2025

Tutorial VinciBot

(Kit AI Visión)

Susana Oubiña Falcón

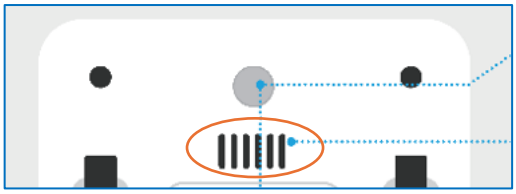
Índice

1. Características

El set educativo de robótica VinciBot Coding está diseñado para estudiantes de 8 años en adelante. El robot funciona en plataformas de programación gráfica y Python, y puede usarse en las temáticas de inteligencia artificial e IoT.

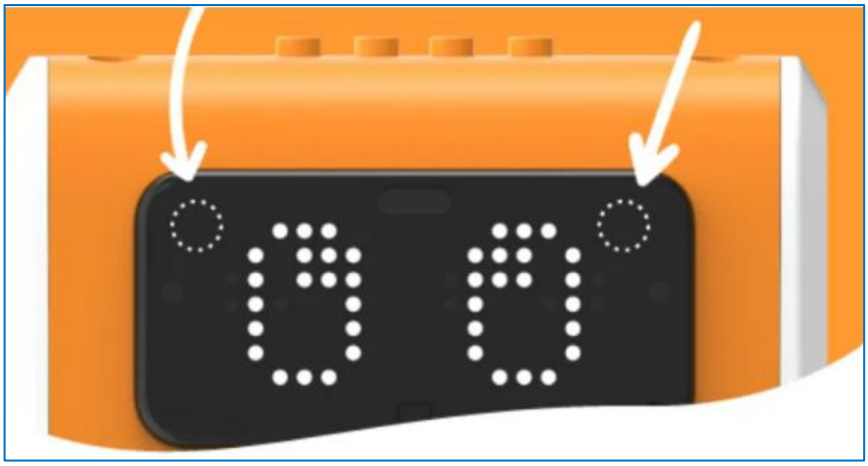
VinciBot es un microcontrolador basado en ESP32-S3 con procesadores de doble núcleo integrados, que funciona a 240 MHz, 16 MB de memoria flash y 8 MB de RAM. Dispone de conexión inalámbrica Wi-Fi y Bluetooth, y es compatible con MicroPython. Sus características técnicas son:

- Matriz de LED (128 LEDs blancos que pueden mostrar imágenes, números y letras, etc.) 6 LED RGB programables.
- Altavoz (Incluye 21 sonidos de instrumentos musicales y permite cargar formatos como wav, mp3, etc.). situado en la parte inferior del robot.



- Comunicación por infrarrojos (1 transmisor, 2 receptores)

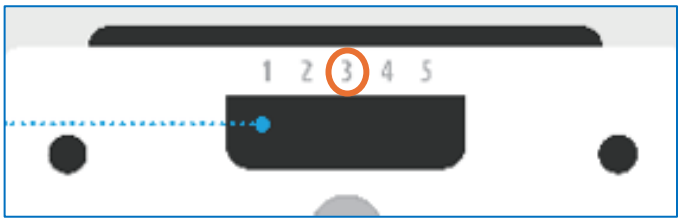
- Sensor de luz (derecho e izquierdo)



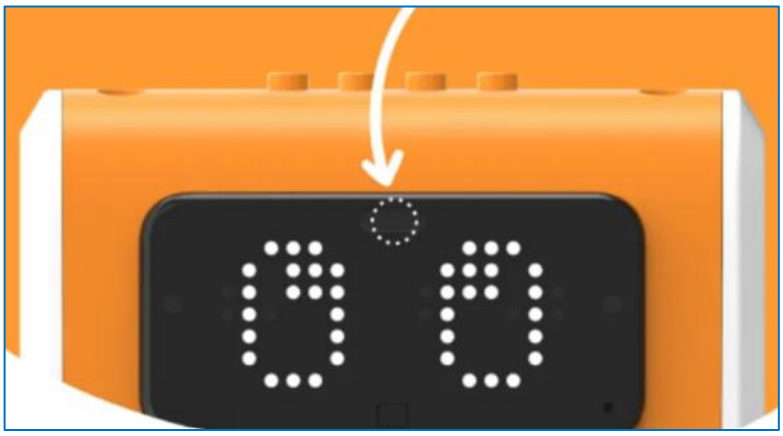
- Sensor de sonido con reconocimiento de voz

- Sigue-líneas de 5 vías

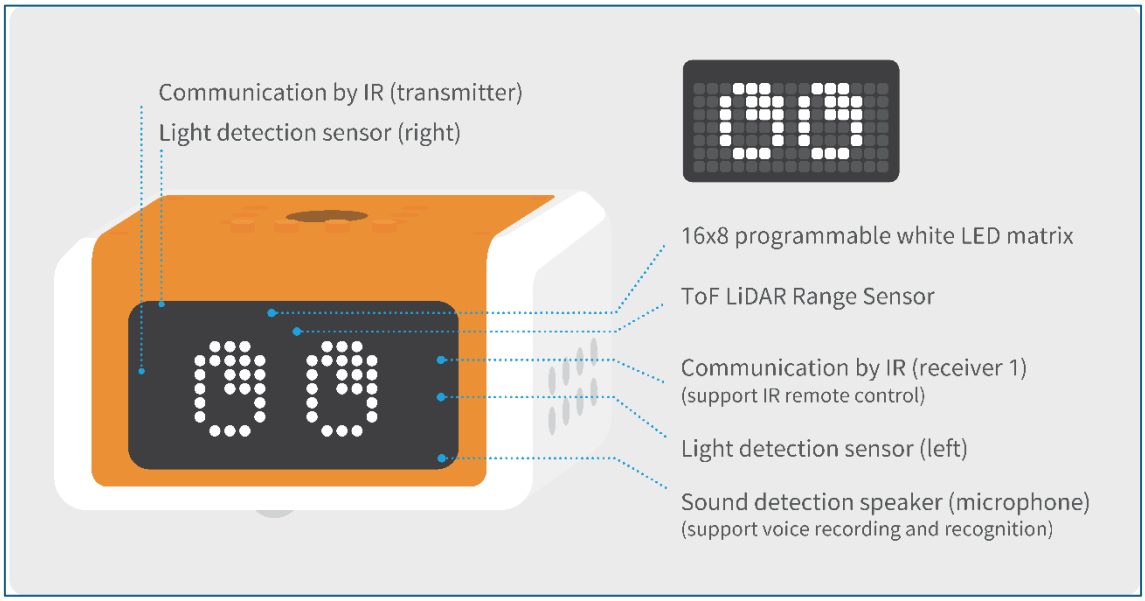
- Sensor de color (sensor número 3 del módulo siguelíneas)

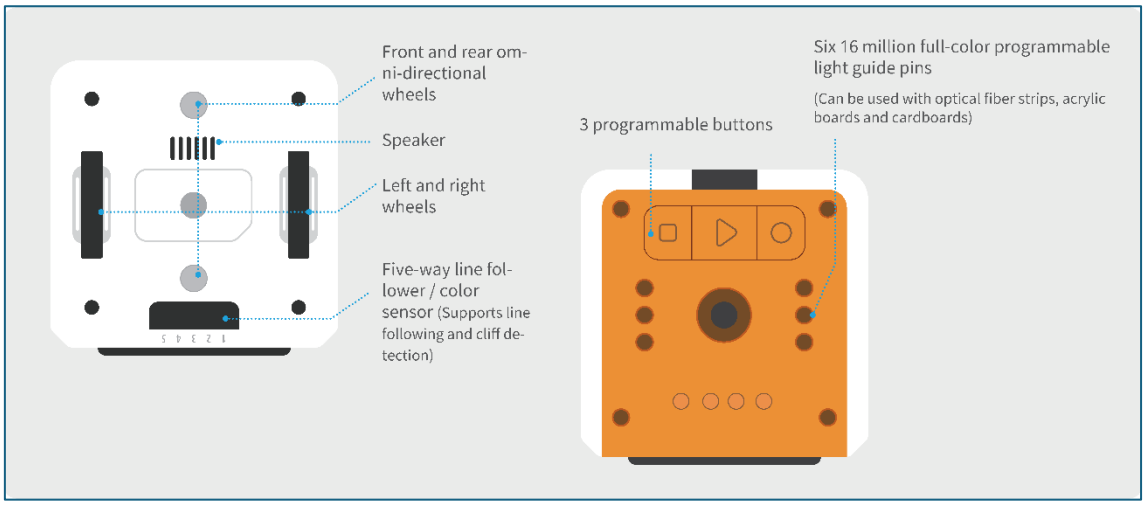


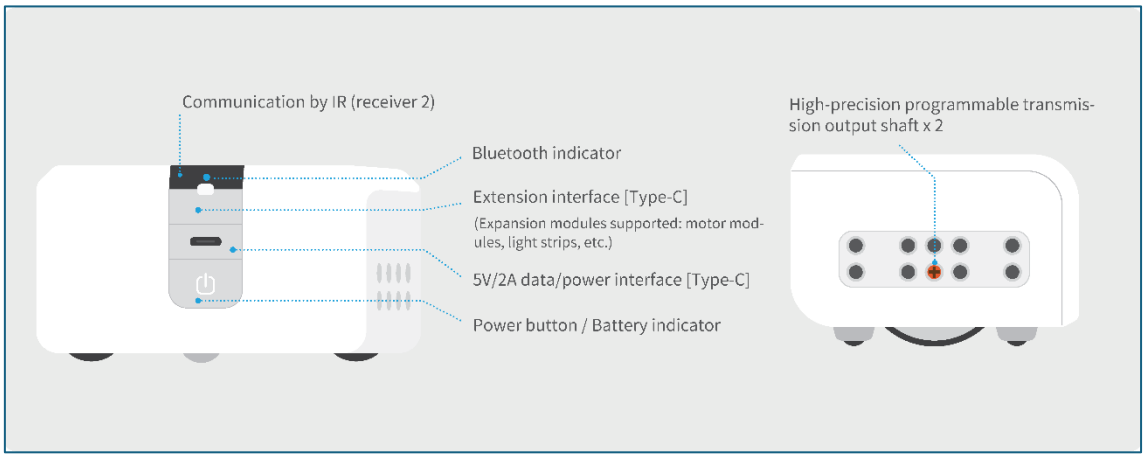
- Tres botones programables (cuadrado, triángulo y círculo)
- Sensor de alcance/obstáculos ToF LiDAR (Visión Artificial de profundidad con rango de medición 0.02~3 m)



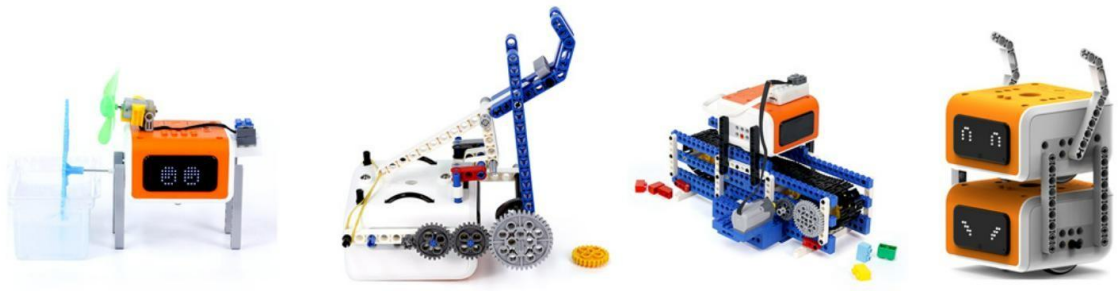
- Conexión inalámbrica Bluetooth y Wi-Fi ó vía cable USB-C
- Almacenamiento USB/USB DCD
- Batería de litio recargable de 1500 mAH (2 horas de tiempo de carga para más de 4 horas de uso).







VinciBot incluye 8 sensores, 2 motores de alta calidad y Visión Artificial para una diversidad de funciones orientadas a los proyectos STEAM. A mayores, puede comunicarse con otros VinciBots e incluso, montarse y acoplarse unos sobre otros.



VinciBot soporta **Tiny Machine Learning**. Esto nos permite trabajar el proceso de la inteligencia artificial, desde la creación del modelo, la adquisición de datos, el entrenamiento, la implementación y por último, la programación.

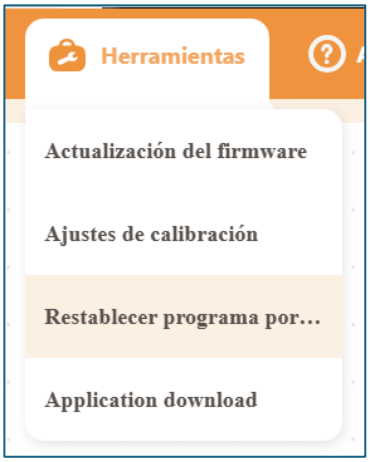
TinyML (Tiny Machine Learning) es una disciplina emergente que permite implementar el **aprendizaje automático** (M learning) en dispositivos con recursos limitados, que presentan restricciones en **memoria, energía y capacidad de procesamiento**. Su principal ventaja es que pueden operar sin conexión a la nube, mejorando la **privacidad** y reduciendo la **latencia**. Esto

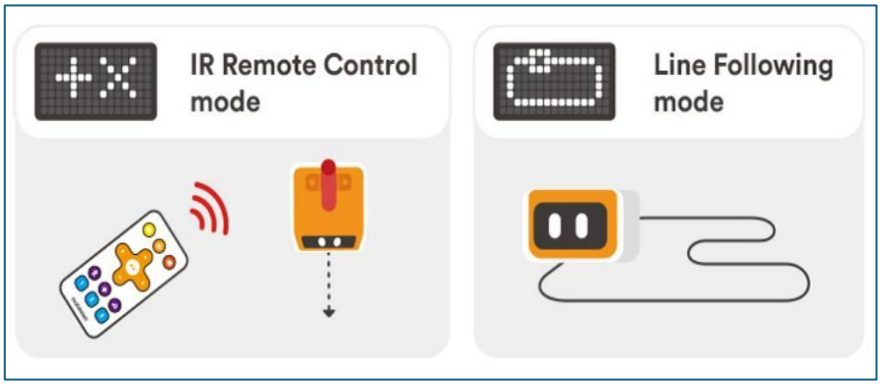
hace que podamos almacenar los datos en el propio robot y hacer pruebas sin tener conexión a Internet.

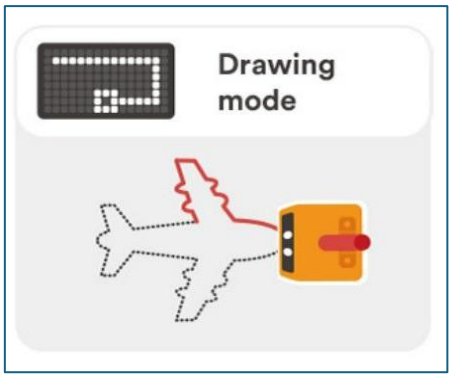
2. Modos de funcionamiento preconfigurados

El robot dispone de **tres modos de uso preconfigurados**: Por mando a distancia IR, seguidor de líneas y dibujo. Se accederá a cada modo a través del mando a distancia, pulsando la techa  seguida de uno de los números 1, 2 o 3 del mando.

Obviamente, el robot debe tener instalado el programa por defecto y esto puede hacerse desde el software Matata Studio (https://vinci.matatastudio.com/) en la opción Herramientas > Restablecer programas por defecto:







Modo Mando IR

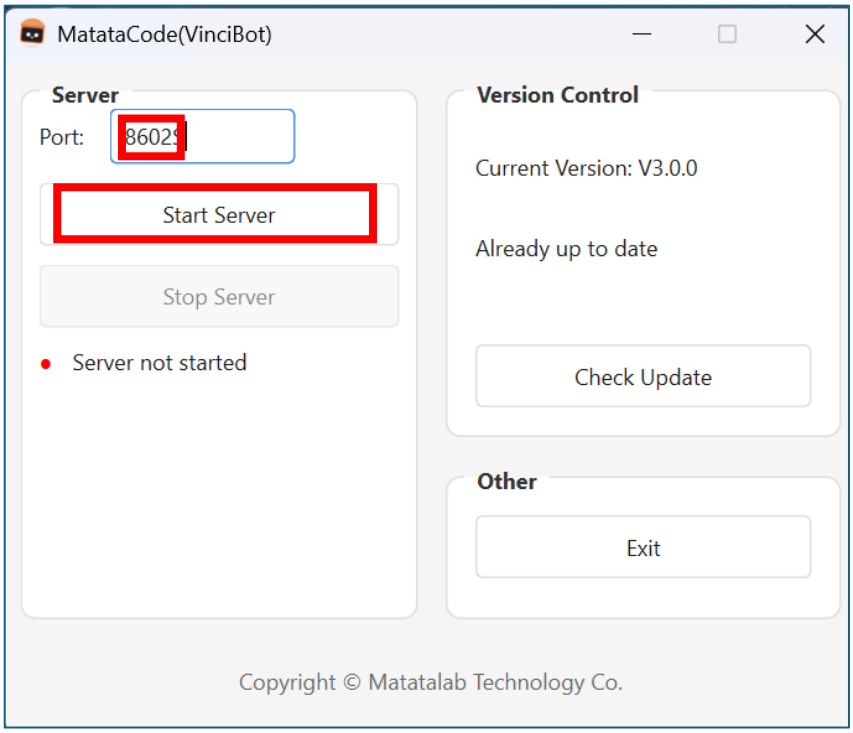
Tras escoger el modo 2 o 3, debemos pulsar la techa de inicio/parada del mando para comenzar o parar ese modo de funcionamiento.

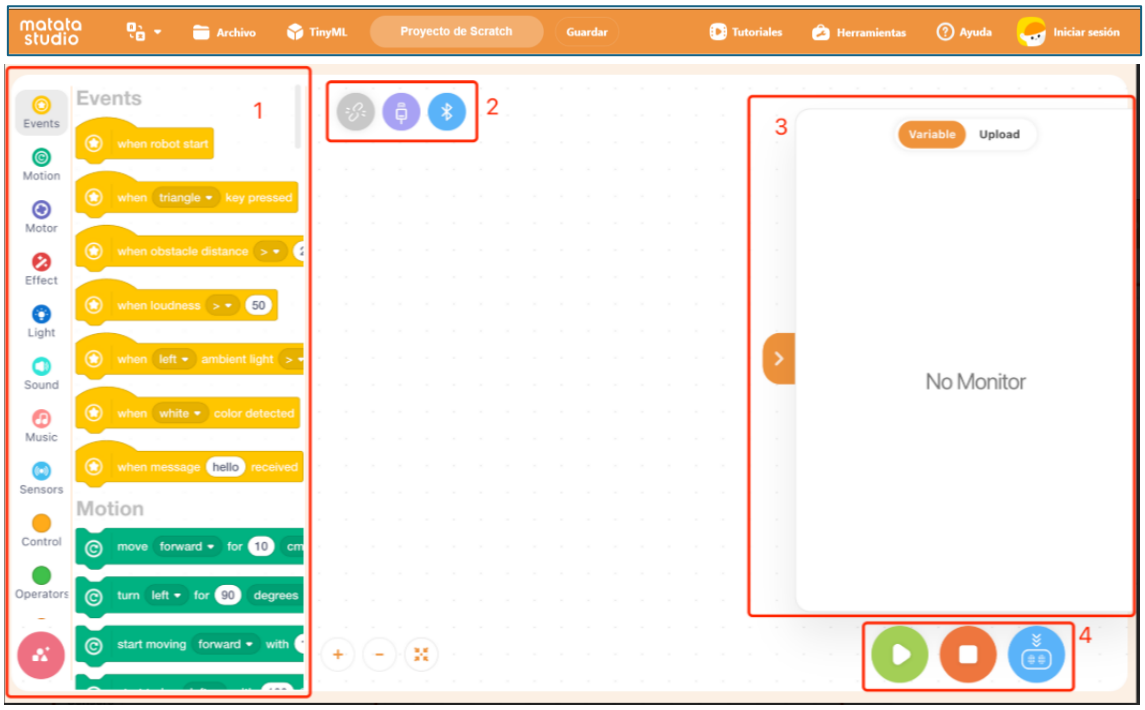
Modo seguidor de línea

Modo dibujar

3. Software Matata Studio

Para programarlo utilizamos el software Matata Code Studio, ya sea en su versión online (https://vinci.matatastudio.com/) u offline. La versión offline realiza una conexión al puerto 8602 por localhost:

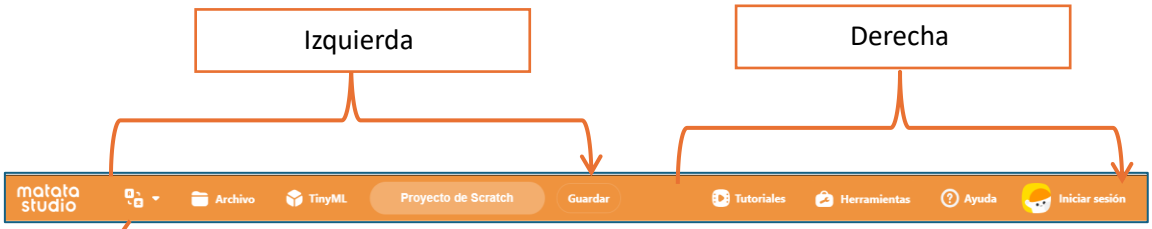




En la imagen observamos cuatro zonas diferenciadas:

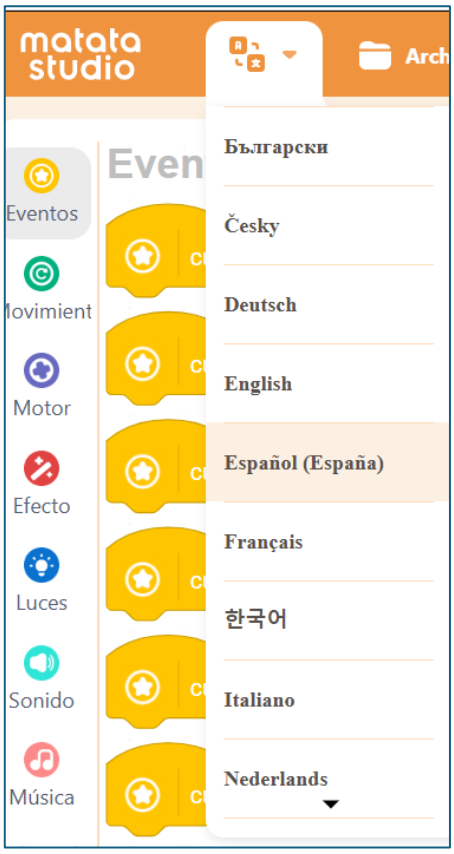
1. Zona de bloques de programación, con todos sus comandos
2. Zona de modos de conexión con el robot: Buetooth o USB
3. Monitor/extensión de variables/Python: Puede visualizar los valores de las variables y descargar el firmware de la extensión necesaria para la programación en esta área. También podemos cambiar de bloques a código Python.
4. Área de ejecución del programa: podemos elegir ejecutar el programa en línea (botón verde), detener la ejecución del programa en línea (botón rojo) o cargar el programa en el dispositivo local (botón azul).

El menú superior dispone de las siguientes opciones, que separaré entre izquierda y derecha:

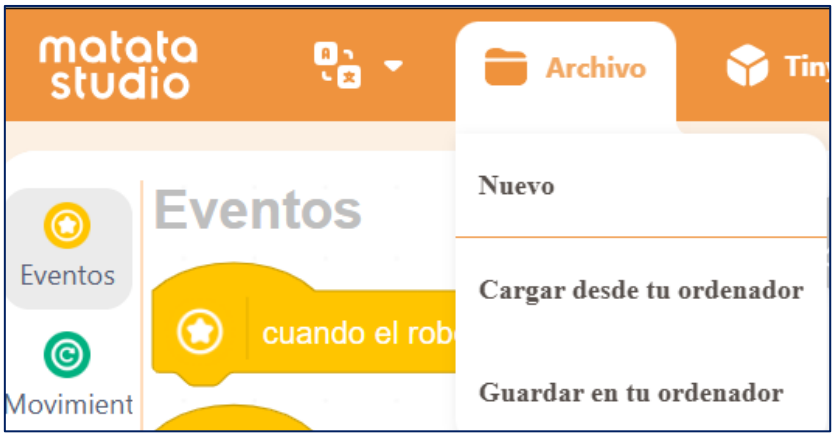


Comenzando con el menú izquierdo, nos encontramos, por orden, las siguientes opciones:

Idiomas:



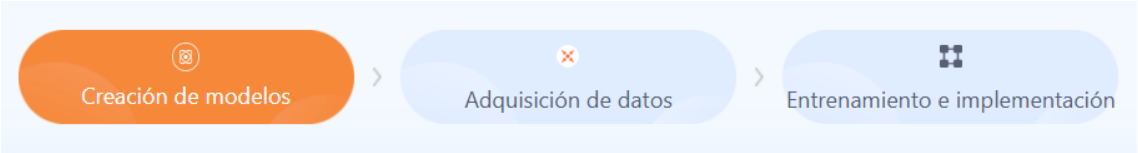
Archivo: Cargar o guardar un programa.



Tiny ML: Necesitamos estar logueados en el software y esto podemos hacerlo cubriendo los datos del icono Iniciar Sesión del menú superior derecho 

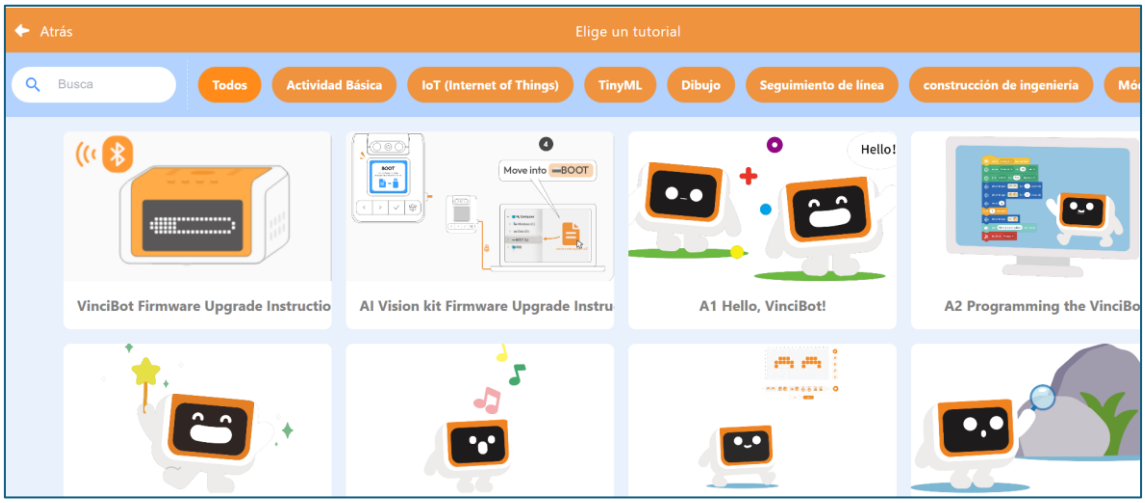
Tras iniciar sesión, podemos acceder a la opción de IA Tiny ML. Podemos enseñar al robot a reconocer imágenes, reconocer sonidos y reconocer lo que la casa comercial denomina objetos IoF.

Las fases son las típicas: Escogemos el tipo de modelo, introducimos los datos y lo entrenamos y comprobamos su funcionamiento.



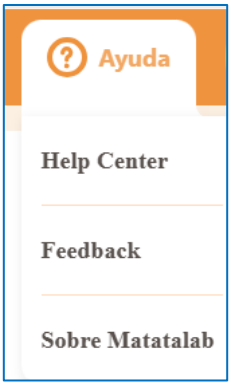
En el menú superior derecho nos encontramos con las siguientes opciones:

Tutoriales: son muy buenos para aprender

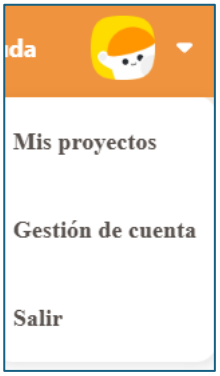


Herramientas: Para actualizar el firmware, calibrar los sensores de línea, restablecer el programa de fábrica o para acceder al lugar de descarga de la aplicación para tablet o móvil.

Ayuda: Nos redirige al centro de ayuda de Matatalab y links de la casa Matatalab.



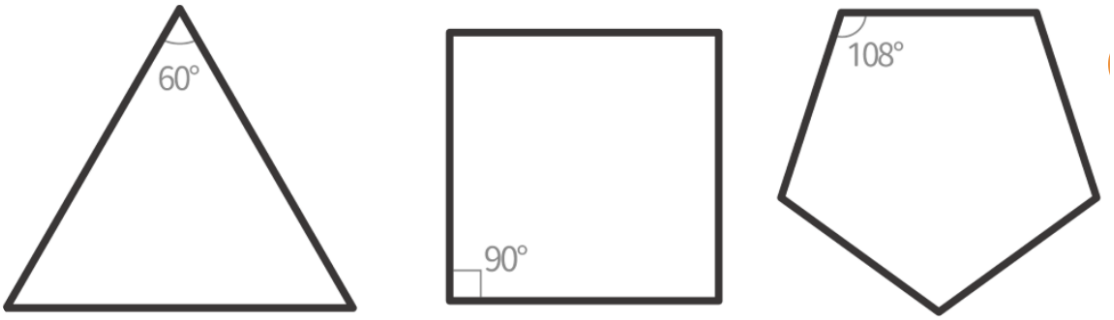
Iniciar sesión: Indispensable para trabajar con IA. Tras disponer de un usuario podemos guardar los programas en la nube de Matatalab



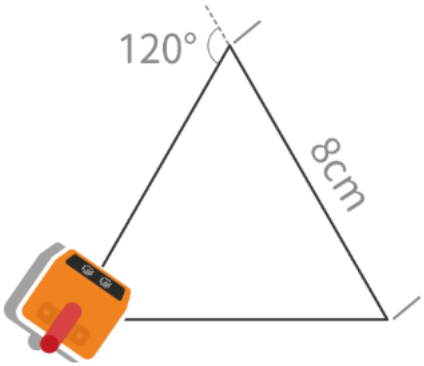
4. Ejemplos de programación

4.1. Dibujando polígonos regulares

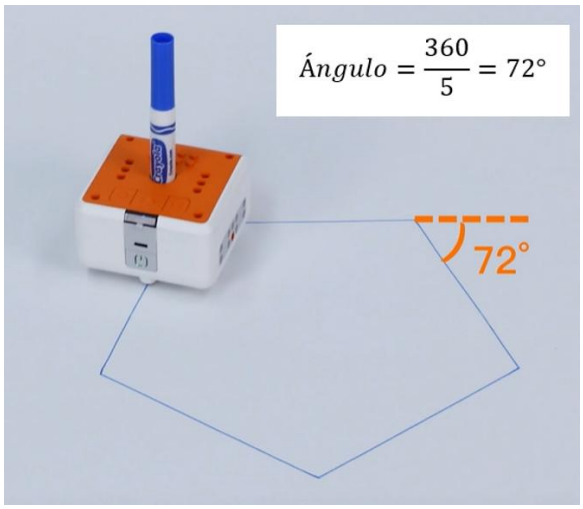
Podemos establecer una fórmula genérica que determine el ángulo de giro necesario a la hora de pintar polígonos regulares.



Si conocemos el ángulo interno, podemos descubrir el externo, que es el que realmente necesita el robot para crear la figura:



$$Ángulo = \frac{360}{3} = 120°$$



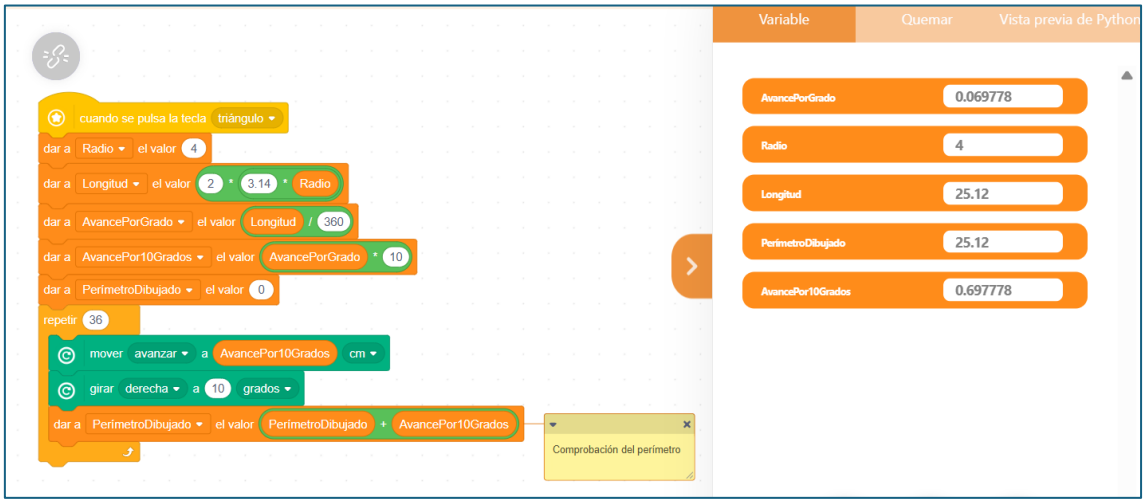
Por lo tanto, la fórmula que debemos programar es la siguiente:

$$Ángulo\ de\ giro = \frac{360}{número\ de\ lados\ del\ polígono}$$

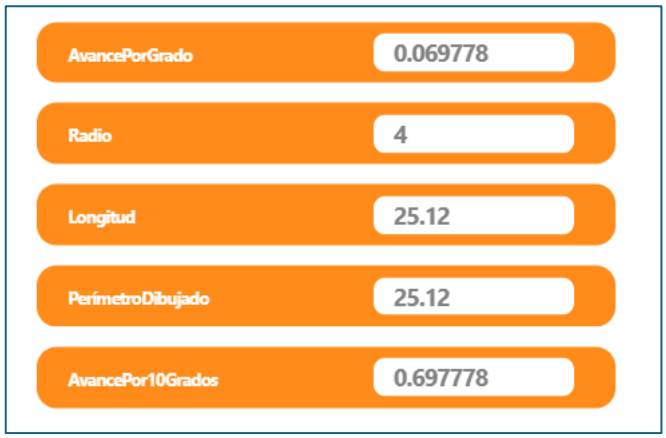
De forma genérica, podemos crear un bloque que dependa de una variable que llamaremos "NumeroLados". En el siguiente script se dibujará un pentágono regular (NumeroLados=5) de lado 6cm. El avance y giro del robot dependerá del número de lados que deba pintar. Por este motivo incluimos un bloque de repetición que dependerá de la variable "NumeroLados".

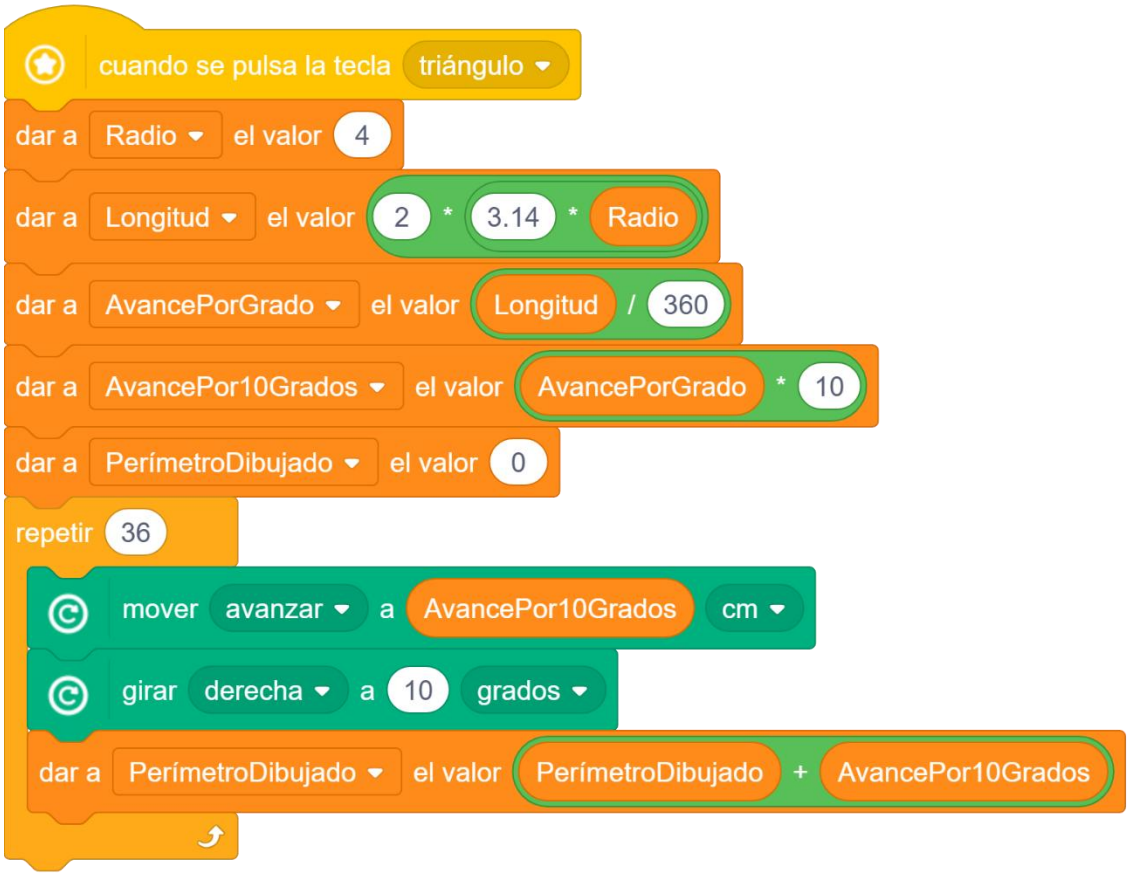
Podemos mejorarlo incluyendo en el bloque la medida del lado del polígono regular:

¿Cómo dibujaríamos un círculo sin que pareciera un polígono de 360 lados?



Como el avance que debe realizar el robot por grado es un valor numérico muy pequeño, se le ha dicho al robot que avance gire 10 grados en cada avance. Por ese motivo, el bucle o repetición se ejecutará 36 veces.

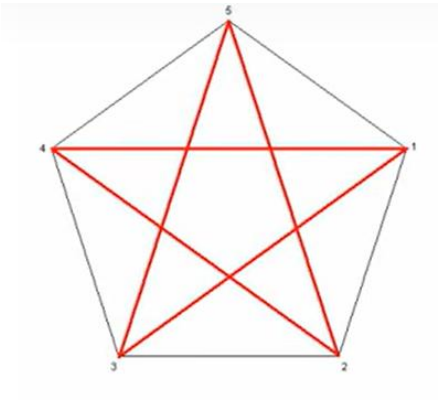




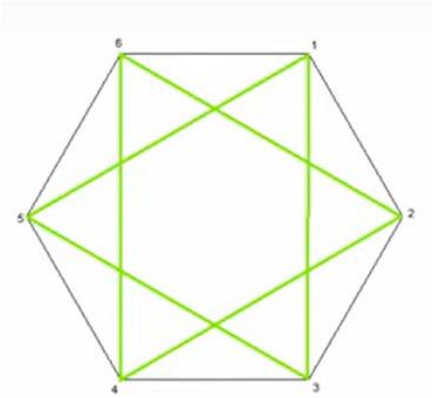
4.2. Dibujando estrellas

Las estrellas dependen de dos parámetros o valores: el número de puntas que tiene la estrella y el salto que se hace (dentro de un polígono) a la hora de pintarlas.

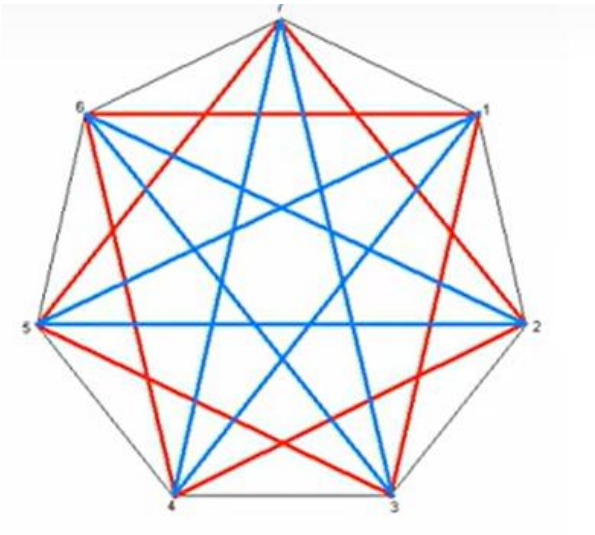
Por ejemplo, si queremos pintar una estrella de 5 puntas, partiremos de un pentágono regular. Si queremos que el salto sea 2, saltaremos un vértice del pentágono a la hora de pintar la estrella. Por lo tanto, la ruta de nuestro dibujo sería la siguiente: 1-3-5-2-4-1



Si queremos dibujar una estrella se 6 puntas y salto 2, nuestra figura seguiría la siguiente ruta: 1-3-5-1 y 2-4-6-2



En la siguiente imagen vemos una estrella de 7 punto y salto 2 en color rojo, así como, una estrella de 7 puntas y salto 3 en azul:



Observamos que existe una relación entre el número de puntas de la estrella y los saltos:

$$1 < salto < \frac{número\ de\ puntas}{2}$$

Por este motivo, una estrella de 7 puntas (7/2=3.5) puede tener salto 2 o 3. Es decir, no se puede hacer una estrella regular de 7 puntas y salto 4.

Además, observamos que el ángulo de giro que debe realizar el robot es:

$$Ángulo\ de\ giro = \frac{360° \cdot salto}{número\ de\ puntas}$$

A la hora de programar el dibujo de una estrella, podemos crear un bloque cuyas variables sean "NumeroPuntas" y "Salto", de la siguiente forma:

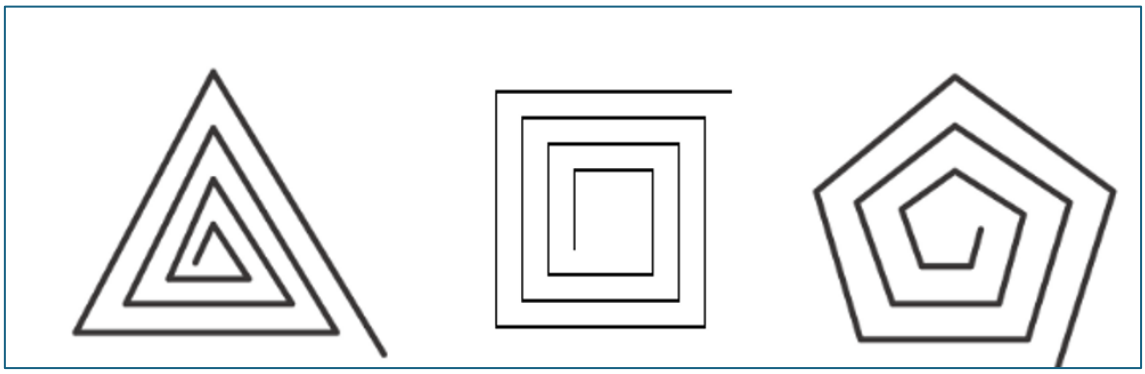
Usando este bloque, si deseamos dibujar una estrella de 5 puntas y salto 2, añadiríamos este pequeño script:

Para una estrella de 7 lados y salto 3:

Para una estrella de 8 lados y salto 3:

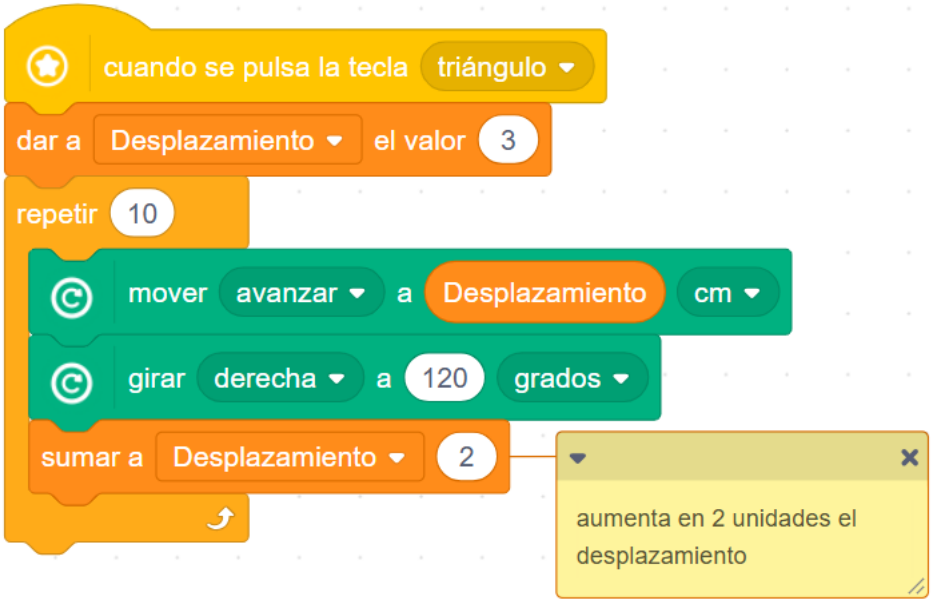
4.3. Dibujando trayectorias espirales

Denominamos trayectorias espirales a figuras como las que se pueden ver en la siguiente imagen:

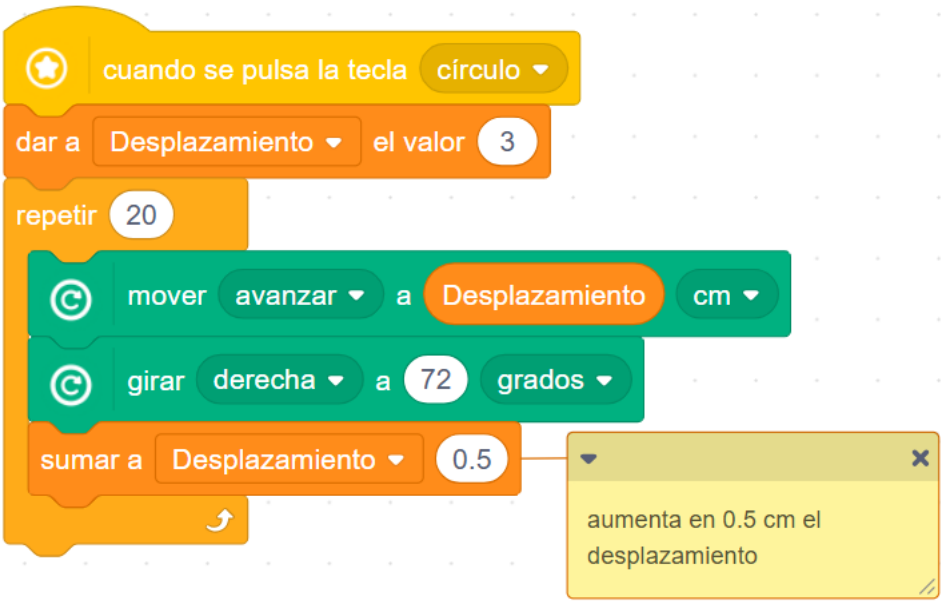


Comencemos con una trayectoria espiral cuadrada. Cada 2 movimientos, aumentamos el desplazamiento en 2cm siendo el ángulo de giro 360°/4=90°.

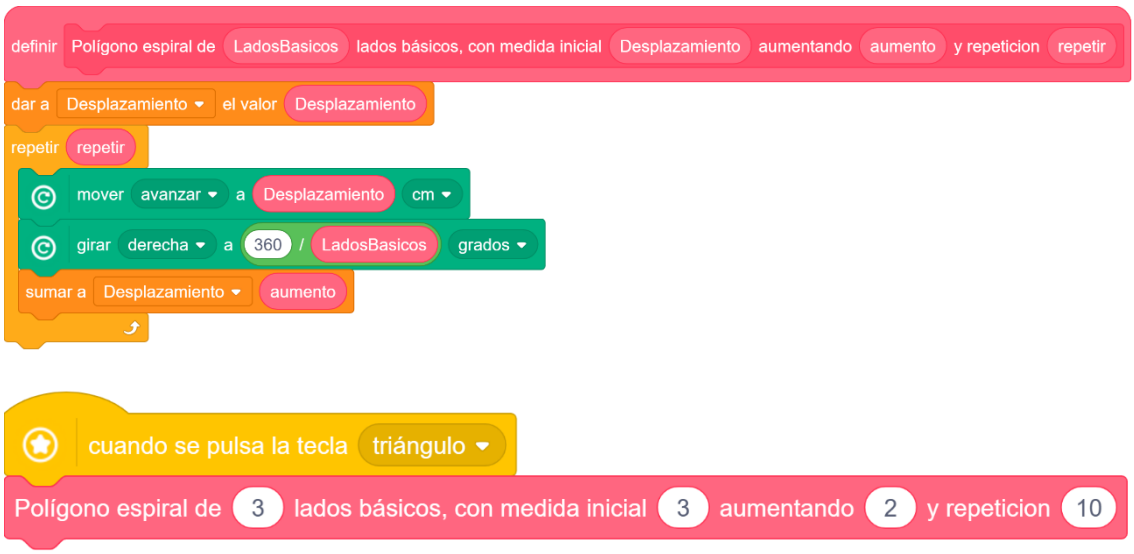
El siguiente script muestra la programación de una trayectoria espiral triangular. En cada desplazamiento se aumenta la longitud del lado en 2cm siendo el ángulo de giro 360°/3=120°.



El siguiente script muestra la programación de una trayectoria espiral pentagonal. En cada desplazamiento se aumenta la longitud del lado en 2cm siendo el ángulo de giro 360°/5=72°.



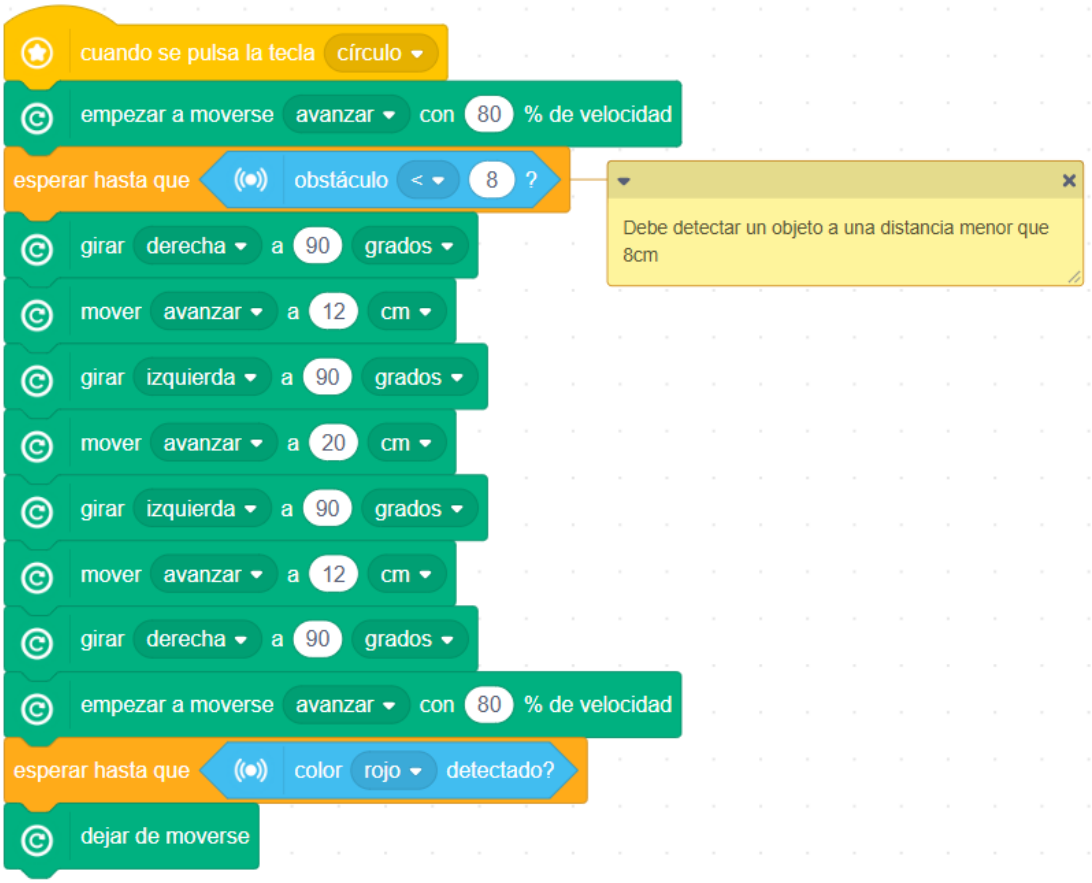
Podemos generalizar para todos una función que dependa del ángulo de giro (unido al número de lados de la figura básica), del número de lado básico, de las repeticiones, de la distancia inicial y del aumento:



4.4. Robot que evita obstáculos

Supongamos el siguiente circuito (ver imagen): el robot sale del punto 1, esquiva el objeto y se para cuando detecta el color rojo en el suelo.

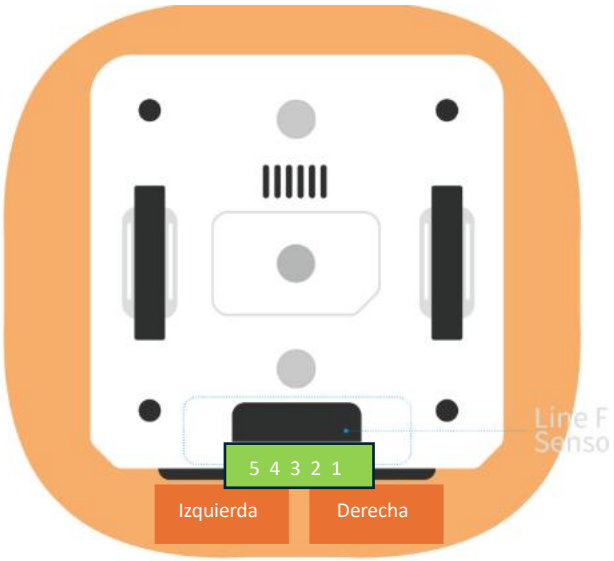
Un programa que cumple con las necesidades podría ser el siguiente:



4.5. Seguidor de línea

El robot VinciBot dispone de cinco sensores de seguimiento de línea debajo del robot, de los cuales, los números 1, 2, 4 y 5 son sensores en escala de grises, y el número 3 es un sensor de color o RGB. Estos 5 sensores se pueden calibrar desde el software (Herramientas > Ajustes de calibración)

Los cinco pueden detectar la intensidad de la luz reflejada en colores blanco y negro.



Caso 1: Línea negra ancha (sensores de línea 1 y 5)

Si el circuito de línea (negra o blanca) es una línea ancha, se pueden usar los sensores de seguimiento de línea números 1 y 5 para lograr el seguimiento de esa línea.

Usando los sensores seguidores de línea números 1 y 5, se pueden dar cuatro situaciones diferentes con VinciBot moviéndose a lo largo de la línea. A saber, el robot completamente en la línea (que es la situación ideal), completamente fuera de la línea, inclinado a la izquierda de la líneao inclinado a la derecha de la línea.

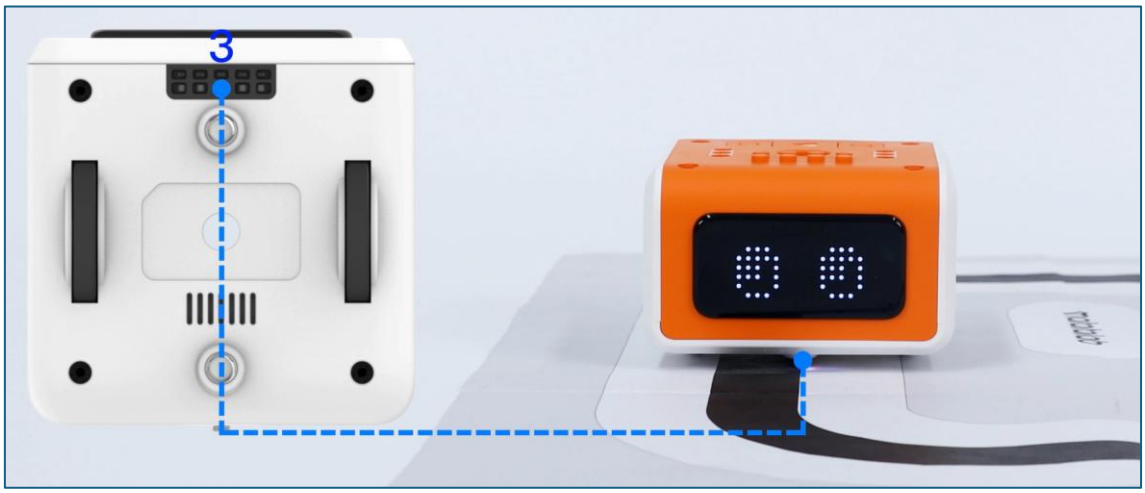
Si comprobáramos los valores de luz reflejada correspondientes a los sensores de seguimiento de línea 1 y 5 en esas cuatro situaciones, veríamos que, según el principio de reflexión de la luz, el negro absorberá una fuente de luz, por lo que el valor de luz reflejada será relativamente bajo cuando los sensores vean negro; por el contrario, el valor de luz reflejada blanca será alta cuando los sensores vean blanco. En la siguiente tabla se resumen todas las opciones:

Luz Reflejada/Casos	Completamente en la línea	Completamente fuera de la línea	Inclinado a la izquierda	Inclinado a la derecha
Sensor izquierdo (5)	< 40	> 80	Izq > Der (1<5)	Der > Izq (1>5)
Sensor derecho (1)	< 40	> 80		
Acción que debe realizar el robot	Avanzar recto	Ir hacia atrás	Girar a la derecha	Girar a la izquierda

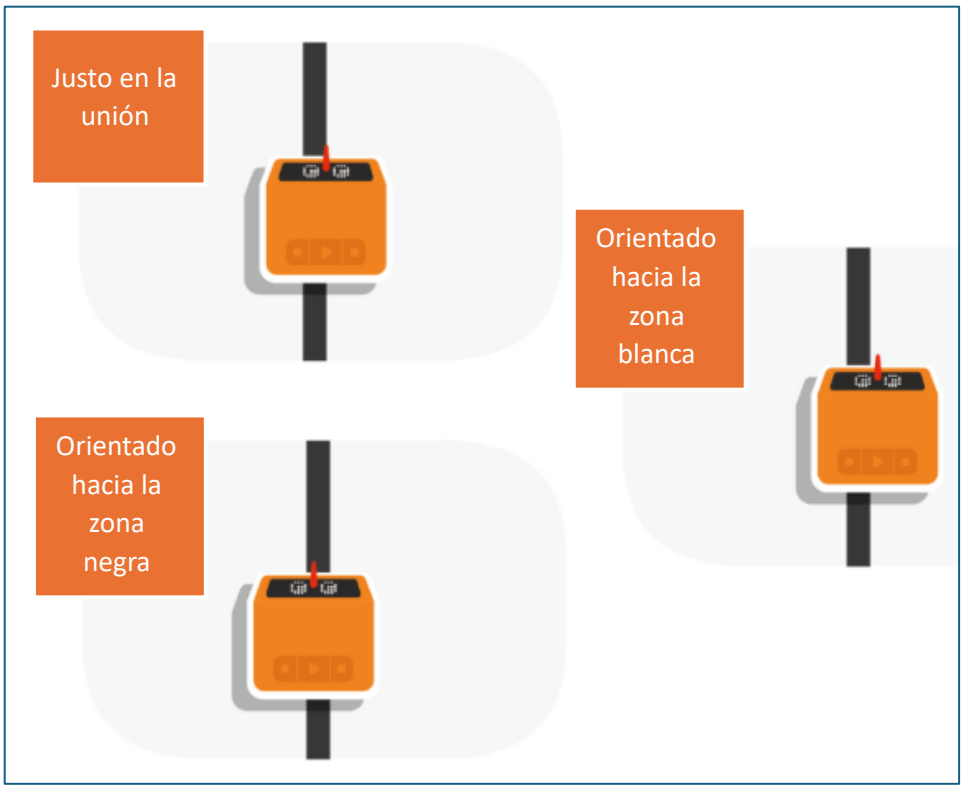
Ojo: El motor izquierdo avanza con un valor numérico negativo, ya que está en oposición de fase con el motor derecho, que avanza con un valor positivo.

Caso 2: Sensor de línea RGB (número 3)

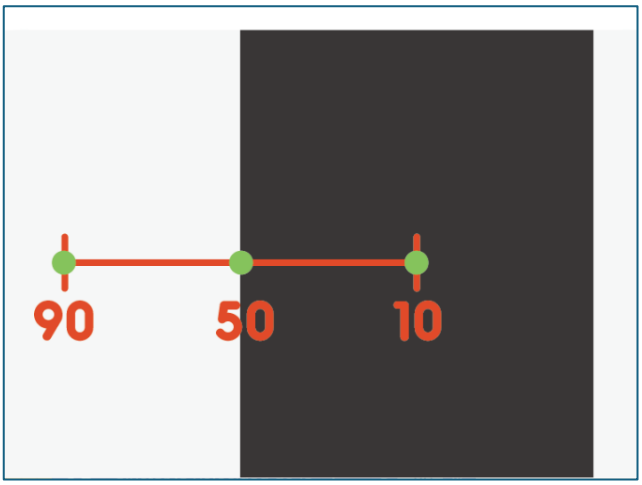
Vamos a hacer un programa seguidor de línea usando el sensor de línea RGB, que es el sensor siguelíneas número 3. A diferencia del programa de seguimiento de línea del caso 1, que utiliza los sensores números 1 y 5 (que hacen que VinciBot recorra la línea central), al usar el sensor de color para recorrer la línea, VinciBot recorre la intersección de los colores blanco y negro.



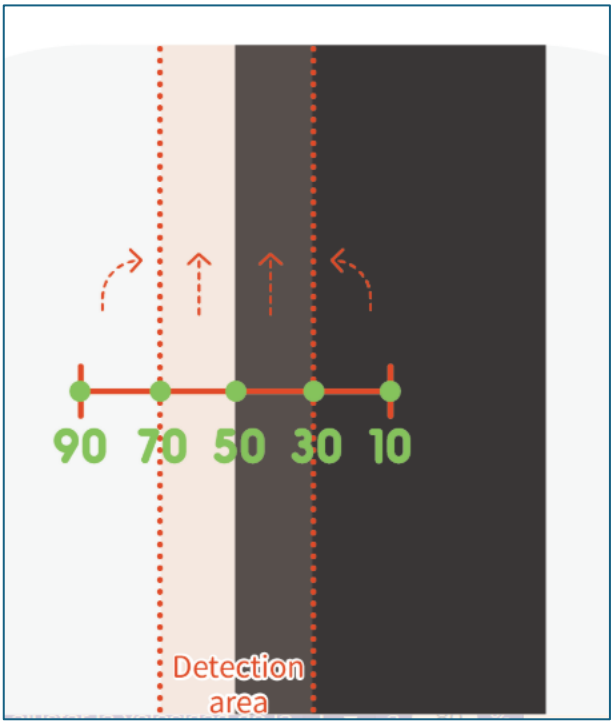
En este caso se pueden dar 3 situaciones lo largo de la línea de unión: justo en la unión de negro y blanco; que esté orientado hacia la parte blanca; o que esté orientado hacia la parte negra.



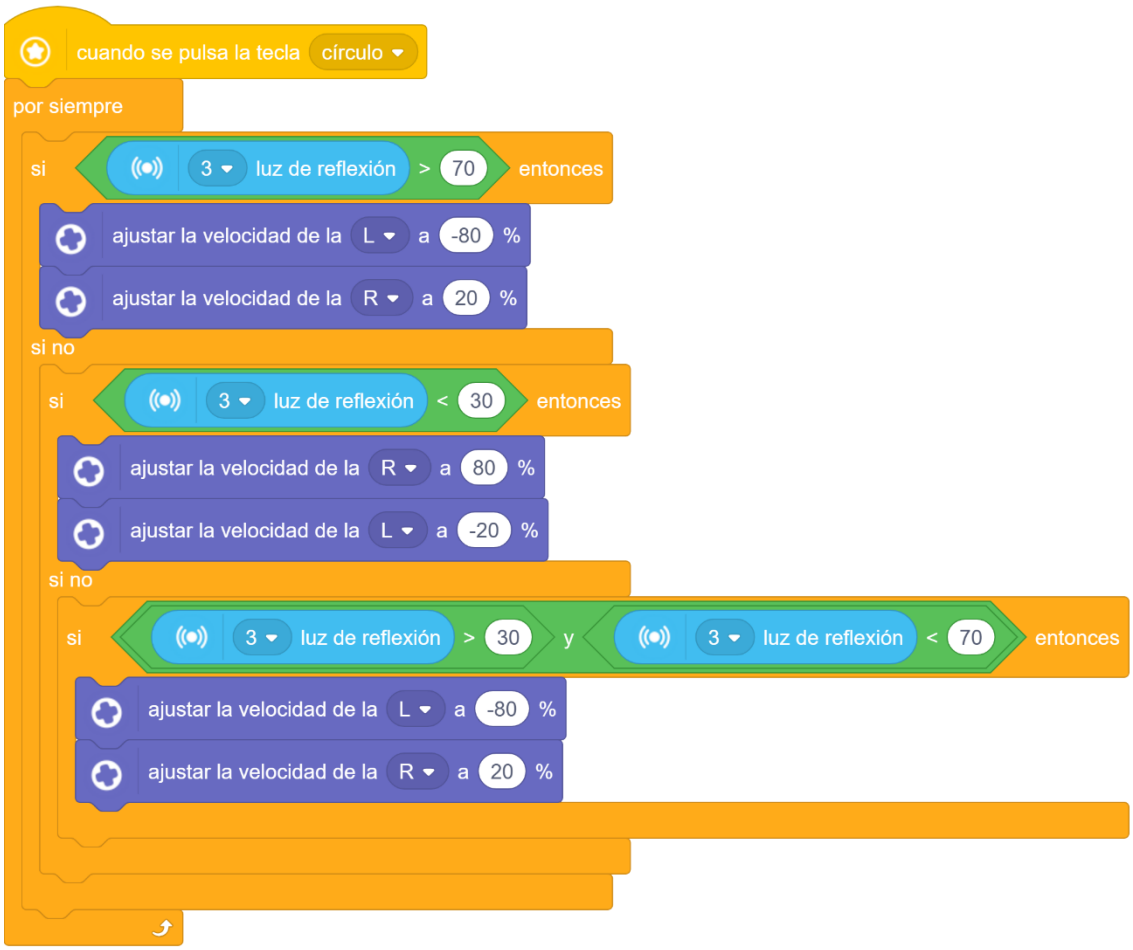
Podemos comprobar que el valor de la luz reflejada en la línea negra es de aproximadamente 10, mientras que el valor en la línea blanca es de aproximadamente 90. Por lo tanto, se puede calcular que el valor en la unión del negro y el blanco es de aproximadamente (10+90)/2=50.



Es más, la línea negra la podemos dividir en cuatro partes (como se muestra a continuación). En este caso, VinciBot debería avanzar recto en la zona que se representa como área de detección y, si está fuera de esa área, el robot debería girar a la izquierda o a la derecha.

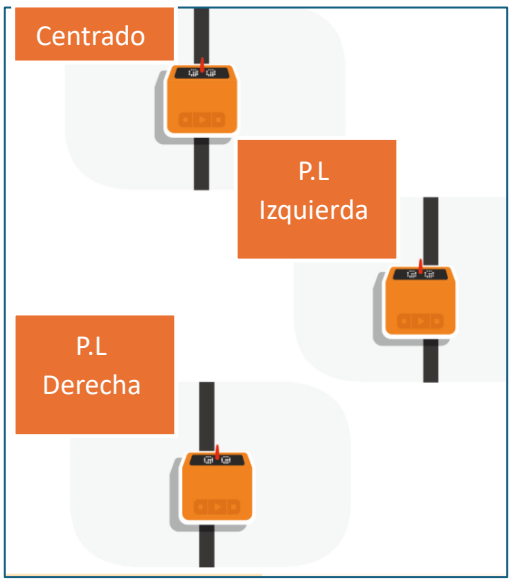


Cuando VinciBot se desvía hacia la zona blanca (es decir, cuando el valor de la luz reflejada es mayor que 70), debe girar a la derecha para retomar la línea. Por el contrario, cuando VinciBot se desvía hacia la zona negra (es decir, cuando el valor de la luz reflejada es menor que 30), debe girar a la izquierda. Pero mientras se mueva entre las dos zonas centrales (es decir, cuando el valor de la luz reflejada es mayor que 30 y menor que 70 - área de detección -), el robot debe avanzar en línea recta. Un script que ejecuta estas acciones es el siguiente:

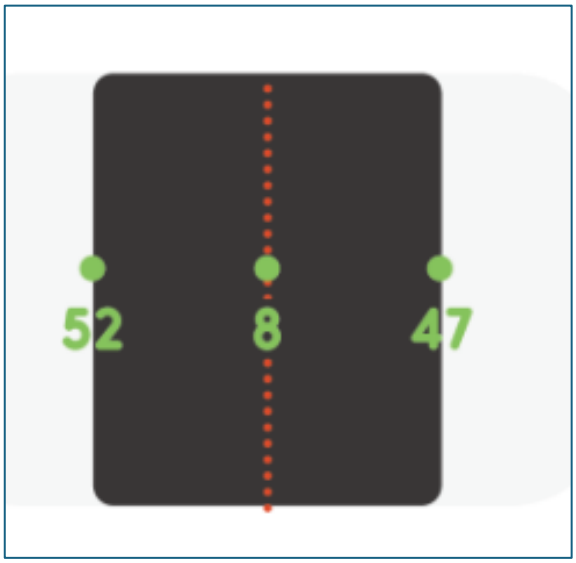


Caso 3: Línea estrecha. Sensores de línea centrales (2, 3 y 4)

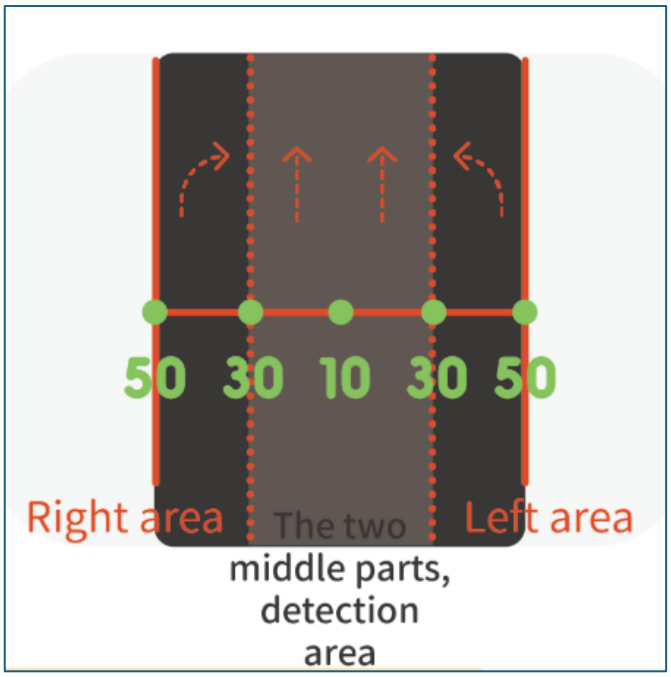
En este caso usaremos los sensores de línea números 2, 3 y 4. A la hora de seguir la línea, podemos encontrarnos con cualquiera de estas tres situaciones: Que la línea la siga perfectamente (estando ella en el sensor 3), que pierda la línea (P.L) por la izquierda o que la pierda por la derecha.



Podemos determinar que los valores de los tres sensores son muy similares. Si tomamos el sensor número 3 como ejemplo: el valor de la luz reflejada es de aproximadamente 10 cuando VinciBot está en el medio de la línea negra, y el valor de la luz reflejada es de aproximadamente 50 cuando las uniones de blanco y negro están en los lados izquierdo y derecho.



Esa línea negra podemos dividirla en más partes. Por ejemplo, en cuatro partes, como se muestra a continuación. VinciBot tendrá la situación ideal de seguir la línea en las dos áreas centrales:

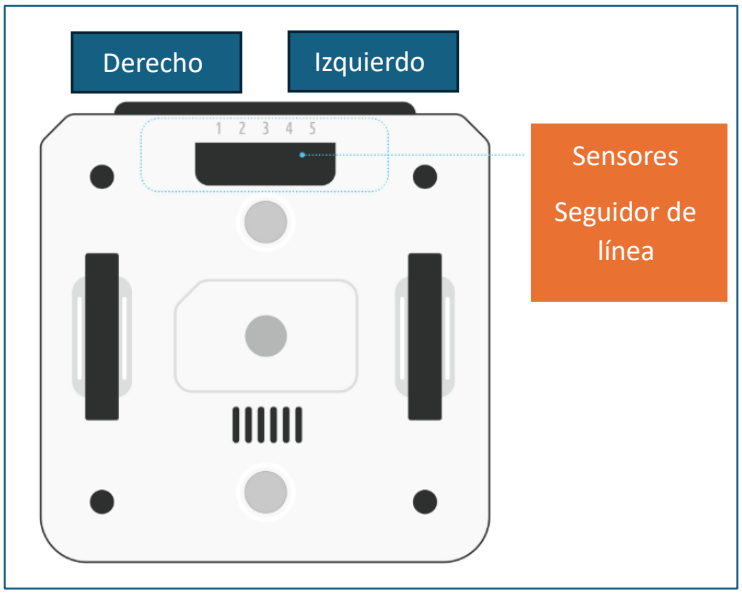


Si se desvía hacia el área izquierda o derecha, girará a la derecha o izquierda, respectivamente, para enderezarse: el valor de la luz reflejada del área de cálculo redondeada es de aproximadamente 10-30. Cuando la luz reflejada del sensor número 3 es inferior a 30, el robot debe avanzar. Pero cuando el sensor número 2 obtiene un valor inferior a 30, VinciBot se desvía

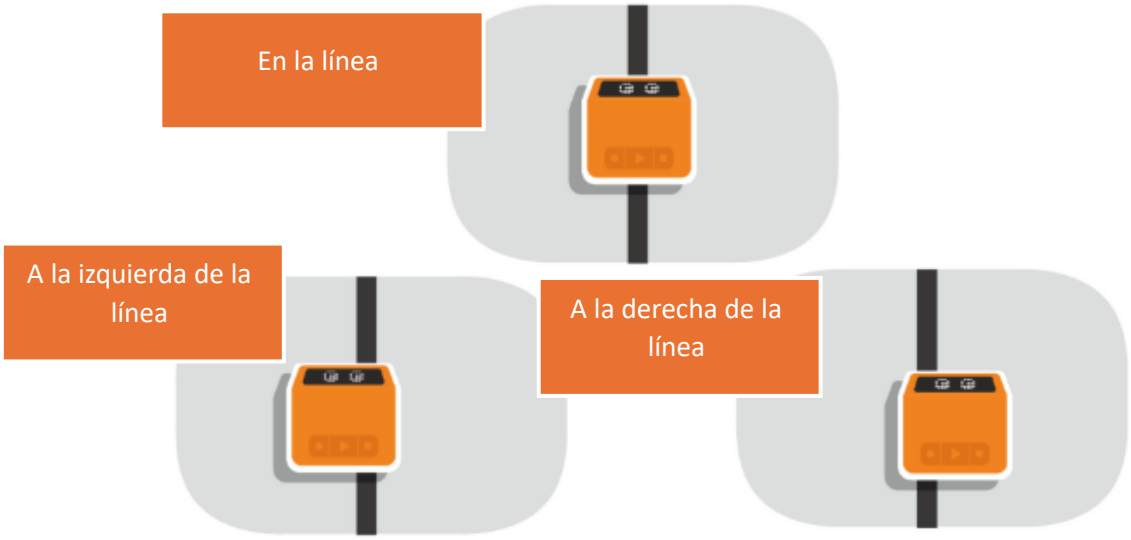
hacia la izquierda y necesita girar a la derecha. Y cuando el sensor 4 obtiene un valor inferior a 30, el robot se desvía hacia la derecha y necesita girar a la izquierda.

Por lo tanto, un posible script de este seguidor de línea sería el siguiente:

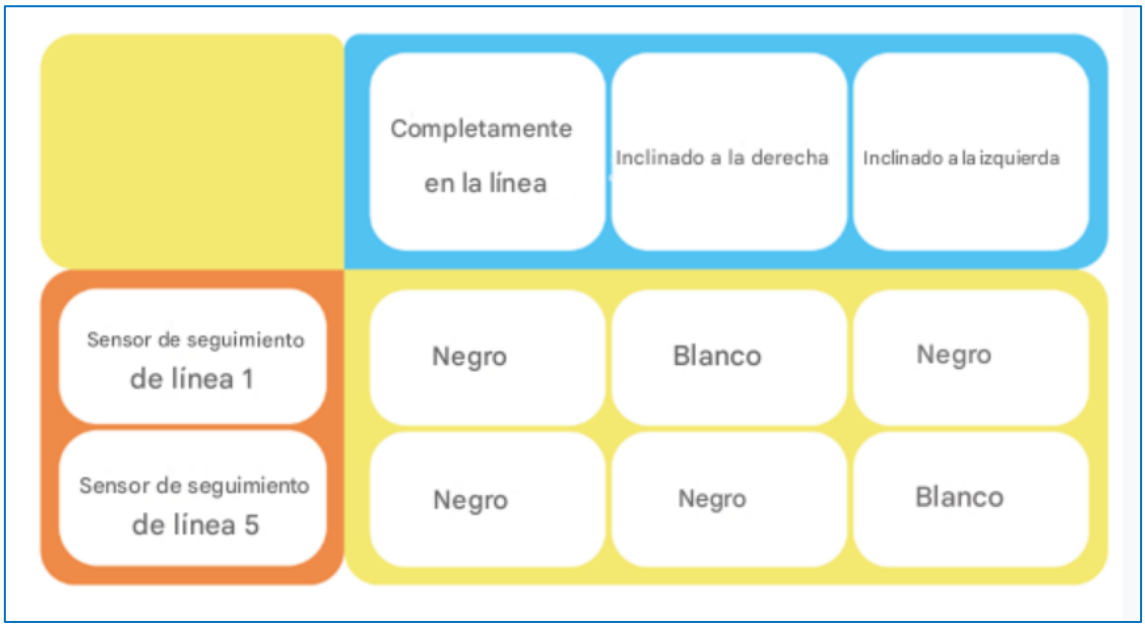
Como ya he indicado, el robot dispone de cinco sensores de seguimiento de línea situados debajo del robot, de los cuales 1, 2, 4 y 5 son sensores en escala de grises, mientras que el número 3 es un sensor de color RGB. Pero todos pueden detectar la intensidad de la luz reflejada en blanco o negro. Al moverse a lo largo de líneas más anchas, se pueden usar los sensores de seguimiento de línea números 1 y 5 para lograr un mejor seguimiento de línea.



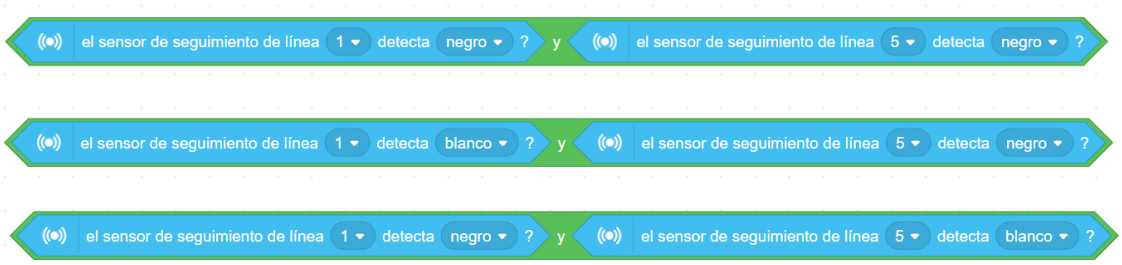
Analizando los sensores 1 y 5, encontramos tres situaciones en las que el robot puede seguir una línea: el robot se encuentra completamente en la línea, inclinado hacia la izquierda de la línea o inclinado hacia la derecha de la línea.



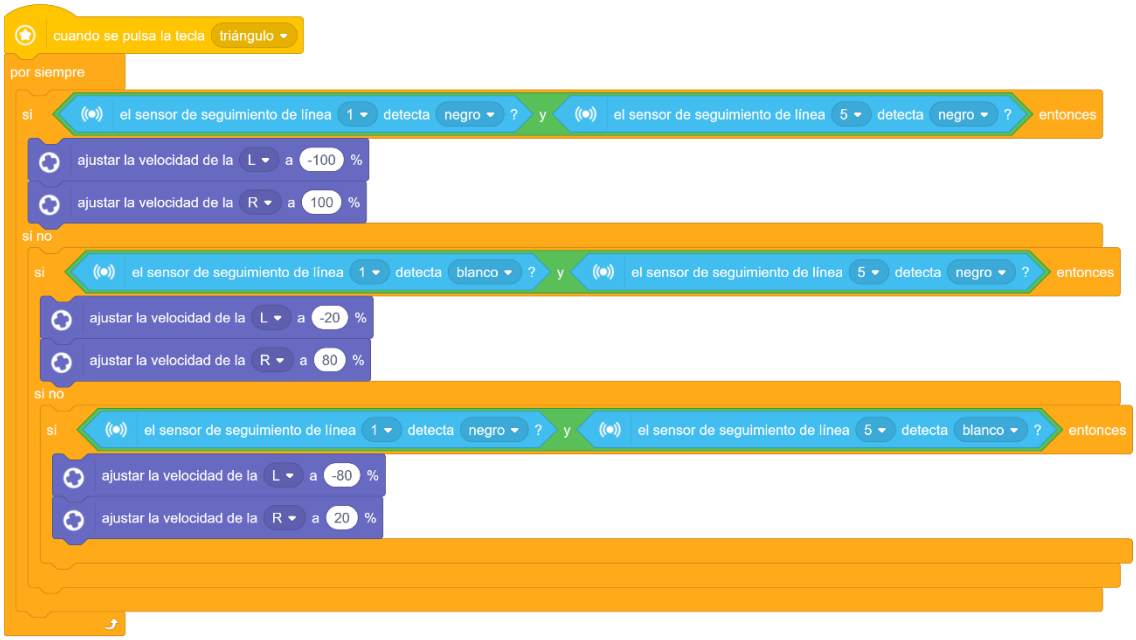
Podemos determinar una regla para los valores de luz reflejada a la izquierda y a la derecha en las tres situaciones.



Esas tres situaciones pueden verse representadas por estas condiciones:



Las acciones del robot para cada situación se perfilan en el siguiente script:



Si decidiéramos usar los sensores de línea números 2 y 4, el script sería el siguiente:

Notas: Cuando tenemos un circuito con una línea estrecha debemos usar los sensores de línea 2 y 4. El ancho de la línea debe ser la distancia entre los sensores de línea 2 y 4.

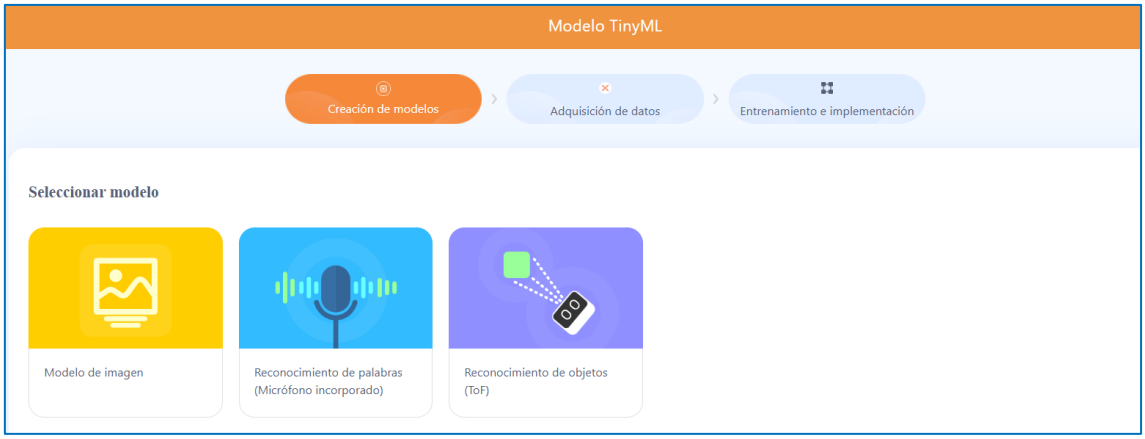
4.6. Reconocimiento de letras (Machine Learning)

Este robot dispone de la capacidad de aprender (Tiny Machine Learning) y lo realiza a través de cuatro pasos: creación del modelo, introducción de un conjunto de datos de entrenamiento, entrenar el modelo y testearlo y finalmente, programarlo.

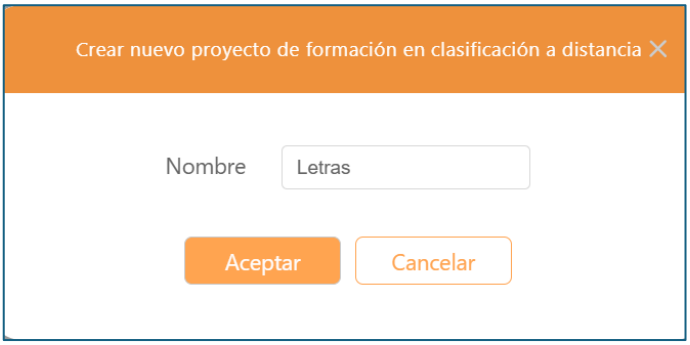
En este ejemplo vamos a intentar que el robot reconozca las letras B y W.

- Paso 1: creación del modelo

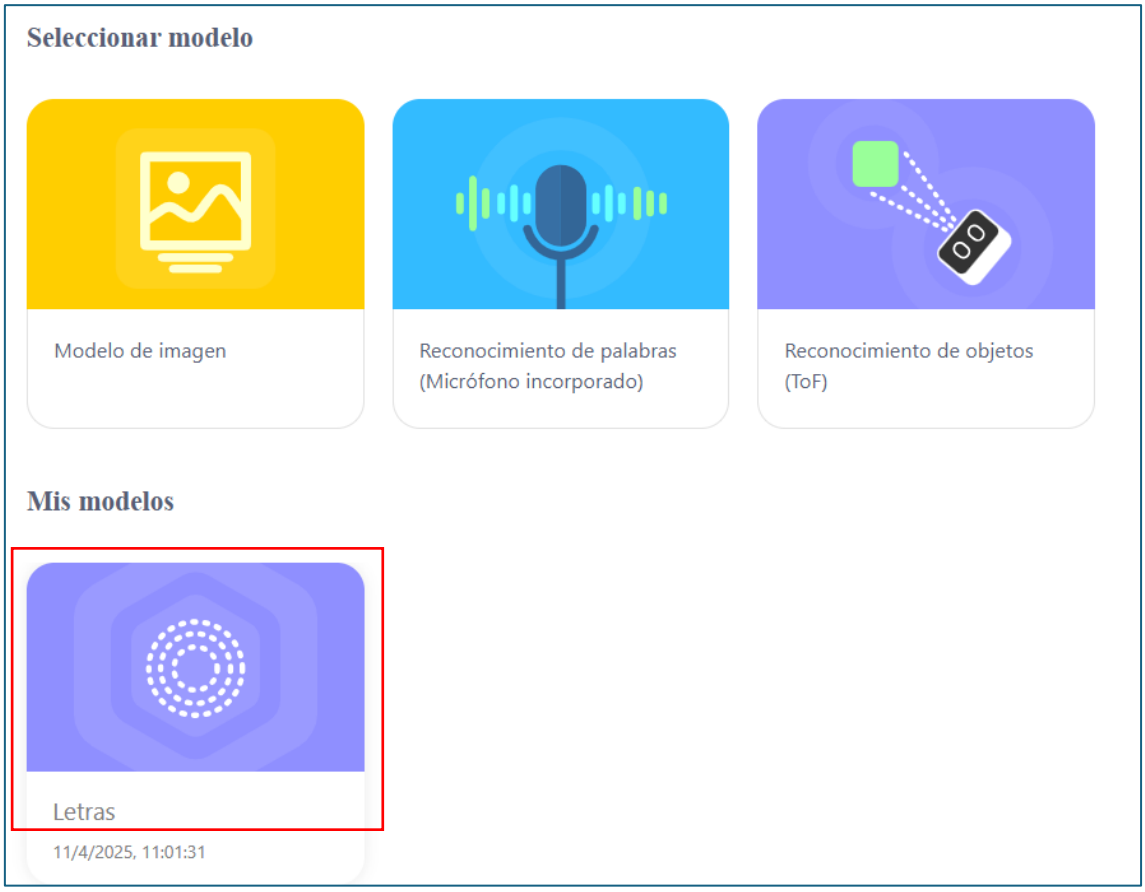
Dentro de la interfaz de programación, al seleccionar la opción “Tiny ML” TinyML podemos escoger entre varios modelos de aprendizaje disponibles: reconocimiento de imágenes, reconocimiento de palabras (audios) y reconocimiento de objetos (utilizando el sensor ToF del robot que determina distancias con gran precisión usando un haz de luz infrarroja).



Para nuestro ejemplo (reconocer las letras B y W), optamos por el modelo de reconocimiento de objetos (aprovechando el sensor ToF) y le asignamos al modelo el nombre "Letras".



Podemos ver el modelo recién creado ("Letras") en la sección "Mis modelos":



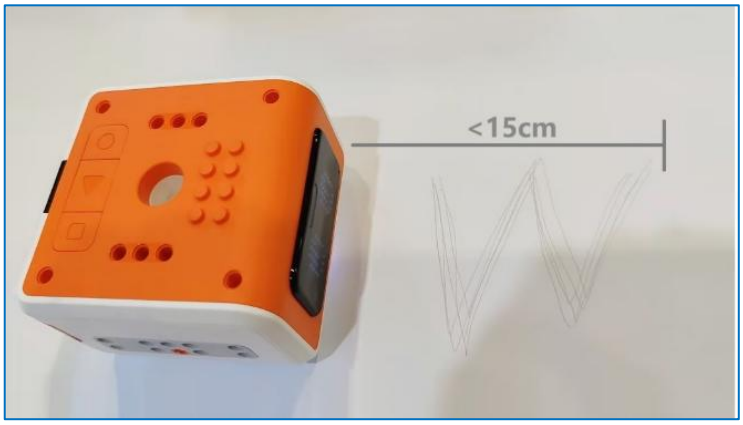
- Paso 2: Adquisición de datos.

No se puede entrenar un modelo de aprendizaje sin datos de los cuales aprenda. Para introducir los datos con los que aprenderá nuestro modelo, dentro del modelo "Letras" comenzamos seleccionando el recuadro "Editar".

Se nos abre una nueva pantalla en la cual comenzaremos a cubrir los nombres de las clases que vamos a crear y la duración o tiempo para la recopilación de datos. Debemos incluir, como mínimo, 4 datos en cada categoría o clase definida, pero, obviamente, cuantos más datos mejor. Crearemos tres categorías que se llamarán “B”, “W” y “Nada” y en cada una introducimos los datos.

¿Por qué podemos emplear el modelo de reconocimiento de objetos (sensor ToF integrado) para reconocer letras manuscritas? Esto es posible porque la distancia entre nuestra mano y el sensor ToF fluctúa en tiempo real mientras escribimos.

El método de recopilación de datos de VinciBot implica tener muy en cuenta lo siguiente: **La distancia entre VinciBot y la letra escrita a mano debe ser inferior a 15 cm.** Y, durante cada ciclo de recopilación de datos, la letra «W» debe **escribirse a mano repetida y continuamente**.



Una vez confirmada la exactitud de los datos en las tres categorías, hacemos clic en "Siguiente paso" para avanzar a la siguiente etapa.

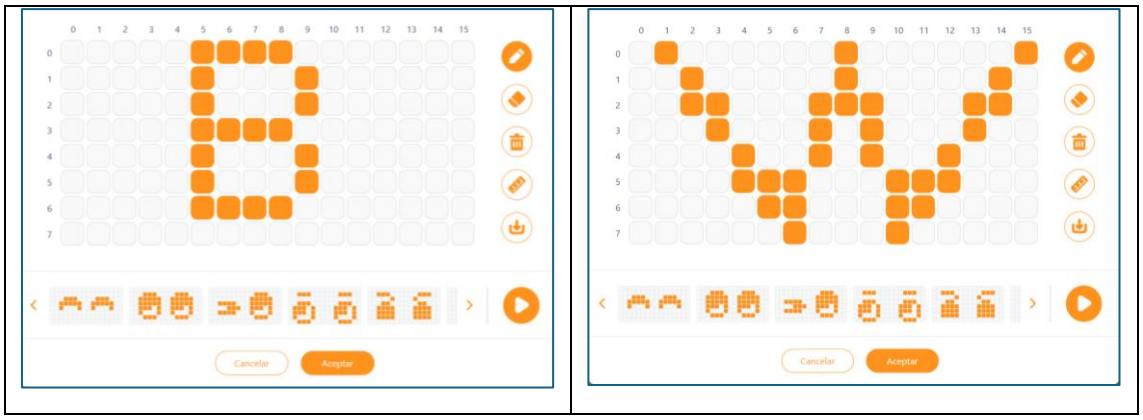
- Paso 3: Entrenamiento y comprobación.

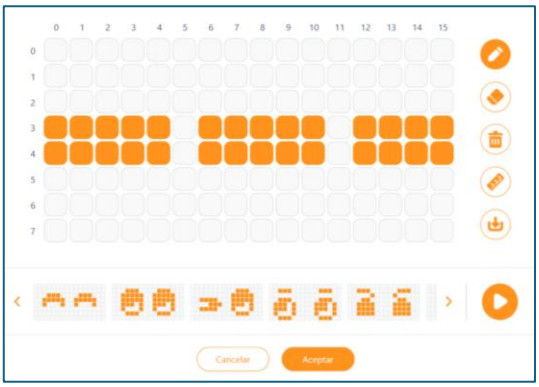
En este paso sólo debemos hacer clic en el recuadro "Iniciar entrenamiento". La red neuronal comenzará a entrenar el modelo y, al finalizar el entrenamiento, se generará un informe que detalla la precisión, la pérdida de datos, la matriz de confusión y el rendimiento del modelo. Una mayor precisión nos indicará que los resultados de reconocimiento de letras de VinciBot son más exactos, con menos errores. Finalmente, si esta precisión es adecuada, hacemos clic en "Implementar modelo" para poder usar el modelo entrenado.

- Paso 4: Programación

Sólo nos resta programar el modelo que hemos creado y que vemos que funciona ya que ha sido testeado en el paso anterior. Un posible script sería el siguiente:

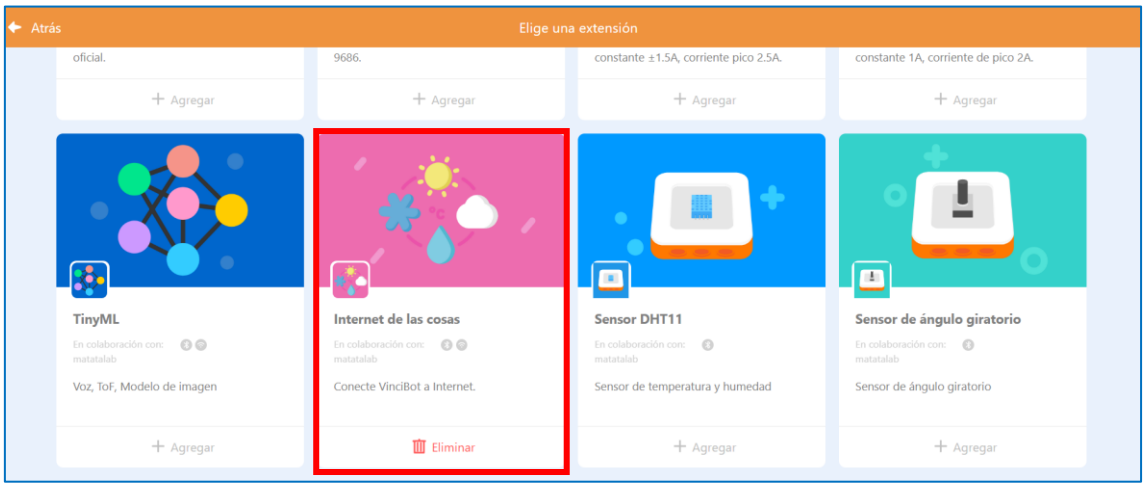
Los dibujos que hemos creado en la matriz de LEDs son los siguientes:





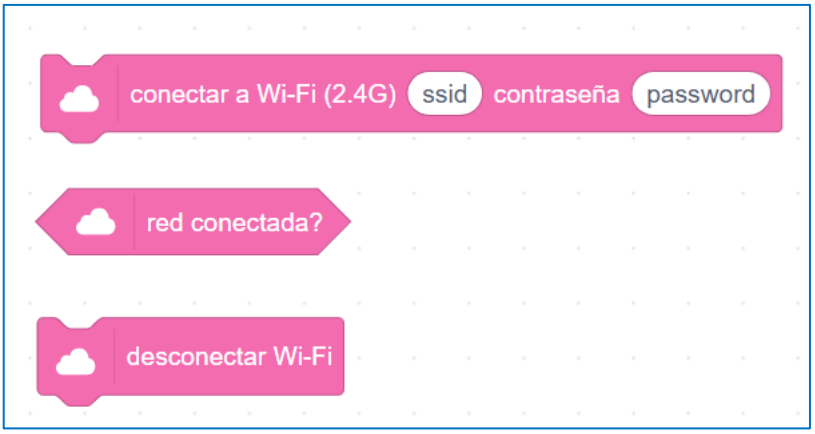
4.7. IoT

El microprocesador de VinciBot se basa en un ESP32-S3 y, por lo tanto, dispone de un módulo WiFi 2.4G para implementar proyectos de IoT (Internet de las cosas). Eta opción requiere que el usuario esté logueado a la nube de Matata Studio y que se incluya la extensión “Internet de las cosas”.

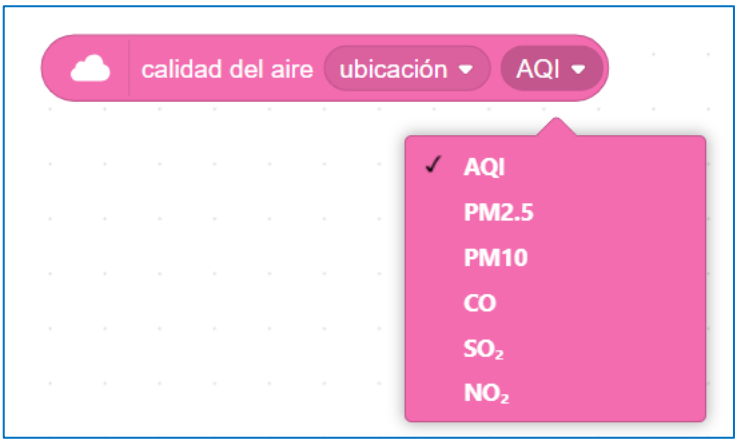


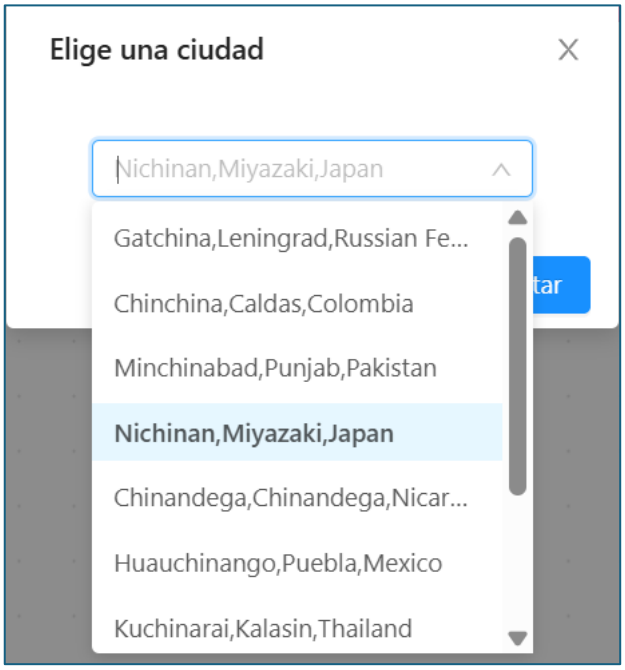
Los bloques de esta extensión nos permiten:

- Conectar o desconectar el robot a una WiFi 2.4G



- Obtener medidas de calidad del aire en ubicaciones concretas (ciudades-ninguna europea). Por lo tanto, no es interesante esta opción en España.

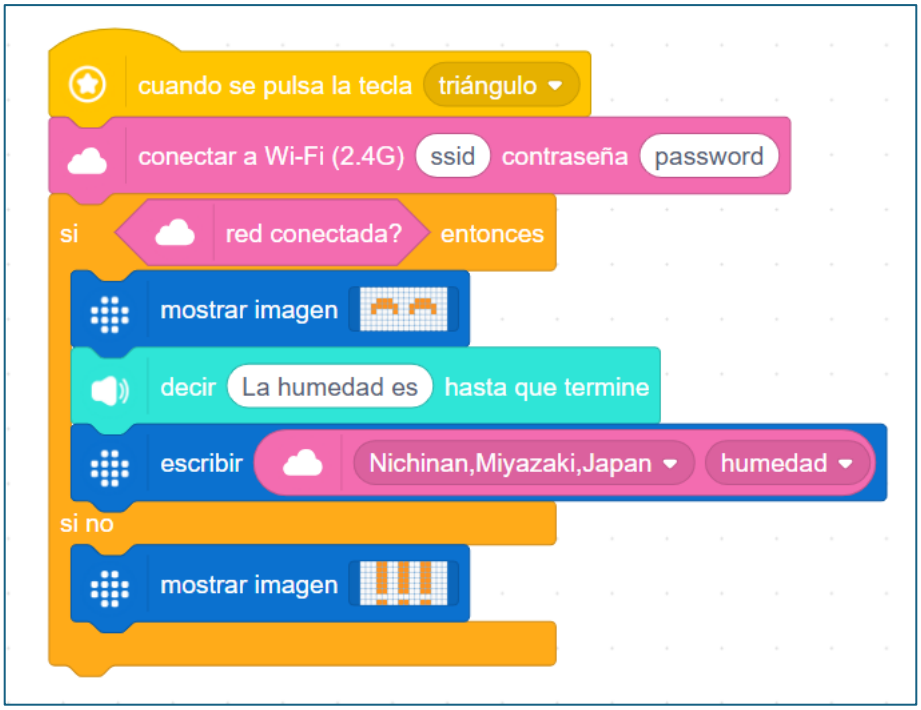




- Obtener datos de T, clima, humedad, y la hora a la que amanece o anochece en una ubicación concreta.

- Enviar (transmitir) o recibir mensajes entre dos o más robots VinciBots

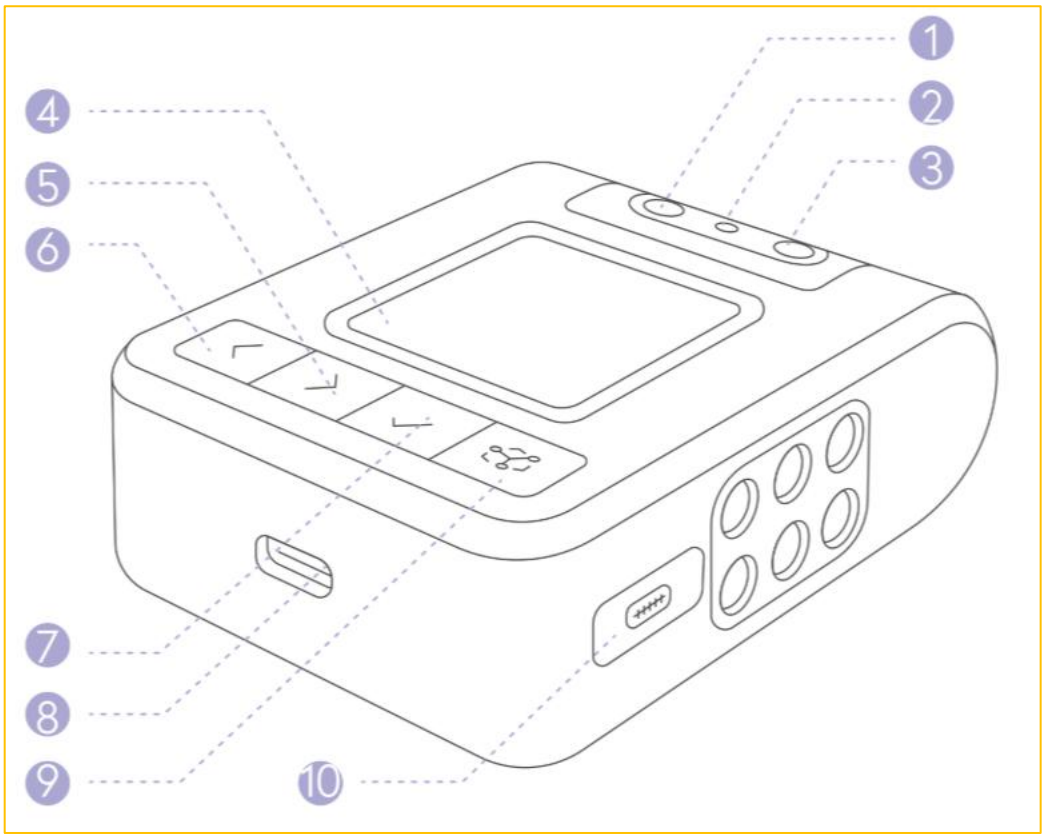
El siguiente script nos ofrece información de la humedad en Japón:



5. Kit AI Visión para VinciBot

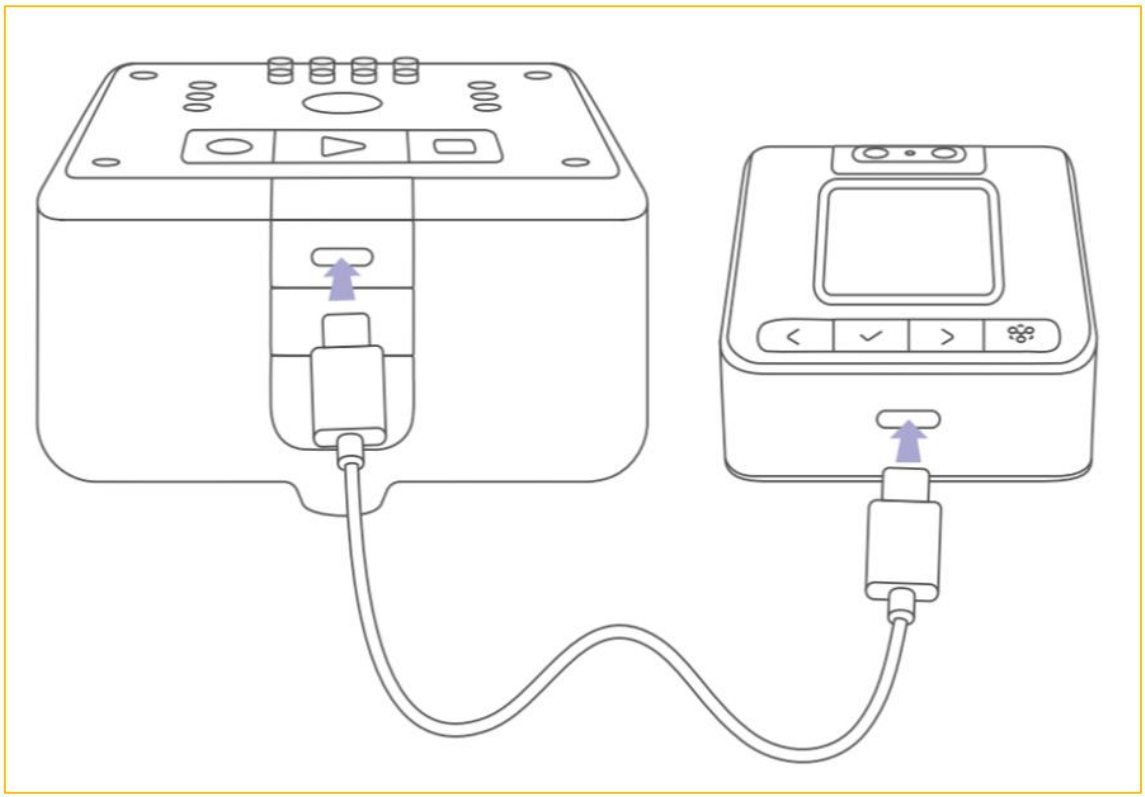
Es una cámara de IA con las siguientes funciones integradas: reconocimiento facial, de caras de gatos, de tarjetas, de color, de escritura a mano y seguidor de línea.

5.1. Información básica



1	Luz de relleno 1
2	Cámara
3	Luz de relleno 2
4	Pantalla LCD a color de 1,5 pulgadas y resolución 240x240
5	Botón de selección derecho
6	Botón de selección izquierdo
7	Botón enter (botón OK)
8	Interfaz de comunicación USB-C
9	Botón de aprendizaje
10	Interfaz de actualización del firmware USB-C

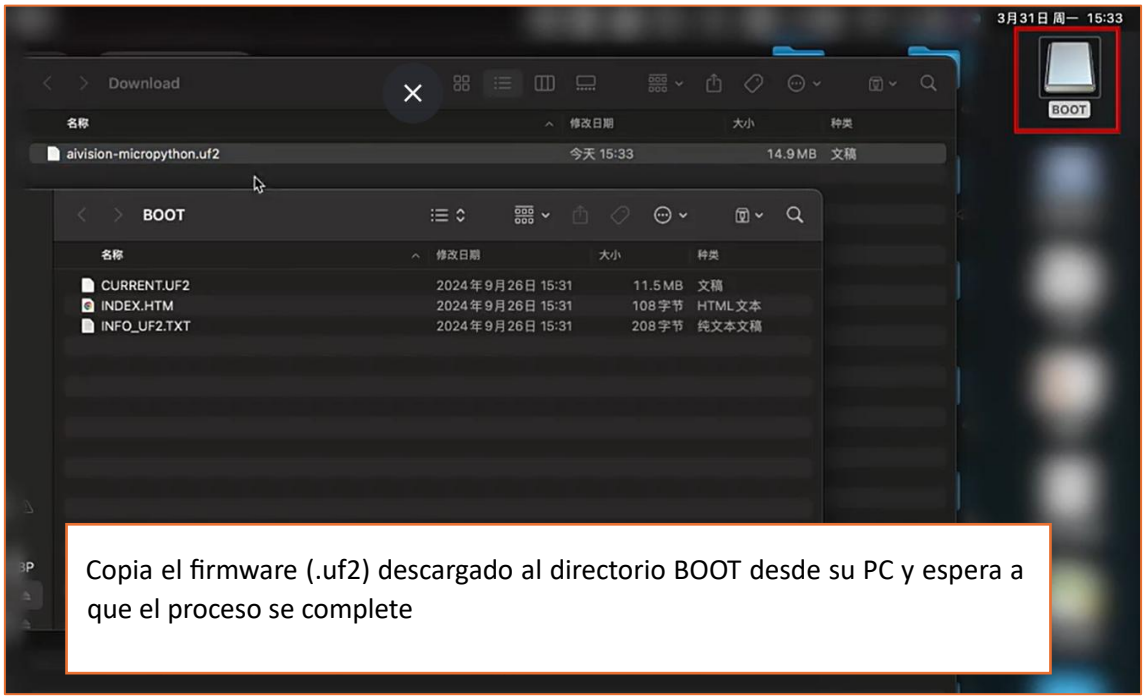
La conexión entre el robot y la cámara de IA se realiza a través de un cable USB-C a USB-C que viene con la cámara, tal y como se muestra en la siguiente imagen:



A veces se necesita actualizar la versión del firmware de la cámara. El número de su versión de observa en la pantalla:

En Herramientas > Actualización de firmware, podemos actualizar el software interno de la cámara de IA, así como, el firmware del robot VinciBot, que debe ser compatible con el de la cámara de IA:

Presione simultáneamente las teclas < y >, luego inserte el cable USB en el puerto USB del módulo AI Vision ubicado detrás de la cubierta plástica gris en el lado derecho.



Copia el firmware (.uf2) descargado al directorio BOOT desde su PC y espera a que el proceso se complete

Pulsando los botones de selección derecho e izquierdo, seguidos de la pulsación del botón OK (o enter), podemos pasar de un modo de funcionamiento a otro:

También tiene un botón de aprendizaje para el reconocimiento de caras y colores. Según como lo presionemos (pulsación corta o prolongada), la cámara realizará una u otra función:

- Una pulsación breve: podemos escoger entre reconocimiento de caras o colores.
- Una pulsación prolongada: Para iniciar el reconocimiento. Tras dejar de presionar esta tecla, el aprendizaje finaliza.

5.2. Documentación y ejemplos

En el siguiente link de YouTube de la casa comercial se muestran algunas actividades o ejemplos. A mayores, la documentación completa del producto podemos encontrarla en el siguiente enlace.