


# PicoBlockly

- [Cuatro programas a elegir](#)
- [Interface](#)
- [Conexión](#)
- [Dos formas de ejecutar los programas](#)
- [PROYECTO BLINK](#)
- [PROYECTO ACTION-REACTION](#)
- [PROYECTO Autonomous Lighting](#)
- [PROYECTO Thermometer](#)
- [PROYECTO Graphic Monitor](#)
- [PROYECTO Dominate the Rhythm](#)
- [PROYECTO Show Your Reaction](#)
- [PROYECTO My Timer](#)
- [PROYECTO Alarm Clock](#)
- [PROYECTO Know Your Color](#)
- [PROYECTO BUZZ WIRE GAME](#)
- [Algo diferente PROYECTO IR](#)
- [PICO COCHE](#)
- [Mapeo](#)
- [Servo](#)
- [Relé](#)
- [Sensor de distancia de ultrasonidos](#)

# Cuatro programas a elegir

Si entramos en <http://rbt.ist/ide> podemos ver cuatro opciones



**PICOJR**  
A simple and visual-based coding platform for kids who are new to robotic coding.

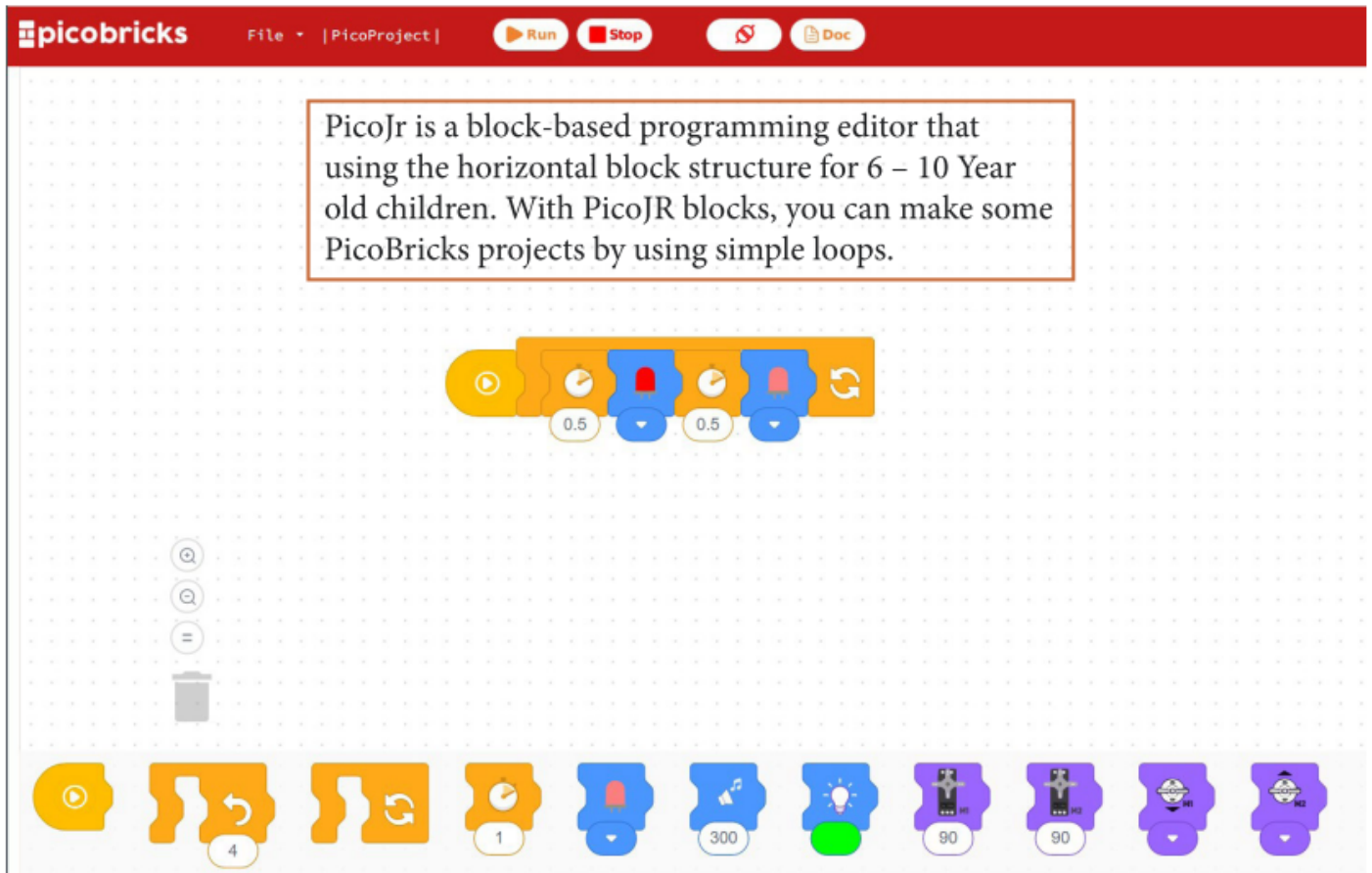
**PICOBLOCKLY**  
Easily code PicoBricks with block coding and see python code at the same time.

**PICOPY**  
Do high-level coding with text-based python.

**SIMULATOR**  
With PicoBricks Simulator you can experience using PicoBricks.

## PICOJR

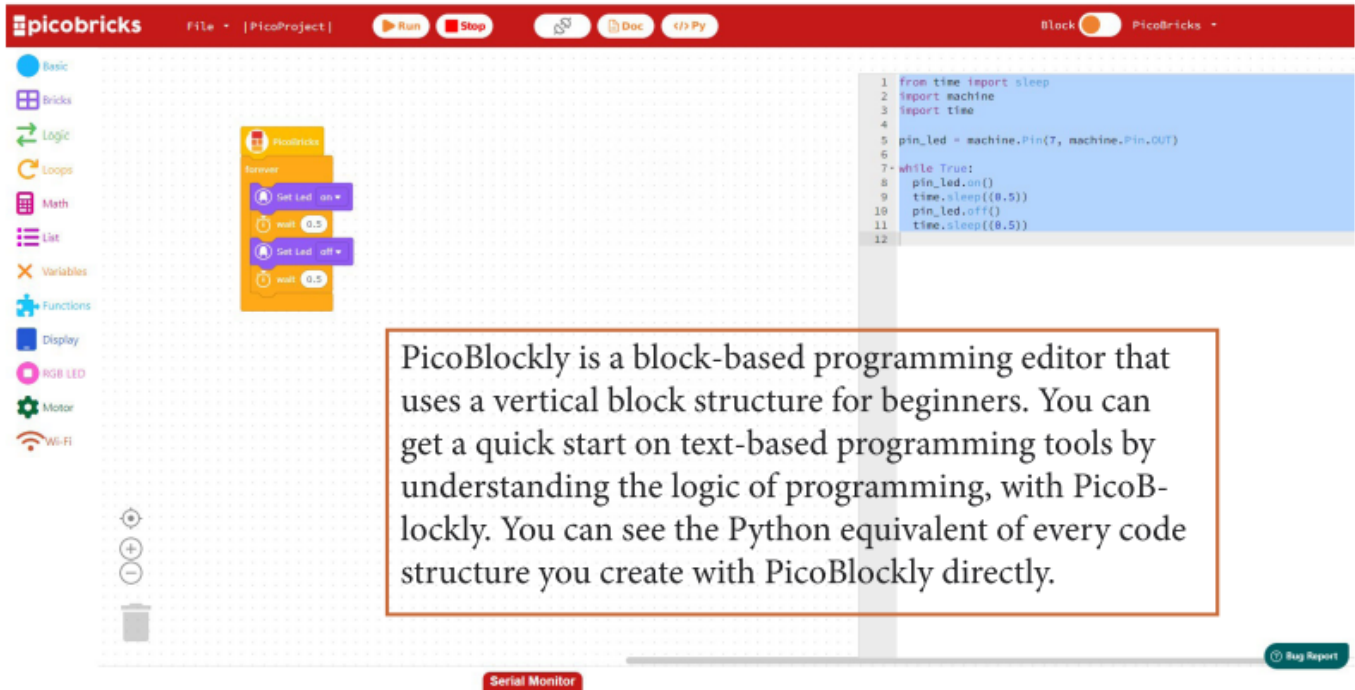
Pensado para programar Picobriks con bloques para etapas de 8 a 10 años con un **mínimo de instrucciones**



Fuente Pico Bricks IDE Book CC-BY-SA <https://picobricks.com/pages/idebook> ver [créditos](#)

## PicoBlockly

Es la opción más recomendada para la mayoría de las etapas



PicoBlockly is a block-based programming editor that uses a vertical block structure for beginners. You can get a quick start on text-based programming tools by understanding the logic of programming, with PicoBlockly. You can see the Python equivalent of every code structure you create with PicoBlockly directly.

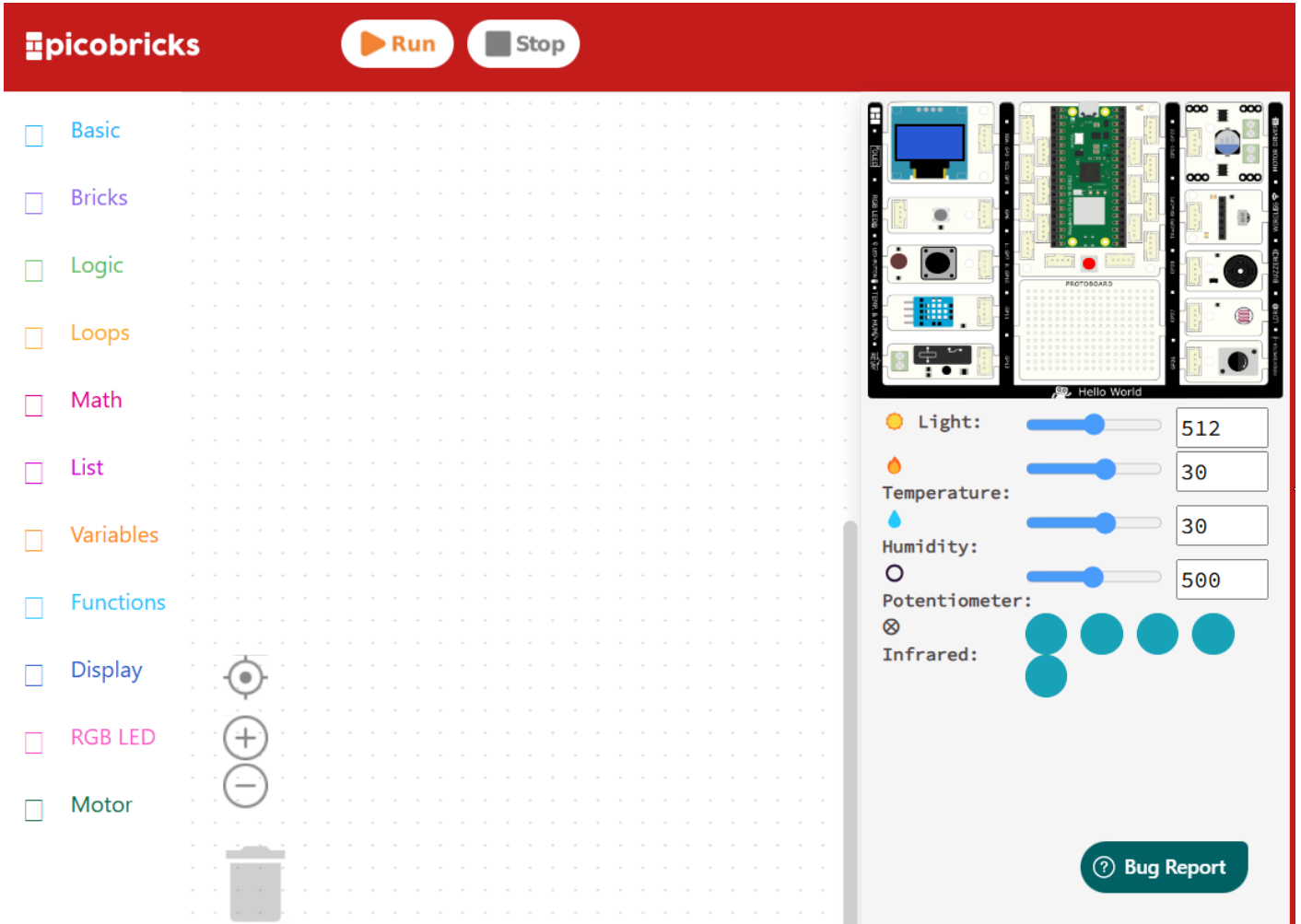
Fuente Pico Bricks IDE Book CC-BY-SA <https://picobricks.com/pages/idebook> ver [créditos](#)

## PicoPy

Para poder editar en Python, no lo intentes por aquí, no va. Para trabajar con Python se trabaja con Thommy que lo trataremos en este curso.

## Pico simulator

Es un [simulador online](#) que permite realizar proyectos sin tener físicamente la Pico bricks

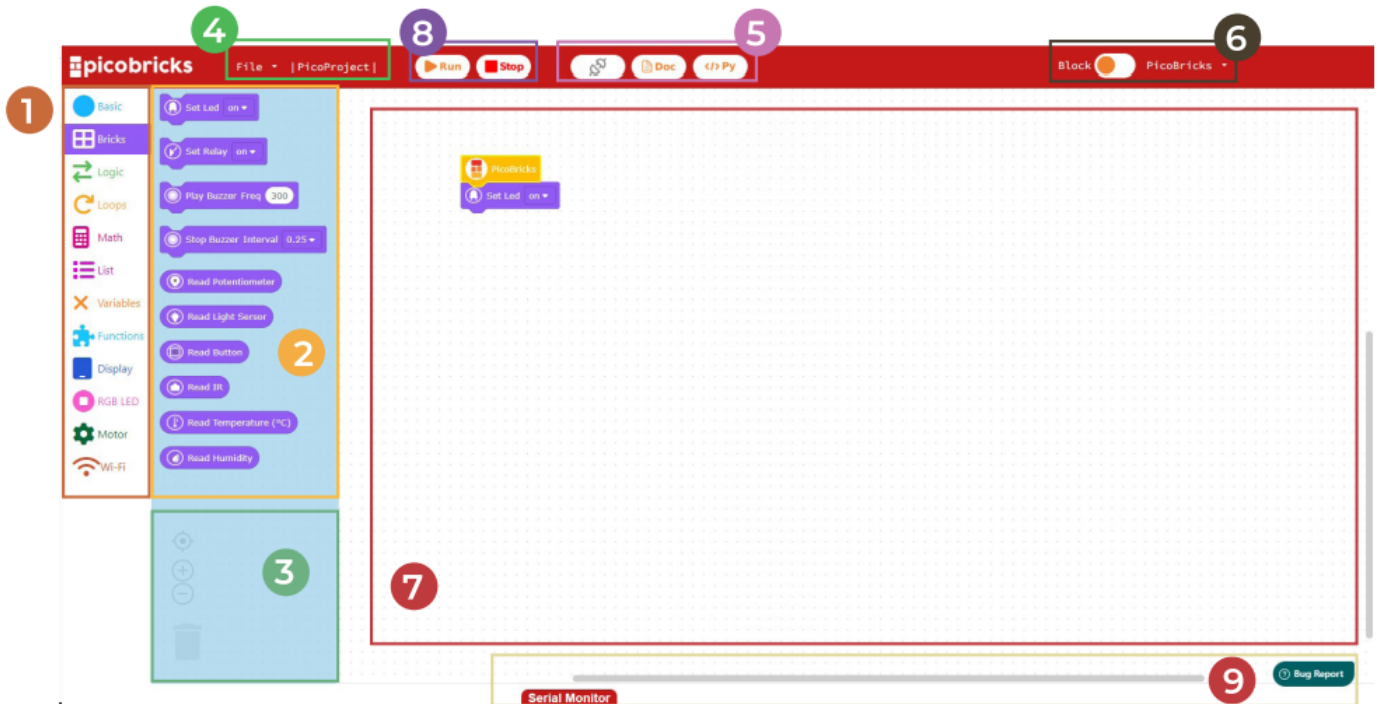


The screenshot shows the Picobricks simulator interface. At the top, there is a red header with the 'picobricks' logo and 'Run' and 'Stop' buttons. On the left, a vertical list of components is shown with checkboxes: Basic, Bricks, Logic, Loops, Math, List, Variables, Functions, Display, RGB LED, and Motor. Below this list are icons for a camera, a plus sign, a minus sign, and a trash can. The main workspace is a grid where a virtual breadboard is placed. The breadboard contains a microcontroller, a display, and various sensors. To the right of the breadboard, a control panel displays real-time data: Light (512), Temperature (30), Humidity (30), Potentiometer (500), and Infrared (represented by four blue circles). A 'Bug Report' button is located at the bottom right of the control panel.

Ojo el simulador no permite gestión de ficheros, es decir, no puedes ni grabar proyectos ni abrirlos, cuando cierras el navegador se pierde todo

# Interface

Cuando abrimos Picoblockly tenemos la siguiente ventana:



Fuente Pico Bricks IDE Book CC-BY-SA <https://picobricks.com/pages/idebook> ver [créditos](#)

1. Donde encontramos las diferentes instrucciones ordenadas por categorías
2. La paleta de instrucciones preparados para elegir y arrastrar a 7
3. Herramienta de zoom, borrar
4. Menú de fichero para grabar los proyectos o abrirlos (todo localmente)
5. Panel operaciones
  1. Botón de conectar, por cable (recomendado) o bluetooth
  2. Botón de proyectos ya preinstalados
  3. Vista de código Python (también en 6 hay una pestaña para pasar a esta vista)
6. Menú de configuración para descargar los firmwares necesarios para la conexión
7. Área donde programamos
8. Start stop tu programa
9. Área del puerto serie donde podemos ver los valores que deseamos

# Conexión

Lo primero que tenemos que hacer es poner el firmware para podernos conectar con Picobriks

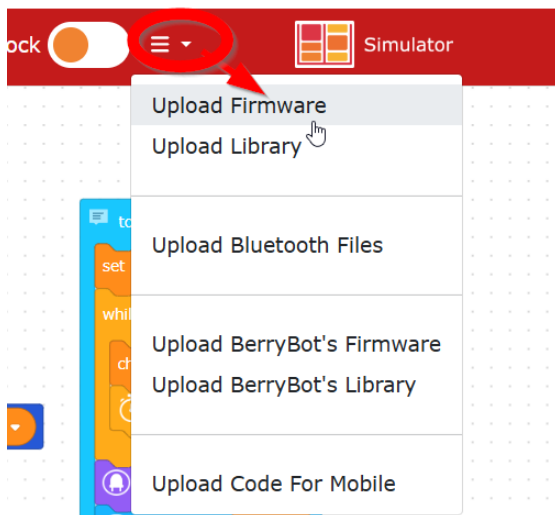
**P: ¿Qué es eso de "firmware"?**

R: No es más que un software que se graba en los chips de la placa.

**P ¿Y por qué se llama así, y no se llama software o programa y en paz?**

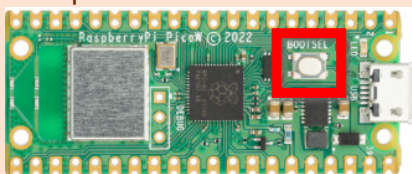
R: Digamos que como se graba en los chips, es un medio camino entre software y hardware, para diferenciarlo del software habitual.

Entramos en el menú y descargamos el firmware

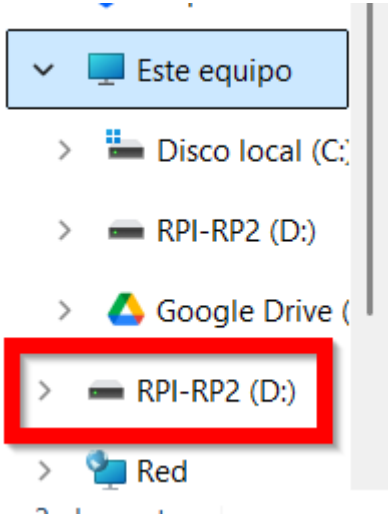


**ATENCIÓN, poner PicoBricks en modo Bootloader**

- 1.-Desconectamos PicoBricks de nuestro ordenador
- 2.- Apretamos el botón BOOTSEL **mientras** lo volvemos a conectar al puerto USB



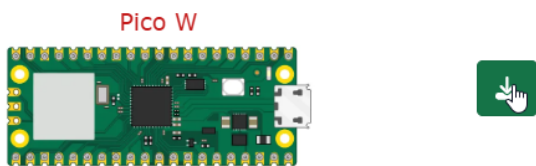
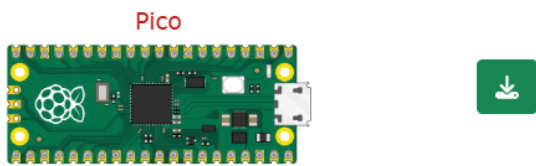
- 3.- Automáticamente aparecerá una nueva unidad de disco en nuestro ordenador (ya puedes soltar BOOTSEL)

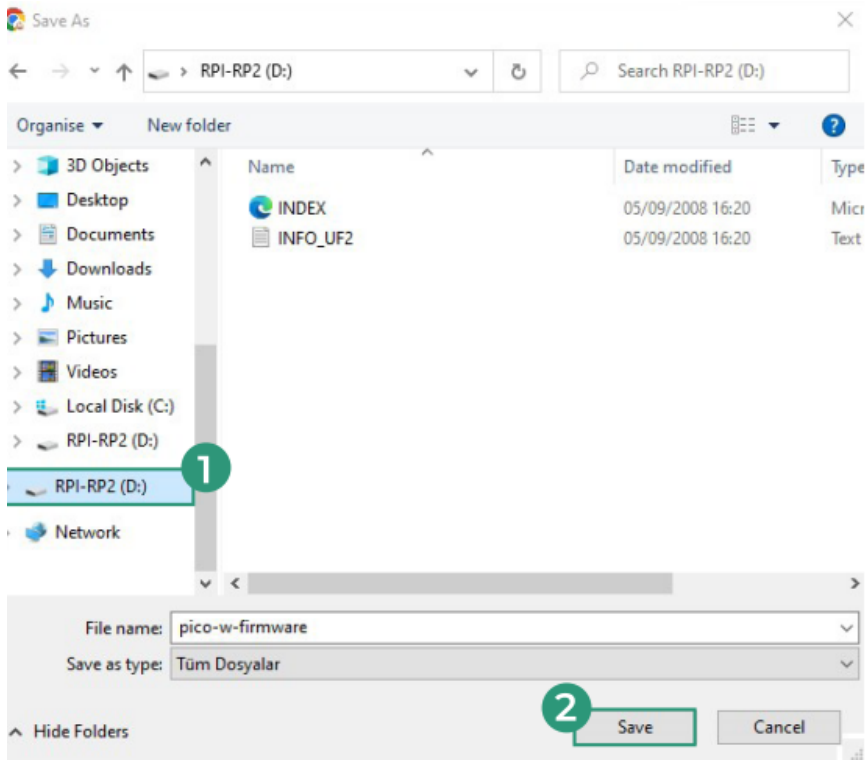


Descargamos el correspondiente al **PicoW** Y LO GRABAMOS EN LA UNIDAD NUEVA en mi caso RPI-RP2 (D:)

### FIRMWARE UPDATE ×

to firmware update hold down the **BOOTSEL** button while plugging the board into USB

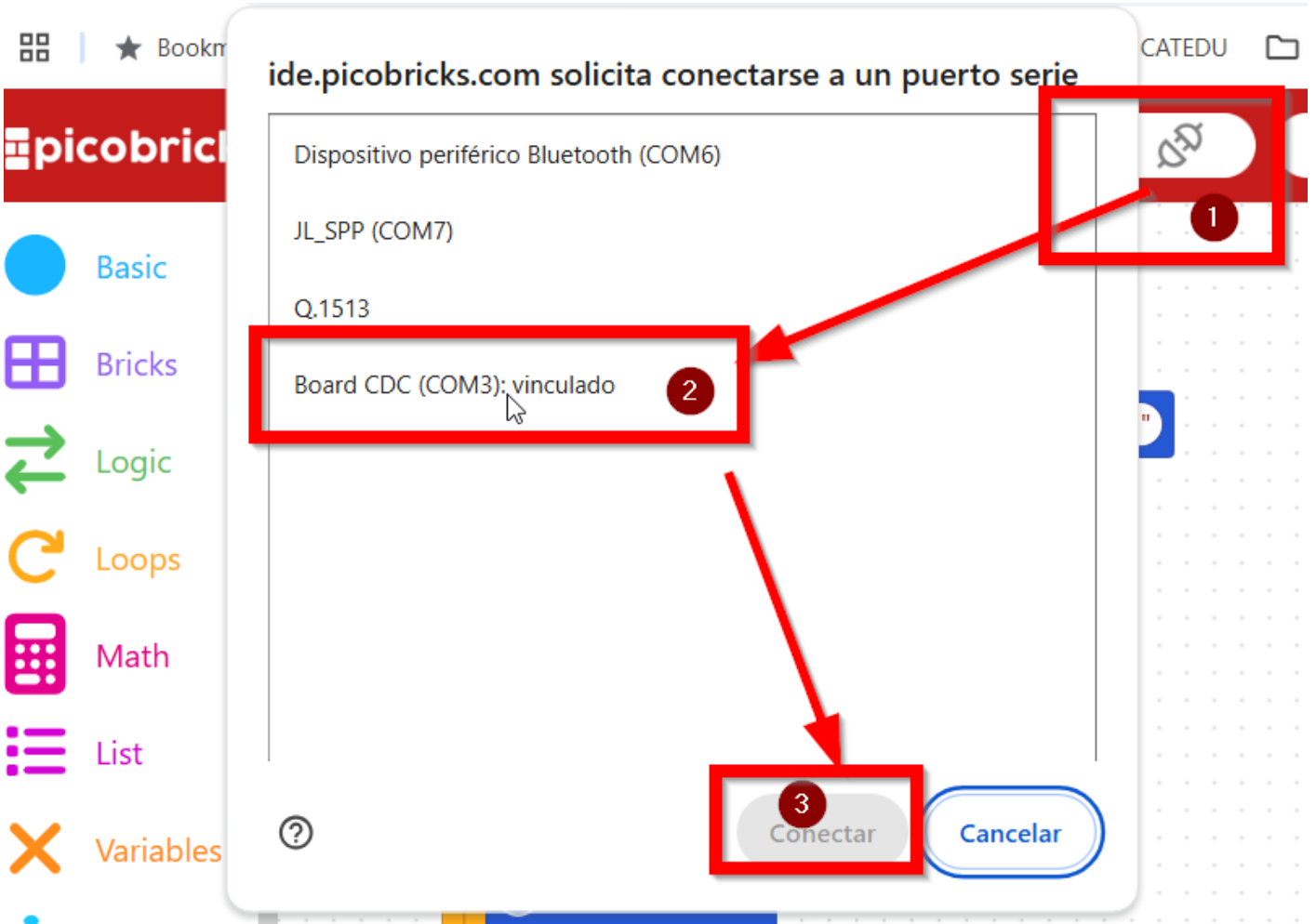




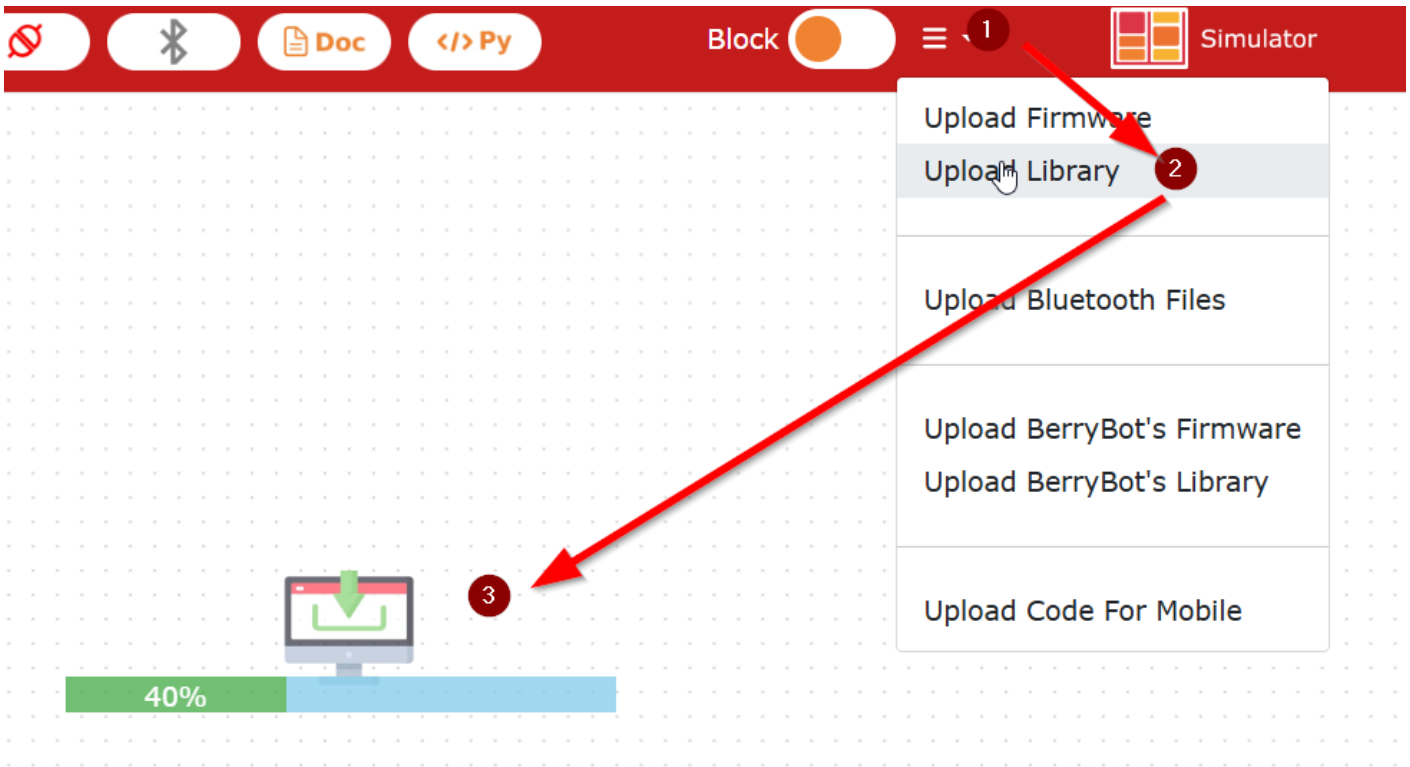
Fuente Pico Bricks IDE Book CC-BY-SA <https://picobricks.com/pages/idebook> ver [créditos](#)

Una vez grabado el firmware, esperamos a que nos salga un mensaje: *Please conect to the board*

Entonces dar a conectar y seleccionar la placa



Una vez conectado, descargamos las librerías en el PicoBricks para poder usar todas las funciones



YA ESTA, esto lo tienes que hacer **SOLO UNA VEZ** mientras uses PicoBloxly, si te pasas a otro programa y te cargas su firmware, tendrás que volverlo a poner.

# Dos formas de ejecutar los programas

Picobriks permite dos formas de trabajar:

[https://docs.google.com/presentation/d/e/2PACX-1vQb1Dv9wN9QK-F6V7yvwDoyzquqwWIGvlyVJr83Yk56kAoYD7bXLnYDm\\_tCQkeAgg/pubembed?start=false&loop=false&delayms=3000](https://docs.google.com/presentation/d/e/2PACX-1vQb1Dv9wN9QK-F6V7yvwDoyzquqwWIGvlyVJr83Yk56kAoYD7bXLnYDm_tCQkeAgg/pubembed?start=false&loop=false&delayms=3000)

La forma más fácil de trabajar es **EN VIVO** es decir, que los programas se ejecuten **desde nuestro ordenador** es la más rápida y para ello necesita que el PicoBricks tenga el Firmware correspondiente dentro (tal y como hemos visto)

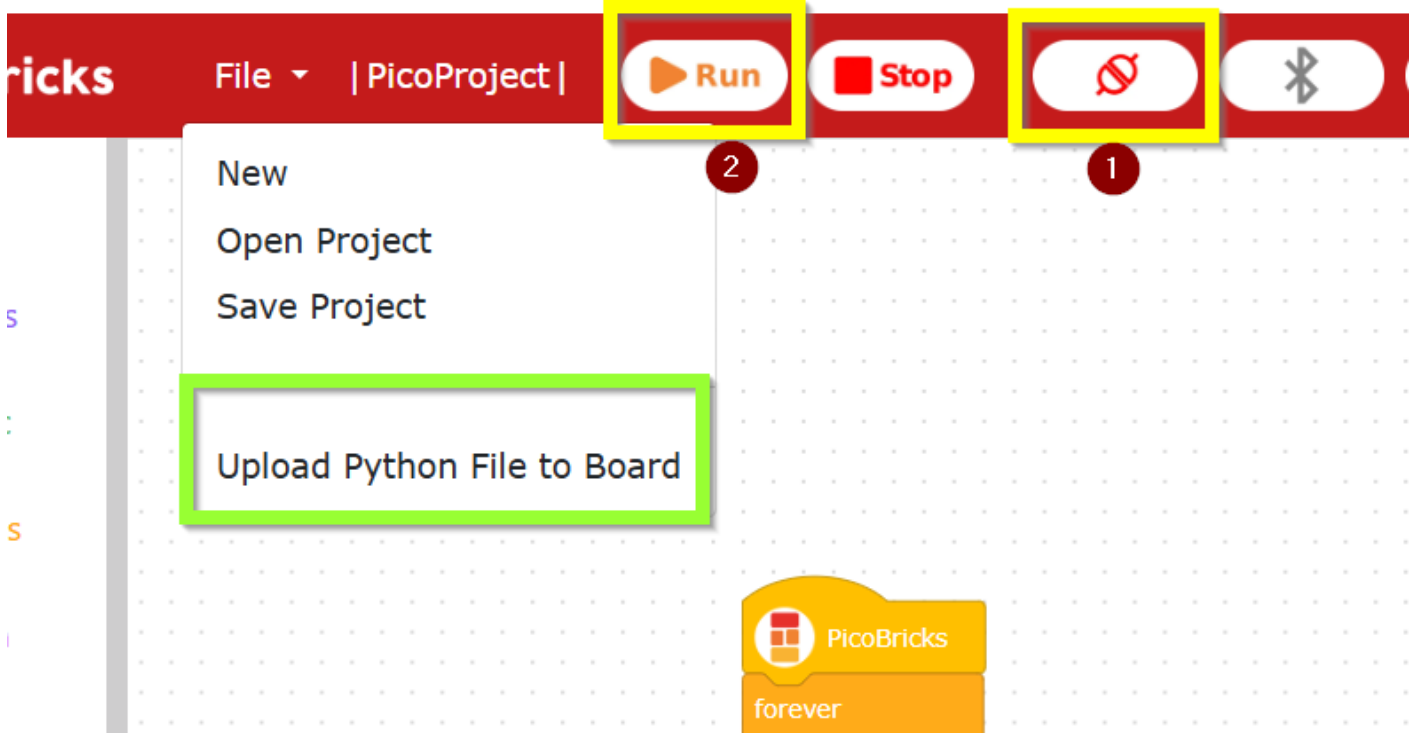
La otra forma de trabajar es **EN CARGA** es decir que los programas se ejecutan **desde dentro de PicoBricks** tiene la ventaja que el programa funciona si necesidad de ordenador. Eso sí, hay que alimentar Picobriks por el cable USB (usando un Powerbank o un cargador de móvil por ejemplo)

**ATENCIÓN** si trabajamos EN CARGA nos "cargamos" el Firmware, por lo que si queremos volver a trabajar EN VIVO tenemos que volverlo a poner tal y como hemos visto

Recomendamos EN VIVO por la rapidez y sencillez. Sólo es aconsejable EN CARGA cuando sean proyectos que precisen que el ordenador no esté. Por ejemplo en el PICO COCHE

<https://libros.catedu.es/books/pico-bricks/page/pico-coche>

- Para trabajar **EN VIVO** tenemos que estar **conectados** (1) y darle al **Run** (2) (recuadros amarillos)
- Para trabajar **EN CARGA** entramos en archivo y cargamos el programa dentro de Picobriks (recuadro verde) **Upload Python File to Board**



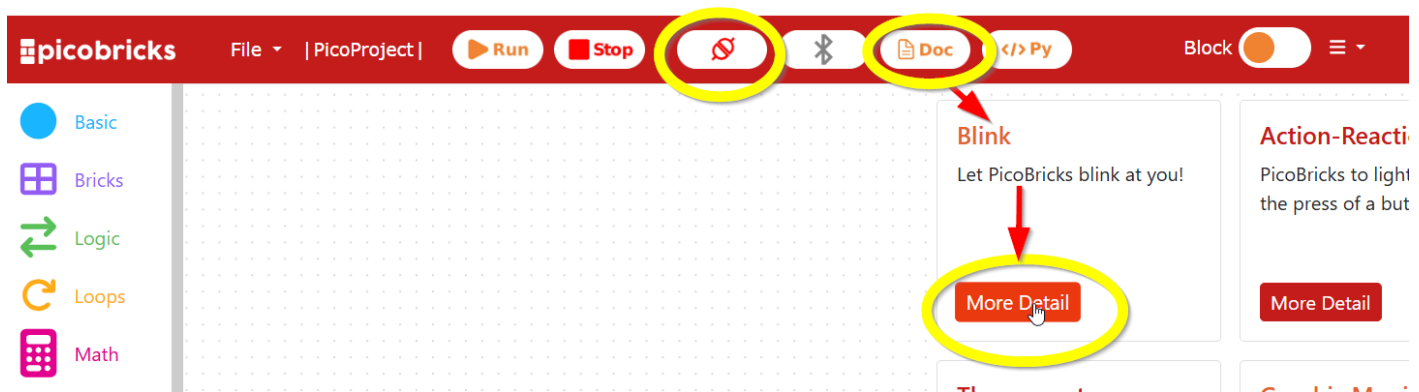
The image shows the PicoBricks IDE interface. At the top, there is a red header bar with the text "PicoBricks" on the left and "File | PicoProject |" in the center. To the right of the header are four buttons: "Run" (with a play icon), "Stop" (with a red square icon), a button with a red circle and slash icon, and a Bluetooth icon. The "Run" and the red circle/slash button are highlighted with yellow boxes. Below the "Run" button is a small red circle with the number "2", and below the red circle/slash button is a small red circle with the number "1". On the left side, there is a vertical toolbar with icons for "New", "Open Project", "Save Project", and "Upload Python File to Board". The "Upload Python File to Board" icon is highlighted with a green box. In the center of the workspace, there is a code block with the text "PicoBricks" and "forever" below it.

# PROYECTO BLINK

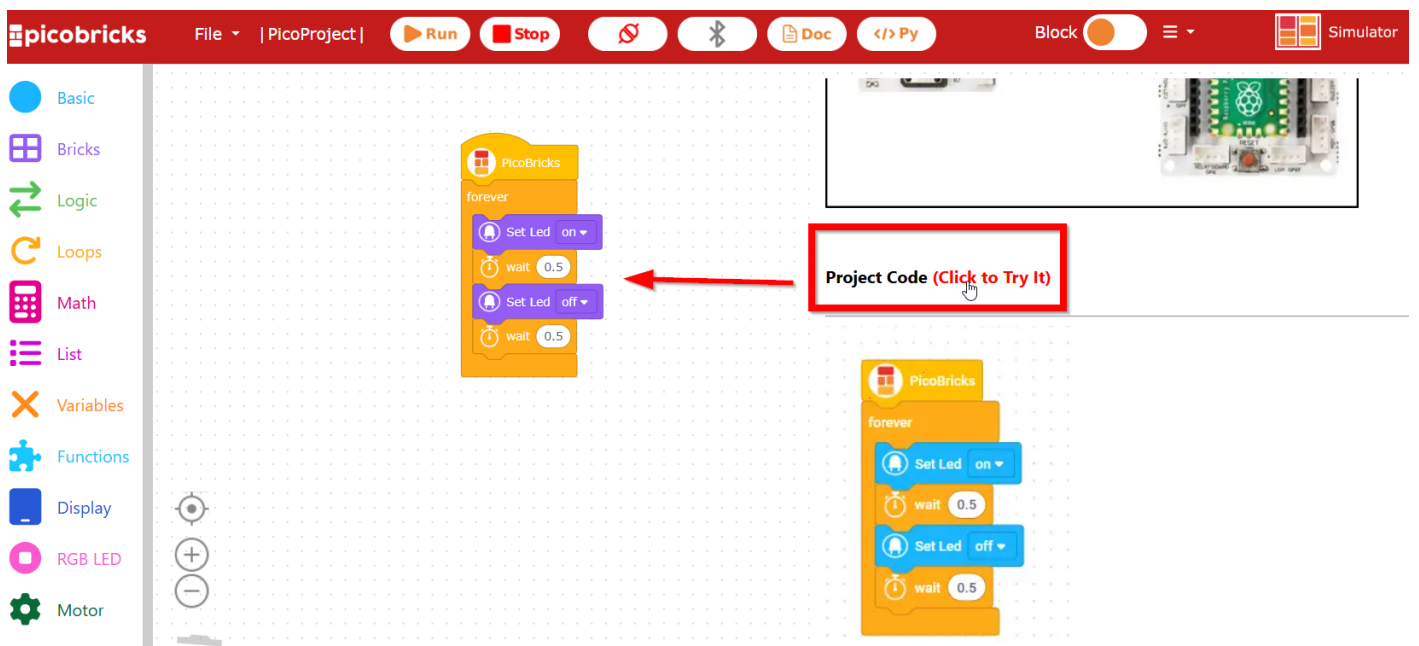
Extraído de *Pico Bricks IDE Book CC-BY-SA* <https://picobricks.com/pages/idebook> ver [créditos](#)

Vamos a realizar nuestro primer proyecto, parpadear el led rojo

Como es un programa predeterminado, lo más cómodo es ir los tutoriales que lo explican bien



Vamos al código y si apretamos en este botón, nos aparece en nuestro panel **si necesidad de hacerlo** pero ojo que a veces está escondido tras la ventana, usar el zoom y navegar



al dar a **RUN** tenemos



[https://www.youtube.com/embed/nYPWAC\\_SHWA](https://www.youtube.com/embed/nYPWAC_SHWA)

# PROYECTO ACTION-REACTION

Extraído de *Pico Bricks IDE Book* CC-BY-SA <https://picobricks.com/pages/idebook> ver [créditos](#)

Ahora será con la interacción del botón. Repetimos los pasos pero con este proyecto:

The screenshot displays the PicoBricks IDE interface. At the top, there is a red menu bar with options: File, PicoProject, Run, Stop, Doc, and </> Py. Below the menu bar, the main workspace shows a code block for a PicoBricks project. The code block starts with a 'forever' loop. Inside the loop, there is an 'if' statement: 'if Read Button = 1 do Set Led on else Set Led off'. The right sidebar contains several project cards. The 'Action-Reaction' card is highlighted with a red border and has a 'More Detail' button. Other cards include 'Blink', 'Thermometer', and 'Graphic Monitor'.

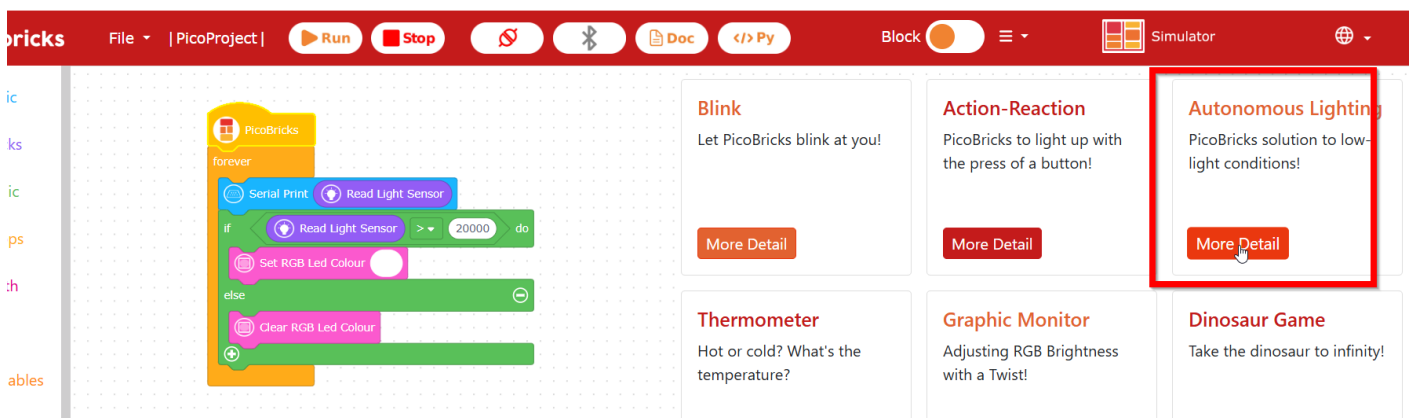
Resultado

<https://www.youtube.com/embed/6cZ-Hk3dnj8>

# PROYECTO Autonomous Lighting

Extraído de *Pico Bricks IDE Book* CC-BY-SA <https://picobricks.com/pages/idebook> ver [créditos](#)

Lo mismo con el siguiente proyecto



Resultado

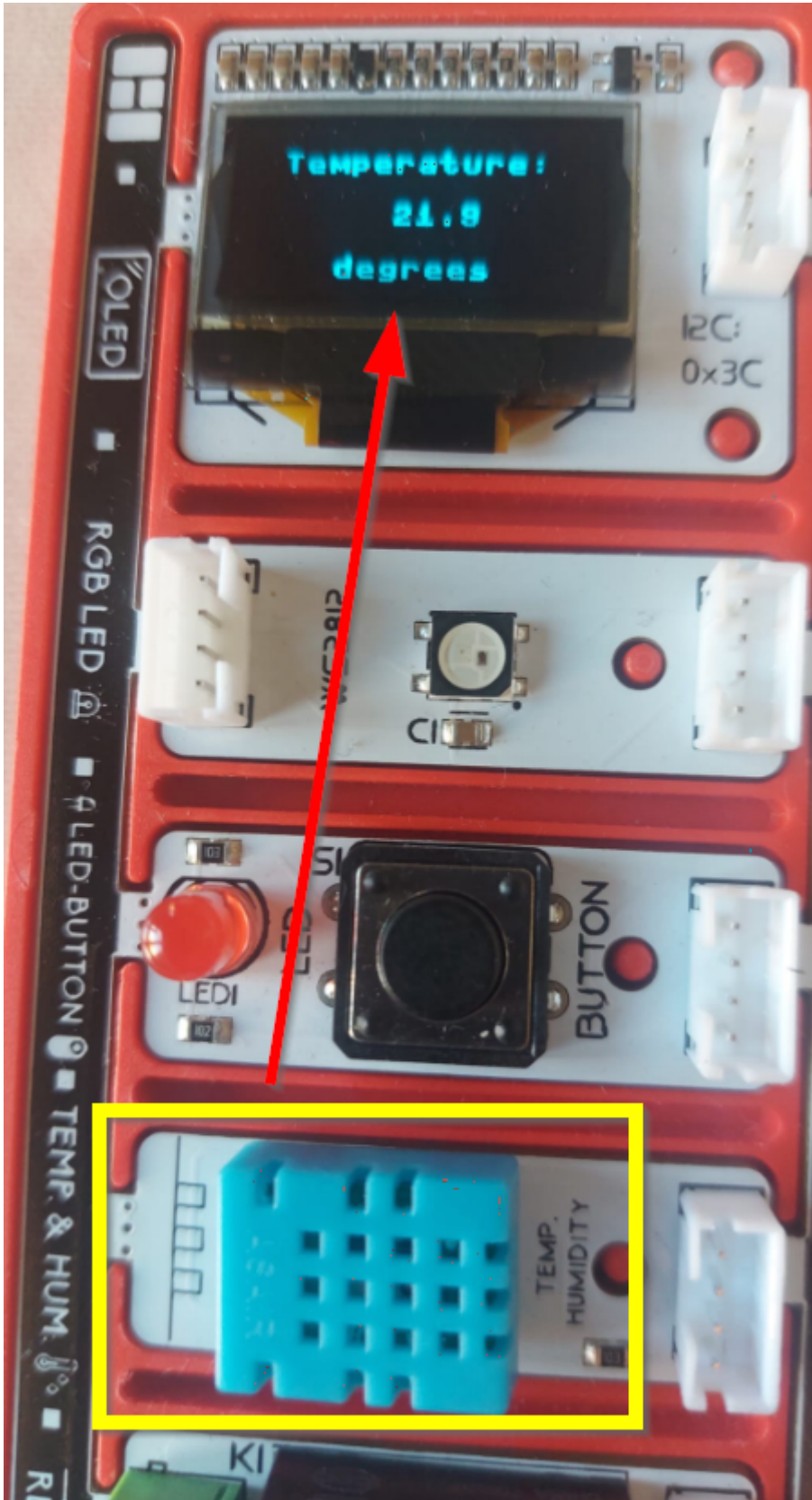
<https://www.youtube.com/embed/sN8Y3boBPAg>

# PROYECTO Thermometer

Extraído de *Pico Bricks IDE Book* CC-BY-SA <https://picobricks.com/pages/idebook> ver [créditos](#)

The screenshot shows the PicoBricks IDE interface. At the top, there is a red toolbar with buttons for 'File', 'Run', 'Stop', 'Debug', 'Bluetooth', 'Doc', and 'Py'. Below the toolbar is a workspace with a grid background. On the left, a 'PicoBricks' block is connected to a 'forever' loop. Inside the loop, the following blocks are stacked: 'wait 1', 'Clear Screen Buffer', 'Write Text to Screen X 15 Y 10 "Temperature: "', 'Write Text to Screen X 55 Y 30 Read Temperature (°C)', 'Write Text to Screen X 35 Y 50 "degrees"', and 'Show Screen Buffer'. On the right, there are two blocks: 'Blink' and 'Thermometer'. The 'Thermometer' block is highlighted with a red border and contains the text 'Hot or cold? What's the temperature?' and a 'More Detail' button.

Si soplamos el aliento sobre el sensor podemos ver como sube la temperatura



Recomendamos este proyecto cargarlo en el PicoBricks y así funciona autónomo sin necesidad de PC, con lo que se puede colocar en el exterior y ver la temperatura que hace simplemente alimentandolo con un PowerBank en el cable USB

P: ¿No sabes cómo se carga el programa en PicoBricks?

R: Porque no te has leído <https://libros.catedu.es/books/pico-bricks/page/dos-formas-de-ejecutar-los-programas>

# PROYECTO Graphic Monitor

Extraído de *Pico Bricks IDE Book* CC-BY-SA <https://picobricks.com/pages/idebook> ver [créditos](#)

The screenshot shows the PicoBricks IDE interface. At the top is a red toolbar with buttons for File, PicoProject, Run, Stop, a refresh icon, a Bluetooth icon, Doc, and a code editor icon. Below the toolbar is a workspace with a grid background. On the left, a code block is visible, starting with a 'PicoBricks' block, followed by a 'forever' loop. Inside the loop, there is a 'set color to round Read Potentiometer 255 65535' block and a 'Set RGB Led Colour R G B' block. On the right, there is a library of project templates. The 'Graphic Monitor' template is highlighted with a yellow border. It is titled 'Graphic Monitor' and has the description 'Adjusting RGB Brightness with a Twist!'. Other templates include 'Blink', 'Action-Reaction', and 'Thermometer'.

Resultado

<https://www.youtube.com/embed/NqovcB6RImA>

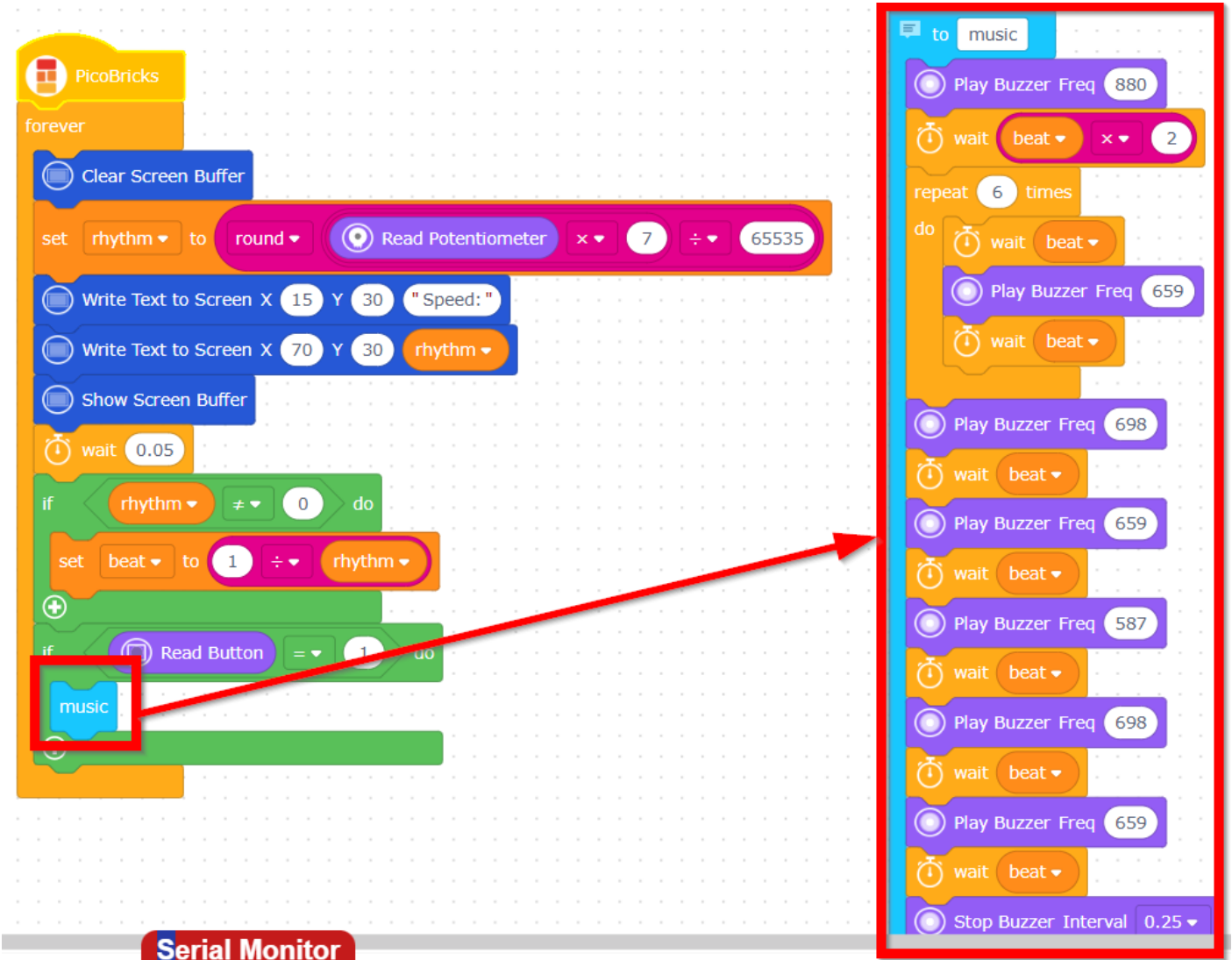
# PROYECTO Dominate the Rhythm

Extraído de *Pico Bricks IDE Book* CC-BY-SA <https://picobricks.com/pages/idebook> ver [créditos](#)

Este proyecto ya es más complejo y recomendamos importarlo desde el tutorial como siempre pues es más largo

The screenshot displays the PicoBricks IDE interface. On the left, a block-based code editor shows a 'forever' loop containing the following blocks: 'Clear Screen Buffer', 'set rhythm to round(Read Potentiometer \* 7 / 65535)', 'Write Text to Screen X 15 Y 30 "Speed:"', 'Write Text to Screen X 70 Y 30 rhythm', 'Show Screen Buffer', 'wait 0.05', 'if rhythm != 0 do' block containing 'set beat to 1 / rhythm', and 'if Read Button = 1 do' block containing 'music'. On the right, a gallery of project cards is visible. The card for 'Dominate the Rhythm' is highlighted with a red border and a red arrow pointing to it. Other visible cards include 'Remember to press the button or buzzer won't stop beeping!', 'kicks in at 20°C!', 'racers?', 'Air Piano', 'Buzz Wire Game', 'Digital Ruler', 'Know Your Color', and 'Automatic Tr'.

Implica la utilización de FUNCIONES



**Serial Monitor**

Y recomendamos leer el tutorial, esta bien explicado en el libro en la página 34;

[https://drive.google.com/file/d/1plad6bjn87FcgHb3cpd1vl-B\\_A25rnfF/preview](https://drive.google.com/file/d/1plad6bjn87FcgHb3cpd1vl-B_A25rnfF/preview)

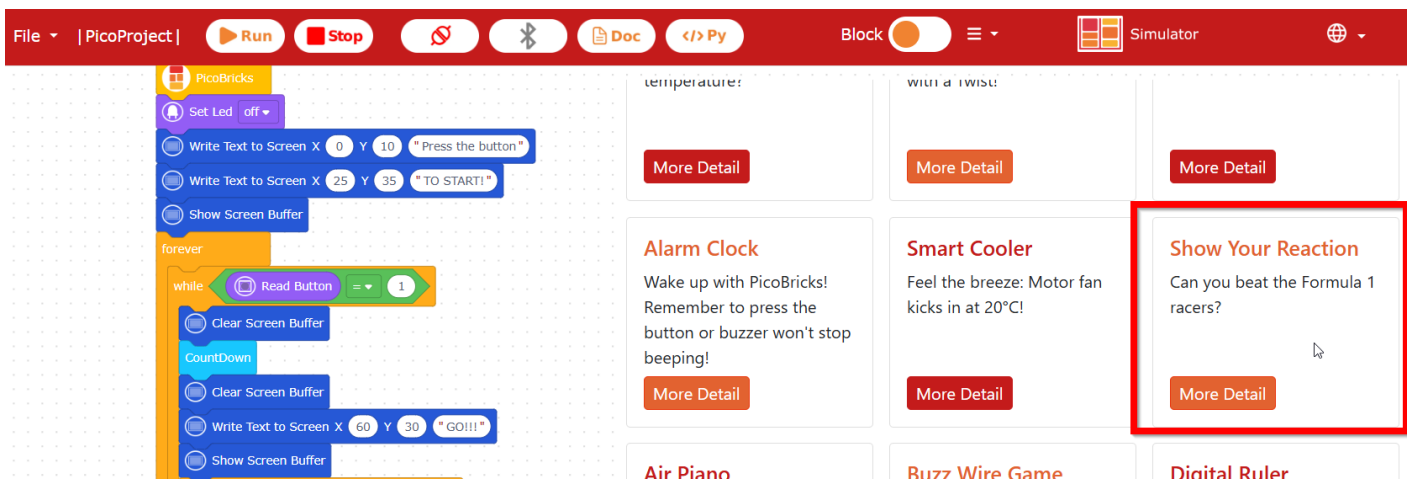
Como se puede ver en el resultado, la primera vez suena la música a un ritmo número 4 pero en la segunda vez subimos con el potenciómetro al ritmo máximo 7 y la música suena más deprisa

<https://www.youtube.com/embed/WynkqehvWuw>

# PROYECTO Show Your Reaction

Extraído de *Pico Bricks IDE Book* CC-BY-SA <https://picobricks.com/pages/idebook> ver [créditos](#)

Cuando nuestro proyecto comience a funcionar, mostraremos un mensaje de bienvenida en el OLED pantalla. A continuación imprimiremos en la pantalla lo que el usuario tiene que hacer para iniciar el juego. Para comenzar el juego, le pediremos al jugador que se prepare contando hacia atrás desde 3 en la pantalla después de presionar el botón. Después del final de la cuenta regresiva, el El LED rojo se encenderá en un tiempo aleatorio entre 2 y 10 segundos. Reiniciaremos el temporizador inmediatamente después se enciende el LED rojo. Mediremos el temporizador tan pronto como el se vuelve a pulsar el botón. Este valor que obtengamos estará en milisegundos. Mostraremos esto en la pantalla como el tiempo de reacción del jugador.



Aquí he ganado pues sólo he tardado 1ms en pulsar el botón

<https://www.youtube.com/embed/CLEUZPzI2vI>

# PROYECTO My Timer

Extraído de *Pico Bricks IDE Book CC-BY-SA* <https://picobricks.com/pages/idebook> ver [créditos](#)

El clásico cuenta atrás pero con la peculiaridad que es fácil de programar con el potenciómetro, hasta las horas !

The image shows the PicoBricks IDE interface. On the left, a block-based program is visible, featuring a 'forever' loop with 'if' statements for 'clue' values 0, 1, 2, and 3, and a 'timerFunction' block. The main workspace is filled with various colored blocks for setting variables, loops, and conditional execution. On the right, a gallery of project cards is displayed, each with a title, a brief description, and a 'More Detail' button. The 'My Timer' card is highlighted with a red border. Other cards include 'Magic Lamb', 'Maze Solver Robot', 'Night And Day', 'Piggy Bank', and 'Smart Greenhouse'. The interface also shows a top menu bar with options like 'File', 'Run', 'Stop', and 'Block', and a 'Simulator' button.

<https://www.youtube.com/embed/QA7Oe8KibCo>

# PROYECTO Alarm Clock

Extraído de *Pico Bricks IDE Book* CC-BY-SA <https://picobricks.com/pages/idebook> ver [créditos](#)

The screenshot shows the PicoBricks IDE interface. The top bar includes a menu (File), the current project name (PicoProject), and several control buttons: Run, Stop, a refresh icon, a Bluetooth icon, a Doc icon, and a code editor icon. There are also a 'Block' toggle and a 'Simulator' button.

The code editor on the left contains the following script:

```
PicoBricks
Write Text to Screen X 25 Y 32 "Good night"
Show Screen Buffer
wait 2
forever
  wait 1
  Clear Screen Buffer
  if Read Light Sensor < 20000 do
    Write Text to Screen X 15 Y 32 "Good Morning"
    Show Screen Buffer
    Set RGB Led Colour
    Play Buzzer Freq 300
  if Read Button = 1 do
    Clear Screen Buffer
    Write Text to Screen X 0 Y 32 "Have a nice day"
    Show Screen Buffer
    Clear RGB Led Colour
```

The project gallery on the right displays several project cards, each with a 'More Detail' button. The 'Alarm Clock' card is highlighted with a red border. The cards include:

- Thermometer: Hot or cold? What's the temperature?
- Graphic Monitor: Adjusting RGB Brightness with a Twist!
- Dinos: Take th...
- Smart Cooler: Feel the breeze: Motor fan kicks in at 20°C!
- Air Piano
- Buzz Wire Game
- Dinit:

[https://www.youtube.com/embed/8Drcl\\_YEsFs](https://www.youtube.com/embed/8Drcl_YEsFs)





<https://www.youtube.com/embed/Tv9z8krs26g>

# PROYECTO BUZZ WIRE GAME

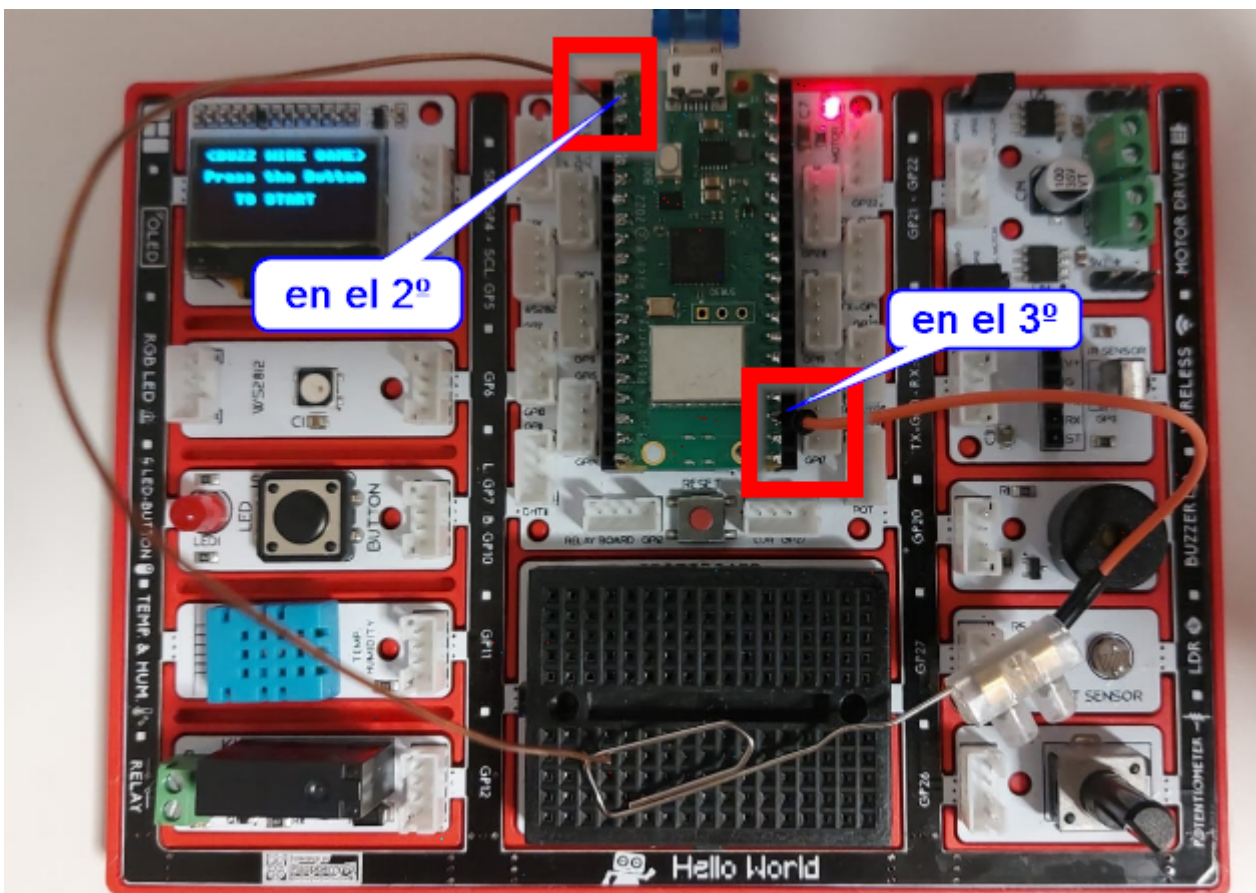
## Enunciado

Este juego es el típico de pasar una arandela por un cable y si toca, suena la alarma

## Hardware

Preparamos :

- un cable pelado, rígido pero fino, conectado en el GP1 (segundo agujero por arriba izquierda) y GND (tercer agujero abajo derecha)
- un cable normal dupond
- un clip unido al cable normal dupond por una regleta pequeña como en la foto.



## Software

El programa lo tienes en los tutoriales Buzz Wire Game

The image shows a screenshot of the PicoBricks simulator interface. At the top, there is a red toolbar with buttons for 'Run', 'Stop', 'Doc', and '</> Py'. A red circle with the number '1' is placed over the 'Doc' button. Below the toolbar, the main workspace is divided into two sections. On the left, there is a code block for a 'while' loop. The code starts with a 'start' block, followed by a 'forever' loop. Inside the loop, the steps are: 'Clear Screen Buffer', 'Write Text to Screen X 20 Y 30 "GAME STARED"', 'Show Screen Buffer', 'Serial Print Read Digital Pin 1', and an 'if' statement. The 'if' statement checks 'Read Digital Pin 1 == 0'. If true, it executes 'Play Buzzer Freq 300', 'Set Led on', and 'Finish'. On the right, there is a text area with the title 'Buzz Wire Game' and a red circle with the number '2' next to it. Below the title, there is a paragraph of text: 'In this project, we will electronically prepare the attention and concentration develop the help of a conductor wire using the buzzer and LED module with Picobricks. Projects don't always have to be about solving problems and making things easier. Y projects to have fun and develop yourself. Attention and concentration are features i to develop.' Below the text is a section titled 'Wiring Diagram' which shows a schematic of a PicoBricks board connected to a buzzer and an LED. A red circle with the number '3' is placed over the 'Project Code (Click to Try It)' link below the diagram.

1

2

3

**Buzz Wire Game**

In this project, we will electronically prepare the attention and concentration develop the help of a conductor wire using the buzzer and LED module with Picobricks.

Projects don't always have to be about solving problems and making things easier. Y projects to have fun and develop yourself. Attention and concentration are features i to develop.

**Wiring Diagram**

Project Code (Click to Try It)

## Resultado

<https://www.youtube.com/embed/CzDTe3UoNcA>

Este proyecto no funciona muy bien con los otros lenguajes de programación, no sé por qué

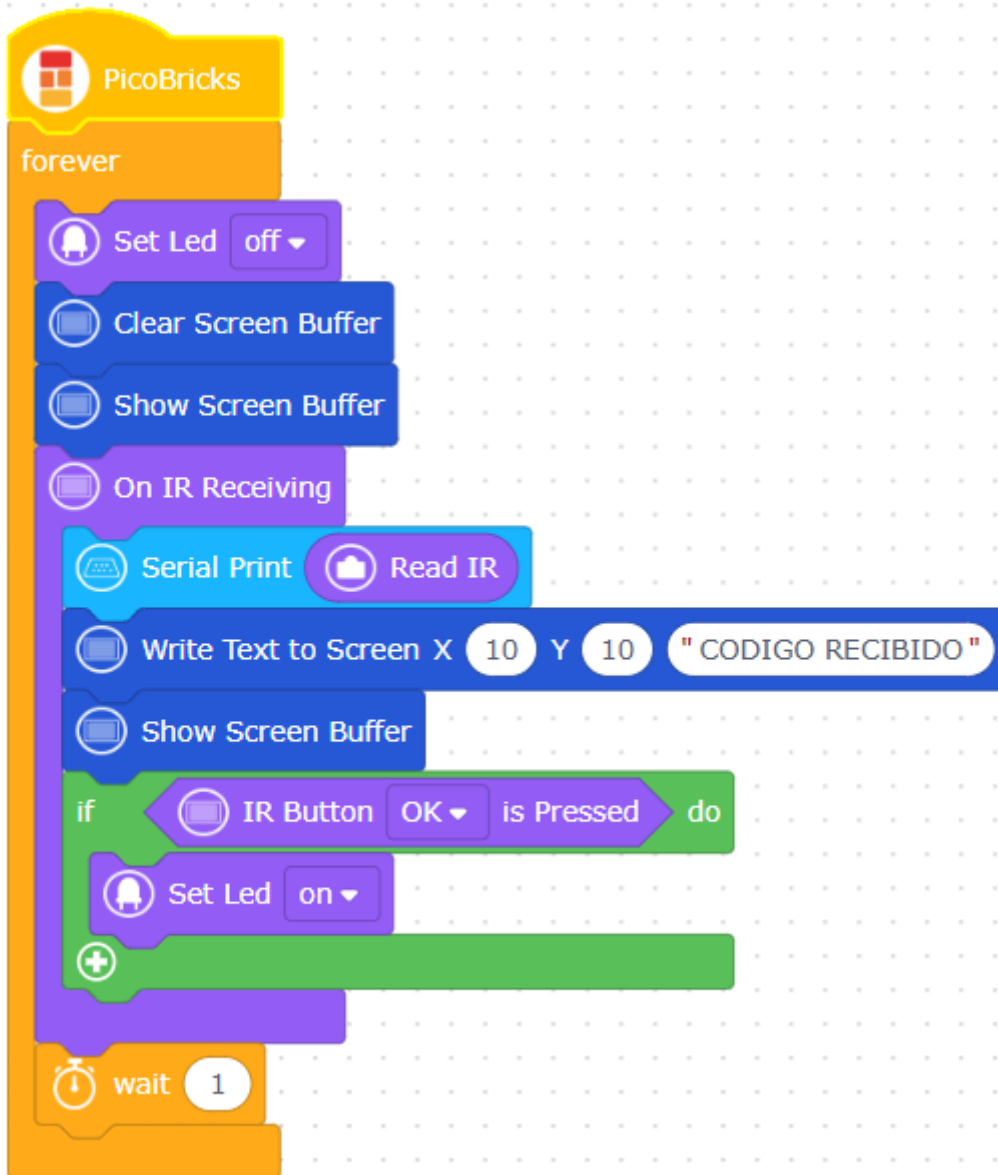
# Algo diferente PROYECTO IR

No hay en los tutoriales ningún proyecto para usar el mando IR, luego este proyecto no pertenece a ninguno de los tutoriales que predetermina PicoBricks. Proponemos el siguiente enunciado

Realizar un programa que:

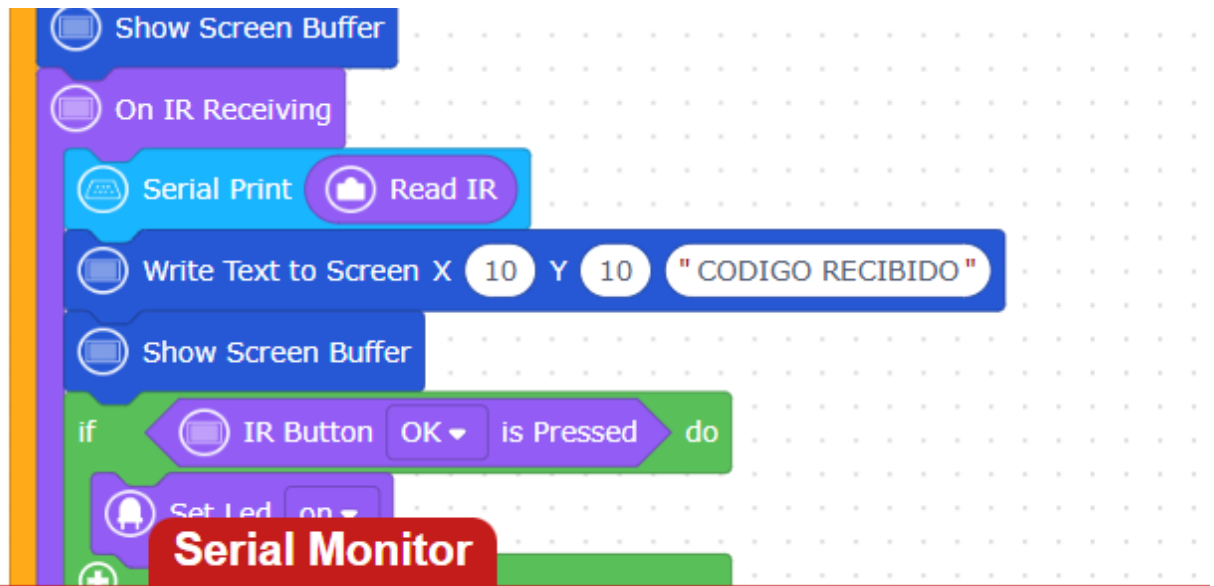
- si se aprieta un botón del mando IR que visualice por la pantalla OLED que ha recibido un código
- visualizará por el puerto serie el código recibido
- si la tecla es OK se encenderá el led rojo

## Solución



## Resultado

Por el puerto serie van apareciendo los códigos de las teclas apretadas en el mando IR



```
>> None
>> Data 15 Addr 0000
>> None
>> Data 1c Addr 0000
>> None
>> Data 44 Addr 0000
>> None
>> Data 46 Addr 0000
>> None
>> Data 0d Addr 0000
```

y en la pantalla OLED se visualizaba que se había recibido un código y si era OK se enciende el led rojo:

<https://www.youtube.com/embed/6kSRjpbTSDg>

# PICO COCHE

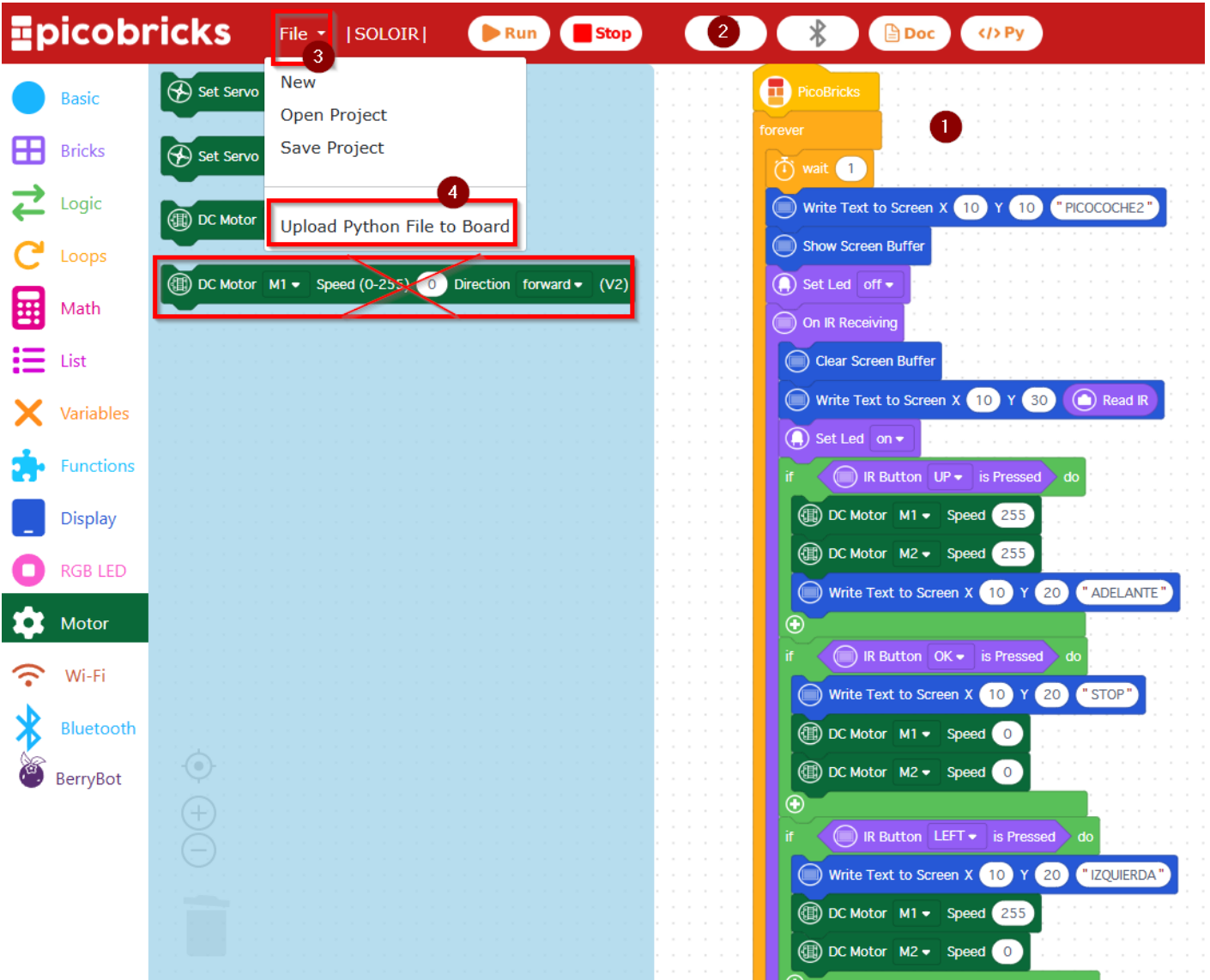
Podemos comprar en cualquier tienda de electrónica por unos 10€ un kit de coche, por ejemplo [aquí](#)



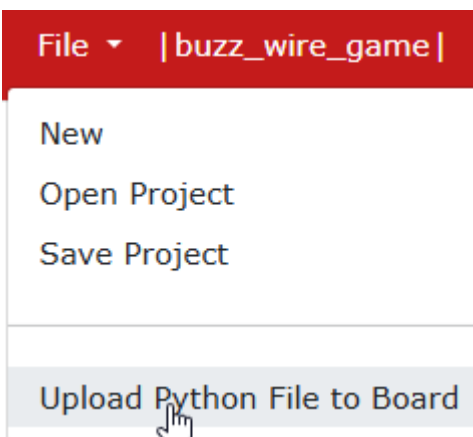
En Picobricks creamos un programa similar al anterior de Infrarrojos

Precaución: No utilizar la instrucción señalada, bloquea nuestra Raspberry Pi Pico W, sospechamos que es para otra versión. Por lo tanto no puede dar marcha atrás

Consejo: Grabarlo en la placa, y así es independiente

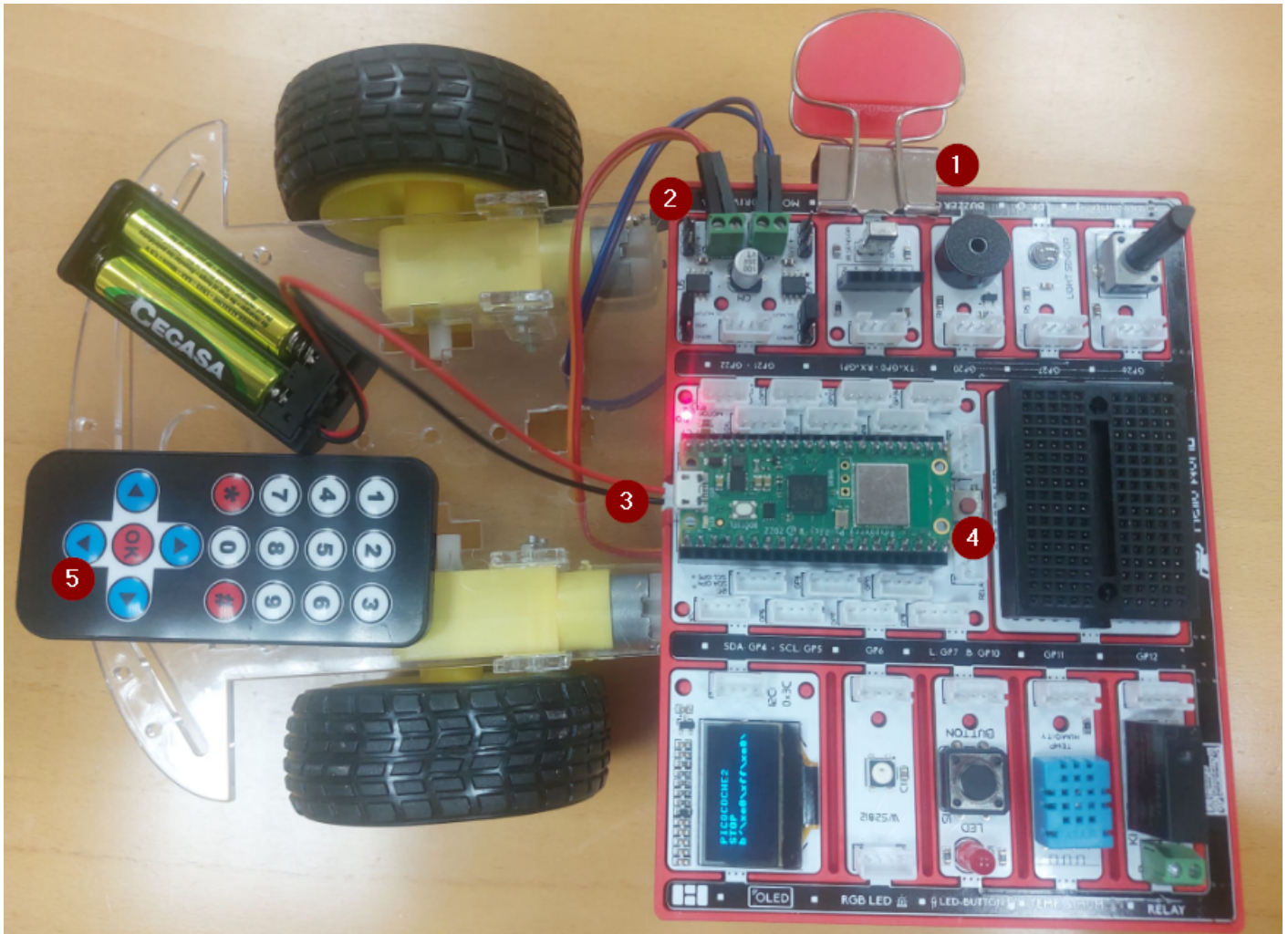


Una vez cargado el programa en la placa Raspberry Pi pico W,

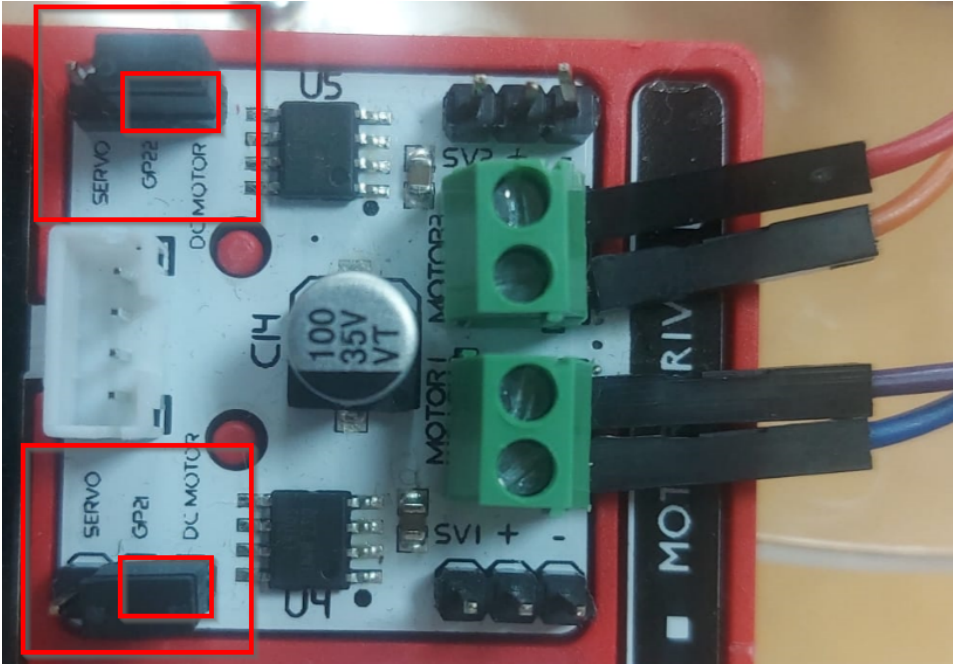


lo montamos en el coche:

1. Ponemos alguna sujeción, en este caso he puesto una pinza, pero puede ser una goma elástica.
2. Conectamos los cables de los motores en los terminales verdes
  1. Si por alguna razón va al revés algún motor, intercambiar los cables
3. Conectamos alimentación
  1. Puede ser también con el cable usb y un powerbank
4. Apretamos el botón reset
5. A jugar



Detalle de conexión, asegúrate que los jumpers están colocados en modo DCMOTOR, es decir GP21 y GP22 tienen que estar conectados con DCMOTOR1 y DCMOTOR2



☐☐ las pilas estaban algo gastadas ...

<https://www.youtube.com/embed/G1snag4AK9M>

# Mapeo

## ¿Qué es eso de "mapeo"?

En la jerga robótica, dicho pronto y mal pero para que se entienda, mapear significa hacer un **cambio de escala**

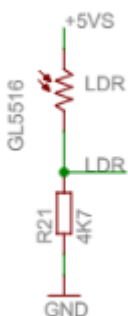
## ¿Cuándo se dan esas situaciones?

**SITUACION A :** Queremos leer un valor de entrada analógica en un Arduino, por lo tanto va de 0-1023 y queremos que se copie en una salida digital PWM de Arduino que va de 0-255

**SITUACION B :** Queremos leer un valor de entrada analógica en un Arduino, por lo tanto va de 0-1023 e interpretarlo en sus valores de voltios. Si suponemos que la placa se alimenta a 5V la variable de salida irá desde 0 a 5V

**SITUACION C :** Queremos leer el valor de un LDR, que tapándolo nos da 917 e iluminándolo al máximo es 1023, lo queremos copiar en una salida digital PWM, o sea que la salida va desde 0 a 255

*Nota: El mínimo de 917 (puede ser otro número, es un valor experimental) es debido a que los LDR van montados en un divisor de tensión como el de la figura, y la resistencia de abajo, siempre se queda algo de tensión*



**SITUACION D :** Queremos según el valor de un joystick conectado a las entradas analógicas de un Arduino (esto pasa en Echidna) se representen en la pantalla de Scratch



2\*220 por 2\*180, es decir

- Eje X : el potenciómetro (vamos a llamarlo *potx*) va de 0 a 1023 y la salida (*ejex*) va de -220 a 220
- Eje Y : el potenciómetro (vamos a llamarlo *poty*) va de 0 a 1023 y la salida (*ejey*) va de -180 a 180

**SITUACION E:** Ídem pero no con el potenciómetro, sino con el acelerómetro (vamos a llamarlo *acel*) que va 250 a 500

**SITUACION F :** Queremos leer un valor de entrada analógica en un Arduino, por lo tanto va de 0-1023 y queremos que se copie en una salida de un servo, por lo tanto lo que necesita es un ángulo que va de 0-180

**SITUACION G :** Ídem que F pero una raspberry por lo tanto GPI va de 0-65.535

## ¿Cómo se consigue mapear?

- Si programas con código ArduinoIDE, tienes la instrucción **map**
- Si no tienes map, por ejemplo, programas con bloques gráficos tipo Scratch, lo tienes que hacer a mano
  - ¿Cómo? Con la ecuación de una recta

Para entendernos :

- **X** será el valor de entrada que tiene unos valores límites **X<sub>1</sub>** e **X<sub>2</sub>**
- **Y** es la variable de salida que queremos y que tiene otros valores límites **Y<sub>1</sub>** e **Y<sub>2</sub>**

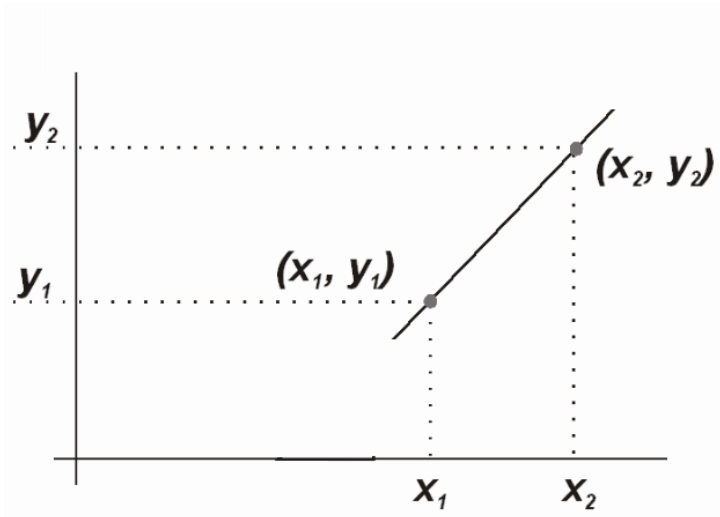
Luego y tiene esta ecuación :

$$y = y_1 + m * (x - x_1)$$

donde m es

$$m = \frac{x_2 - x_1}{y_2 - y_1}$$

Gráficamente



### ¿Una hoja de cálculo para poder hacer esos cálculos?

Sí, claro, en este enlace

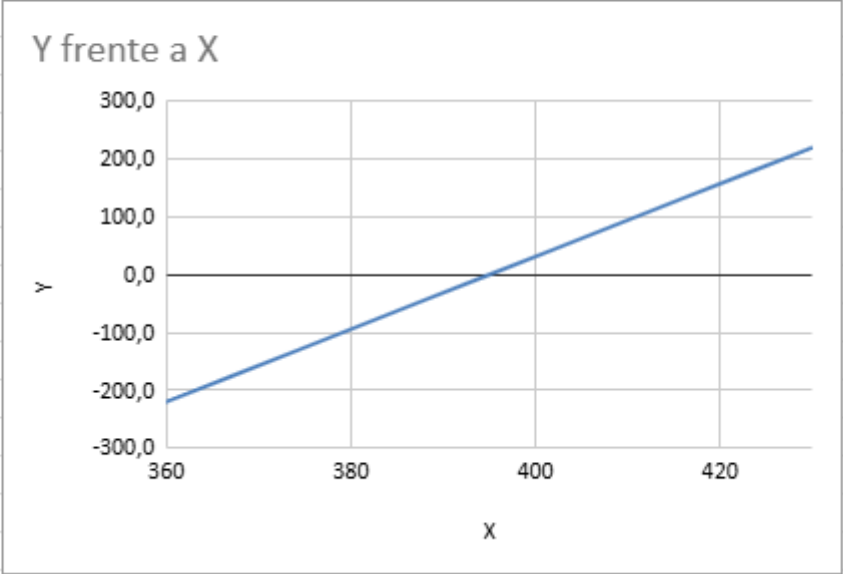
[https://docs.google.com/spreadsheets/d/1qNbaZ2c\\_H1UCNhtvp2LimfWSbaGvZLVI5gjr9Wu0ifU/edit?usp=sharing](https://docs.google.com/spreadsheets/d/1qNbaZ2c_H1UCNhtvp2LimfWSbaGvZLVI5gjr9Wu0ifU/edit?usp=sharing) dale a descargar

Rellena las casillas amarillas, y en las naranjas tienes el resultado de m y n

$y = m * x + n$		m	6,3	x1	360	y1	-220
		n	-2482,9	x2	430	y2	220

X	Y
360	-220,0
367	-176,0
374	-132,0
381	-88,0
388	-44,0
395	0,0
402	44,0
409	88,0
416	132,0
423	176,0
430	220,0

### ¿Me lo puedes hacer para cada situación anterior?

Si claro:

**SITUACION A :** Queremos leer un valor de entrada analógica en un Arduino, por lo tanto va de 0-1023 y queremos que se copie en una salida digital PWM de Arduino que va de 0-255

- Límites de las variables :
  - X de 0-1023
  - Y de 0-255
- Con la instrucción map :  $Y = \text{map}( X, 0, 1023, 0, 255);$
- Sin la instrucción map  $Y = 0.25 * X$  pues  $255/1023 = 0.25$  también podemos escribir  $Y = X/4$

**SITUACION B :** Queremos leer un valor de entrada analógica en un Arduino, por lo tanto va de 0-1023 e interpretarlo en sus valores de voltios. Si suponemos que la placa se alimenta a 5V la variable de salida irá desde 0 a 5V

- Límites de las variables :
  - X de 0-1023
  - Y de 0-5
- Con la instrucción map :  $Y = \text{map}( X, 0, 1023, 0, 5);$
- Sin la instrucción map  $Y = 0.0048 * X$  pues  $5/1023 = 0.0048$  o también podemos escribir  $Y = X/204$  que queda mejor pues  $1023/5=204$  aprox.

**SITUACION C :** Queremos leer el valor de un LDR, que tapándolo nos da 917 e iluminándolo al máximo es 1023, lo queremos copiar en una salida digital PWM, o sea que la salida va desde 0 a 255

- Límites de las variables :
  - X de 917-1023
  - Y de 0-255
- Con la instrucción map :  $Y = \text{map}( X, 917, 1023, 0, 255);$
- Sin la instrucción map  $Y = 2.4 * X$  pues  $255/(1023-917) = 2.4$

**SITUACION D :** Queremos según el valor de un joystick conectado a las entradas analógicas de un Arduino (esto pasa en Echidna) se representen en la pantalla de Scratch 2\*220 por 2\*180, es decir

- Eje X : el potenciómetro (vamos a llamarlo *potx*) va de 0 a 1023 y la salida (*ejex*) va de -220 a 220

- Eje Y : el potenciómetro (vamos a llamarlo *poty*) va de 0 a 1023 y la salida (*ejey*) va de -180 a 180

- EJEX
  - Límites de las variables :
    - *potx* de 0-1023
    - *ejex* de -220 a +220
  - Con la instrucción `map` : `ejex = map( potx, 0, 1023, -220, 220);`
  - Sin la instrucción `map` `ejex = -220 + 0.43*potx` pues  $(220 - (-220)) / 1023 = 0.43$
- EJY
  - Límites de las variables :
    - *poty* de 0-1023
    - *ejey* de -180 a +180
  - Con la instrucción `map` : `ejey = map( poty, 0, 1023, -180, 180);`
  - Sin la instrucción `map` `ejey = -180 + 0.35*poty` pues  $(180 - (-180)) / 1023 = 0.35$

**SITUACION E:** Ídem pero no con el potenciómetro, sino con el acelerómetro (vamos a llamarlo *acel*) que va 250 a 500

- EJEX
  - Límites de las variables :
    - acelerómetro *acel* de 250-500
    - *ejex* de -220 a +220
  - Con la instrucción `map` : `ejex = map( acel, 250, 500, -220, 220);`
  - Sin la instrucción `map` `ejex = -220 + 1.76*(acel-250)` pues  $(220 - (-220)) / (500 - 250) = 1.76$
- EJY
  - Límites de las variables :
    - acelerómetro *acel* de 250-500
    - *ejey* de -180 a +180
  - Con la instrucción `map` : `ejey = map( acel, 250 500, -180, 180);`
  - Sin la instrucción `map` `ejey = -180 + 1.44*(acel-250)` pues  $(180 - (-180)) / (500 - 250) = 1.44$

**SITUACION F :** Queremos leer un valor de entrada analógica en un Arduino, por lo tanto va de 0-1023 y queremos que se copie en una salida de un servo, por lo tanto lo que necesita es un ángulo que va de 0-180

- Límites de las variables :



- X de 0-1023
- Y de 0-180
- Con la instrucción map :  $Y = \text{map}( X, 0, 1023, 0, 180)$ ;
- Sin la instrucción map  $Y = 0.17 * X$  pues  $180/1023 = 0.17$  también podemos escribir  $Y = X/5.7$  pues  $1023/180=5.7$

### **SITUACION G : Idem que F pero una raspberry por lo tanto GPI va de 0-65.535**

- Límites de las variables :
  - X de 0-65535
  - Y de 0-180
- Con la instrucción map :  $Y = \text{map}( X, 0, 65535, 0, 180)$ ;
- Sin la instrucción map  $Y = 0.00274 * X$  pues  $180/65535 = 0.00274$  pero es más cómodo al revés  $Y = X/364$  pues  $65535/180=364$

# Servo

Una de las aplicaciones más utilizadas de los sistemas de control por ordenador y en la robótica están asociados con los motores, que permiten accionar o mover otros componentes, como puertas, barreras, válvulas, ruedas, etc. Uno de los tipos que vamos a ver en este capítulo son los servos, hay de dos tipos:

- El **servomotor** o **servos convencionales** que posee la capacidad de posicionar su eje en un ángulo determinado entre 0 y 180 grados en función de una determinada señal.
- **Servo de rotación continua** Son servos por fuera igual que los anteriores, pero pueden girar 360º y se controlan por tiempo

Por defecto cuando se dice **servo**, es un **servomotor** o **servo convencional**

## Servomotores o servos convencionales



Los servos son un tipo especial de motor en el que se añade un circuito lógico electrónico que permite un control mucho más preciso que a un motor normal de corriente continua. Esto les permite posicionar el eje en un ángulo determinado.

El hardware interno se compone de un potenciómetro y un circuito integrado que controlan en todo momento los grados que gira el motor. De este modo, en nuestro caso, desde Arduino, usando las salidas digitales PWM podremos controlar fácilmente un servo. Lo ideal es conectarlo a 6V pero trabajan bien en los 5V del Arduino.

Hay muchos modelos, en robótica educativa cuestan entre 1-5€, el más común es el SG90, muy barato, pero tiene muy poca fuerza, el MG90S tiene algo más, si queremos algo más, ya tiene que ser el MG996R pero ya este modelo **NO se puede conectar directamente al Arduino o Raspberry**, el pico de energía que necesita, provoca el reinicio de la placa. Incluso varios pequeños SG90.

Si quieres saber más, te recomendamos <https://www.luisllamas.es/controlar-un-servo-con-arduino/>

Ejemplos de uso de servos:

- [Curso Arduino con código](#)
- [Curso brazo robótico del mClon con nanoArduino](#)
- [Apertura de barrera por ultrasonidos en curso Arduino con ArduinoBlocks](#)
- [Tractor entrando en el corral Arduino con EchidnaShield](#)
- [Apertura de puerta Domótica con Arduino](#)
- [Servo con PicoBrick](#)
- [Apertura ventana y puerta en Smarth Home para microbit](#)
- [Smart Agriculture Kit para micro:bit](#)

## Enunciado

Vamos a aprovechar el concepto de **mapeo** con el siguiente enunciado

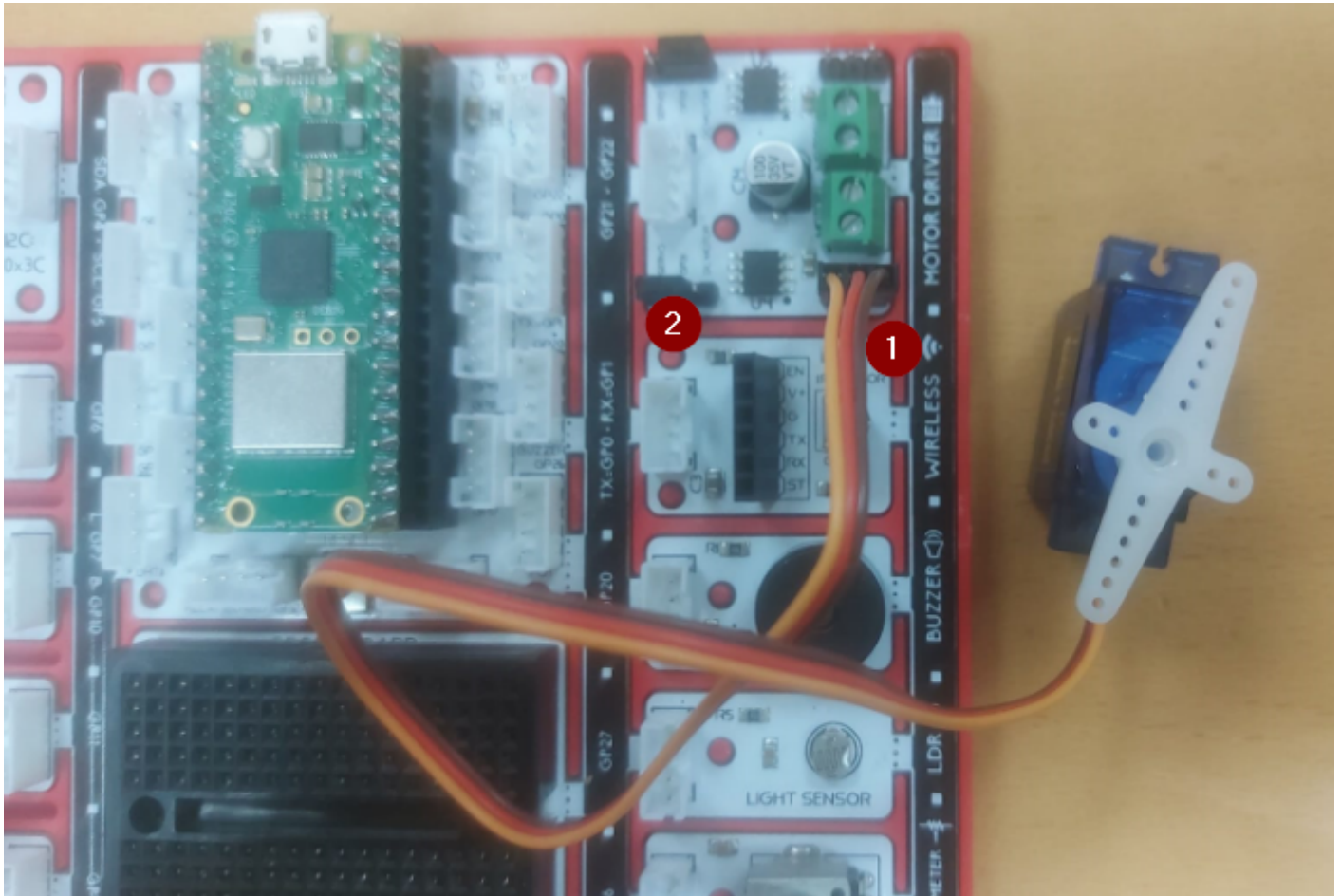
Queremos que se mueva un servo motor (ángulo de 0º a 180º) en función del potenciómetro, conectado a GP27 por lo tanto sus valores van de 0 a 65535 (que es 2 elevado a 16)

## Hardware

1. Conectamos un servo en el slot correspondiente

Ojo, fíjate que tienes que conectar el pin - con el marrón. Mira la foto

2. Movemos el Jumper a la posición que conecte GP21 con Servo

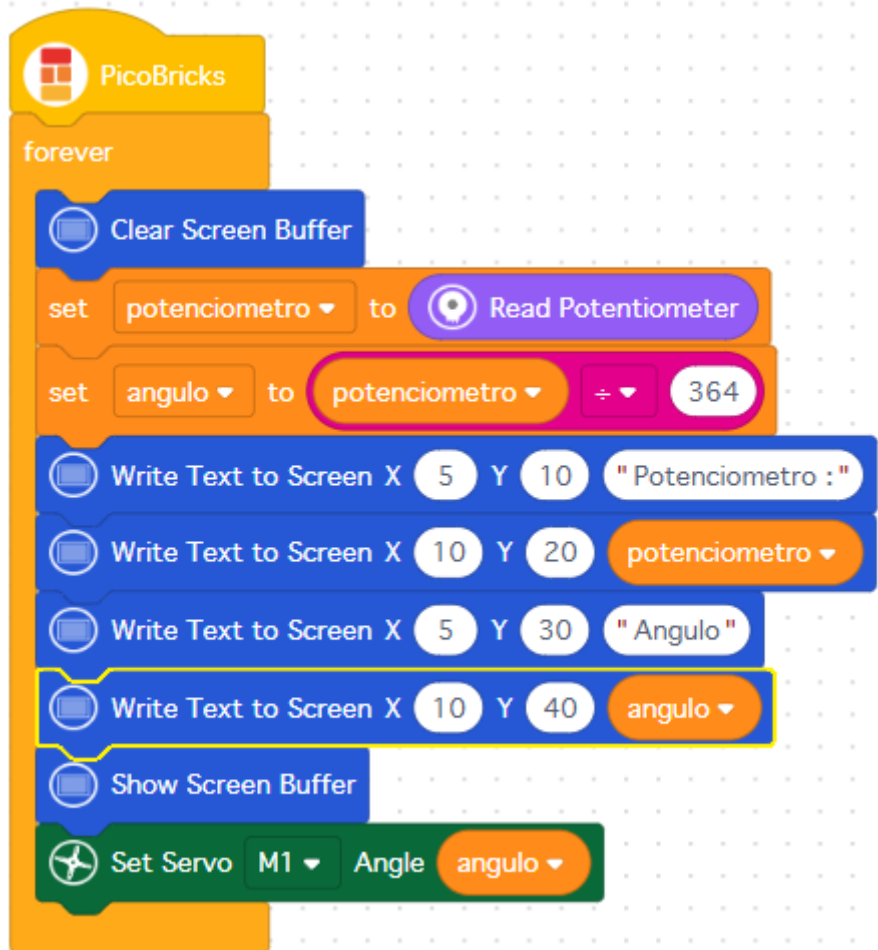


## Mapeo

Utilizaremos la situación G de **mapeo** <https://libros.catedu.es/books/pico-bricks/page/mapeo> por lo tanto dividiremos la posición del potenciómetro por 364

## Software

El programa es el siguiente:



No utilizar esta instrucción



Set Servo M1 Angle 180 (V2)

## Resultado

<https://www.youtube.com/embed/zspyvn8PN4g>

**Te atreves a...** manejar dos servos, por ejemplo con el mando a distancia de IR

# Relé

## Enunciado

Buscamos un circuito que cuando haya luz, se encienda y cuando hay oscuridad se apaga. Es al revés de una **luz crepuscular** es decir, que cuando sea de noche se encienda y si es de día que se apague automáticamente

P: **¿Por qué lo hacemos al revés?.**

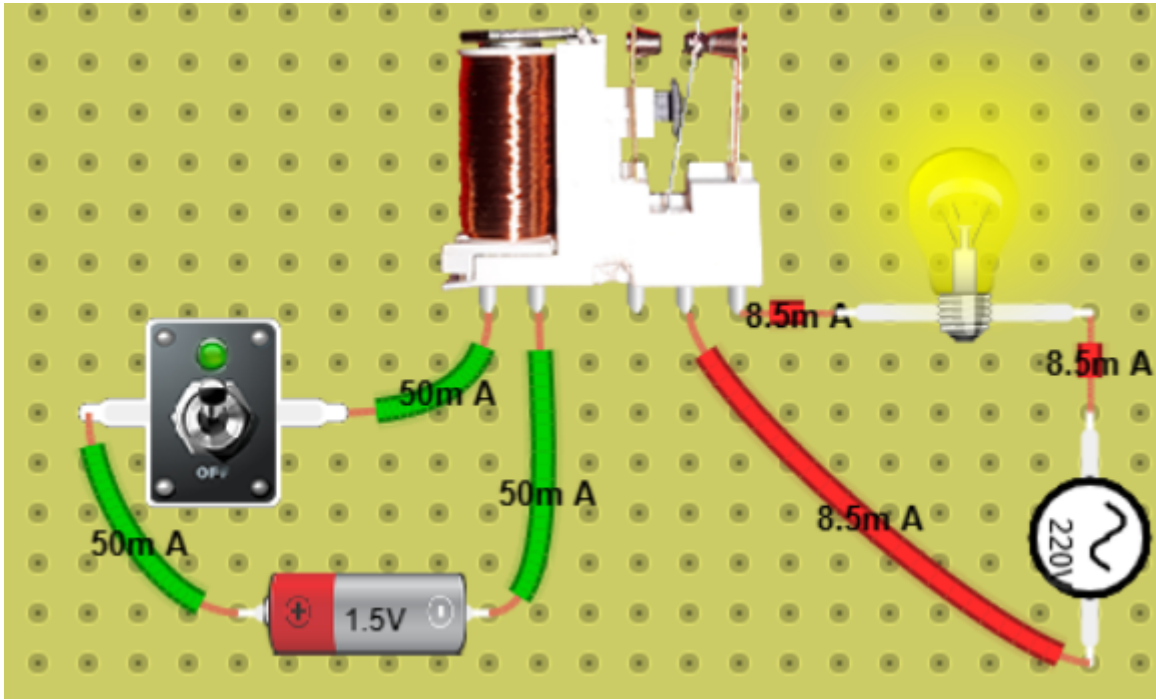
R: Por que nos evitamos **retroalimentaciones** es decir, si hay oscuridad, se enciende pero como se enciende es de día, por lo que se apaga pero como es de noche se enciende, ... etc... y comienza a hacer clack, clack, clack, clack, clack, clack, clack, clack,...

P **¿Pero entonces cómo se soluciona en las luces públicas?**

R: El sensor LDR no se dirige a las luces a encender, o incluso con una pantalla que haga sombra.

## Relé

Un relé es un interruptor activado por un electroimán, lo que permite independizar los circuitos. En el dibujo se ve que el circuito rojo de 220V esta separado del verde, de sólo 1.5V. Pero es el circuito verde que al funcionar, hace que el electroimán mueva el interruptor del relé y encienda la bombilla. El objetivo es que he podido encender una bombilla de 220V sin tocar los 220V peligrosos. En el circuito verde, el interruptor puede ser un Arduino. Experimentalo en este [simulador](#).

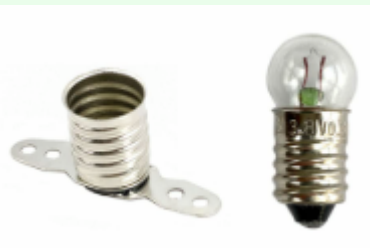


## Circuito con Picobriks

### ATENCIÓN,

- NO RECOMENDABLE PARA PRIMARIA.
- NO DEJAR A LOS ALUMNOS SOLOS CON ESTE CIRCUITO.
- SE UTILIZAN TENSIONES DE 220V
- EL CONECTOR VERDE DEL RELÉ ES MUY PEQUEÑO:
  - los dos bordes están muy próximos, NO UTILIZAR CABLE CON HILOS peligro de que algún hilo cortocircuite
  - no utilizar potencias superiores a 20W

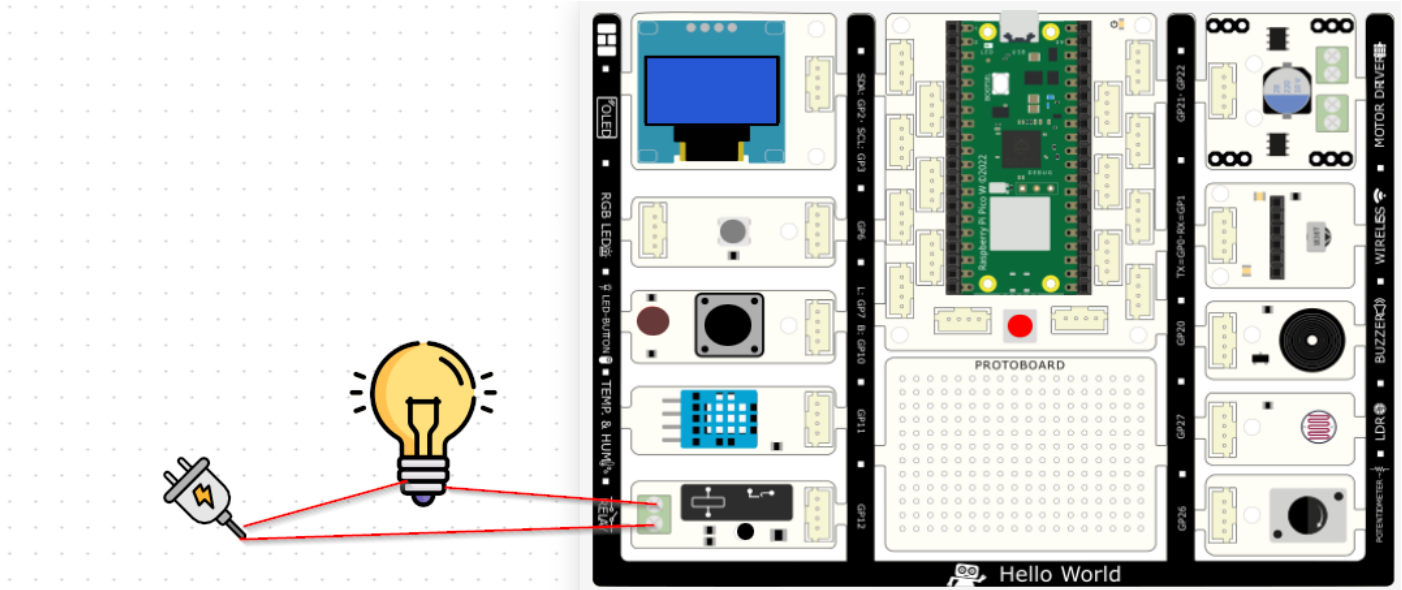
Si no quieres estos peligros,



na pila y la bombilla doméstica por

una pequeña de maquetas :

El circuito que se propone es utilizar el relé para que cierre un circuito que encienda una bombilla. El circuito con los cables rojos estarían a 220V, luego **peligro !!!**



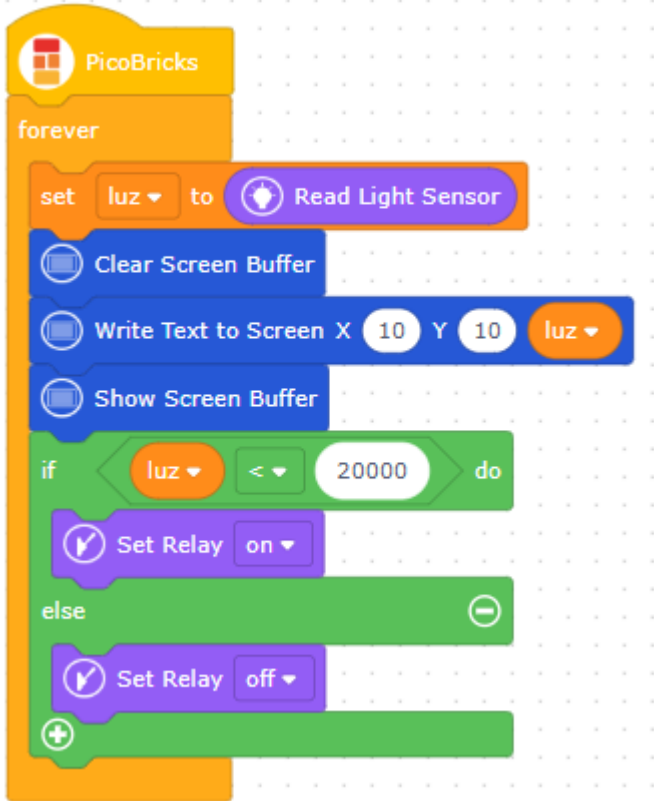
## Programa

El programa lee la intensidad luminosa y lo visualiza por la pantalla OLED. Si es inferior a 20.000 eso quiere decir que hay luz, por lo tanto que cierre el circuito.

```

PicoBricks
forever
  set luz to Read Light Sensor
  Clear Screen Buffer
  Write Text to Screen X 10 Y 10 luz
  Show Screen Buffer
  if luz < 20000 do
    Set Relay off
  else
    Set Relay on
  
```

si quieres hacerlo crepuscular como es en la vida real, cambia el estado del relé pero aleja el sensor de la bombilla o pon una pantalla de sombra



## Resultado

<https://www.youtube.com/embed/7CYORlz7Ze8>

# Sensor de distancia de ultrasonidos

## ¿Qué es el sensor de distancia HC-SR04?

Es un sensor digital de distancias por ultrasonidos capaz de detectar objetos y calcular la distancia a la que se encuentra en un rango de 2 a 350 cm. Su uso es tan sencillo como enviar el pulso de arranque y medir la anchura del pulso de retorno.

**No es un sensor preciso**, con una ligera inclinación de la superficie ya da lecturas erróneas pero es muy barato

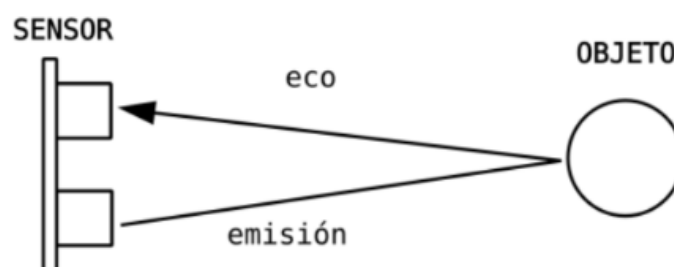
El más común es el **HC-SR04** que tiene 4 pines de conexión: **VCC** **Trig** (Disparo del ultrasonido) **Echo** (Recepción del ultrasonido) y **GND** aunque en algunos modelos como el de [ElecFreaks](#) tiene 3 pines. Integra Trig y Echo en uno sólo.

La distancia se calcula con esta fórmula:

**Distancia en cm = {(Tiempo en segundos entre Trig y el Echo) \* (V.Sonido 34000 en cm/s)} / 2**

Si programas en código, tienes que utilizar la fórmula anterior, previamente tienes que programar el cálculo del tiempo entre una emisión de un pulso en Trig y la respuesta en Echo.

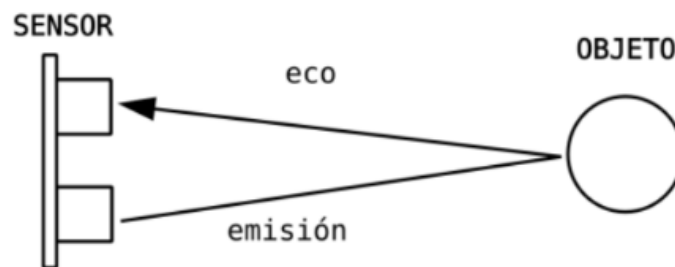
bloques, no es necesario, seguro que hay un bloque que lo hace todo



Ejemplos de uso:

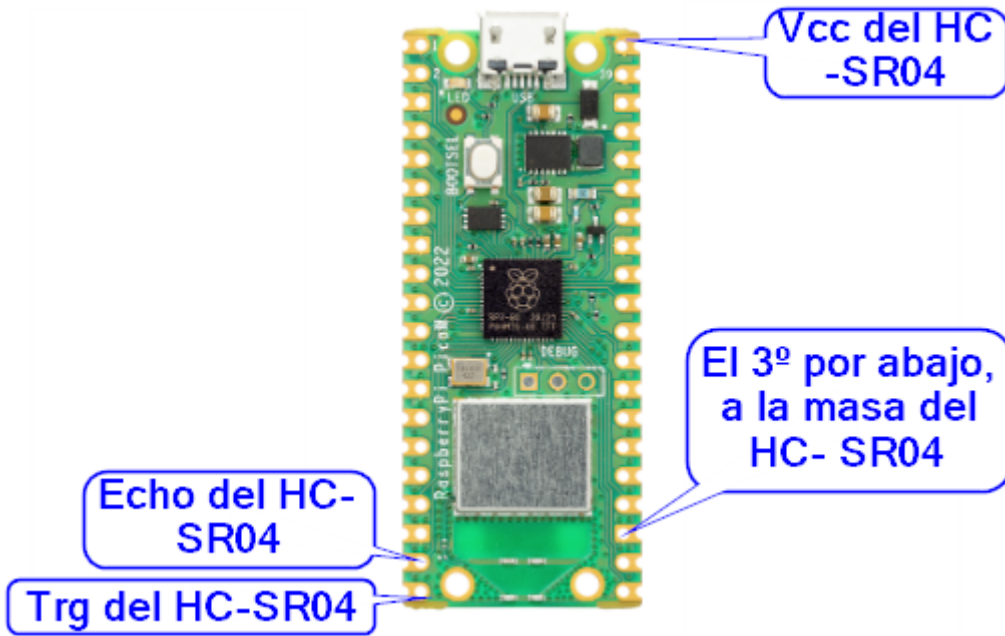
- [Alarma en Domótica con Arduino](#)
- [Piano invisible en Arduino con ArduinoBlocks,](#)

- [Sensor parking en Arduino con ArduinoBlocks](#)
- [Piano invisible en Arduino con mBlock](#)
- [Sensor parking en Arduino con mBlock](#)
- [Sensor de distancia de ultrasonidos con Picobricks](#)



## Conexión con la Raspberry Pi Pico E

Si miras <https://libros.catedu.es/books/pico-bricks/page/que-es-pico-bricks> verás el esquema para poder conectar los pines del HC-SR04 con la Raspberry, te lo mostramos aquí



La conexión de Echo y de Trg es arbitraria, lo hemos puesto en GP14 y GP15 que es la que recomienda el programa Picobricks

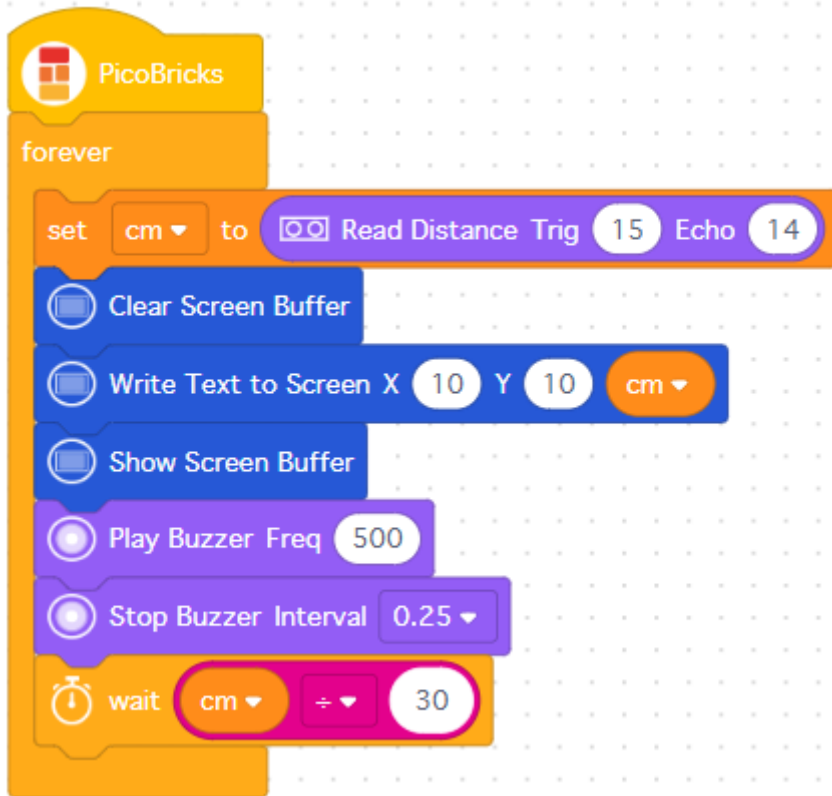


## Enunciado

Mostrar por la pantalla de OLED la distancia y a la vez que suene un radar como en los asistentes de parking de los coches

## Programa

Para hacer el pitido intermitente en función de los centímetros del objeto, se utiliza una pausa, como la espera en cm convertirla a segundos es muy grande, se divide por 30, puedes modificar este valor a tu gusto



## Resultado

<https://www.youtube.com/embed/ZV5cjfenSE>

**Te atreves a...** Juntar el Pico coche con un sensor de distancia de tal manera que vaya autónomo evitando los obstáculos como un romba

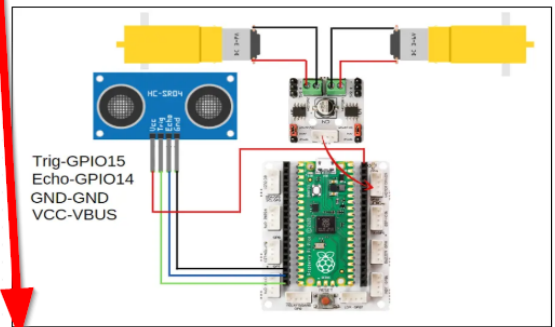
**¿No te atreves?** Pues aquí tienes la solución:



**Maze Solver Robot**

In the maze solving robot project, we will use the 2WD robot car kit that con  
Coding education is as old as the history of programming languages. Today,  
education and make it exciting and fun. The first of these is educational rob

**Wiring Diagram**



**Project Code (Click to Try It)**