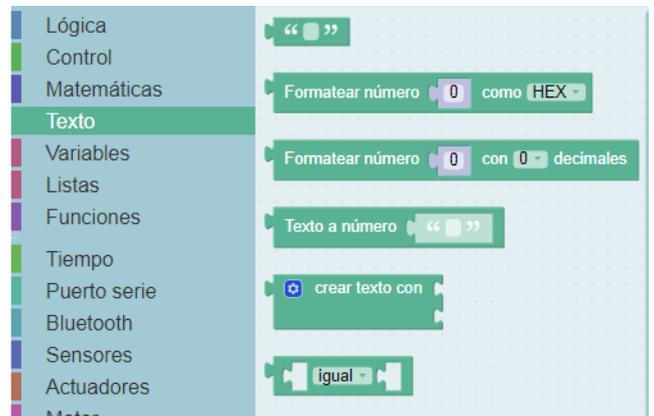
Para que quede un poco más chulo y aparezca "cm" al lado del número que indica la distancia debemos ir al menú "TEXTO" y seleccionar el bloque "Crear texto con"



Y el bloque para escribir texto



Con estos dos nuevos bloques modificamos el programa anterior quedándonos de la siguiente manera;



ArduinoBlocks :: Consola serie

Baudrate: 9600 Conectar Desconectar Limpiar

Enviar

```
13.19 cm
13.19 cm
13.21 cm
13.21 cm
13.21 cm
13.21 cm
14.22 cm
31.05 cm
16.26 cm
21.21 cm
```

El resultado es el siguiente;

# A08. – Sensor de Ultrasonidos III

¿Has pensado como funcionan los avisadores acústicos de los coches para aparcar? ¿e gustaría hacer uno?

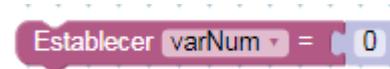
En esta práctica haremos un detector de aparcamiento que a medida que la distancia sea menor, el intervalo entre pitidos sea más pequeño.

Vamos a aprovechar también para introducir el concepto de variables. Lo vimos inicialmente con la práctica del LED en la que incrementábamos y disminuíamos su brillo utilizando una variable "i". En esta ocasión vamos a profundizar un poco más.

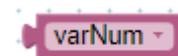
Las variables son elementos muy comunes en programación. Básicamente, crear una variable es darle un nombre a un dato o a una lectura. Por ejemplo, las mediciones de valores de temperatura las podemos guardar en una variable que se llame "Temperatura" o las del sensor de ultrasonidos en una llamada "Distancia", etc. No es obligatorio su uso, pero nos permiten trabajar más cómodamente, además, como podemos personalizar su nombre, ayudan a clarificar el código y utilizar un lenguaje más natural.

Al trabajar con variables vamos a tener dos tipos de bloques principales:

1. El bloque en el que le damos valor a la variable:



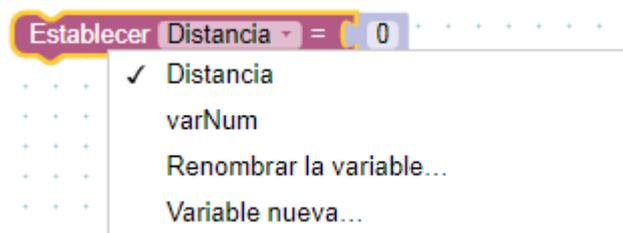
2. Y el bloque de la propia variable creada, para poder insertarla y combinarla con otros bloques:



También podemos personalizar el nombre de la variable, de la siguiente forma:



Una vez creada la nueva variable, podemos seleccionarla haciendo clic sobre el desplegable:



Ten en cuenta que las variables solo pueden estar formadas por una palabra. Si quieres incluir varias palabras, puedes usar el truco de separarlas con una barra baja “\_”, como en el ejemplo, “Valor\_Distancia”.

Vistas las variables vamos con la actividad. Vamos a hacer que el tiempo entre pitidos dependa de la distancia, para ello debemos crear dos variables; a la primera la llamaremos “Distancia” y a la segunda “Tiempo\_Pitidos”.

```

Bucle
  Establecer Distancia = 0
  Establecer Tiempo_Pitidos = 0
  
```

A la variable “Distancia” le daremos el valor del sensor de ultrasonidos y a la variable “Tiempo\_Pitidos” le daremos un valor aplicando una fórmula matemática en la que primero convertiremos todo a un número entero y después lo multiplicaremos por un factor (12 por ejemplo). Esto es porque el tiempo se mide en milisegundos y la distancia se da en cm, por ejemplo; una distancia de 50 cm nos dará un tiempo entre pitidos de  $50 \times 12 = 600$  ms. Se necesita poner lo de número entero y que la distancia puede tener decimales y el bloque “Esperar” no los admite. De todo modos puedes cambiar ese factor por otro que se ajuste más.

```

Bucle
  Establecer Distancia = Distancia
  Establecer Tiempo_Pitidos = Número entero Distancia × 12
  
```

El programa final sería el siguiente;

```

Bucle
  Establecer Distancia = Distancia
  Establecer Tiempo_Pitidos = Número entero Distancia × 12
  Zumbador Ms 100 Tono 1000
  Esperar Tiempo_Pitidos milisegundos
  
```

Vemos en el programa que la frecuencia con la que se realiza cada ciclo, que equivale a la frecuencia con la que damos cada pitido, depende directamente de la distancia del [KeyBot](#) al objeto.

En el programa anterior hay un inconveniente, y es que el robot siempre va a pitar, aunque el objeto que esté delante se encuentre a mucha distancia. Ahora vamos a poner una nueva condición para que este pitido solo comience cuando el objeto esté a una determinada distancia mínima. Para el ejemplo que se propone, se ha probado con 50cm, pero, igual que antes, podéis ajustarlo como queráis.

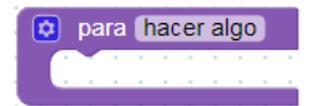
```

Bucle
  Establecer Distancia = Distancia
  Establecer Tiempo_Pitidos = Número entero Distancia × 12
  si Distancia < 50
  hacer
    Zumbador Ms 100 Tono 1000
    Esperar Tiempo_Pitidos milisegundos
  sino
    Zumbador Ms 100 Tono 0
  
```

## A09. – Funciones

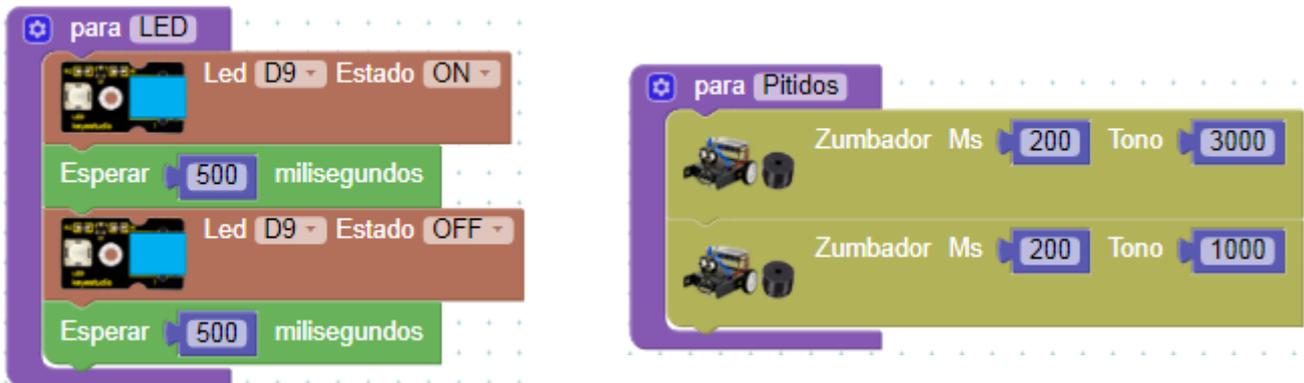
Una función es simplemente un conjunto de instrucciones a las que damos un nombre, para no tener que repetir las en diferentes partes del programa y para clarificar y ordenar el código. Al crear una función, se genera automáticamente un bloque de dicha función, que ya puede ser insertado en cualquier parte del programa.

Para ello, tenemos que usar el bloque “**para**” que encontramos en el apartado “**Funciones**” de **ArduinoBlocks**.

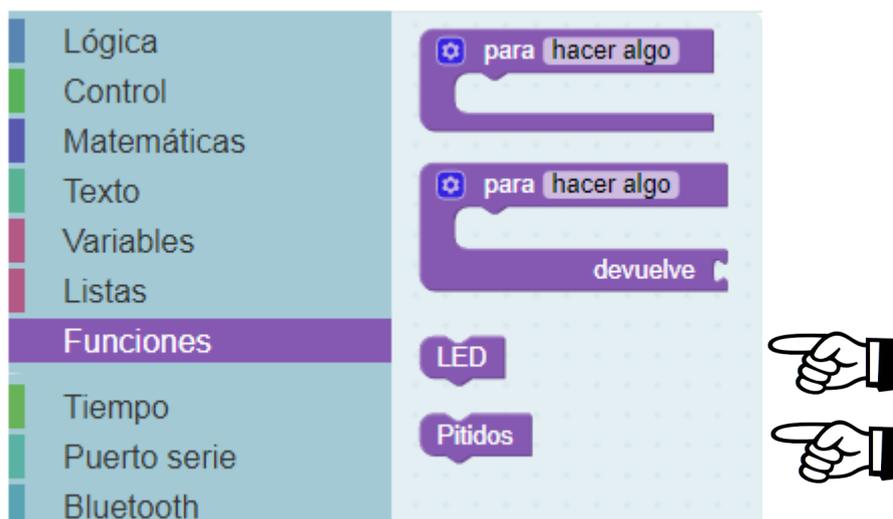


Primero tenemos que darle un nombre a la función, e incluir dentro de la misma, el conjunto de instrucciones que queremos que realice el robot cada vez que la usemos.

Por ejemplo, podemos definir dos funciones; la primera la llamaremos “LED” y en ella encenderemos y apagaremos el LED del Pin D9. La segunda función la llamaremos “Pitidos” y su finalidad será activar el zumbador con dos pitidos distintos. Quedarían así;



Si ahora vamos a la sección “**Funciones**” del panel izquierdo de **ArduinoBlocks**, encontraremos ya las dos nuevas funciones creadas, disponibles para ser seleccionadas e insertadas en cualquier parte del código.

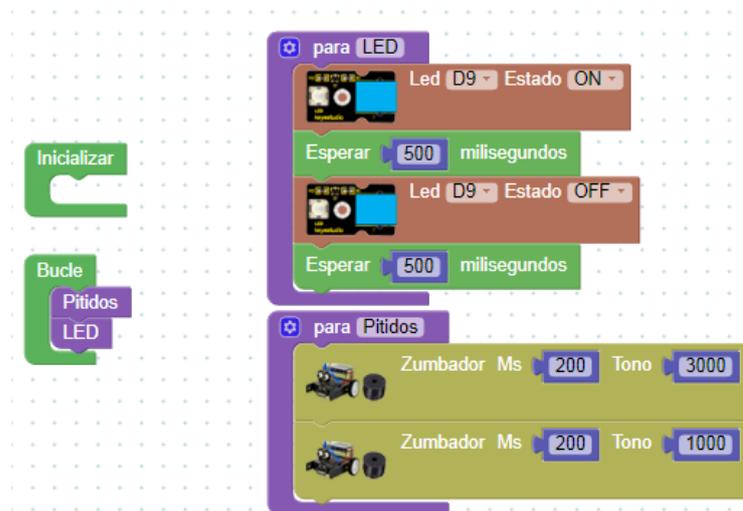


Es importante destacar que las funciones no se definen dentro del “Bucle”. Se crean fuera y luego se insertan dentro en los momentos en los que queramos que se ejecuten.



Cuando creamos un programa pequeño (de pocos bloques), no suele merecer la pena definir funciones, ya que no ahorra mucho tiempo. Pero cuando realizamos programas más extensos o con partes iguales que se repiten, es una buena práctica hacer uso de ellas, tanto por comodidad, como por claridad del programa realizado.

El programa completo sería el siguiente;



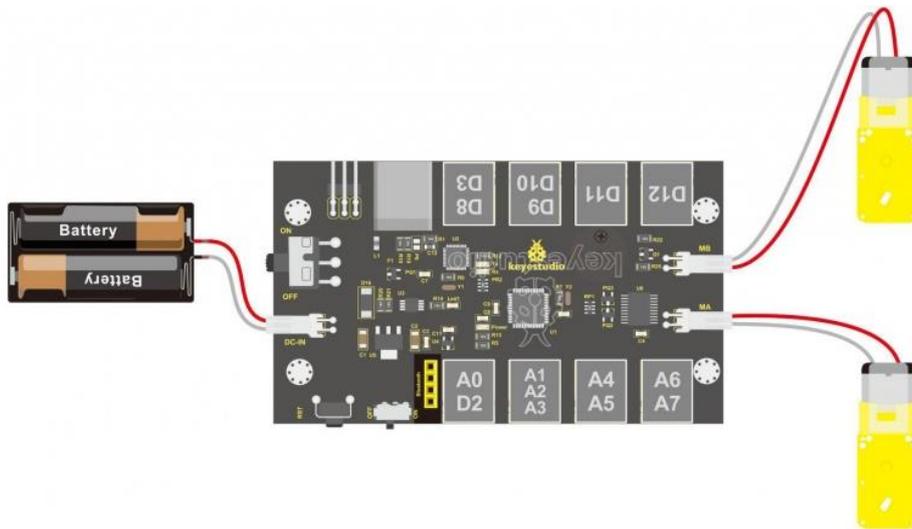
Este ejemplo lo hemos hecho para entender como se hacen las funciones. Prueba a variar el programa y déjalo como el siguiente;



## A10. – Movimientos con KeyBot

Llegado a este punto vamos a empezar a conocer como podemos hacer movimientos con nuestro **KeyBot**. Para realizar movimientos el **KeyBot** dispone de dos motorreductores CC. (Izquierdo y Derecho) con los que conseguiremos 5 movimientos básicos; Adelante, Atrás, Giro Derecha, Giro Izquierda y Paro. En la placa de control existen dos conexiones establecidas para conectar los motores directamente. Internamente estas conexiones son las siguientes en el microprocesador;

	Interface de dirección del motor	Interface de velocidad del motor
Motor A (Izquierdo)	<b>D4</b>	<b>D5</b>
Motor B (Derecho)	<b>D7</b>	<b>D6</b>



!!!RECUERDA!! El motorreductor que tiene el cable más corto es el que se conecta a Pin MA (Motor Izquierdo) y el que tiene el cable más largo al Pin MB(Motor dercho).

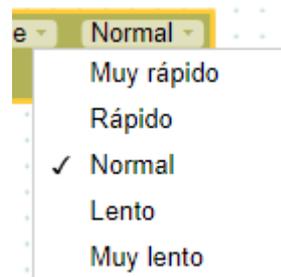
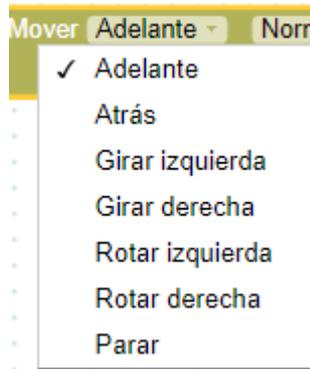
En el menú KeyBot de **ArduinoBlocks** existen dos bloques específicos para controlar el movimiento del robot.

The screenshot shows the 'KeyBot' menu in the ArduinoBlocks software. The menu items are:

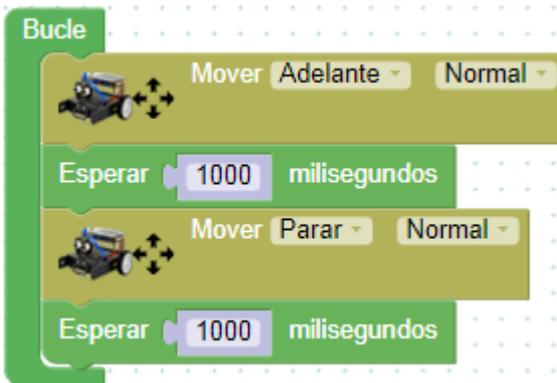
- Mover Adelante - Normal
- Motor Izquierda - Adelante - Velocidad 255
- Distancia
- Línea negra detectada - Izquierda
- Zumbador Ms 500 Tono 1000

Two hand icons are pointing to the 'Motor Izquierda' and 'Zumbador' blocks.

Con el primer bloque se programan los movimientos del robot, es decir, se programan los dos motores a la vez. En sus menús podremos controlar tanto el tipo de movimiento como la velocidad con la que se ejecutan.



Vamos con el primer programa con el cual el robot avance 1 segundo y se pare otro. ¿A qué es muy sencillo? Practica combinando movimientos y velocidades.



¿Qué diferencia encuentras entre “Giro” y “Rotar”? Efectivamente, en el “Giro” la rueda interna está parada, mientras que en “Rotar” gira en sentido contrario.

Ahora vamos a intentar que en su movimiento, el KeyBot realice un cuadrado. Lo primero que deberás calcular es el tiempo que tarda en hacer un giro de 90°. En este caso son 1100 milisegundos, pero eso es con la alimentación del USB del ordenador (5V). Para realizar movimientos, no necesitamos el USB, lo normal es encender el robot con su interruptor y alimentarlo con las pilas. Esta velocidad también cambiará en función del estado de las baterías. En resumen; deberás calcular los tiempos de giro y avance haciendo pruebas en cada caso Y recuerda que también puedes cambiar la velocidad, *deberás buscar el equilibrio entre tiempo y velocidad.*



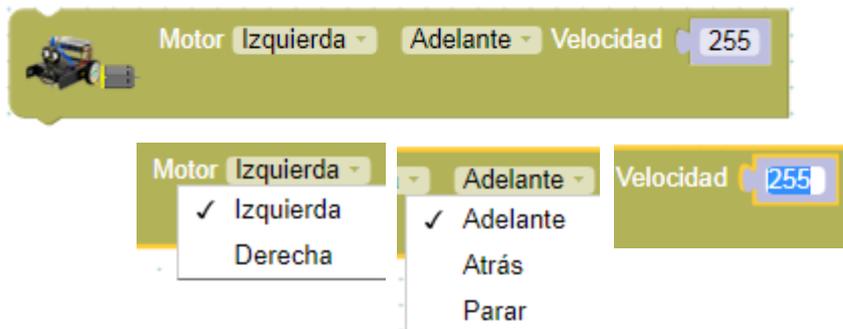
Este programa anterior es muy repetitivo, ya que realiza 4 veces avanzar y girar. En programación existe una función que es la de “repetir XX veces y hacer”. En **ArduinoBlocks** la encontramos dentro del menú “**Control**”.



Usando ese bloque, el programa anterior quedaría de la siguiente manera;



En **ArduinoBlocks** existe otro bloque para el control individual de cada motor. Debemos indicar que motor vamos a utilizar, que dirección de movimiento va a tener y la velocidad. La velocidad lleva un control por PWM que va de 0 hasta 255.



Un valor de 255 equivale a tener el motor al 100%, un valor de 127 al 50%.

El PWM lo vimos cuando se explicó el control de brillo de un LED.

Por lo general valores más bajos de 90-100 no son capaces de mover el motor. Ahora vamos a ponerlo en práctica;

Primero movemos el motor izquierdo;

Haz lo mismo con el motor derecho.

Y a ahora con los dos motores a la vez;

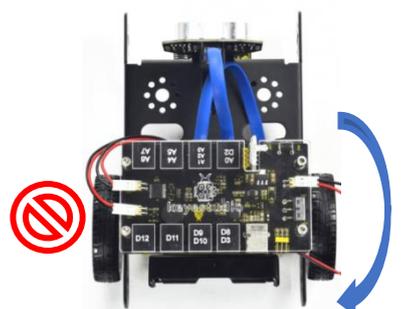
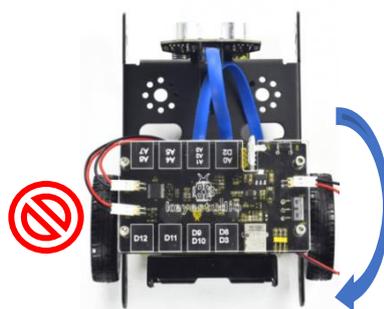
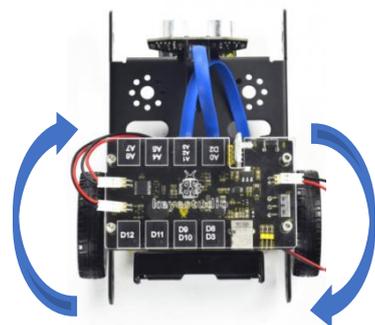
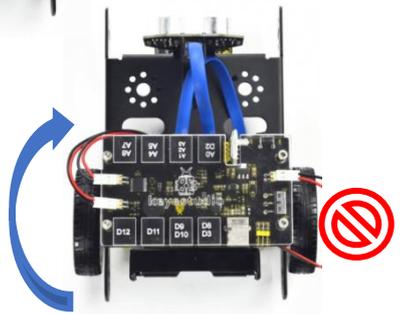
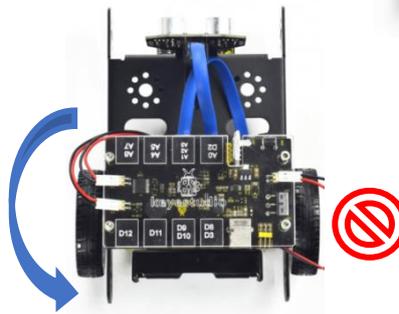
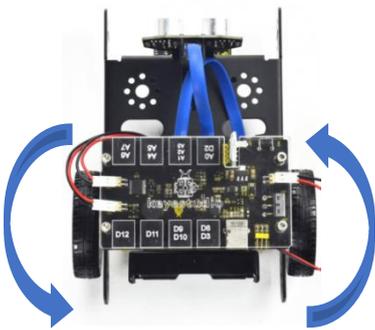
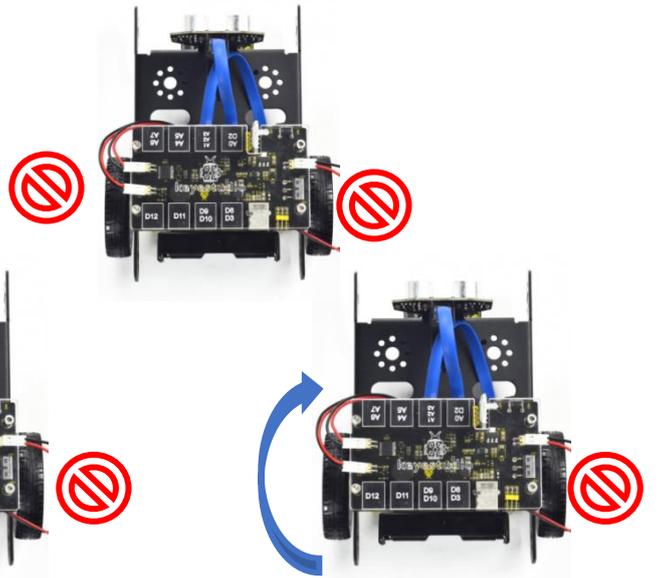
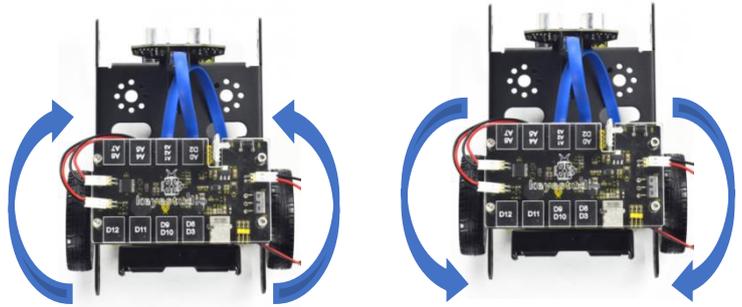
```

Bucle
  Motor Izquierda Adelante Velocidad 255
  Esperar 1000 milisegundos
  Motor Izquierda Parar Velocidad 255
  Esperar 1000 milisegundos
  Motor Izquierda Atrás Velocidad 255
  Esperar 1000 milisegundos
  
```

```

Bucle
  Motor Izquierda Adelante Velocidad 255
  Motor Derecha Adelante Velocidad 255
  Esperar 1000 milisegundos
  Motor Izquierda Parar Velocidad 255
  Motor Derecha Parar Velocidad 255
  Esperar 1000 milisegundos
  Motor Izquierda Atrás Velocidad 255
  Motor Derecha Atrás Velocidad 255
  Esperar 1000 milisegundos
  
```

Realiza diferentes órdenes cambiando los motores, las direcciones y las velocidades creando giros rápidos, lentos,...



## A11. – Movimientos con KeyBot II

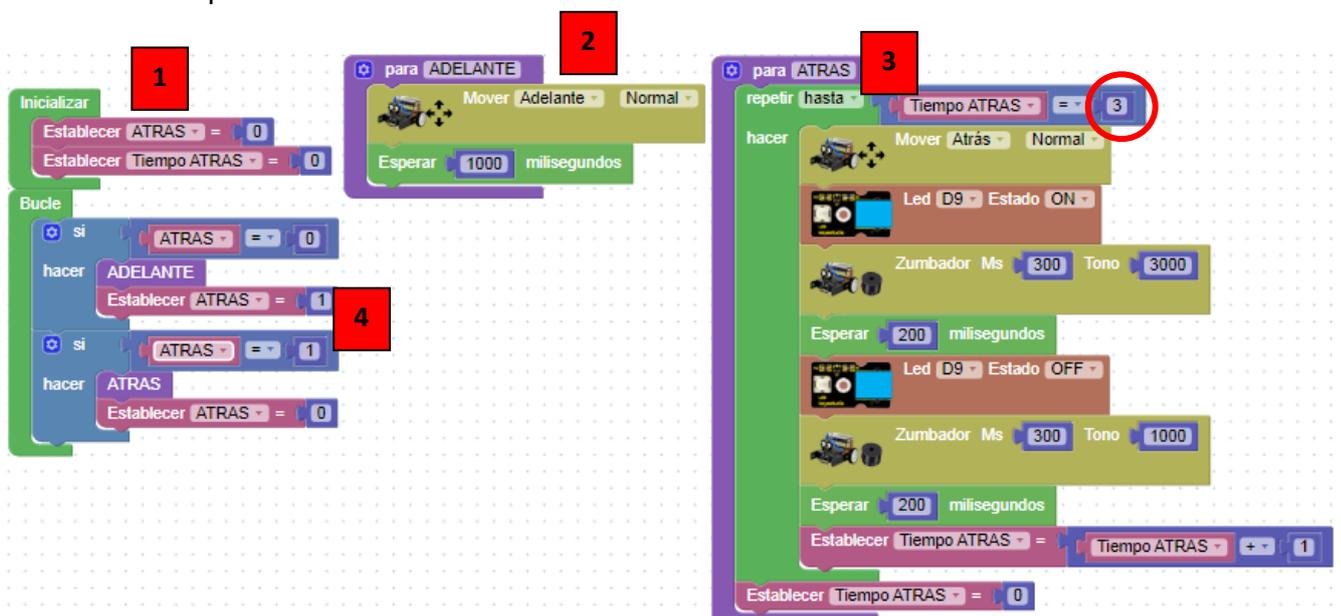
En esta práctica vamos a combinar una actividad vista anteriormente con el movimiento del robot. Vamos a hacer que cuando el robot de marcha atrás se encienda el LED y emita un pitido, tal y como hacen los camiones y maquinaria industrial.

Para ello conectamos nuestro LED en el Pin D9, el zumbador recuerda que es el Pin D13 y el motor izquierdo en MA y el derecho en MB.

Inicialmente parece un programa sencillo, pero no lo es tanto. Una de las desventajas que tiene Arduino y la mayoría de microprocesadores programados es que no pueden hacer tareas paralelas, van realizando las órdenes en cascada, una detrás de otra y según el orden en el que están programadas. Para intentar solventar esta “desventaja” el siguiente programa lo hemos realizado de la siguiente manera;

- En primer lugar se han creado dos “**VARIABLES**” (1); “**ATRÁS**” y “*TiempoAtras*” y se han inicializado con un valor igual a 0.

- En segundo lugar se han creado dos “**FUNCIONES**”; “**ADELANTE**” y “**ATRÁS**”. La función “**ADELANTE**” (2) es muy sencilla y sólo indica el movimiento del robot hacia adelante y el tiempo. La función “**ATRÁS**” es un poco más complicada. En ella existe el bloque de “**repetir hasta .... hacer**” que está en el menú **CONTROL**. Este bloque funciona de la siguiente manera; es un bucle que repite un número determinado de veces todas las acciones que existan en su interior. En nuestro caso ir hacia atrás, encender el LED, emitir un sonido durante 300 ms, esperar 200 ms, apagar el LED, emitir otro sonido durante 300 ms, esperar 200 ms y establecer en la variable “*TiempoATRÁS*” una suma de una unidad. De este modo al ejecutarse por primera vez su valor será 1, la segunda vez será 2, la tercera 3, y así sucesivamente. Tal y como está, el programa se repetirá 3 veces cumpliendo la condición de “*TiempoATRÁS*” =3 (Valor que podemos cambiar como si fuera el tiempo. Fíjate que un ciclo completo dura 1 seg. (300+200+300+200), por lo que ese 3 en realidad equivaldría a 3 seg). Al salir del ciclo el programa vuelve al valor 0 a la variable “*TiempoATRÁS*”.



.- Y por último, vamos con el Bucle principal (4). En él hay dos condicionales; en el primero cuando cumpla la condición que la variable “ATRAS” sea 0, el robot realizará la función **ADELANTE** y al finalizar está cambiará el estado de la variable por 1. En ese momento se cumplirá el segundo condicional y el robot realizará la función **ATRÁS**. Al finalizar esta función la variable volverá al valor de 0 por lo que el robot reiniciará su marcha hacia adelante.



```

Bucle
si (ATRAS = 0)
  hacer
    ADELANTE
    Establecer ATRAS = 1
si (ATRAS = 1)
  hacer
    ATRÁS
    Establecer ATRAS = 0
  
```

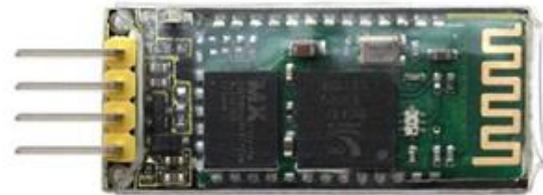
# A13. – Módulo Bluetooth

En esta actividad aprenderemos a controlar nuestro robot **KeyBot** con un móvil Android a través del módulo Bluetooth. El Bluetooth es un método inalámbrico de transmisión de datos. La tecnología Bluetooth es una tecnología estándar inalámbrica que permite el intercambio de datos de corto alcance entre dispositivos fijos, dispositivos móviles y redes de edificios de área personal (ondas de radio UHF en la banda ISM de 2.4 a 2.485 GHz).

Hay dos tipos de módulos Bluetooth de uso común en el mercado, los modelos HC-05 y HC-06. La diferencia entre ellos es que el HC-05 es maestro-esclavo, quiere decir que además de recibir conexiones desde un PC, Tablet o móvil Android, también es capaz de generar conexiones hacia otros dispositivos bluetooth. El HC-06 solo puede funcionar en modo esclavo, que solo puede aceptar el comando superior. El Bluetooth de nuestro **KeyBot** es HC-06.

La conexión del Bluetooth es directamente en la placa de control en los pines amarillos. **OJO!!!** Al tener la placa protectora de metacrilato **sólo** nos permite colocar el Bluetooth en una posición. El módulo de comunicaciones Bluetooth tiene 4 conexiones:

- .- Vcc : Alimentación 5V (+).
- .- Gnd: Alimentación 0V (-).
- .- Tx: Transmisión de datos.
- .- Rx: Recepción de datos.



**IMPORTANTE!!!** Para cargar los programas siempre es necesario desconectar el módulo Bluetooth, si lo dejamos conectado dará problemas al cargar.

El Bluetooth del **KeyBot** es Bluetooth 2.0. Actualmente, sólo es compatible con los dispositivos Android. No es compatible con dispositivos Apple. Después de instalar el asistente en serie, primero debemos conectar el dispositivo, abrir el Bluetooth móvil, buscar un dispositivo Bluetooth. Si encuentra un dispositivo Bluetooth llamado HC-06, emparéjalo e ingresa la contraseña 1234, finalmente deberías de ver el dispositivo emparejado a tu móvil.

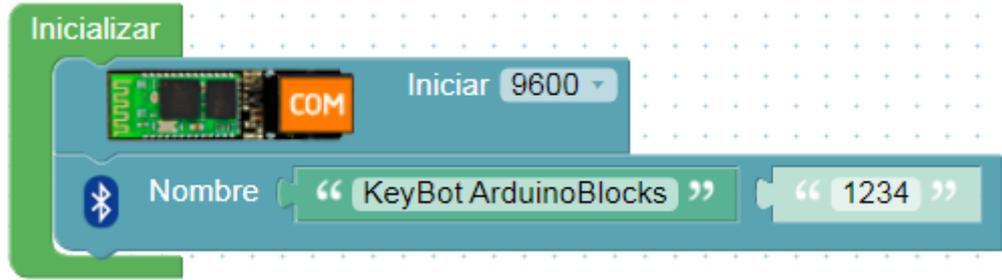
Cuando el Bluetooth está desconectado hay un pequeño led rojo parpadeando, cuando está emparejado el led rojo se queda fijo.

Comenzamos con la programación, vamos a realizar una primera práctica muy sencilla en la que vamos a encender y apagar el LED del Pin D9.

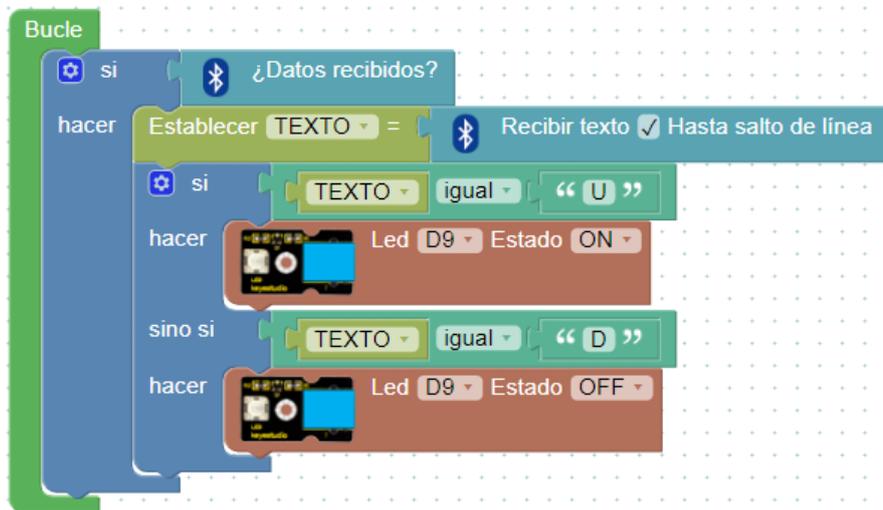
En **ArduinoBlocks** disponemos de un menú específico para el Bluetooth con todos los bloques necesarios para programarlo.



Comenzaremos “Iniciando” el Bluetooth. Y fijaremos una velocidad de comunicación de 9600 Bauds.



A continuación realizaremos el siguiente programa en el cual al recibir la orden “U” enviada desde el móvil se encienda el LED y al recibir la orden “D” se apague.



### FALTA LA APP PARA EL MOVIL

Te puedes crear tu propia aplicación para el móvil utilizando APPInventor (<https://appinventor.mit.edu/>). Una plataforma online en la que puedes programar tus propias aplicaciones de modo muy similar a Scratch.



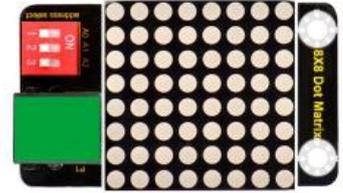
**MIT**  
APP INVENTOR

El siguiente material es opcional y no viene incluido en el Kit básico.

## A14. – Matriz de LED 8 x 8

Para realizar la siguiente práctica es necesario disponer de una Matriz de EASY Plug LED 8x8 I2C (Ref. [KS0139](#)).

Este elemento se conecta a través de un bus llamado I2C, en el **KeyBot** el puerto I2C es el conector A4/A5 y también se conecta con los terminales RJ11.



I2C es un protocolo para poder conectar varios dispositivos de forma muy fácil con un par de cables para los datos y otros dos para alimentación y tierra o GND.

Además, existen Hubs para poder conectar más de uno a la vez. Keyestudio Módulo EASY Plug hub I2C ([KS0390](#))



La matriz de LED 8x8 dispone de un total de 64 leds con los cuales podemos hacer infinidad de caras, iconos, letras, números,... hay cantidad de opciones ya prediseñadas y **ArduinoBlocks** da la opción de crearlas por ti mismo.

Como no podía ser de otra manera **ArduinoBlocks** tiene un menú específico para programar la matriz de LED 8x8 en modo **KeyBot**.



The screenshot shows the ArduinoBlocks software interface. On the left, a vertical menu lists various categories: Lógica, Control, Matemáticas, Texto, Variables, Listas, Funciones, Tiempo, Puerto serie, Bluetooth, Sensores, Actuadores, Motor, Pantalla LCD, Pantalla OLED, **LedMatrix 8x8** (highlighted), Memoria, and KeyBot. A hand icon points to the 'LedMatrix 8x8' option. The main workspace displays several blocks for controlling the LED matrix:

- Iniciar I2C 0x70 v1**: A block to initialize the I2C connection.
- Rotación 0°**: A block to set the rotation angle.
- Limpiar**: A block to clear the matrix.
- Bitmap**: A block to load a custom bitmap.
- Bitmap Face normal**: A block to load a pre-defined face.
- Píxel X 0 Y 0 Led ON**: A block to turn on a specific pixel.
- Línea X1 0 Y1 0 X2 7 Y2 7 Led ON**: A block to turn on an entire line.

Additional blocks for 'Inicializar' and 'Bucle' are also visible on the right side of the workspace.

Lo primero que debemos hacer es INICIALIZAR la LedMatrix 8x8. Podemos poner hasta 4 distintas utilizando el Hub de expansión por ello debemos indicar de cual se trata (1, 2, 3 o 4). Lo segundo que debemos hacer es dar la dirección que por defecto será 0x70 y por último hay dos modelos v1 y v2 (utiliza al v2, si ves que los iconos no están centrados cambia al v1).



Una vez inicializada la matriz de LED vamos con el Bucle. En esta ocasión haremos que salgan dos caras, una FELIZ y otra TRISTE con una espera de 1 seg. entre ellas.

Fíjate en la cantidad de iconos de los que dispones;

- Face normal
- Face happy
- Face sad
- Face angry
- Eye normal
- Eye medium
- Eye small
- Eye closed
- Eye look left
- Eye look right
- ✓ Eye look up
- Eye look down
- Eye angry left
- Eye angry right
- Eye sad
- Eye surprise
- Icon heart
- Icon user
- Icon clock
- Icon arrow up
- Icon arrow down
- Icon arrow left



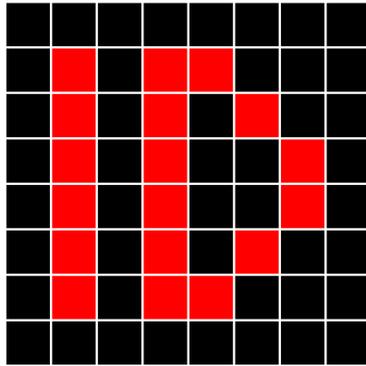
Prueba a cambiar y combinar unos cuantos de ellos cambiando también los tiempos de esperas.

Otro bloque muy curioso y con el que podrás crear tus propios iconos es el bloque de "Bitmap". Vamos a ver cómo se utiliza;



Seleccionando “Ayuda” con el botón derecho, se nos abre otra ventana/pestaña del navegador con la opción de dibujar lo que deseamos:

### LedMatrix - Bitmap Data

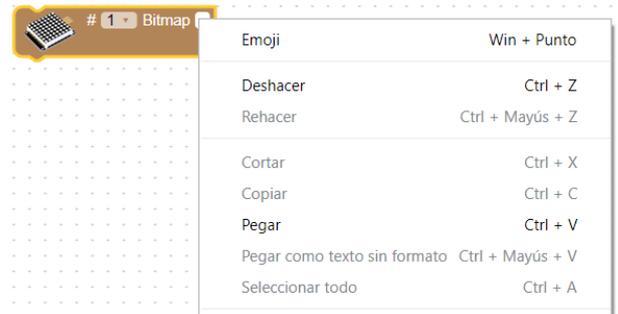


Clear Fill Copy data:

```
B000000000,B01011000,B01010100,B01010010,B01010010,B01010100,B01011000,B00000000
```

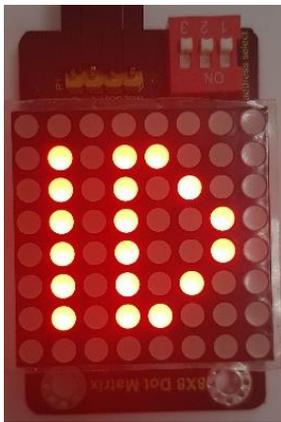


Al hacer click sobre cada cuadradito negro se selecciona y pasa a color rojo. Una vez creado el dibujo que queramos, clicamos en “Copy data:” y volviendo a la ventana de programación sobre el bloque “Bitmap”, en cuadro en blanco, le damos a “Pegar”.



Este es el resultado final.

También puedes hacer gif animados como por ejemplo el típico hombre andando del semáforo en verde de peatones. ¿Te animas?



## A15. – Pantalla LCD 2x16

Para realizar esta práctica es necesario disponer de Keyestudio EASY Plug Módulo LCD 1602 I2C (Ref. [KS0137](#))

Esta pantalla LCD al igual que la Matriz de led 8x8 se comunica por I2C

En la pantalla podemos enviar datos como los valores de los distintos sensores, temperatura, luminosidad y por ejemplo la distancia detectada por el sensor de “Distancia”.

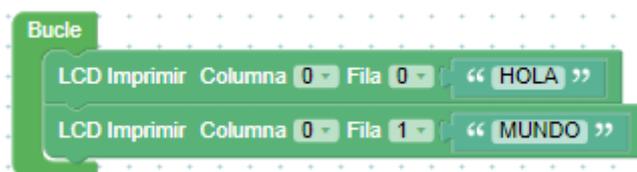


En el bloque “Pantalla LCD” encontramos la función “LCD Iniciar”, sirve para indicar que vamos a usar la pantalla, que hay que poner al bloque “LCD Inicializar”. Usaremos 2x16 si es una pantalla de 2 filas x 16 columnas y 4x20 si es de 4 filas y 20 columnas. El segundo parámetro establece el ADDR que lo dejaremos por defecto en 0x27\*.



Seguidamente dentro del “Bucle” incluiremos la función “LCD Imprimir”. Se puede “imprimir” los datos en la primera fila=0 o en la segunda fila=1 y empezar a escribir en la columna deseada desde la primera columna=0 hasta la última columna=15 o 19 (según modelo).

Vamos con nuestro particular “Hola Mundo”. Realiza este programa;



Ahora cambia los valores de “Columna” a 5 y 4 en el primer y segundo bloque de imprimir respectivamente. ¿Entiendes los conceptos de Columnas y Filas?

Bien, vamos a ver los valores que nos lee el sensor de ultrasonidos por la pantalla LCD. Creamos una variable que la llamaremos “Distancia” a la cual daremos los valores del sensor de ultrasonidos y después “imprimiremos” esos valores. Dejaremos un tiempo de espera para que nos de tiempo a ver el valor y por último “limpiaremos” la pantalla que se vuelva a escribir con la pantalla vacía y no quede nada debajo. Puedes hacer la prueba de no poner ese último bloque y comprobar lo que pasa.

```

Inicializar
  LCD Iniciar 2x16 ADDR 0x27 *
  I2C

Bucle
  Establecer Distancia = Distancia
  LCD Imprimir Columna 0 Fila 0 Distancia
  Esperar 750 milisegundos
  LCD Limpiar
  
```

Mejorando un poco el programa anterior nos podría quedar como sigue;

```

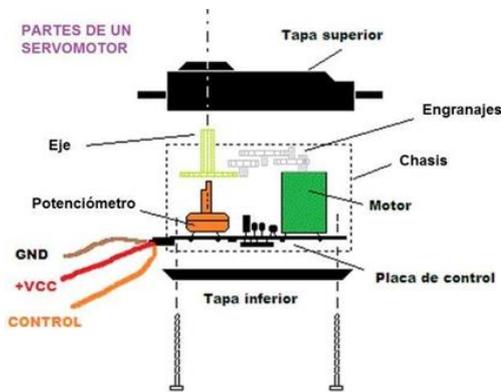
Inicializar
  LCD Iniciar 2x16 ADDR 0x27 *
  I2C

Bucle
  Establecer Distancia = Distancia
  LCD Imprimir Columna 0 Fila 0 " La distancia es: "
  LCD Imprimir Columna 0 Fila 1 crear texto con Distancia " cm "
  Esperar 1500 milisegundos
  LCD Limpiar
  
```

# A16. – Controlar un servomotor

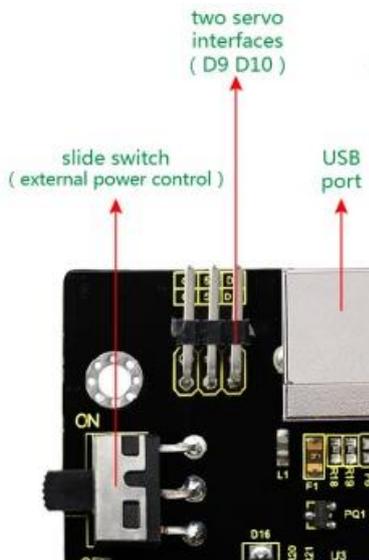
En esta actividad vamos a necesitar un servomotor (Ref. [KS0194](#) o [KS0209](#)).

Un elemento muy utilizado en robótica y el mundo de Arduino es un servomotor, es un motor especial que puede posicionar su eje en un ángulo determinado y lo puede mantener en esta posición. Para funcionar sólo necesita alimentación GND, VCC (5voltios) y una señal de control.



Los servomotores estándar sólo pueden girar 180°, aunque en el mercado podemos encontrar de 270° y de 360° (giro continuo).

En el menú “Motor” encontramos un bloque que se llama “Servo”. Con él tenemos la posibilidad de controlar el servomotor, indicando los grados de rotación (Ángulo) que queremos y el tiempo de retardo (Tiempo que tarda en ir de una posición a otra).



La placa de control del **KeyBot** tiene disponibles los pines D9 y D10 para conectar 2 servomotores. Como estamos usando el D9 para el LED usaremos el D10. Para conectarlo fíjate en el color de los cables; el marrón es el GND, el rojo el 5 v y el amarillo el control. Además existen adaptadores para conectar tantos servos como quieras. Keystudio Adaptador Easy Plug a 3 Pines. Referencia ([KS9006](#))



Por ejemplo, una buena idea es programar, por primera vez, el Ángulo a 0° para descubrir el punto de origen y a partir de aquí montar alguno de los accesorios que vienen con el servo para poder visualizar la rotación del eje.

```

Inicializar
  Servo [Servo] Pin [D10] Grados Ángulo [0°] Retardo (ms) [100]
  
```

Ahora ya podemos practicar con distintos grados, y observar donde apunta la “flecha” del eje.

Un servomotor nos puedes servir para accionar una pala para nuestro KeyBot, para mover un brazo de un robot humanoide, las posibilidades son muchas.



Accesorios servo

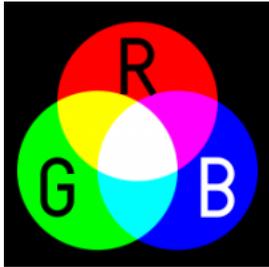
```

Inicializar
  Servo [Servo] Pin [D10] Grados Ángulo [0°] Retardo (ms) [100]

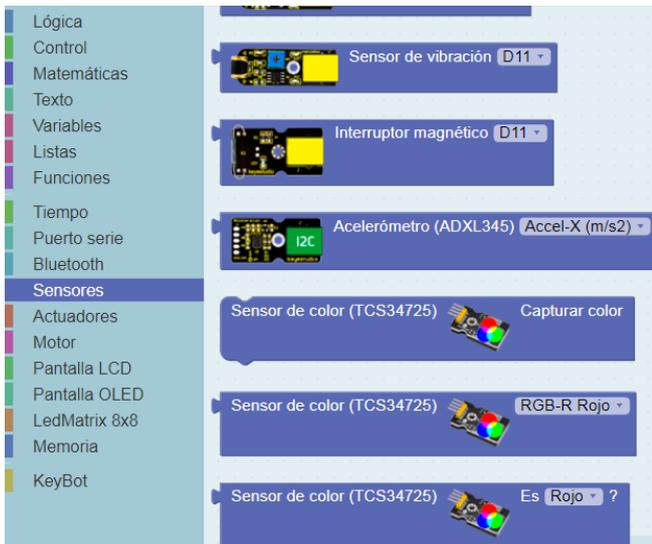
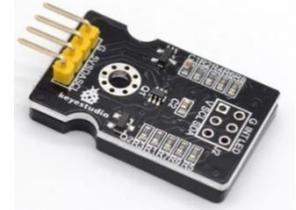
Bucle
  Servo [Servo] Pin [D10] Grados Ángulo [0°] Retardo (ms) [1000]
  Servo [Servo] Pin [D10] Grados Ángulo [90°] Retardo (ms) [1000]
  Servo [Servo] Pin [D10] Grados Ángulo [180°] Retardo (ms) [1000]
  Servo [Servo] Pin [D10] Grados Ángulo [90°] Retardo (ms) [1000]
  Servo [Servo] Pin [D10] Grados Ángulo [0°] Retardo (ms) [1000]
  
```

# A17. – Sensor de Color

En esta práctica vamos a ver el funcionamiento del sensor de color RGB TCS34725 (Ref. [K0407](#)). Este sensor se comunica con la placa de control del **KeyBot** utilizando la interfaz de comunicación I2C. El chip del sensor de color TCS34725 proporciona valores de retorno digital rojo, verde, azul (RGB) y sensible a la luz.



Para poder utilizar este sensor en el **KeyBot** es necesario el Módulo EASY Plug Hub I2C ([KS0390](#)).

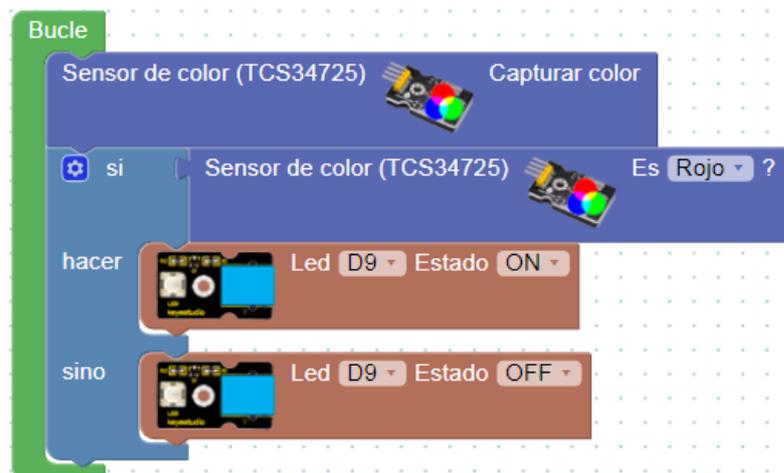


**ArduinoBlocks** tiene 3 bloques para utilizar el sensor de color. Estos bloques los podemos encontrar dentro del menú **“Sensores”**.

Para empezar utilizaremos el sensor de color de tal forma que cuando detecte en color rojo encienda el LED y sino que se apague.

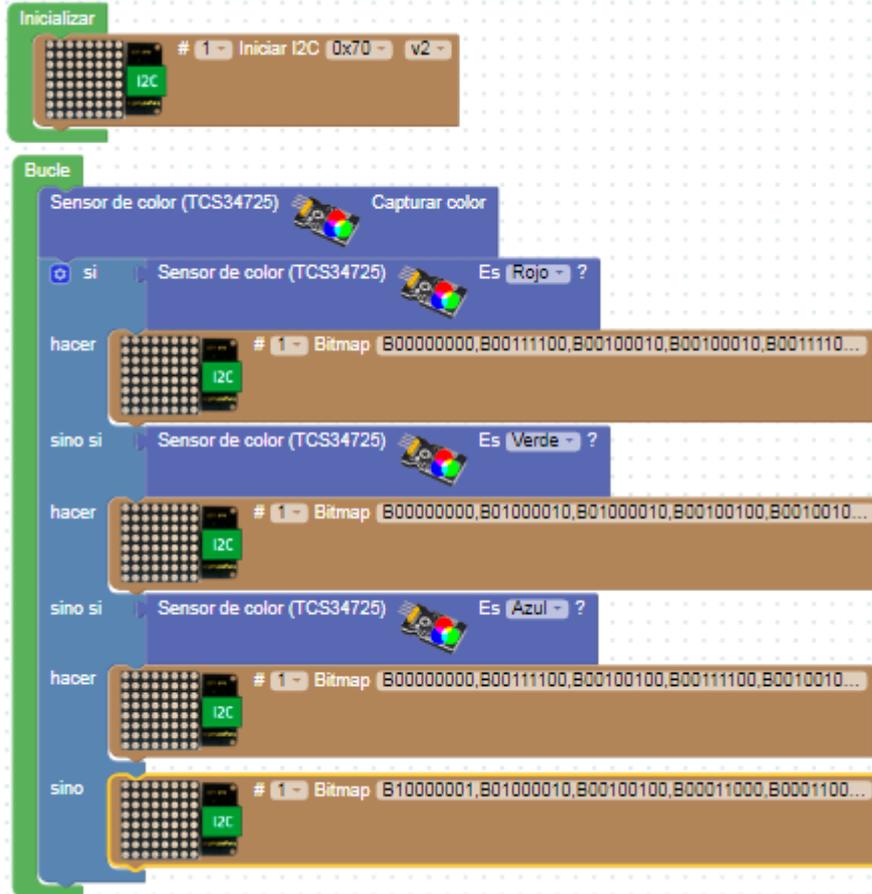
Recuerda el LED va al Pin D9 y el sensor de color al Pin A4-A5

Al inicio del Bucle debemos poner el bloque de “Capturar Color”, para después meter un condicional. Prueba el siguiente programa;



El sensor de color debe estar muy cerca (prácticamente pegando) a la superficie de la que se desea capturar el color para que su lectura sea eficaz.

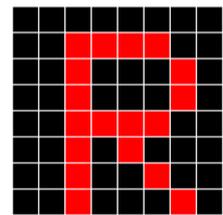
Veamos otro ejemplo; en este caso detectaremos 3 colores (Rojo, Azul y Verde) para que en la Matriz de Led 8x8 aparezca R, A y V en cada caso. Gracias a Hub de expansión podremos conectar el sensor de color y la Matriz de led al Pin I2C (A4/A5).



Recuerda que debes inicializar la matriz de led en la versión que le corresponda (generalmente la V2).

Utiliza la ayuda del bloque **Bitmap** para "dibujar" las letras que necesites, después copia el código y pégalo.

LedMatrix - Bitmap Data



B00000000,B00111100,B00100010,B00100010,B00111100,B00100010,B00100010,B00100010,B00100010,B00100010

También puedes usar la pantalla LCD para que en ella aparezca escrito el color que está detectando.

Este sería el código...



## Proyectos con el KEYBOT completo

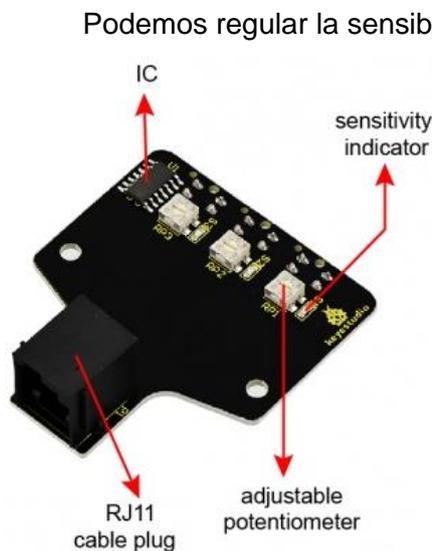
En esta parte del manual veremos proyectos más completos en los que se combinen diferentes sensores y actuadores para que nuestro robot **KeyBot** pueda realizar tareas más o menos complejas de manera totalmente autónoma.

### P1. – KeyBot seguidor de líneas

Una de las funcionalidades más típicas de los robots educativos es realizar con ellos seguidores de líneas autónomos, aunque en la industria también existen muchos ejemplos de este tipo de robots que siguen líneas en grandes fábricas para transportar cualquier tipo de cosas.

Como sensor necesitaremos el sensor seguidor de línea del **KeyBot** que dispone de tres sensores TCRT5000 (S1 (derecha), S2 (centro) y S3 (Izquierda)) recuerda que ya lo explicamos en la [actividad N° 4](#)

Antes de nada dos curiosidades; la primera es que la luz infrarroja que emiten los sensores no la podemos apreciar directamente, pero si miramos los sensores a través de la cámara de un móvil sí que podremos verla claramente. Haz la prueba.



Podemos regular la sensibilidad de cada sensor girando hacia un lado u otro los potenciómetros. Debemos hacer esto para calibrar los sensores en función de su distancia al fondo

y según la iluminación (las luces fluorescentes pueden influir). Junto a los potenciómetros, cada sensor TCRT5000 tiene un pequeño LED rojo. Cuando el fondo es de color blanco, el LED rojo está encendido, cuando el fondo es negro el LED rojo está apagado. Gracias a esto podremos calibrar los sensores de una manera muy sencilla con un pequeño destornillador plano.



Comencemos a ver que valores son las que envían los sensores a la placa de control según el fondo sea blanco o negro. Para ello vamos a crear unas variables especiales, **Variables Booleanas**. Estas variables son especiales ya que no pueden tomar cualquier tipo de valor, únicamente pueden tomar dos valores 0 o 1.



Bien; generemos tres variables; Sensor IZQ, Sensor CEN y Sensor DER y la igualaremos a su sensor correspondiente;

```

Bucle
  Establecer Sensor IZQ = Línea negra detectada Izquierda
  Establecer Sensor CEN = Línea negra detectada Centro
  Establecer Sensor DER = Línea negra detectada Derecha
  
```

Ahora vamos a hacer las lecturas de sus valores utilizando la consola serie;

```

Bucle
  Establecer Sensor IZQ = Línea negra detectada Izquierda
  Establecer Sensor CEN = Línea negra detectada Centro
  Establecer Sensor DER = Línea negra detectada Derecha
  Enviar crear texto con " S3 = " Sensor IZQ
  Esperar 1000 milisegundos
  Enviar crear texto con " S2 = " Sensor CEN
  Esperar 1000 milisegundos
  Enviar crear texto con " S1 = " Sensor DER
  Esperar 1000 milisegundos
  
```

Carga el programa y abre la consola serie y mira las lecturas moviendo el robot entre el blanco y el negro;



ArduinoBlocks :: Consola serie

Baudrate: 9600 Conectar Desconectar Limpiar

Enviar

```

S1 = 0
S3 = 1
S2 = 1
S1 = 1
S3 = 1
S2 = 1
S1 = 1
S3 = 0
S2 = 1
S1 = 1
S3 = 0
S2 = 0
S1 = 0
S3 = 1
S2 = 1
S1 = 0
S3 = 1
  
```

Blanco = 0

Negro = 1

ArduinoBlocks :: Consola serie

Baudrate: 9600 Conectar Desconectar Limpiar

Enviar



Comprueba las lecturas que obtienes en función del color del fondo.

*\*Podrías hacer el mismo programa directamente con los bloques de los sensores, sin necesidad de crear las variables Booleanas.*

Visto esto, vamos con la lógica de un robot seguidor de línea. Lo primero que debemos tener presente es el grosor de la línea. En el caso del **KeyBot** una línea negra de 2,5 cm de ancho es suficiente para que los tres sensores “entren” a la vez, es decir, que los tres sensores detecten la línea. Según los sensores vayan detectando blanco o negro nos encontraremos con los siguientes siete casos que vamos a ver;



Caso A: Robot dentro de línea.

Acción: Ir recto.



Caso B: Robot se va hacia la derecha.

Acción: Girar ligeramente a la izquierda.



Caso C: Robot se va hacia la izquierda.

Acción: Girar ligeramente a la derecha.



Caso D: Robot se va mucho a la derecha.

Acción: Girar a la izquierda.



Caso E: Robot se va mucho a la izquierda.

Acción: Girar a la derecha.



Caso F: Robot pierde la línea a su izquierda.

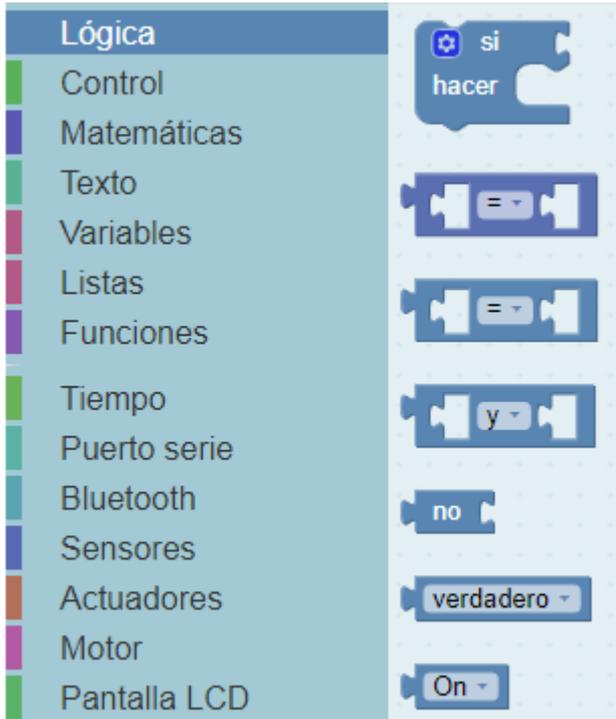
Acción: Continuar con la acción anterior.



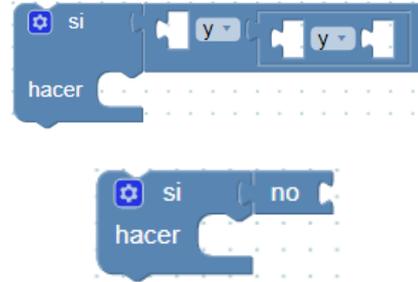
Caso G: Robot pierde la línea a su derecha.

Acción: Continuar con la acción anterior.

Comenzamos con la programación del seguidor de línea, recuerda que cuando el sensor detecta blanco su respuesta es un 0 y cuando detecta línea negra su respuesta es 1. En este primer ejemplo vamos a programar el robots con los primeros 5 casos (A, B, C, D y E) dejando los casos en los que no detecta nada para el siguiente ejemplo.

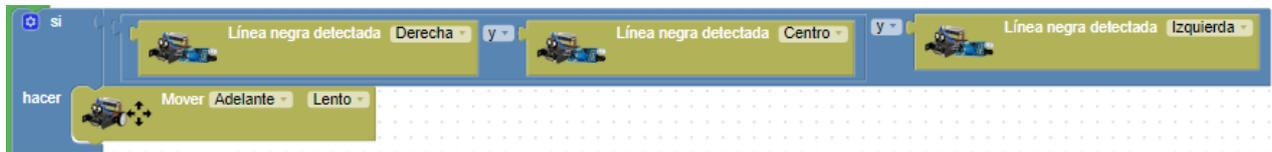


Para realizar esta programación necesitaremos varios de los bloques que aparecen en el menú **“Lógica”**. Las sentencias condicionales, el bloque de **“y”/“o”**, y el bloque de negación **“no”**. Ya que, al ser tres sensores, son varias condiciones las que se tienen que cumplir a la vez.

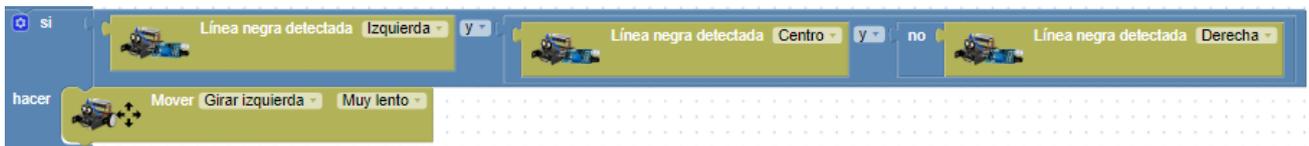


El bloque **“NO”** en la **“Lógica”** de **ArduinoBlocks** lo que hace es negar lo que se coloqe detrás de él.

.- En el **CASO A**, los tres sensores detectan negro y por lo cual el robot deberá ir recto. En este caso la parte del programa sería la siguiente;



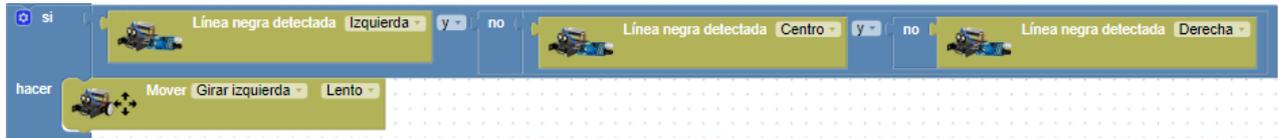
.- En el **CASO B**, S3 y S2 detectan negro y S1 blanco, el robot se sale ligeramente hacia la derecha por lo que deberá girar ligeramente a la izquierda. En este caso la parte del programa sería la siguiente;



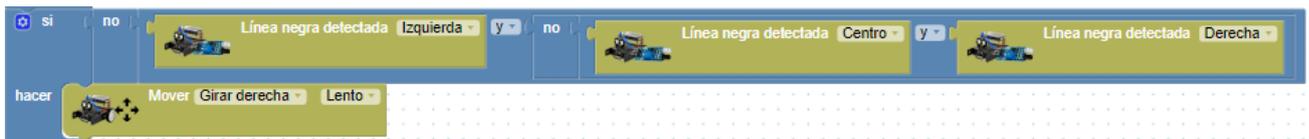
.- En el **CASO C**, S2 y S1 detectan negro y S3 blanco, el robot se sale ligeramente hacia la izquierda por lo que deberá girar ligeramente a la derecha. En este caso la parte del programa sería la siguiente;



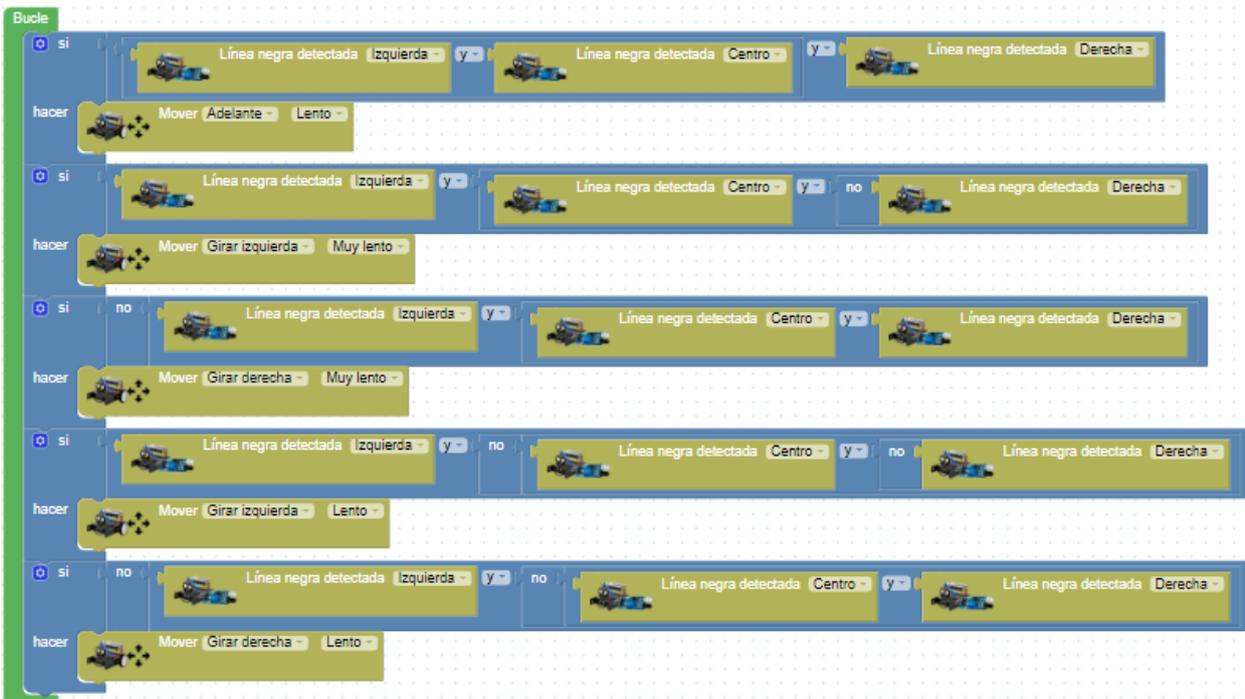
- En el **CASO D**, S3 detecta negro y S2 y S1 blanco, el robot se sale hacia la derecha por lo que deberá girar a la izquierda. En este caso la parte del programa sería la siguiente;



- Por último, en el **CASO E**, S1 detecta negro y S2 y S3 blanco, el robot se sale hacia la izquierda por lo que deberá girar a la derecha. En este caso la parte del programa sería la siguiente;



El programa completo sería el siguiente;



Fíjate que las velocidades están puestas como “Lento” y “Muy Lento”. En función de cada circuito y de como de cerradas sean las curvas prueba a cambiar esas velocidades para que que tu **KeyBot** sea como **UsainBot**.

Se podría simplificar el programa ya que cuando el robot se sale de la línea por un lado y el sensor del centro continua sobre la línea negra, obligatoriamente el sensor opuesto también estará sobre negro por lo que se podría quitar su condición. De esa manera el programa simplificado quedaría de así;

```

Bucle
├── si no Línea negra detectada Derecha y Línea negra detectada Centro y Línea negra detectada Izquierda
│   └── hacer Mover Adelante Lento
├── si no Línea negra detectada Centro y Línea negra detectada Derecha
│   └── hacer Mover Girar derecha Muy lento
├── si no Línea negra detectada Derecha
│   └── hacer Mover Girar derecha Lento
├── si no Línea negra detectada Centro y Línea negra detectada Izquierda
│   └── hacer Mover Girar izquierda Muy lento
└── si no Línea negra detectada Izquierda
    └── hacer Mover Girar izquierda Lento
    
```

Hasta ahora, sólo hemos analizado los casos en los que al menos un sensor tocaba la línea negra, ahora vamos a realizar el programa completo incluyendo los casos en los que ningún sensor detecta línea, es decir, todos están en fondo blanco.

Para ello lo que debemos hacer es que si se encuentran los tres sensores en línea blanca, el robot deberá ejecutar la última acción que estaba realizando en el último momento antes de perder la línea. Este último movimiento lógicamente será un giro, por lo que deberá girar en el último sentido que giraba justo antes de perderse. Para ello, cada vez que ejecute un movimiento, tiene que memorizar que lo ha ejecutado.

Vamos a utilizar variables de texto para memorizar cuál ha sido el último movimiento realizado y la llamaremos “ÚLTIMO ESTADO”. Pero pensemos un poco, realmente los últimos movimientos que nos interesan memorizar son dos, los dos en los cuales sólo un sensor detecta la línea negra. En ese caso vamos a programar lo siguiente;

```

si Línea negra detectada Derecha
hacer
  Mover Girar derecha Normal
  Establecer ULTIMO ESTADO = "GIRO DERECHA"
    
```

**Caso G**

```

si Línea negra detectada Izquierda
hacer
  Mover Girar izquierda Normal
  Establecer ULTIMO ESTADO = "GIRO IZQUIERDA"
    
```

**Caso F**

Por último; dentro de la condición de que los tres sensores no detecten negro introduciremos dos condicionales más para que cumplan los dos estados anteriormente citados y que realicen la acción que estaban ejecutando.

El programa completo quedaría de la siguiente manera;

Ya tienes listo un robot seguidor de línea. Ahora puedes controlar cada motor individualmente y ajustando la dirección y los PWM podrás realizar programas adaptados a cada circuito para sacar el máximo rendimiento a tu **KeyBot**. ¿Te atreves?...



## P2. – KeyBot explorador autónomo

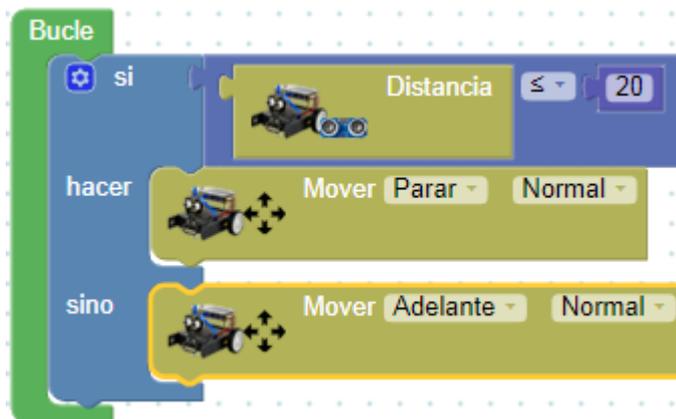
En este proyecto vamos a usar el sensor de ultrasonidos como “ojos” del **KeyBot** y haremos que funcione autónomamente explorando su entorno sin chocar. Para evitar el choque iremos haciendo varias estrategias, de la más simple a la más complicada.

Primero haremos un programa para que el robot siempre vaya hacia adelante y que cuando encuentre un objeto, a menos de 20cm, se pare. Y si el objeto desaparece debe volver a avanzar.

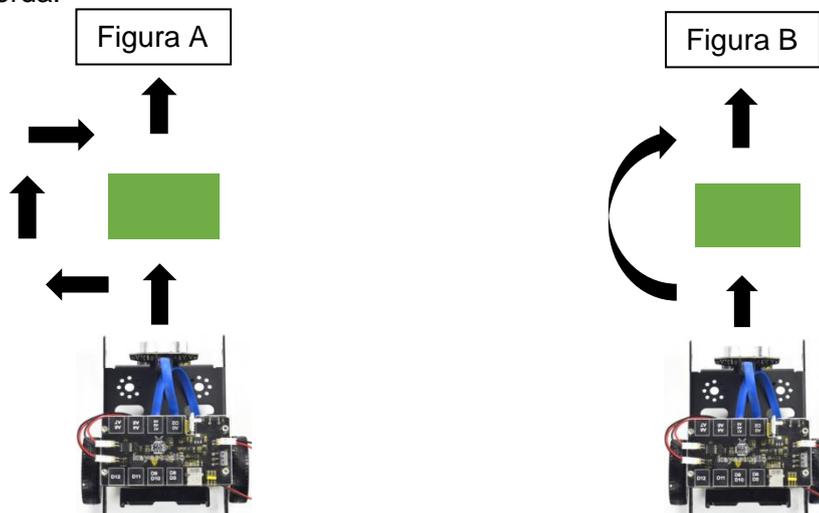
El bloque se encuentra en el apartado “*KeyBot*” de **ArduinoBlocks**:



El programa es muy sencillo, fíjate;



Continuamos ampliando el programa y en esta ocasión vamos a hacer que cuando **KeyBot** detecte un obstáculo, lo esquive y continúe realizando el mismo trayecto. Esquivar el obstáculo se puede resolver de dos maneras diferentes; la primera es realizando movimientos rectos con ángulo de 90° entre ellos (Figura A) y la segunda es realizando un movimiento circular alrededor del objeto (Figura B), en ambos casos el movimiento se podría realizar tanto por la derecha como por la izquierda.



Para resolver este proyecto crearemos una función que la llamaremos “ESQUIVAR” y en ella escribiremos todas las órdenes necesarias para esquivar el objeto.

```

Bucle
  si Distancia ≤ 20
  hacer ESQUIVAR
  sino Mover Adelante Normal
    
```

Los tiempos que tienen que estar ejecutándose cada acción dependen del robot, del tipo de suelo y el rozamiento, del estado de carga de las baterías, del tamaño del objeto a esquivar, de la velocidad del movimiento... es decir, esos tiempo varían, no van a ser los mismos para todos los robots, ni para todos los casos. Una forma de calcular esos tiempos es descomponer el programa en cada una de las acciones e ir calculando uno por uno esos tiempos. Por ejemplo;

```

Bucle
  Mover Rotar izquierda Muy lento
  Esperar 625 milisegundos
  Mover Parar Muy lento
  Esperar 2000 milisegundos
    
```

```

para ESQUIVAR
  Mover Rotar izquierda Muy lento
  Esperar 625 milisegundos
  Mover Adelante Muy lento
  Esperar 700 milisegundos
  Mover Rotar derecha Muy lento
  Esperar 625 milisegundos
  Mover Adelante Muy lento
  Esperar 1000 milisegundos
  Mover Rotar derecha Muy lento
  Esperar 625 milisegundos
  Mover Adelante Muy lento
  Esperar 700 milisegundos
  Mover Rotar izquierda Muy lento
  Esperar 625 milisegundos
    
```

Aquí, calculo el tiempo para que el robot realice una rotación de 90° hacia la izquierda. Cuando consiga el movimiento exacto guardo ese tiempo y calculo el tiempo del siguiente movimiento.

Ya hemos visto como realizar un seguidor de línea y un robot esquiva obstáculos con tu **KeyBot**, ¿Te atreves a combinar las dos funcionalidades?, Es decir, programar el robot para que siga una línea negra y si encuentra un obstáculo lo esquive y continúe siguiendo la línea negra. A por ello, ¡Ánimo!!!

## P3. – KeyBot Marcha Imperial “Star Wars”

Las notas musicales son, simplemente, sonidos con una frecuencia determinada. Por tanto, conociendo la frecuencia correspondiente a cada nota, y configurando los tiempos adecuados de cada una (junto con los silencios) podemos programar una melodía.

Frecuencia de las notas musicales en Hertzios;

		0	1	2	3	4	5	6	7	8
n=1	do		32,7	65,41	130,81	261,63	523,26	1046,5	2093	4186
n=2	do#		34,65	69,3	138,59	277,18	554,37	1108,7	2217,5	4434,9
n=3	re		36,71	73,42	146,83	293,66	587,33	1174,7	2349,3	4698,6
n=4	re#		38,89	77,78	155,56	311,13	622,25	1244,5	2489	4978
n=5	mi		41,2	82,41	164,81	329,63	659,26	1318,5	2637	5274
n=6	fa	21,826	43,65	87,31	174,61	349,23	698,46	1396,9	2793,8	5587,7
n=7	fa#	23,125	46,25	92,5	185	369,99	739,99	1480	2960	5919,9
n=8	sol	24,5	49	98	196	392	783,99	1568	3136	6271,9
n=9	sol#	25,96	51,91	103,83	207,65	415,3	830,61	1661,2	3322,4	
n=10	la	27,5	55	110	220	440	880	1760	3520	
n=11	la#	29,14	58,27	116,54	233,08	466	932,33	1864,7	3729,3	
n=12	si	30,87	61,74	123,47	246,94	493,88	987,77	1975,5	3951,1	

Estas son las notas de la “Marcha imperial de Star Wars”.

- Función *Primera parte*:

La	La	La	Fa	Do	La	Fa	Do	La
----	----	----	----	----	----	----	----	----

```

para Parte 1
  repetir 3 veces
    hacer
      Zumbador Ms 300 Tono 349.23
      Esperar 150 milisegundos
      Zumbador Ms 150 Tono 523.25
      Zumbador Ms 400 Tono 440
      Esperar 200 milisegundos
      Zumbador Ms 350 Tono 349.23
      Esperar 150 milisegundos
      Zumbador Ms 150 Tono 523.25
      Zumbador Ms 600 Tono 440
  
```

- Función Segunda parte:

Mi	Mi	Mi	Fa	Do	La	Fa	Do	La
----	----	----	----	----	----	----	----	----

Se ha realizado con un pulso de 600 ms.

```

para Parte 2
  repetir 3 veces
    hacer
      Zumbador Ms 400 Tono 659.26
      Esperar 200 milisegundos
      Zumbador Ms 300 Tono 698.46
      Esperar 150 milisegundos
      Zumbador Ms 150 Tono 523.25
      Zumbador Ms 400 Tono 415.3
      Esperar 200 milisegundos
      Zumbador Ms 350 Tono 349.23
      Esperar 150 milisegundos
      Zumbador Ms 150 Tono 523.25
      Zumbador Ms 800 Tono 440
  
```

Lo que vamos a hacer es que **KeyBot** desfile mientras esté tocando la “Marcha Imperial”;

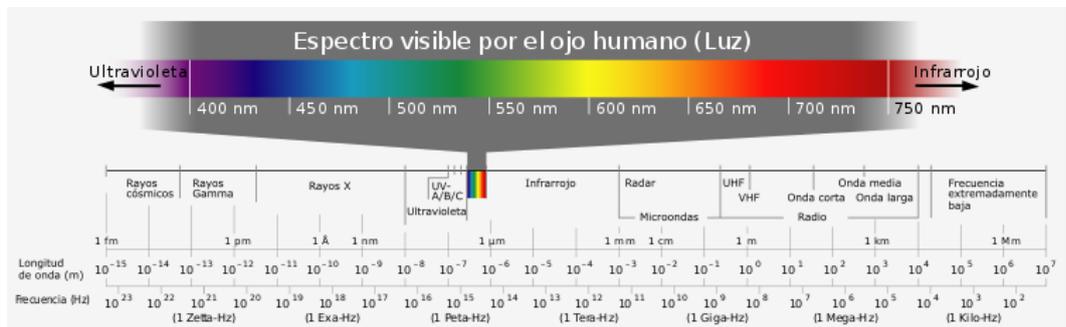
```

Bucle
  Mover Adelante Muy lento
  Parte 1
    Mover Rotar derecha Lento
    Esperar 800 milisegundos
    Mover Adelante Muy lento
  Parte 2
    Mover Rotar izquierda Lento
    Esperar 600 milisegundos
  
```

# P4. – KeyBot el músico de los colores

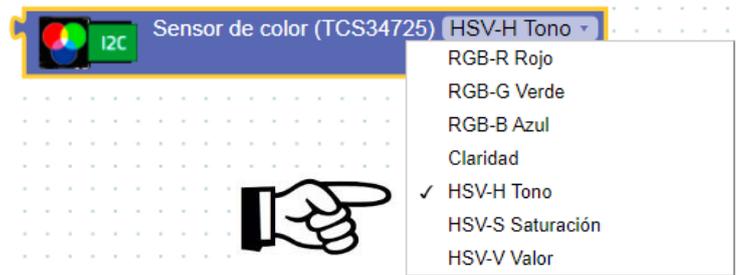
En la actividad en la que utilizamos el sensor de color trabajamos con el bloque correspondiente a los siete colores fijados por **Arduinoblocks**. Seguramente en algún momento hayáis tenido alguna mala lectura o no ha detectado correctamente el color, es normal. Según la [Wikipedia](#), “*El color es la impresión producida por un tono de luz en los órganos visuales, o más exactamente, es una percepción visual que se genera en el cerebro de los humanos y otros animales al interpretar las señales nerviosas que le envían los fotorreceptores en la retina del ojo, que a su vez interpretan y distinguen las distintas longitudes de onda que captan de la parte visible del espectro electromagnético*”. ¿Puedes ver algún color en una habitación con la luz apagada?, ¿Los colores de los paisajes son iguales durante todas las horas del día, al amanecer, al atardecer,...?

La respuesta es no. El color cambia según la luz ambiental, el material, la distancia a la que se aprecia,... Por consiguiente

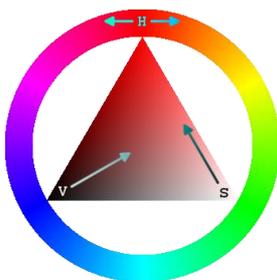


para tener el máximo de fiabilidad usando el sensor de color lo ideal es que lo calibremos en cada situación en la que lo vamos a usar. Por ejemplo, el **KeyBot** funciona perfectamente en un circuito en nuestra aula detectando colores, pero cuando lo sacamos a la calle “ya no funciona” o “hace cosas raras”. ¿Qué ha cambiado? La luz, pasamos de tener una luz artificial de fluorescentes o leds a una luz natural. Eso hará que la respuesta del sensor sea diferente.

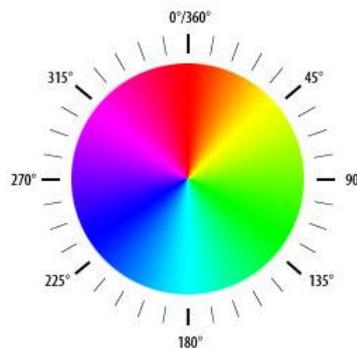
Para solucionar este problema tenemos el siguiente bloque; en el que podemos elegir colores RGB o HSV.



Utilizaremos el HSV-H Tono. El modelo HSV (del inglés Hue, Saturation, Value – Matiz, Saturación, Valor) define un modelo de color en términos de sus componentes como una rueda de color.



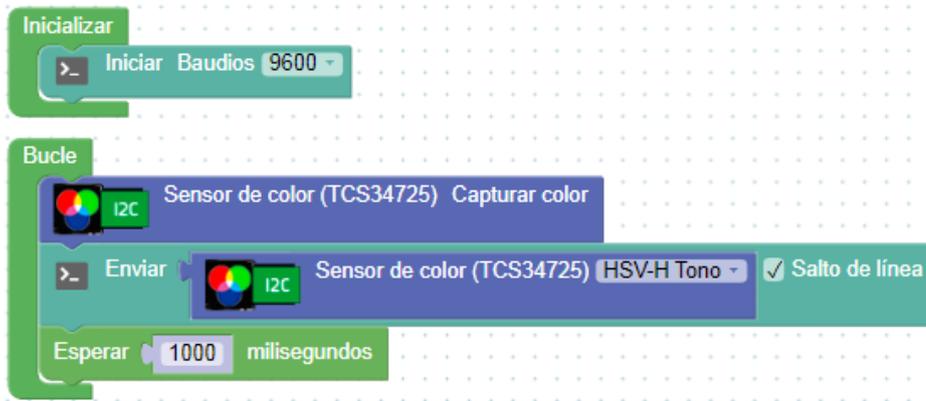
[Wikipedia](#)



En esta rueda de color los colores se definen en grados, por ejemplo el color verde oscila entre los 90° y 135°, el rojo entre los 350° y 10°, el azul entorno a los 225°,.... Utilizando este

modelo y de una forma muy sencilla podremos definir exactamente los colores que tenemos y los ángulos de cada uno de ellos.

Para ello vamos a realizar unas lecturas por el puerto serie con el siguiente programa:



Abrimos la consola serie y pasamos el sensor por diferentes colores, el resultado será similar al siguiente:

Las lecturas que hemos obtenido nosotros para cada color son las siguientes; (recuerda que esos valores serán similares, pero no iguales).

- .- ROJO:341-345
- .- NARANJA: 346-352
- .- AMARILLO:68-75
- .- VERDE: 112-118
- .- CIAN: 214-219
- .- AZUL:239-245
- .- VIOLETA:302-308

### ArduinoBlocks :: Consola serie

Baudrate:  Conectar Desconectar Limpiar

Enviar

---

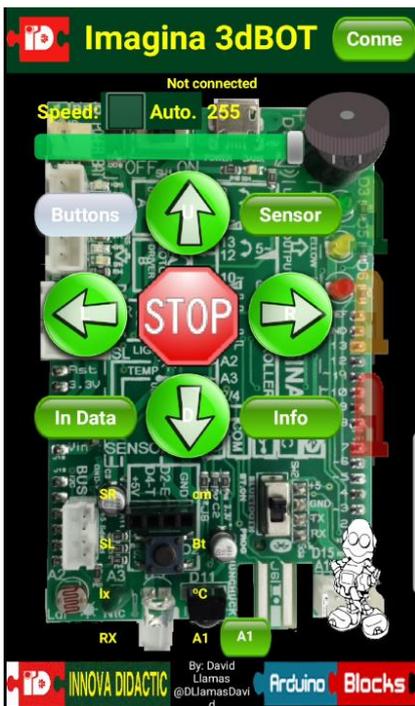
```

210.90
216.62
216.15
115.56
349.09
216.00
317.14
212.50
349.29
311.43
70.00
70.00
287.37
282.00
282.00
282.00
    
```

va a ,En este proyecto vamos a utilizar el sensor de color para hacer que **KeyBot** emita notas musicales según el color del fondo por donde circula.

## P5. – KeyBot controlado por bluetooth

### PENDIENTE APLICACIÓN



Para la gestión de las órdenes en el móvil, disponemos de la App “[Imagina 3dBot](#)” en Google Play, una aplicación hecha expresamente para nuestro robot.

Para decidir qué letras o números son los que hacen avanzar o girar, tenemos que consultar el apartado de información de la App. Teniendo en cuenta dicha información, vamos a hacer un programa que lea los datos recibidos por bluetooth y ejecute un movimiento en función de la letra recibida.

Además de los bloques ya conocidos, vamos a leer los datos procedentes del bluetooth con el bloque “Recibir byte”, también ubicado en el apartado “Bluetooth”.

Trabajar con lecturas en bytes nos obliga a leer los caracteres en código ASCII. Este código es una forma de codificar en números, los caracteres y símbolos del lenguaje. Tranquilos porque ArduinoBlocks dispone de un bloque para hacer la traducción inmediata, por lo que no tendremos que hacer ningún paso extra.



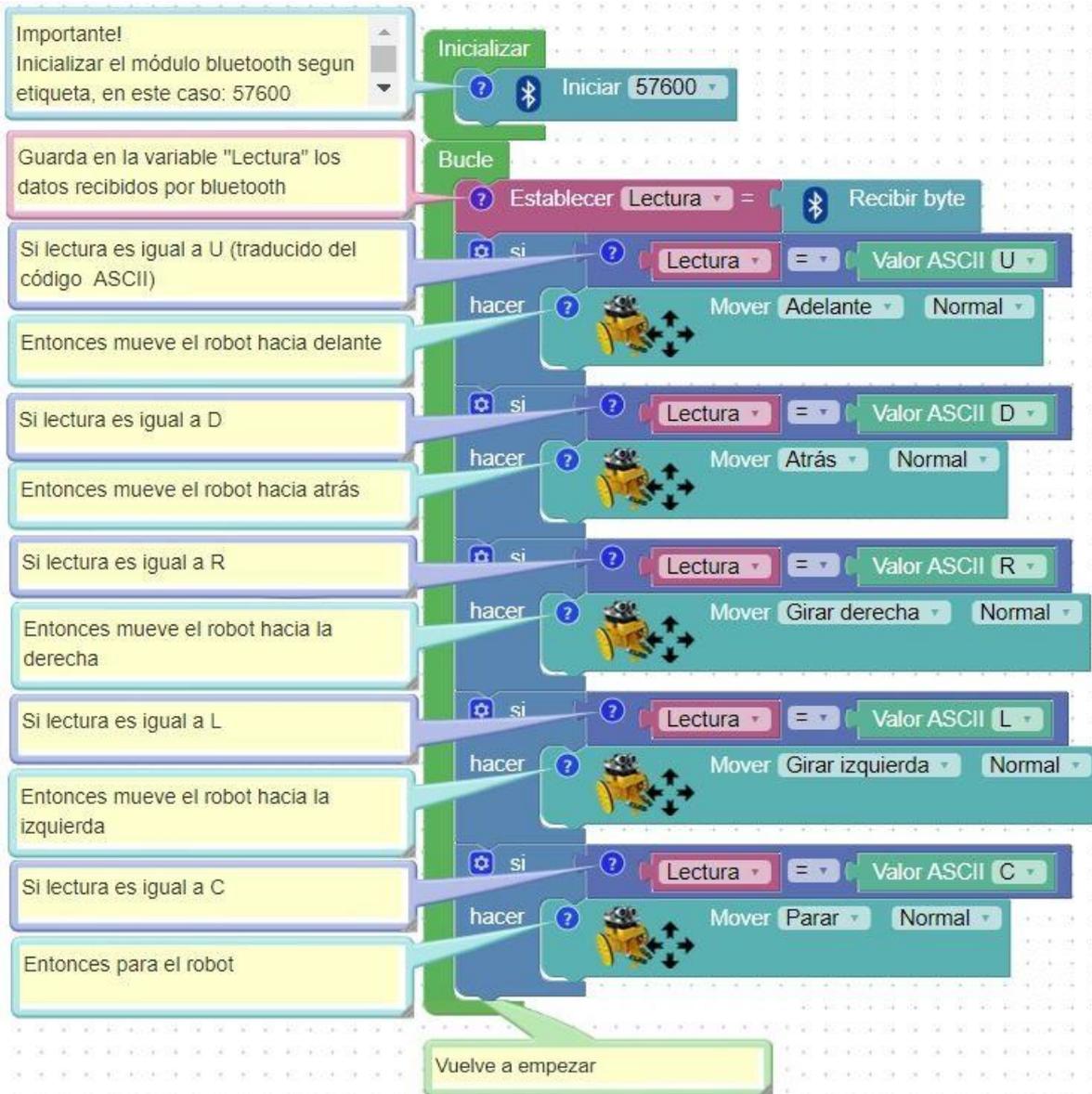
Si la App indica que cuando pulsas la flecha de movimiento hacia delante, envía por bluetooth una *U*, por ejemplo, nosotros tendremos que seleccionar una *U* en la programación en ArduinoBlocks, mediante el siguiente bloque:



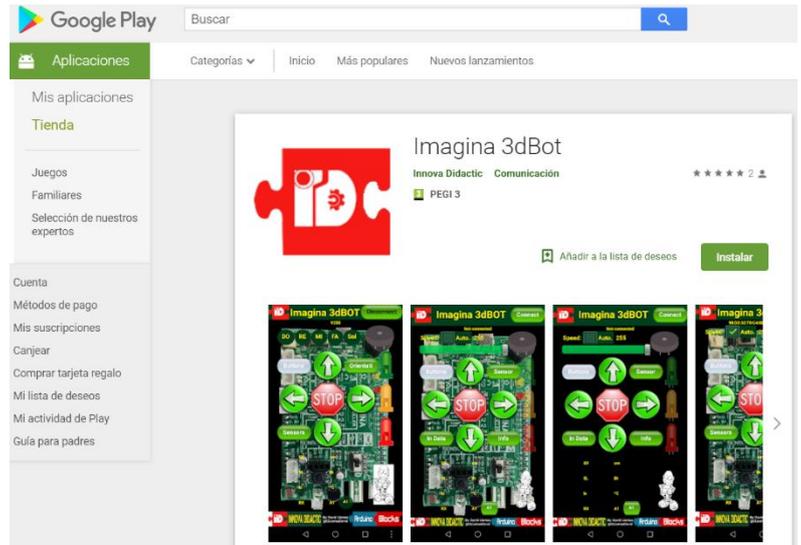
El bloque para traducir los datos leídos en bytes (*Valor ASCII*), se encuentra al final del menú “Texto”.

A16: Control del robot con el móvil.

Para la programación en ArduinoBlocks queda de la siguiente manera:

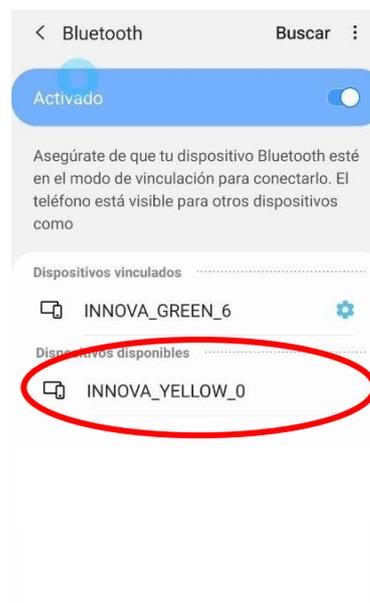
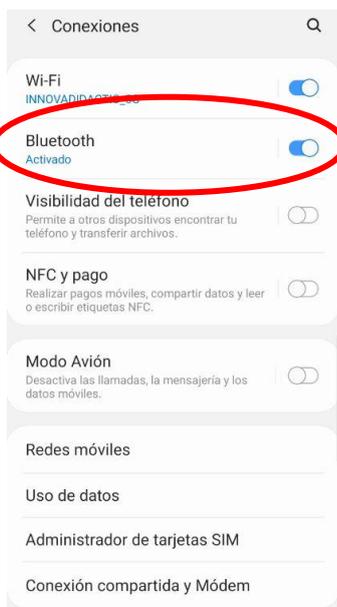


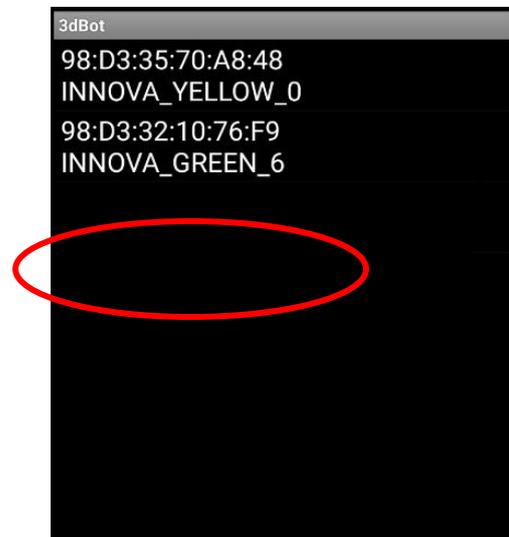
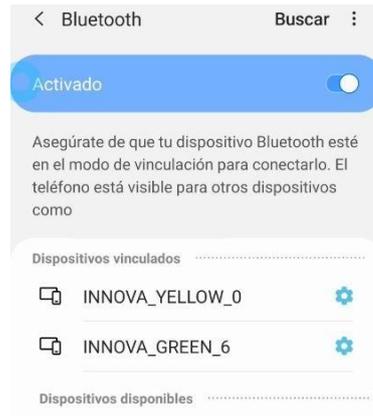
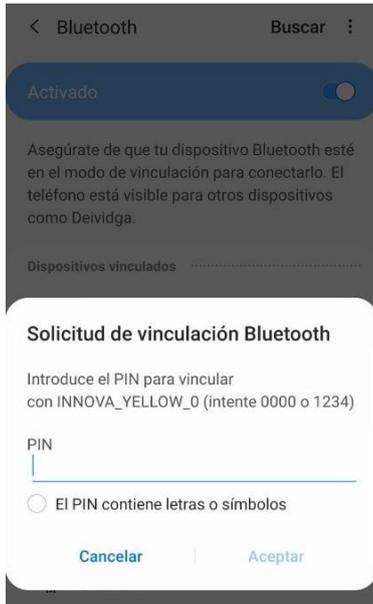
Y en el móvil instalaremos la App llamada [Imagina 3dBot](#) que nos



descargaremos de GooglePlay.

Para vincular el móvil con nuestro robot, activaremos la función bluetooth de nuestro teléfono. Cuando nos aparezcan los distintos “Dispositivos disponibles”, seleccionaremos nuestro robot según la etiqueta del módulo ubicado debajo de la placa, por ejemplo, INNOVA\_YELLOW\_0 e introducimos la contraseña 1234. Una vez aceptada ya debería aparecer el robot en “Dispositivos vinculados”.





Ahora repetimos la operación de conexión en la App Imagina 3dBot, solo se debe pulsar sobre el botón "Connect" situado al lado del título de la aplicación, y volver a seleccionar el nombre de nuestro robot, "INNOVA\_YELLOW\_0":

¡Hasta aquí el tutorial para el robot **KeyBot**!

Lo que hemos visto es solo una pincelada del mundo Arduino. Hay infinidad de proyectos para crear con la gran cantidad de sensores y actuadores que existen. Consulta los Kits para primaria, secundaria y profesional disponibles en [Innova Didactic](#).

[Material Arduino](#)