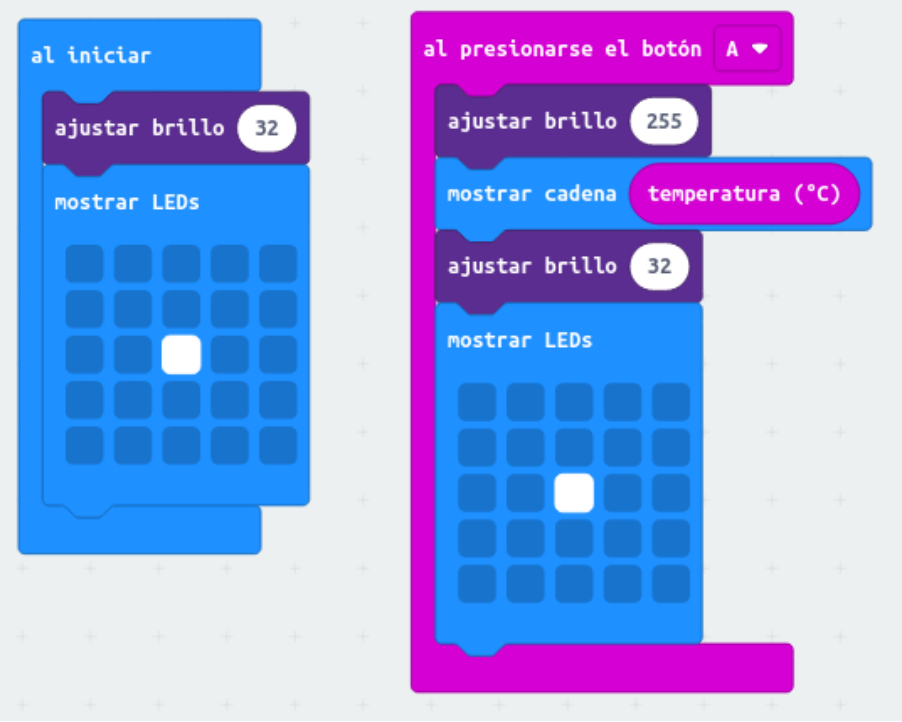
Retos avanzados con micro:bit

Mejorando el termómetro

Programación del termómetro

El siguiente programa proporcionará el valor numérico de la **temperatura ambiente en grados Celsius** cada vez que se pulse el **botón A**. La variable **temperatura (ºC)** se encuentra disponible en el menú **Entrada**.



El evento **al iniciar** comienza encendiendo un punto de la matriz de LED a modo de **piloto de funcionamiento.** El brillo de la pantalla se ajusta a un valor bajo para conseguir un **bajo consumo de energía**.

Por otro lado, **cada vez que se pulse el botón A**, ocurrirá un evento del tipo **al presionarse el botón A** que subirá el brillo de la pantalla al máximo (255) y mostrará la temperatura mediante una cadena de texto deslizante, para volver más tarde a dejar encendido el piloto de funcionamiento a bajo brillo.

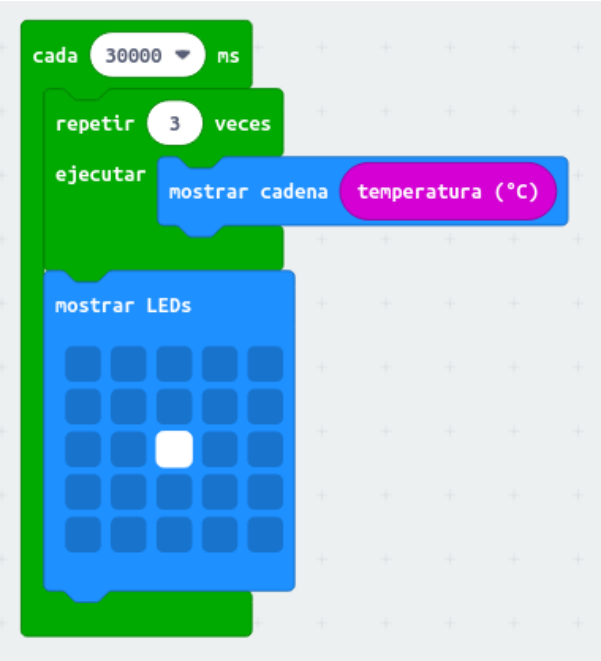
> **La temperatura indicada será algo superior a la ambiental**. Esto ocurre porque el sensor de temperatura se encuentra en el microprocesador y éste se calienta ligeramente cuando la placa está en funcionamiento. Martínez de Carvajal (2019) establece el error

> medio en 3ºC, por lo que habrá que restar 3 al valor mostrado en pantalla para obtener la temperatura real. La **manipulación de la placa con los dedos** también contribuye al calentamiento y al error en la medida de la temperatura.

Podemos añadir un **evento de tiempo** para que el termómetro muestre la temperatura cada cierto tiempo. Para ello debemos usar el evento **cada ms**, dentro del menú **Bucles**.

Cada 30000 ms, o cada **30 segundos**, el programa mostrará la temperatura aunque no haya sido pulsado el botón A. Dentro del bucle de tiempo no se sube el brillo, así que los dígitos se mostrarán con bajo brillo para ahorrar batería.

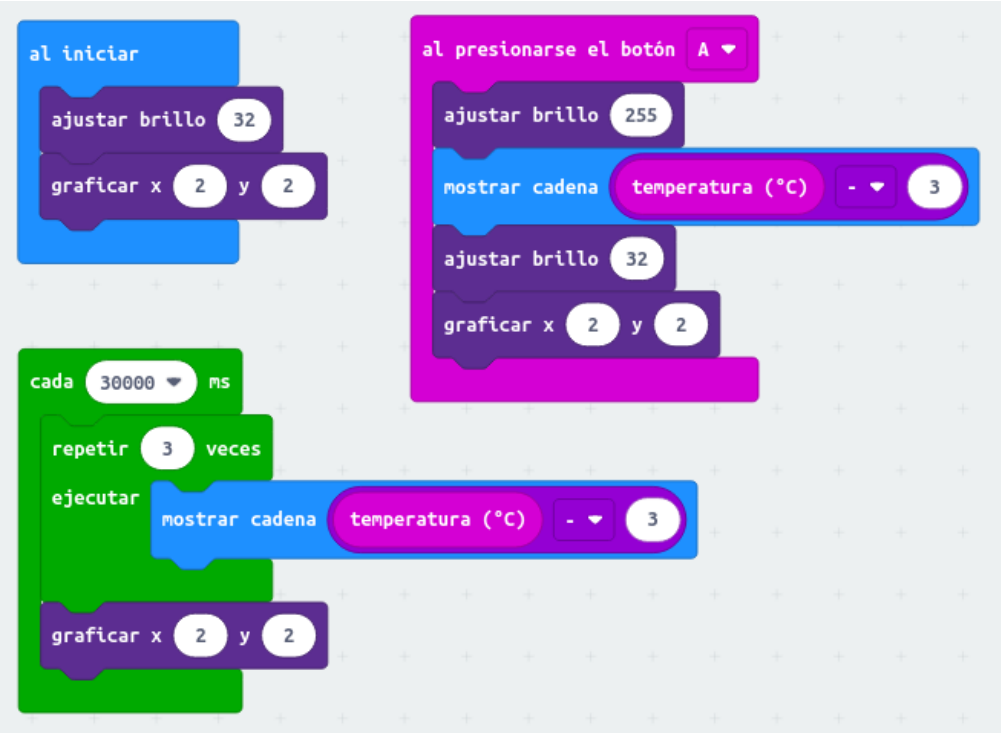
Es posible mejorar un poco más el programa haciendo que micro:bit muestre tres veces la temperatura cada 30 segundos. Podríamos repetir sin más la sentencia **mostrar cadena temperatura (ºC)** tres veces dentro del bucle de tiempo, pero en su lugar vamos a usar un bucle del tipo **repetir veces** , que también se encuentra en el menú **Bucles**.



Nótese que el programa ejecuta un bucle cada 30 segundos, y que dentro de ese bucle se ejecuta otro bucle que muestra la temperatura tres veces seguidas. Al hecho de introducir un bucle dentro de otro se le llama **anidar bucles**.

Mejorando la lectura del sensor de temperatura

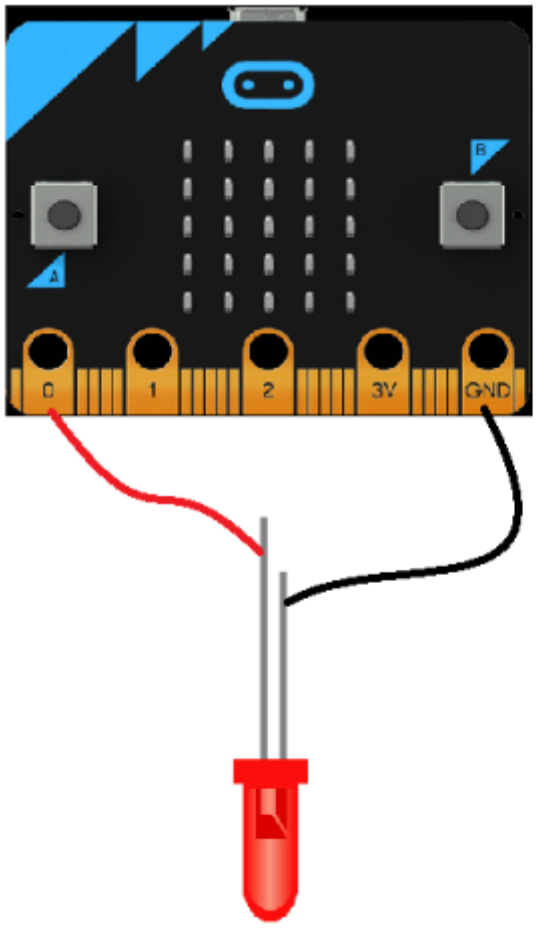
Para corregir el error de 3 grados Celsius en la lectura del sensor, bastará con restar 3 al valor de la variable **temperatura (ºC)**. El menú **Matemática** contiene bloques para realizar operaciones aritméticas. Si se usa el bloque de resta **-** dentro de los bloques **mostrar cadena** resulta sencillo realizar la corección necesaria. El código del termómetro completo quedará:



Los dos bloques **mostrar LED** han sido sustituidos por dos bloques **graficar x y** para conseguir que el código sea algo más compacto.

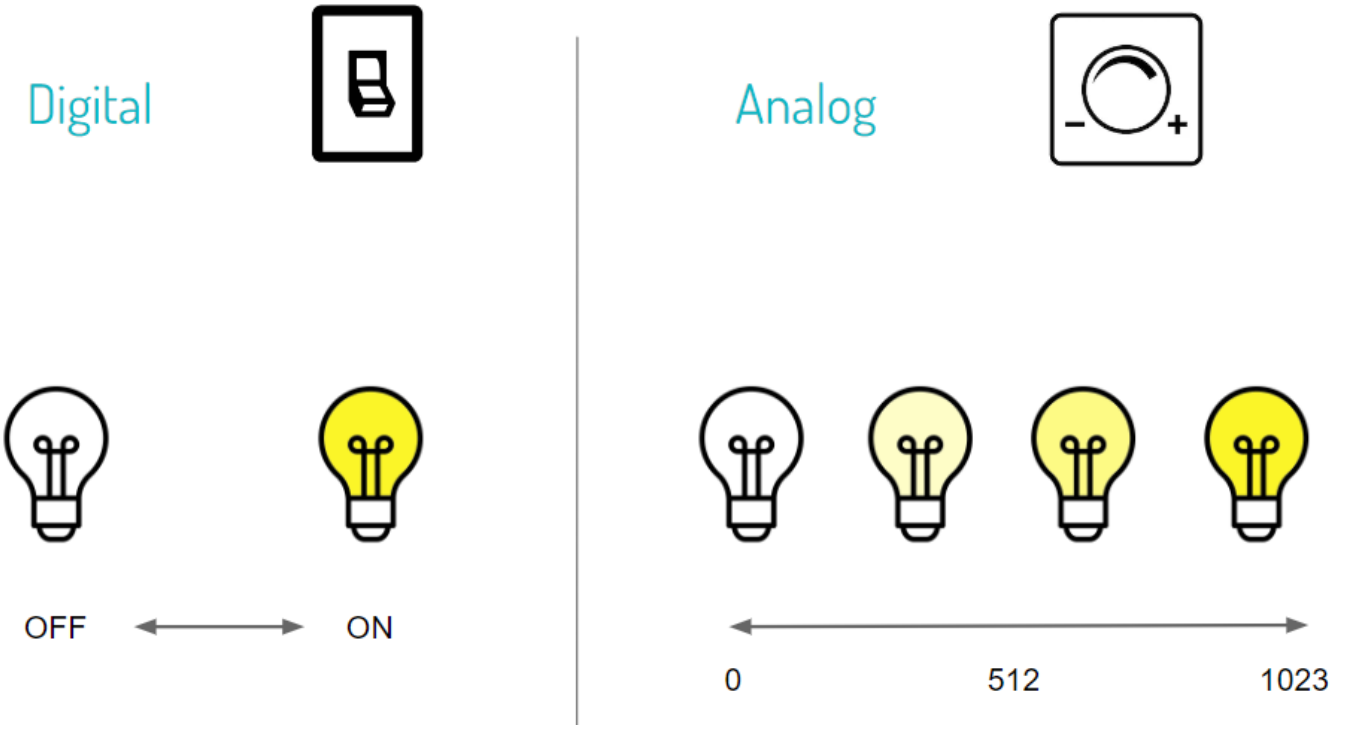
Aprender con un led la diferencia entre analógico y digital

Vamos a conectar un led, el pin corto (-) al GND y el otro al pin0



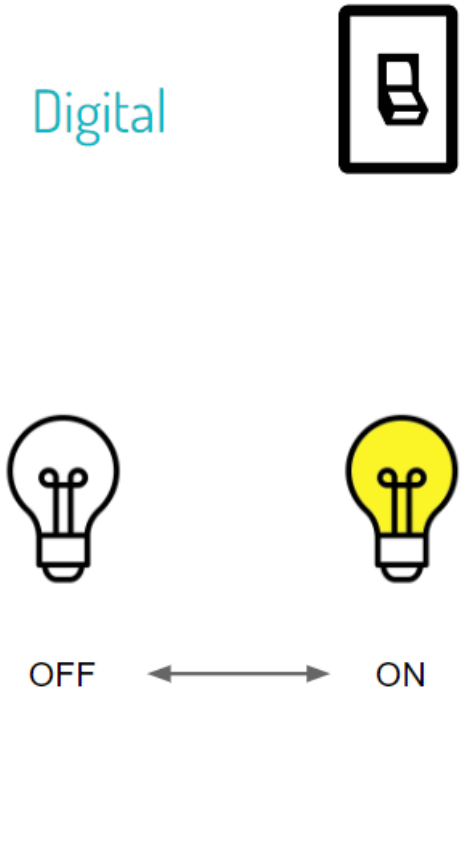
Ulrich Pedersen Dah & Ture Reimer-Mattesen Center for Underisningsmidler CPU

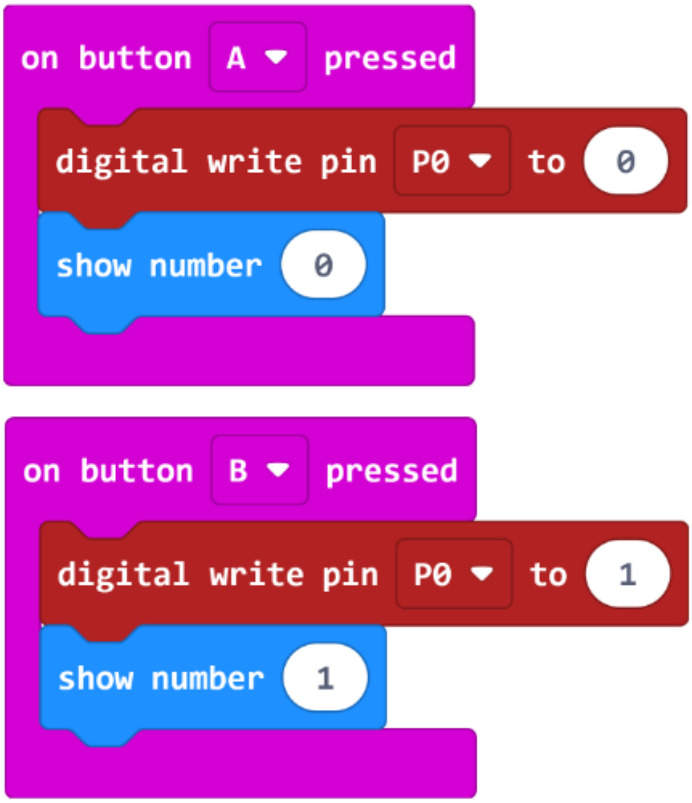
Vamos a ver la diferencia entre estos dos métodos de encender y apagar la luz



Ulrich Pedersen Dah & Ture Reimer-Mattesen Center for Underisningsmidler CPU

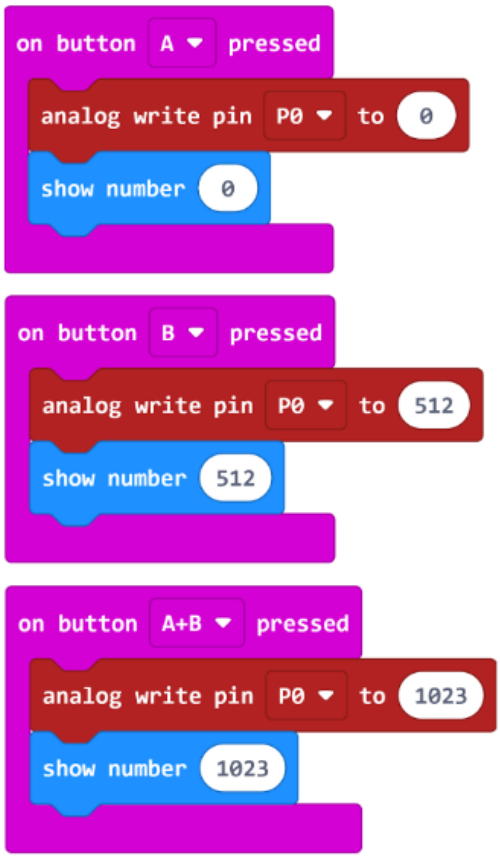
El programa en digital es sencillo

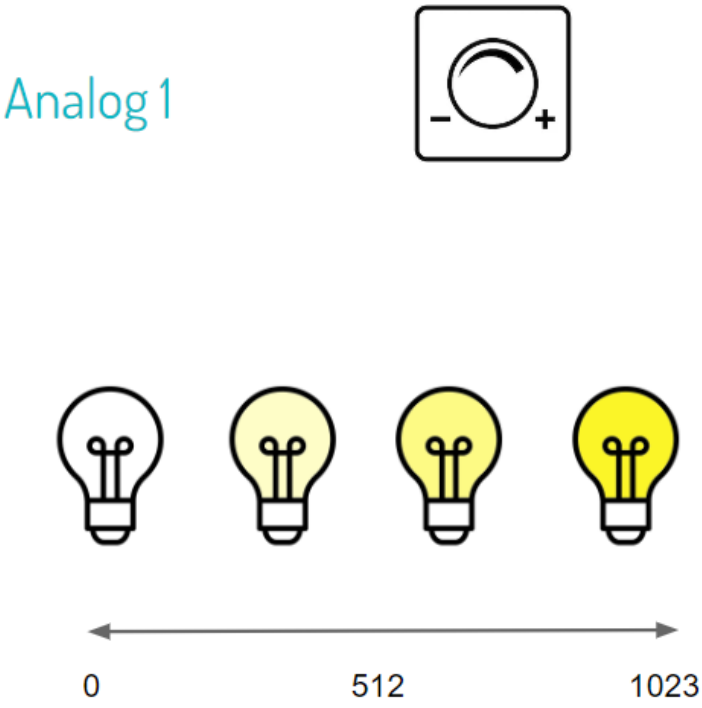




Ulrich Pedersen Dah & Ture Reimer-Mattesen Center for Underisningsmidler CPU

Y el programa en analógico





Ulrich Pedersen Dah & Ture Reimer-Mattesen Center for Underisningsmidler CPU

Bloques de entradas: sonómetro, magnetómetro y acelerómetro con una sola línea de código

Sensor de sonido

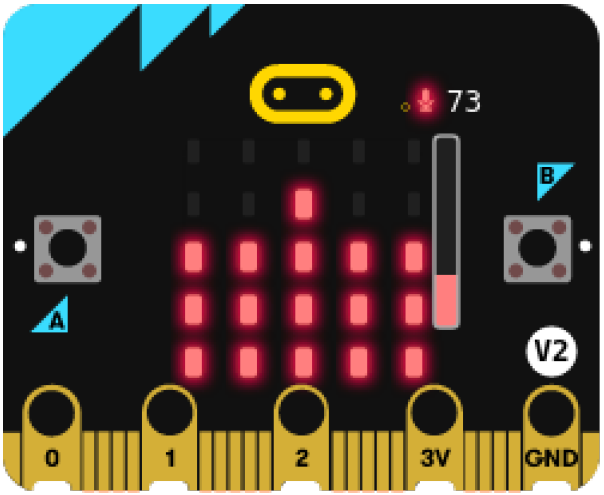
Para usar los sensores integrados de micro:bit no es necesario cargar ni inicializar bibliotecas de código. Las medidas de los sensores se encuentran disponibles en el menú **Entrada** en forma de **variables**. En el lenguaje de bloques las variables se representan mediante rectángulos de extremos redondeados.

La versión 2 de micro:bit dispone de un **micrófono** que además de grabar sonidos puede medir el nivel de ruido. La **variable nivel de sonido** nos dará lecturas entre 0 (nivel mínimo de sonido) y 255 (nivel máximo). Estos niveles no se corresponden con ninguna unidad física, como el dB por ejemplo, y deben usarse con fines comparativos.

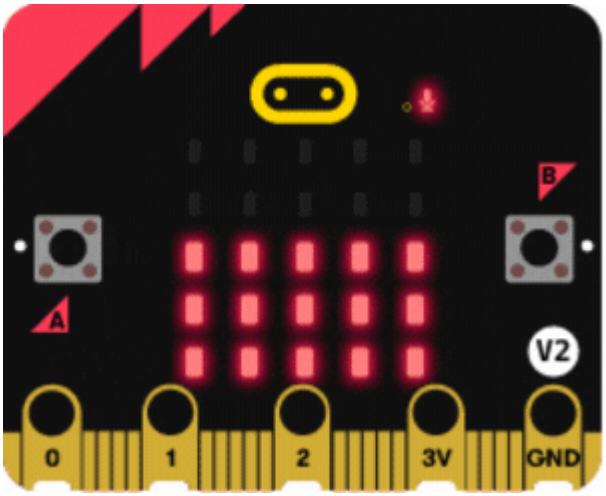
> La razón de que algunos sensores de micro:bit proporcionen medidas entre 0 y 255, es que con un byte (8 bits) sólo se pueden representar $2^8 = 256$ números distintos, es decir, el 0 y los 255 primeros números naturales.

El bloque **plot bar graph of** del menú **LED** permite construir un sencillo medidor de sonido ambiente. Como 255 es un valor muy alto de intensidad de sonido, ajustamos el rango de medida de la barra, **up to**, a la mitad, es decir, a 128. Así la barra reflejará mejor el sonido de una voz o el sonido ambiental normal.

La variable **nivel de sonido** debe arrastrarse desde el menú **Entrada**. En el momento en el que la variable haya sido introducida en el programa, el simulador de micro:bit cambiará, mostrando una **barra ajustable** que simulará el nivel de sonido captado por el micrófono. El valor numérico del nivel de sonido simulado también será mostrado al lado del **LED** del micrófono.



Tras descargar el programa en la placa real, la matriz de LED representará continuamente el sonido recogido por el micrófono en forma de barra vertical. El **LED del micrófono** iluminado indicará que micro:bit está captando sonido.



Una tarjeta micro:bit ejecutando este programa puede agotar un par de pilas alcalinas IEC R03 (AAA) en unas 40 horas (Frost 2018). Para **ahorrar energía** y prolongar la autonomía del medidor podemos reducir tanto el **brillo de la pantalla** como el **número de medidas por segundo** que realiza el sensor. Para conseguir esto último introduciremos en el bucle **para siempre** un bloque **pausa (ms)**. Si el bloque se ajusta a 100 ms, el sensor sólo realizará 10 mediciones del nivel de sonido cada segundo.

Magnetómetro y acelerómetro

Con una mínima modificación, el código anterior puede usarse para monitorizar aquellas **magnitudes que puedan variar rápidamente**. Por ejemplo, podemos usar el sensor integrado de campo magnético (magnetómetro) para medir el campo magnético de la Tierra, el de una imán o el de una masa de hierro.

Podemos acceder al sensor mediante la variable **fuerza magnética (µT)**, que proporciona la **inducción magnética** medida en **microtesla**. Al cargar el programa, micro:bit comenzará a medir el campo magnético terrestre que varía, según la localización, entre 25 y 65 µT. Nótese que el magnetómetro no limita sus medidas al valor de 255.

Otra medida interesante es la de la **aceleración de la placa**. La variable de acceso al acelerómetro se llama **aceleración (mg)** y proporciona las aceleraciones medidas en milésimas de g. Cuando la placa esté en reposo medirá la **aceleración de la gravedad terrestre**, que es

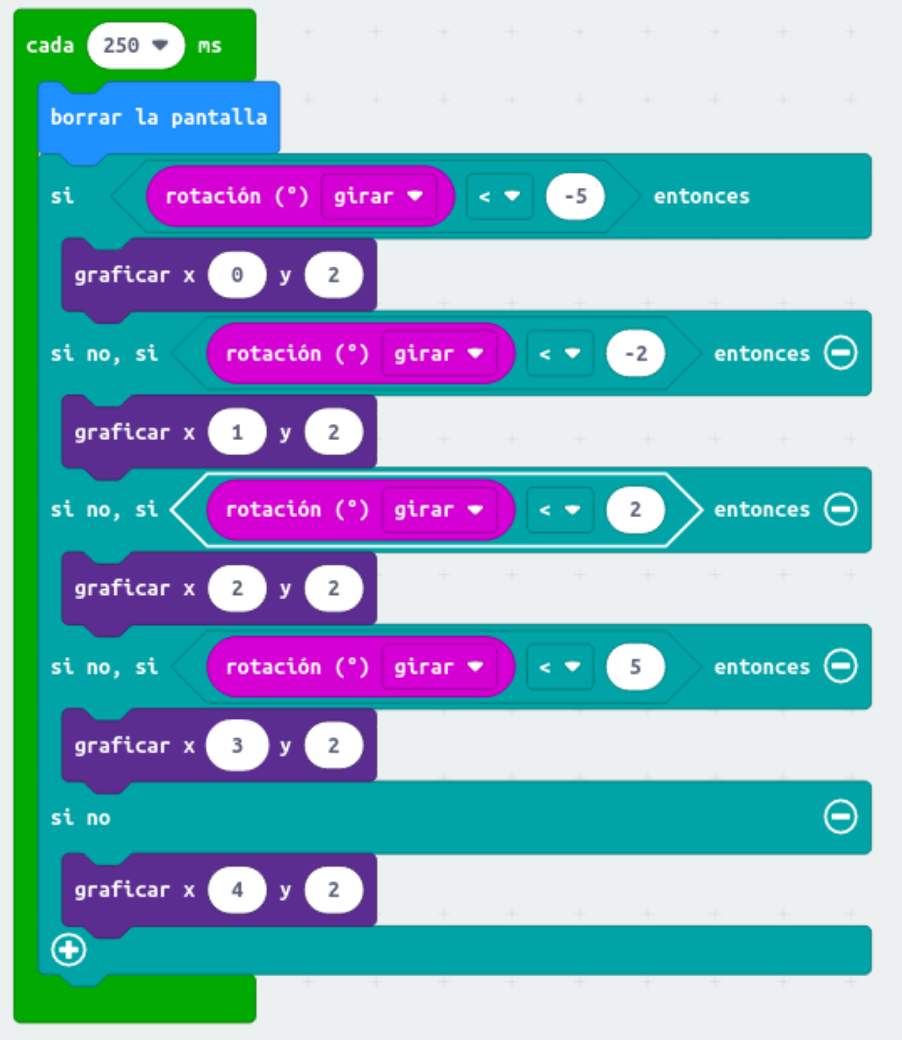
de 1 g. Los movimientos bruscos de la placa en cualquier dirección deberían alterar el valor medido.

Nivel de burbuja usando el giroscopio

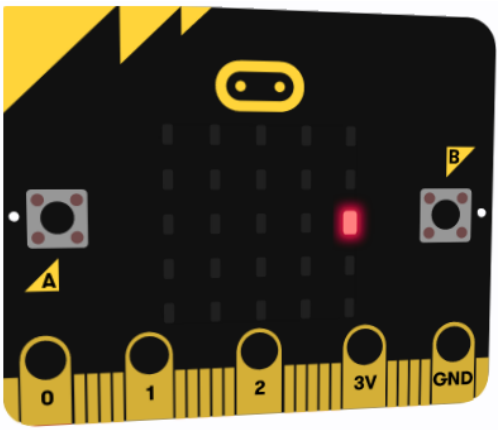
Mediante el **sensor de fuerza**, micro:bit puede determinar para cada uno de los tres ejes coordenados las proyecciones de la aceleración de la gravedad y, a partir de ellas, el **giro de la placa** con respecto al plano horizontal.

Vamos a usar la medida del giro de la placa para programar un sencillo nivel de burbuja. La burbuja será un punto luminoso en la pantalla **LED** de micro:bit. Cuando el punto se encuentre en el centro de la pantalla, cuyas coordenadas son (2,2), micro:bit estará nivelado. Si micro:bit está desnivelado hacia la izquierda o hacia la derecha, el punto se dibujará desplazado en esas direcciones.

La estructura del código, compuesto por múltiples sentencias condicionales, es muy similar al de la brújula analógica presentada en el apartado anterior.



El código consta de **un único evento temporal** dentro del cual se evalúa el giro de la placa cada 250 ms y que, en función del ángulo de inclinación, enciende el punto correspondiente. Por ejemplo, si la placa se inclina hacia la izquierda con una rotación inferior a -5º, se encenderá el punto situado más a la izquierda, cuyas coordenadas son (0,2). En caso contrario, si la placa está incilinada hacia la izquierda menos de -2º, se encenderá el siguiente punto, de coordenadas (1.2), y así sucesivamente.



Lectura puerto USB por MakeCode o Coolterm

Podemos enviar datos por el puerto USB y visualizarlos en el ordenador

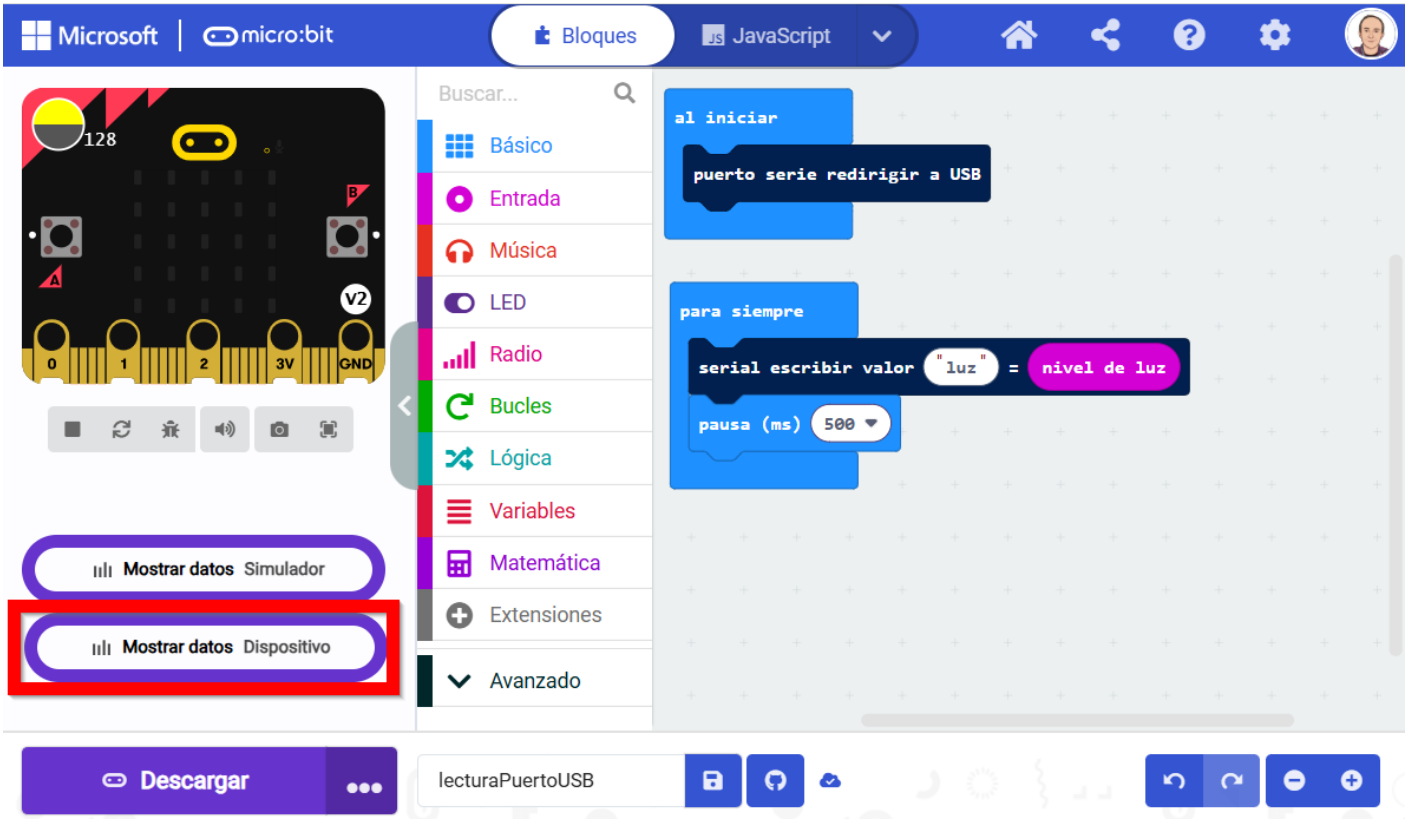
> **Hemos elegido el sensor de luz, pero PUEDE SER CUALQUIER SENSOR**

https://makecode.microbit.org/S14202-21125-85484-72930

https://makecode.microbit.org/#pub:S14202-21125-85484-72930

METODO VISUALIZACIÓN EN EL MISMO MAKECODE

Debajo del simulador podemos ver una evolución de los datos que lee



y el resultado es muy visual

https://www.youtube.com/embed/sh79ImiP0Gw

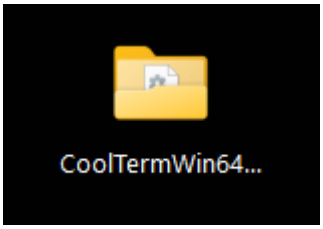
METODO COOLTERM

Este método es utilizando un programa que lo podemos descargar en esta página

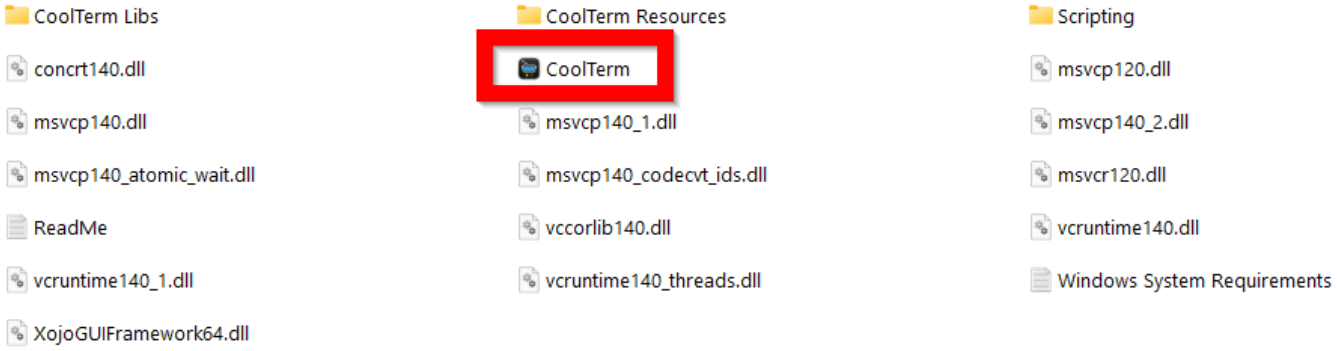
https://freeware.the-meiers.org/

Application	Version	Description
CoolTerm macOS: Intel/ARM Win: Intel 64Bit Intel 32Bit ARM 64Bit Linux: Intel 64Bit Intel 32Bit Linux ARM: Raspberry Pi Raspberry Pi 64Bit	2.4.0 05/19/2025	CoolTerm is a simple serial port terminal application (no terminal emulation) that is geared towards hobbyists and professionals with a need to exchange data with hardware connected to serial ports such as servo controllers, robotic kits, GPS receivers, microcontrollers, etc. Written in Xojo. **LINUX and Raspberry Pi:** The LINUX and Raspberry Pi versions are not "officially" (meaning: "not well") supported. While almost everything is expected to work as expected, only minimal testing using virtual machines has been performed to confirm that all the features work properly. The LINUX and Raspberry Pi builds have been posted here as a courtesy to the users that asked for it. Please use these builds at your own risk. Please use the forums to share your experiences with other users. **Legacy Builds:** • **macOS 32-Bit:** Version v1.7.0 is the last 32-bit build for macOS. • **OS X Universal Binary (PPC/Intel):** Version v1.4.7 is the last version of CoolTerm available as a universal binary supporting OS X 10.6 or older • **Windows XP:** Version v1.4.4 is the last build that supports Windows XP. **Windows 7,8:** Version v2.0.1 is the last build that supports Windows 7 (with SP1) and Windows 8. It can be downloaded here: 64-Bit, 32-Bit. **Older Versions:** Older versions of CoolTerm can be found here.

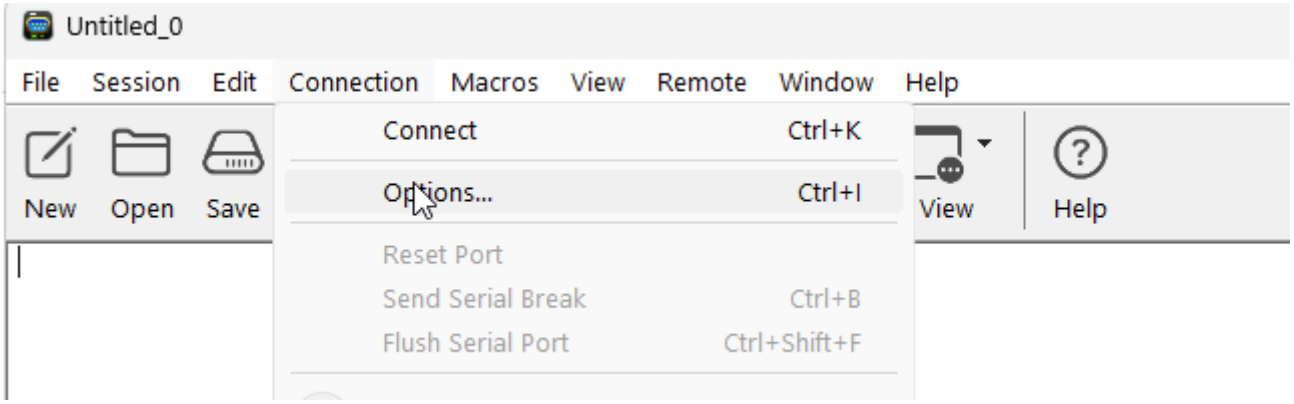
Es un programa libre, y portable, es decir es una carpeta con un ejecutable y programas accesorios



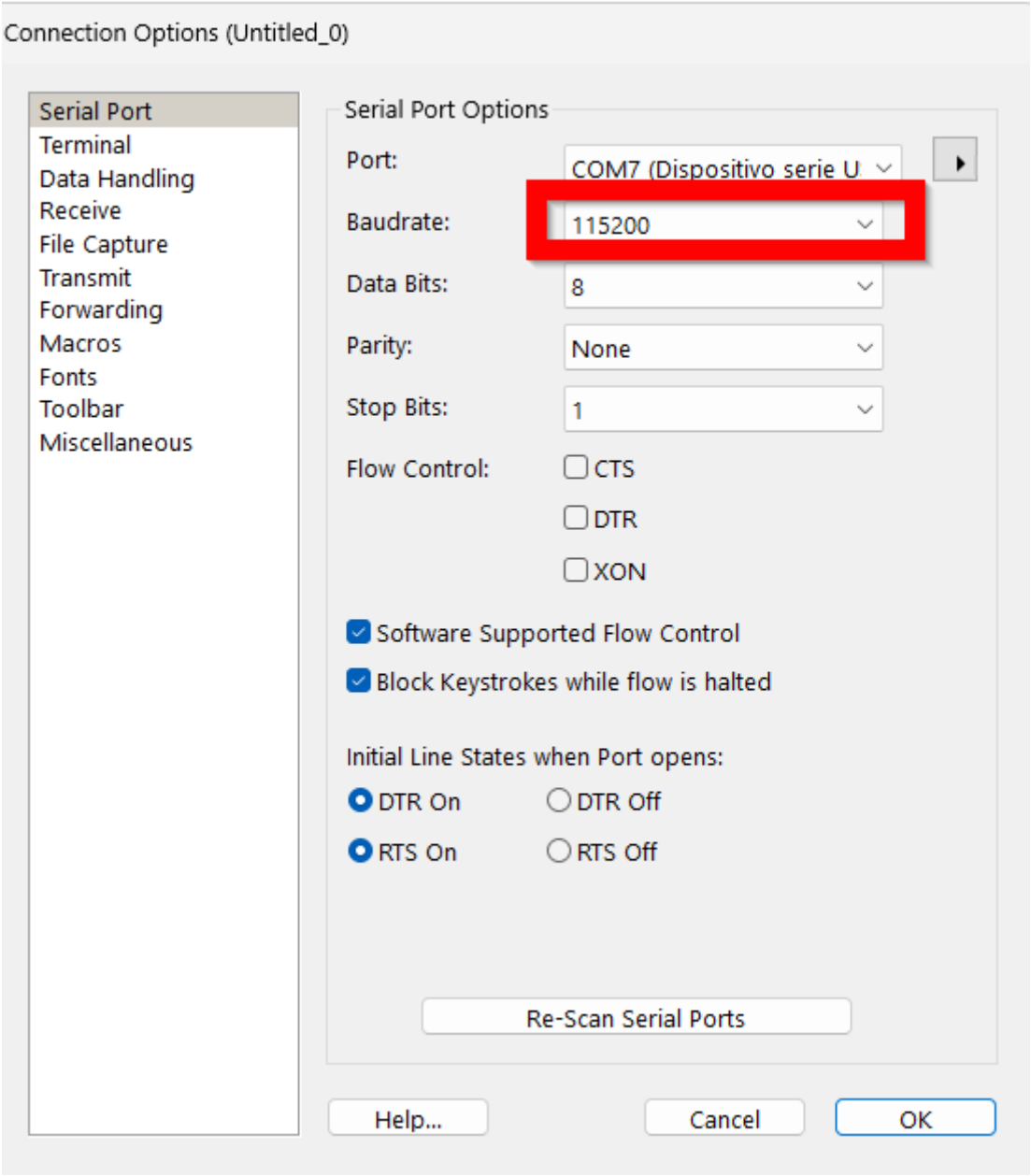
lo ejecutamos



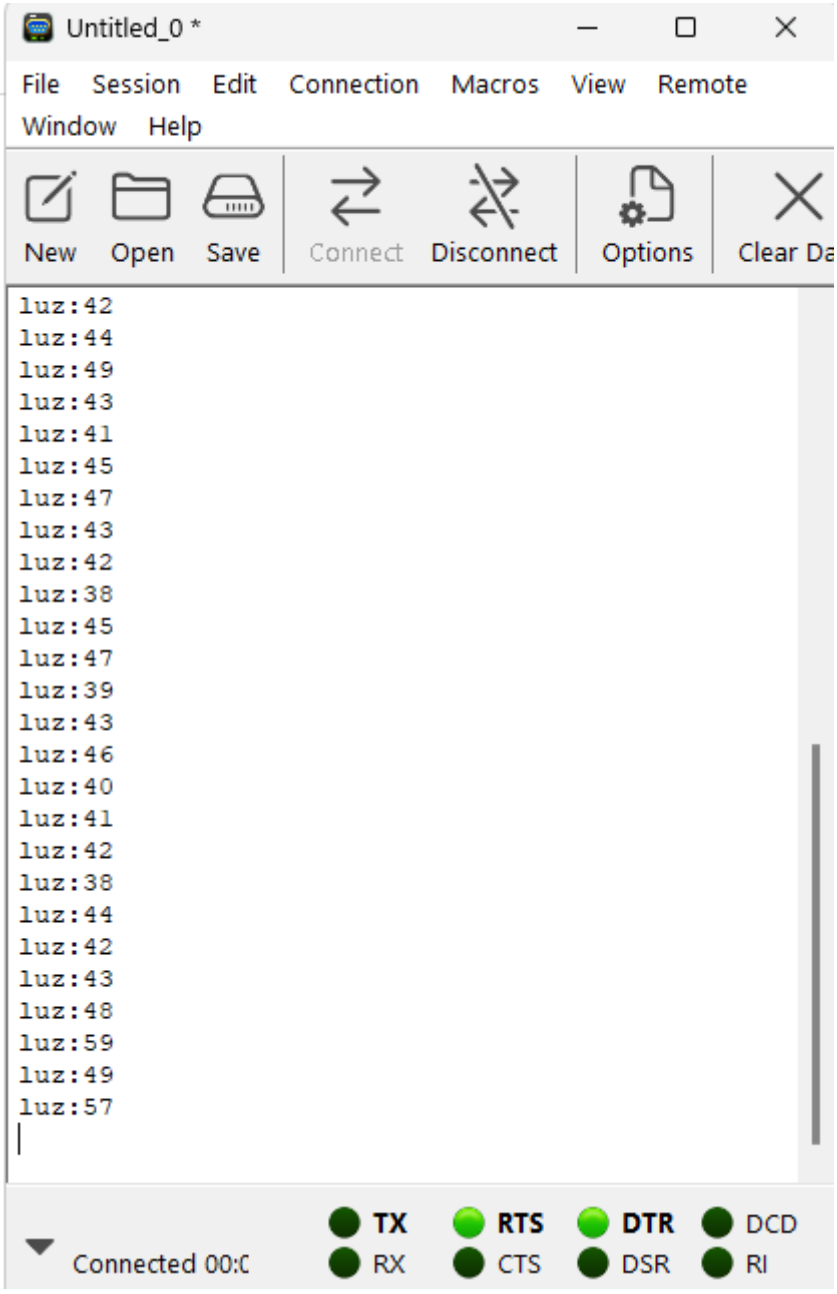
En Connection - Options



Subimos la velocidad a 115.200 baudios



Al darle a **conectar** se ven los datos numéricamente



> LA VENTAJA DE COOLTERM ES QUE LEE CUALQUIER DISPOSITIVO (MICRO:BIT, ARDUINO, ECHIDNA....)

BLUETOOTH un poco de teoría

ONDAS

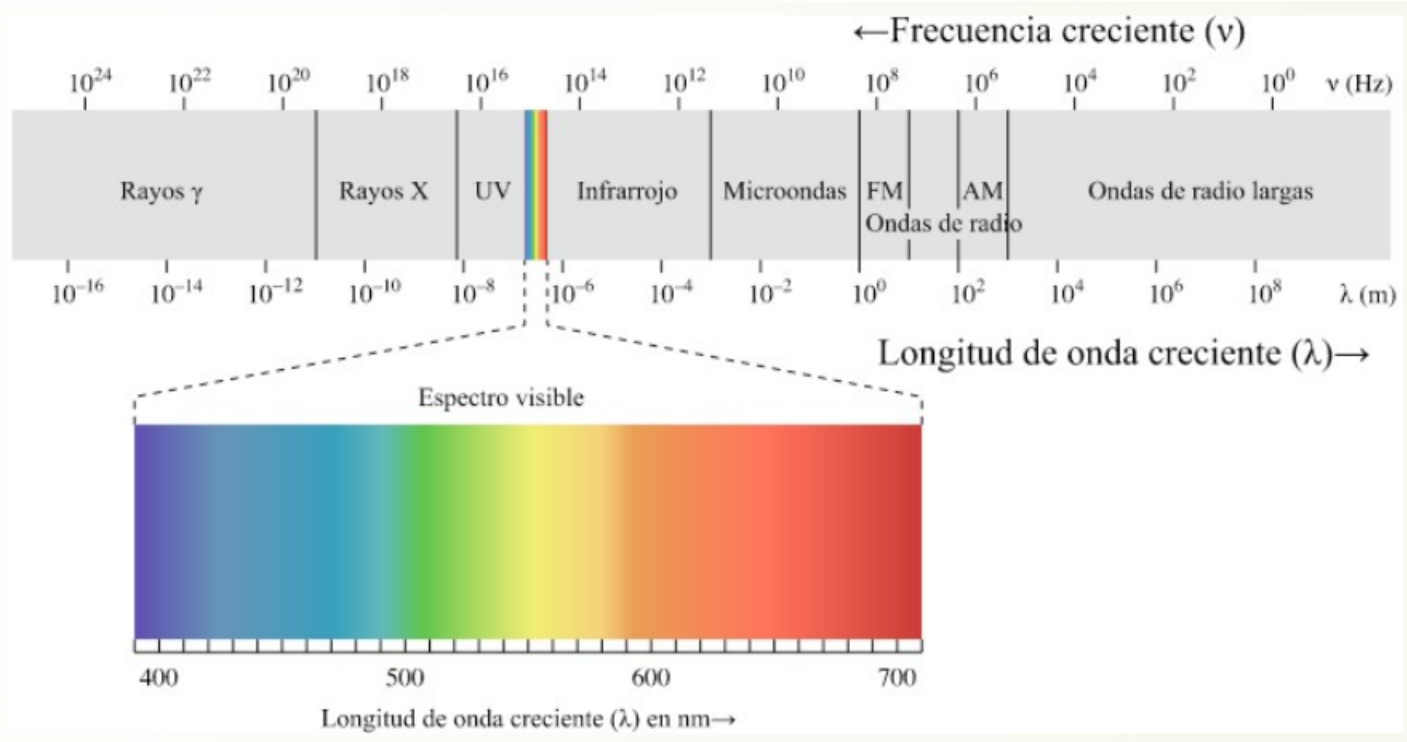
Una onda es una señal que se propaga por un medio. Por ejemplo el sonido, que es una onda mecánica que viaja usando el aire o cualquier otro material. Pero en el caso de las señales eléctricas pueden ser enviadas por el cable o a través del vacío (no necesitan un medio para transmitirse).

Dependen de 3 parámetros principalmente:

- **Amplitud:** altura máxima de la onda. Hablando de sonido representaría el volumen. Si nos referimos a una onda eléctrica estaríamos representando normalmente el voltaje.
- **Longitud de onda λ:** distancia entre el primer y último punto de un ciclo de la onda (que normalmente se repite en el tiempo).
- **Frecuencia f :** Número de veces que la onda repite su ciclo en 1 segundo (se mide en hertzios).
- **Periodo T** es simplemente es la inversa de la frecuencia. T=1/f

La relación entre ellas es muy fácil pues las ondas electromagnéticas viajan a la velocidad de la luz c y si velocidad es espacio/tiempo luego c = λ/T luego **c= λ*f**

Dentro del espectro electromagnético encontramos diferentes tipos de señales dependiendo de las características de su onda.



TRANSMISIÓN INALÁMBRICA: BLUETOOTH.

- Hoy en día, este grupo está formado por miles de empresas y se utiliza no sólo para teléfonos sino para cientos de dispositivos.
- Bluetooth es una red inalámbrica de corto alcance pensada para conectar pares de dispositivos y crear una pequeña red punto a punto, (sólo 2 dispositivos).
- Utiliza una parte del espectro electromagnético llamado "**Banda ISM**", reservado para fines no comerciales de la industria, área científica y medicina. Dentro de esta banda también se encuentran todas las redes WIFI que usamos a diario. En concreto funcionan a 2,4GHz. (Un G son 10^9) luego entre FM y Microondas.

> **¿Sabias que?**
> Su curioso nombre viene de un antiguo rey Noruego y Danés, y su símbolo, de las antiguas ruinas que representan ese mismo nombre.

Hay 3 clases de bluetooth que nos indican la máxima potencia a la que emiten y por tanto la distancia máxima que podrán alcanzar:

CLASE	POTENCIA	DISTANCIA
Clase 1	100 mW	100 m
Clase 2	2,5 mW	10 m
Clase 3	1 mW	1 m

También es muy importante la velocidad a la que pueden enviarse los datos con este protocolo:

Versión	Velocidad
1.2	1 Mbps
2	3 Mbps
3	24 Mbps
4	24 Mbps

> Mbps : Mega Bits por segundo. MBps: Mega Bytes por segundo.
> kb = 1.024 b M = 1.024 k G = 1.024 M

¿Te atreves a calcularlo .... ?

¿Cuántos ciclos por segundo tendrán las ondas que están en la **Banda ISM?** ¿Cuál es el periodo de esas ondas?

Solución
a) f = 2.4G
b) λ=c/f= 12.5cm o sea, las antenas tendrían que ser de esta longitud. Hay muchos trucos para reducirla, una de ellas es la forma de serpiente que puedes ver en el HC-06

¿Te atreves a calcularlo...?

¿A qué distancia y cuanto tiempo tardarían en enviarse los siguientes archivos por Bluetooth?

1. Un vídeo de 7Mb usando versión 2 clase 2
2. Una imagen de 2.5Mb usando versión 3 clase 1
3. Un archivo de texto de 240KB usando versión 1.2 clase 1

Solución
1) 7Mb / 3Mbs = 2.3 seg.
2) 2.5Mb / 24Mbs = 0.1 seg.
3) 240 kB 8b/B = 1.920 kb 1.920 kb / 1.024 = 1.875 Mb 1.875Mb / 1Mbs = 1.875 seg.

¿Bluetooth clásico o Bluetooth Low Energy = BLE?

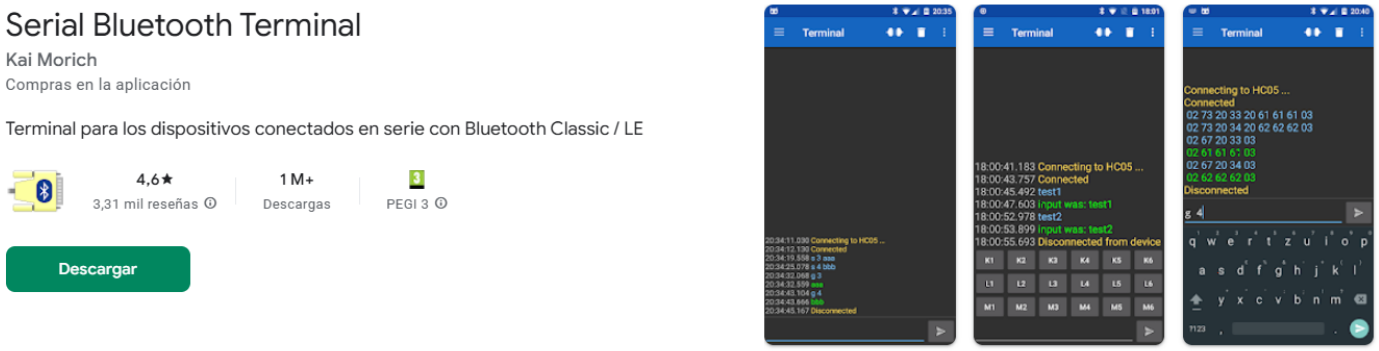
Es un protocolo similar al clásico Bluetooth pero diseñado a consumir menos potencia manteniendo funcionalidad. Su popularidad ha crecido en multitud de dispositivos

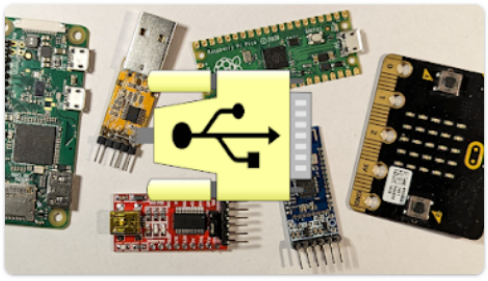
En robótica, el clásico device que utiliza BLE es la **Micro:bit**. Aunque la Micro:bit no tiene Wifi integrada, posee una radiofrecuencia que podemos configurar para Bluetooth (hay que elegir, o utilizar sus comandos de Radio o utilizar comandos de Bluetooth)

Por eso a la hora de elegir la APP tienes que tener en cuenta:

- Si acepta Bluetooth clásico o BLE
- Que la APP acepte leer datos desde el robot como enviar

Nosotros hemos elegido uno sencillo que cumple las dos condiciones (hay muchas APPs) Serial Bluetooth Terminal

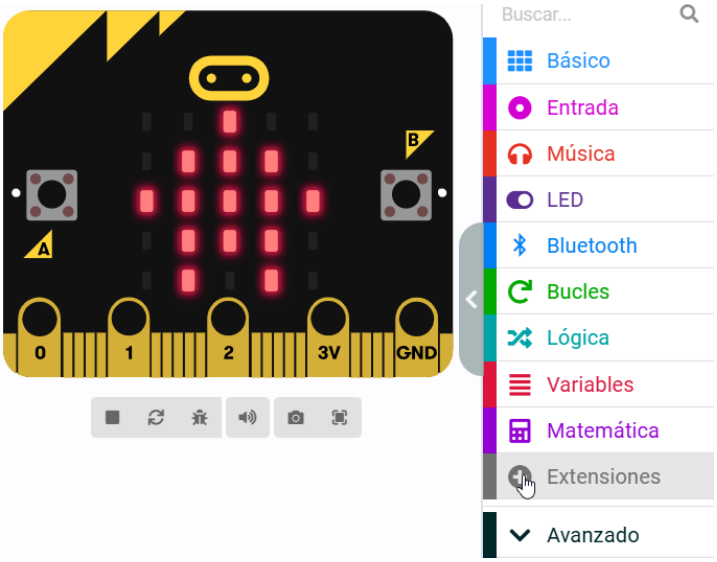




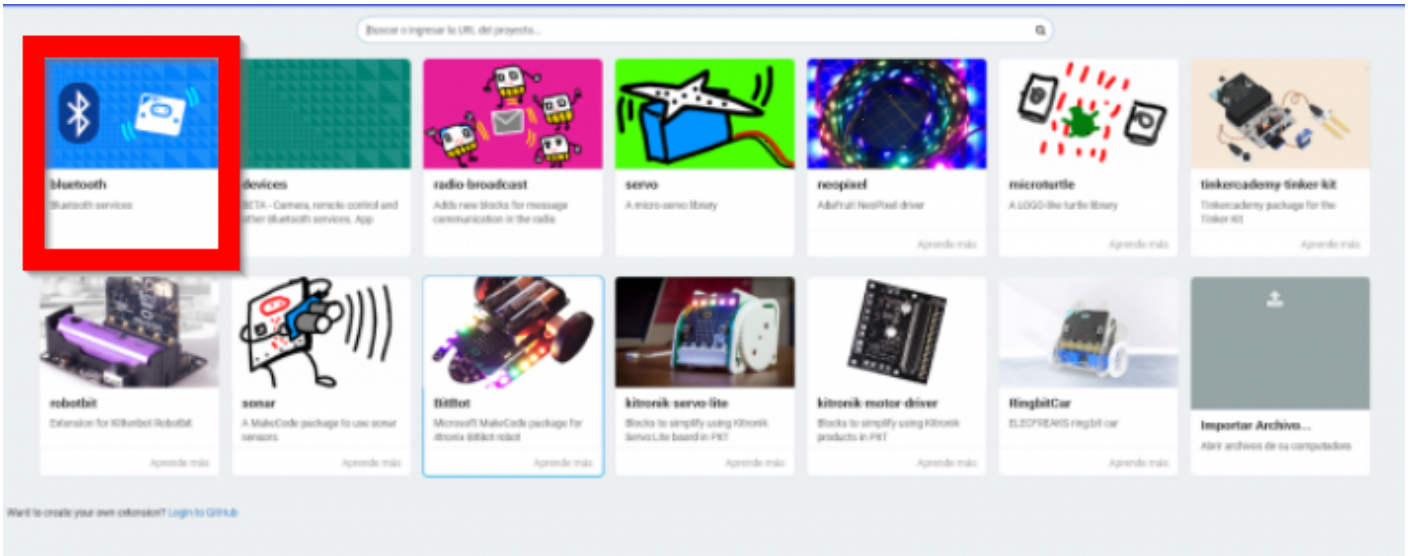
BLUETOOTH extensión en Makecode

En Makecode instalaremos la siguiente extensión

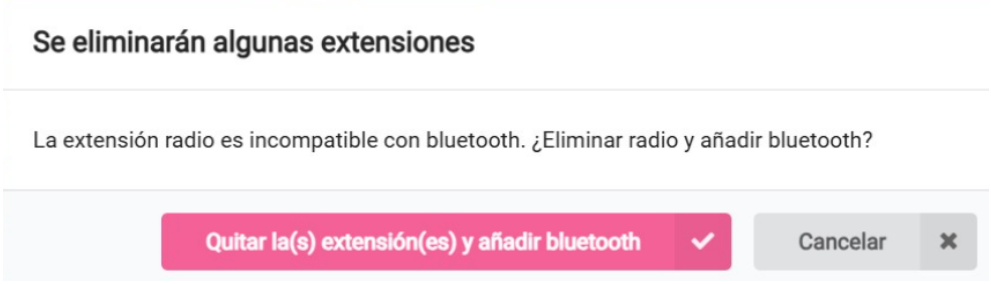
Entramos en **Extensiones**



Buscamos **Bluetooth** y elegimos la esta :

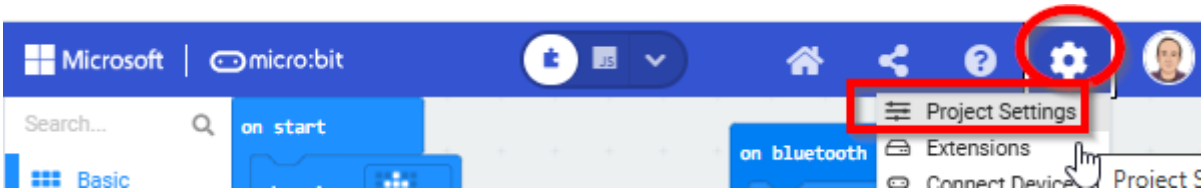


Nos dirá que es incompatible con la radio, y hay que eliminar la radio y poner Bluetotth, aceptamos :

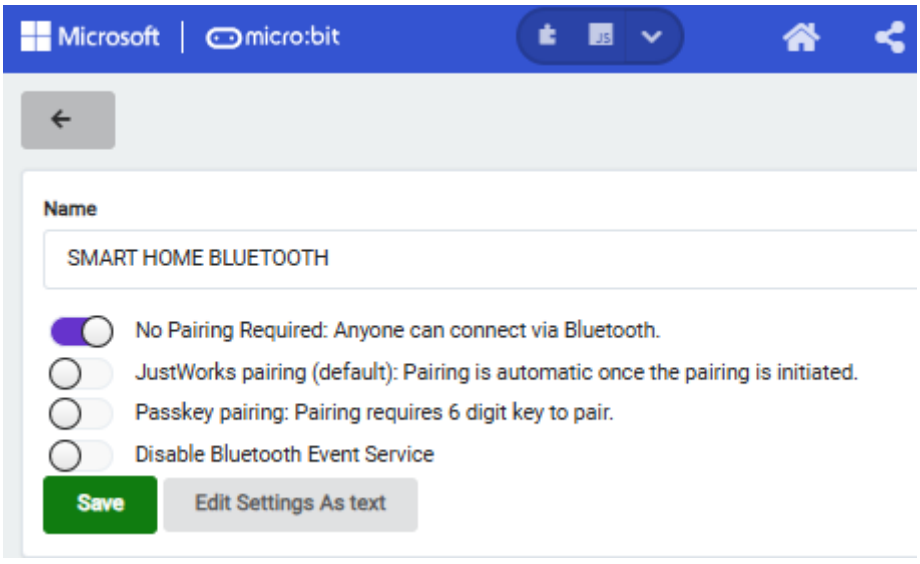


Por si acaso...

En Makecode, si vamos a la rueda dentada - Project settings



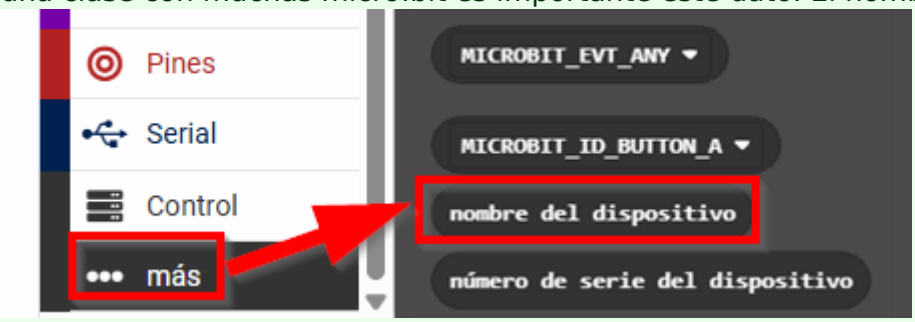
Hay que tener que cualquiera se puede conectar via Bluetooth



BLUETOOTH programa en Makecode

Realizamos un programa que :

- En **inicio**
  - se active el servicio UART para el envío y recepción de mensajes,
  - muestra un mensaje del nombre de la micro:bit, ver más abajo
- **Al conectar Bluetooth** que muestre un check
- **Al desconectar Bluetooth** que muestre X
- **Al recibir datos**, hasta # (puede ser otro carácter) que muestre la frase recibida
- **Al presionar el botón A**
  - Que muestre un mensaje
  - Que muestre la temperatura

> **¿Para qué mostrar el nombre de la micro:bit?** Para saber a qué micro:bit conectarte. En una clase con muchas micro:bit es importante este dato. El nombre del equipo está en
>
> 
>

https://makecode.microbit.org/S60585-58735-21378-05922

https://makecode.microbit.org/#pub:S60585-58735-21378-05922

BLUETOOTH programa Serial Bluetooth Terminal

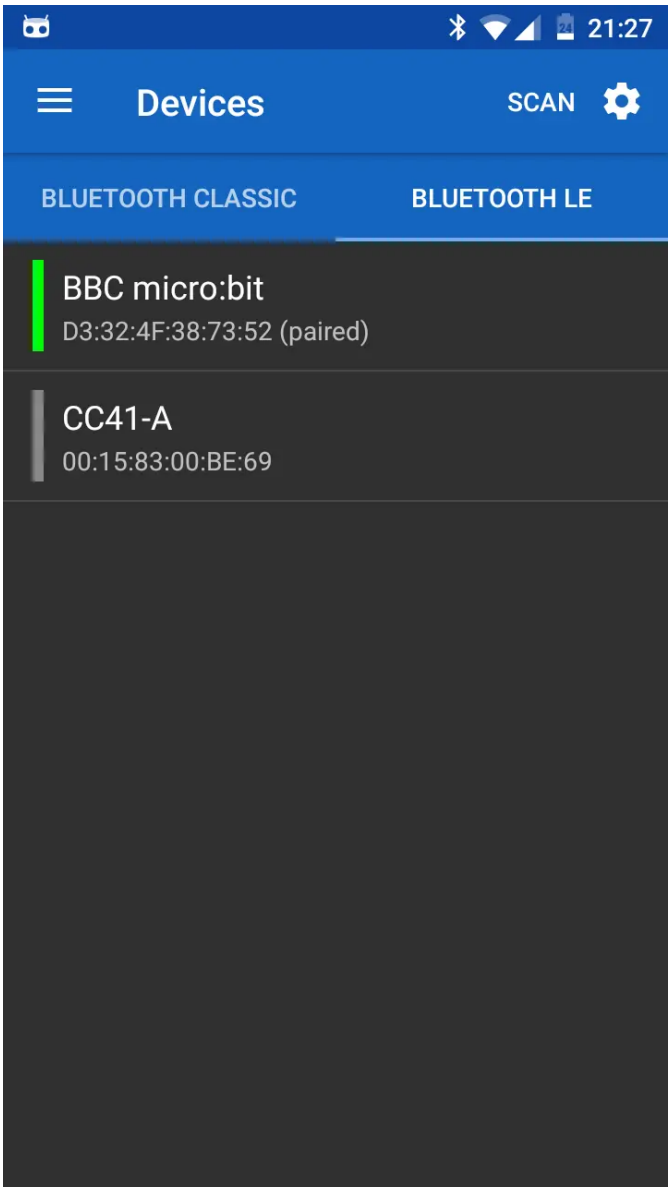
Entramos con el móvil a Google Play e instalamos esta aplicación

https://play.google.com/store/apps/details?id=de.kai_morich.serial_bluetooth_terminal

Tiene la ventaja de

- **Enviar** mensajes
- **Recibir** mensajes
- Permitir conexiones **BLUETOOTH LE** (Low emision) **que es lo que utiliza MICRO:BIT**

Entramos en **Devices** y en **Bluetooth LE** y nos conectamos a la Micro:bit



Una vez conectado, podemos:

- enviar un mensaje, que como hemos definido anteriomente en Makecode tiene que ir entre #
- recibir un mensaje, se visuaizará lo que nos envíe la micro:bit que en Makecode lo hemos programado al apretar el botón A

https://www.youtube.com/embed/H0HDVPmX-tE

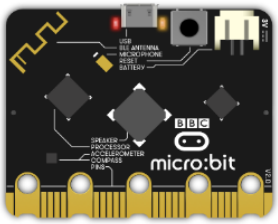
BLUETOOTH con App Inventor. Extensiones

Descargas e instalación

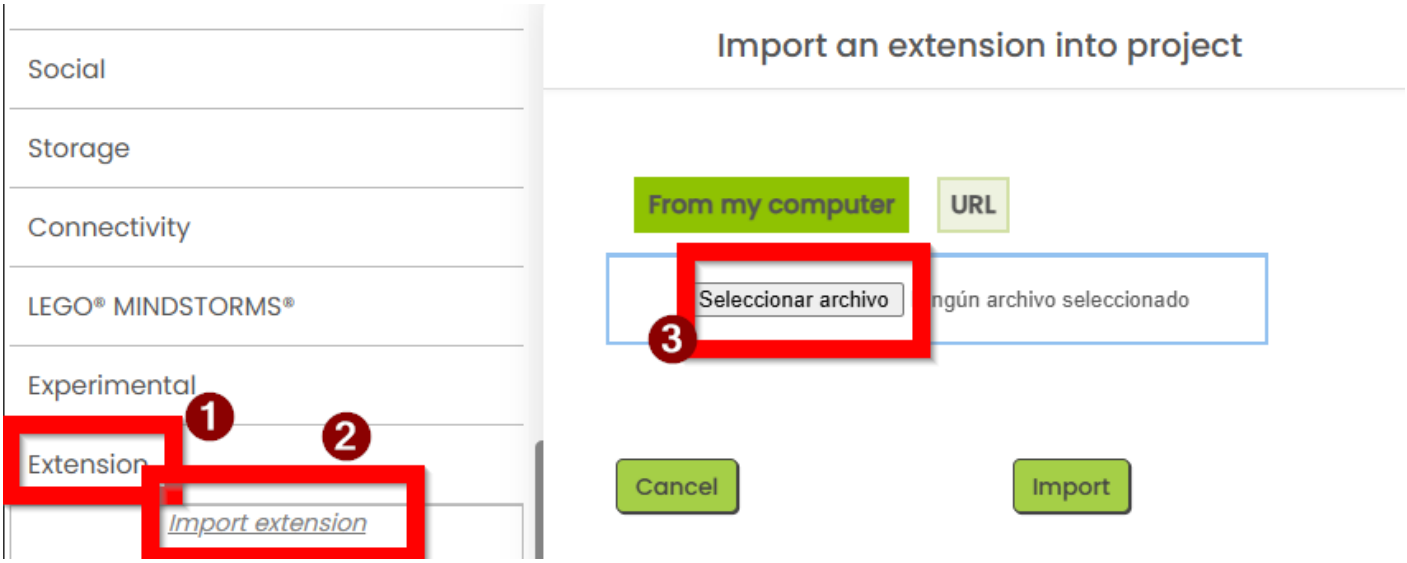
Necesitamos estas extensiones para poder crear una APP que pueda enviar y recibir con nuestra micro:bit

Página de descargas https://iot.appinventor.mit.edu/#/

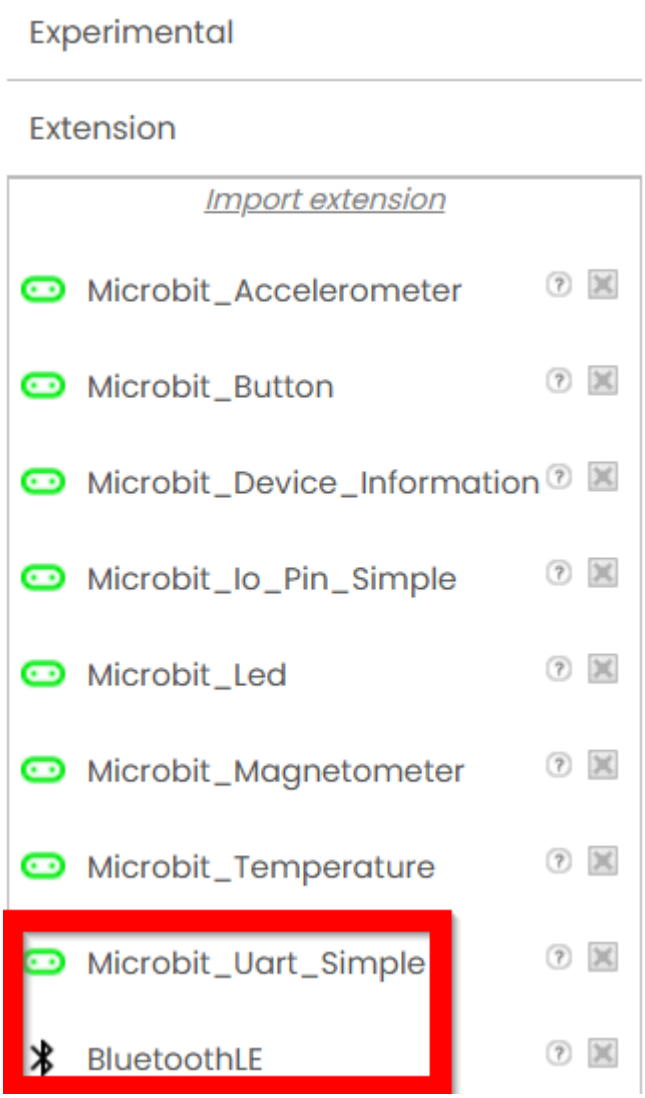
Supported Extensions

Device	Description	Extension
Bluetooth	**Bluetooth low energy** Bluetooth Low Energy, also referred to as Bluetooth LE or simply BLE, is a new protocol similar to classic Bluetooth except that it is designed to consume less power while maintaining comparable functionality.	Download
	**BBC micro:bit** The micro:bit is a computing platform from the BBC. It is an open platform for developing all manner of projects and is programmable by many different editors, including a blocks editor provided by Microsoft. Learn more about the micro:bit at the Micro:bit Educational Foundation's website.	Download

Una vez descargadas, vamos al APP INVENTOR https://ai2.appinventor.mit.edu y las instalamos en extensiones :



Una vez instaladas, se visualizan como extensiones abajo del menú. Las dos últimas son las que utilizaremos:



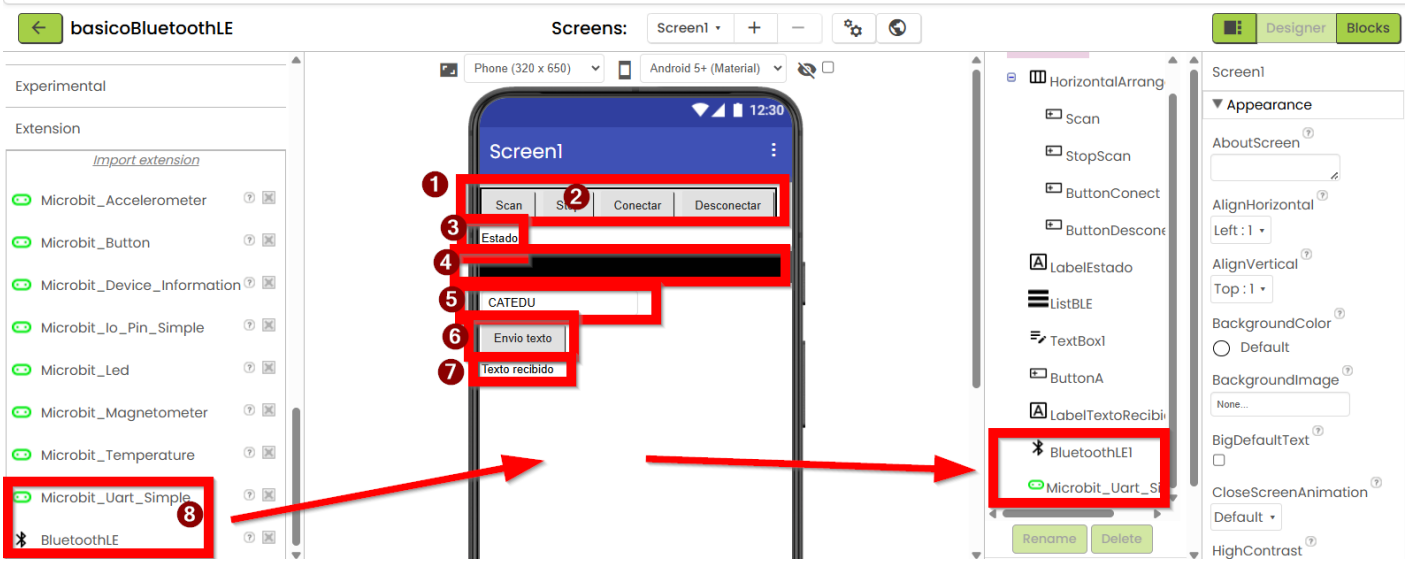
La extensión Bluetooth tiene diversas funciones que tienes su descripción aquí (English). Para poder instalarla, vamos al APP INVENTOR https://ai2.appinventor.mit.edu

BLUETOOTH App Inventor programa

En DESIGNER

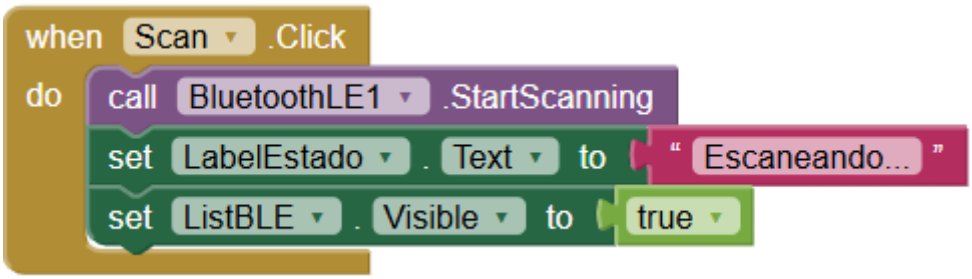
incorporamos:

1. **HorizontalArrangment** para que los botones queden alineados horizontalmente
2. **Botones**
   1. Scan
   2. Stop
   3. Conectar
   4. Desconectar
3. **Label** que dirá el estado de la conexión. Lo llamaremos **LabelEstado**
4. **ListView** que lo llamaremos **ListBLE** donde mostrará los diferentes dispositivos Bluetooth LE que detecta
5. **TextBox** para poner el texto que queramos a enviar a micro:bit
6. Un **botón Enviar** el texto anterior
7. Un **Label** que lo llamaremos **LabelTextoRecibido** que mostrará el mensaje desde micro:bit
8. Añadimos los elementos de las extensiones que hemos instalado anteriormente
   1. Microbit_UART_Simple
   2. BluetoothLE

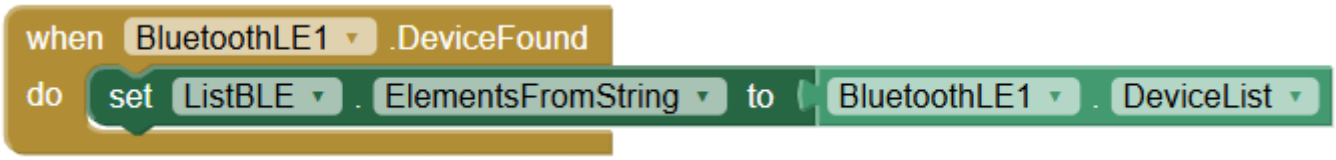


En Blocks

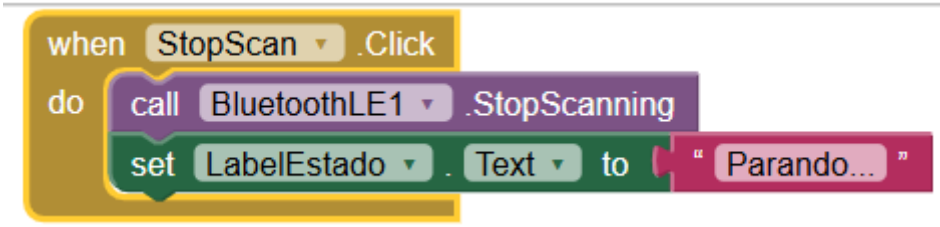
Cuando escaneemos, que el elemento empiece el escaneado y la lista se vuelva visible, además de que LabelEstado diga que esta escaneando



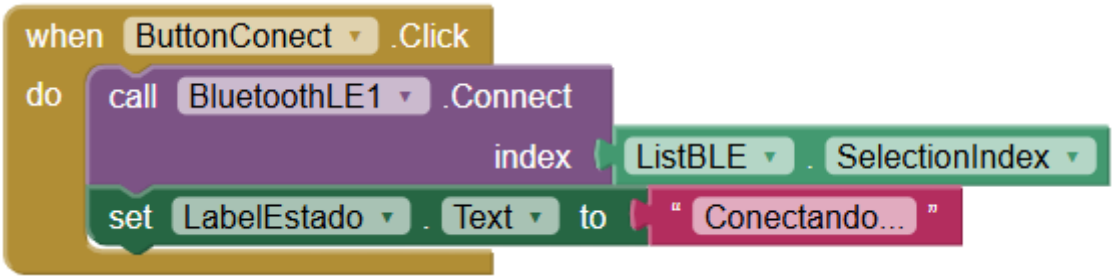
Si ha encontrado un dispositivo, que lo vaya añadiendo a la lista ListBLE



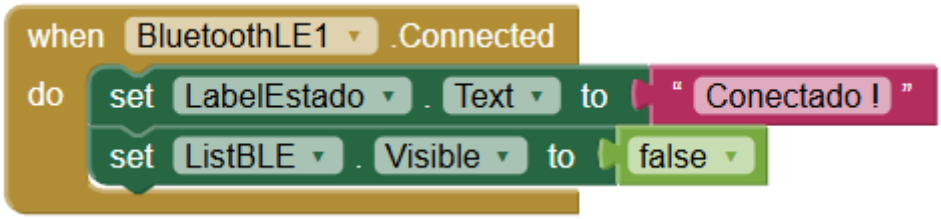
Cuando le digamos que pare, simplemente se lo mandamos al dispositivo y LabelEstado lo informa



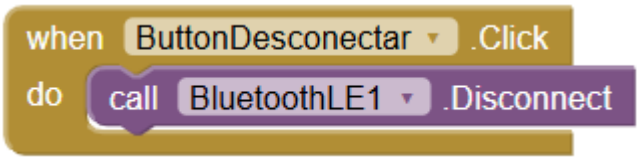
Cuando le demos a conectar, pues conecta con el seleccionado en ListBLE y LabelEstado informa



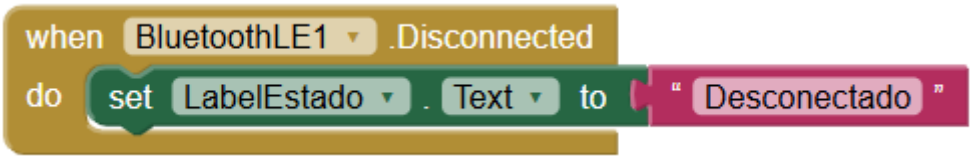
Si conecta, pues LabelEstado informa y ListBLE no es necesaria por lo tanto se oculta, pues entorpece la visión



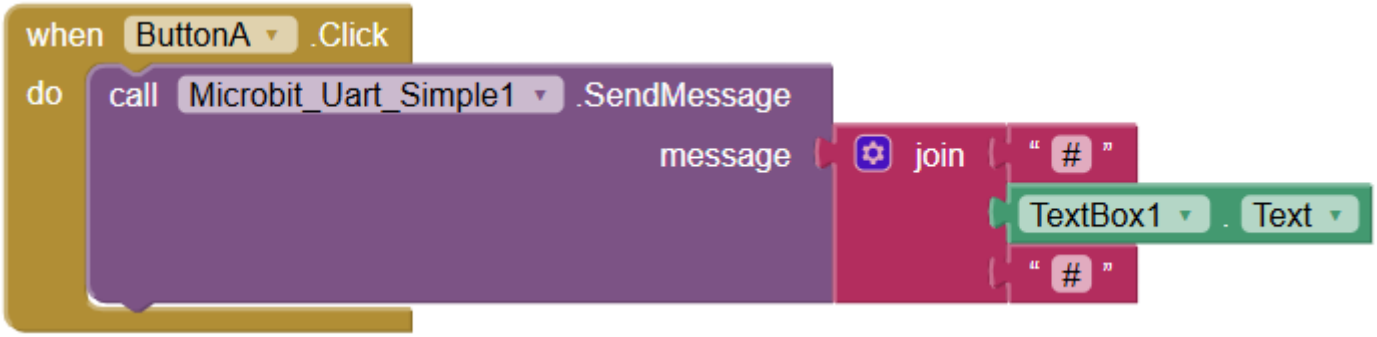
Si queremos desconectar, pues le decimos al elemento BluetoothLE que desconecte



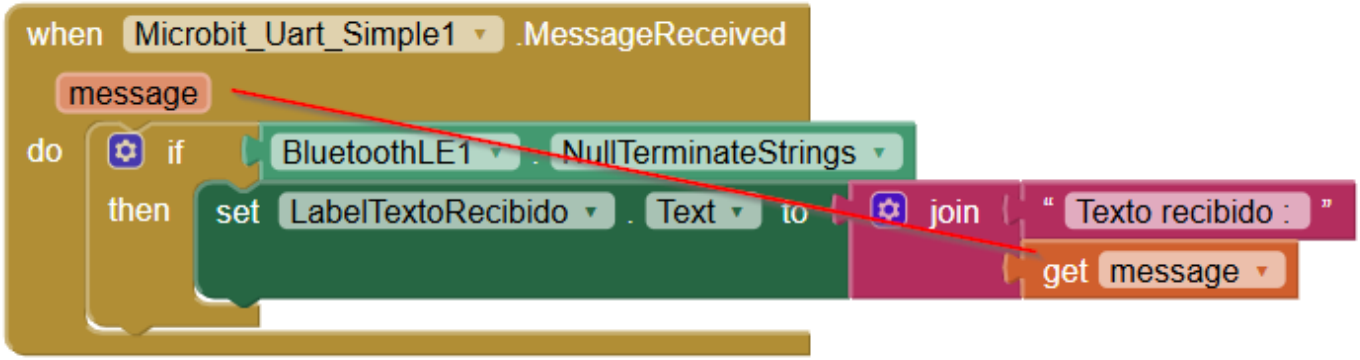
Si se ha desconectado (voluntariamente al dar al botón anterior, o involuntariamente pues el dispositivo se ha desconectado, o esta muy lejos... etc) que informe



Si apretamos el botón enviar, le enviamos el texto que esta en TextBox entre "#" pues así lo hemos definido en el programa Makecode



Si se ha recibido un mensaje, pues que lo visualice, pero primero comprueba que el mensaje no este vacío
NOTA el mensaje "**message**" lo arrastras desde la instrucción "**when**.." tal y como señala la línea roja



basicoBluetoothLE.aia

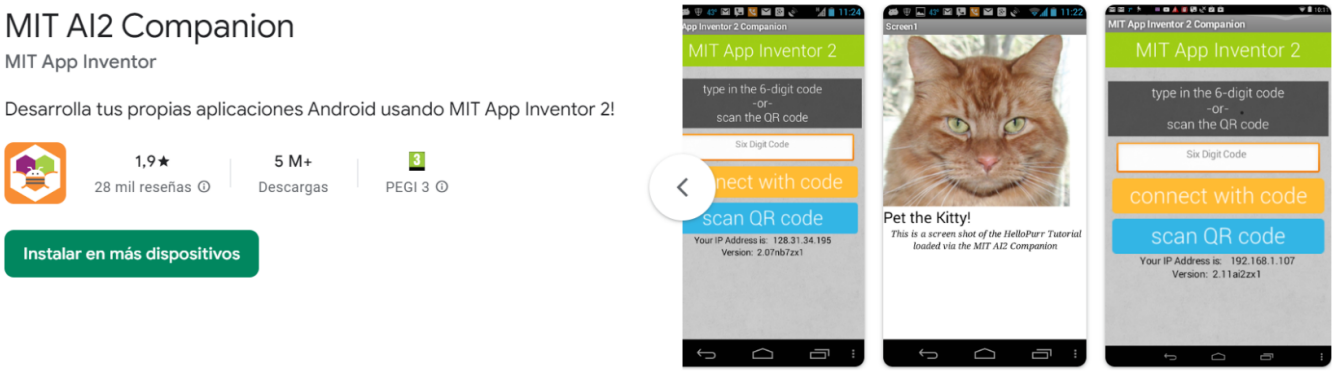
La APP a tu móvil

Tienes dos opciones

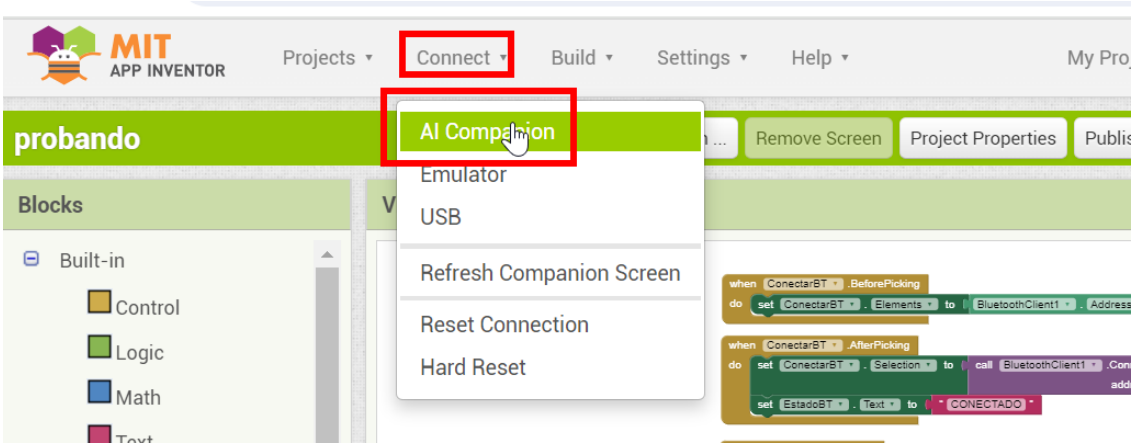
- **EN VIVO CONNECT - AI COMPANION** esta opción es la más rápida, y realmente lo simula a través de la APP INVENTOR.
  - Tienes que tener instalada la APP MIT AI2 COMPANION
  - Se le pasa el código de tu APP a la APP
- **OTRAS OPCIONES**
  - Ver https://appinventor.mit.edu/explore/ai2/setup

OPCIÓN EN VIVO AI COMPANION

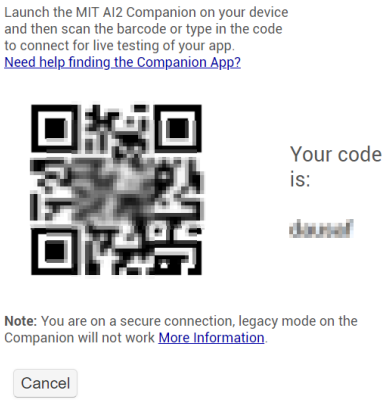
Instalas la APP MIT AI2 COMPANION



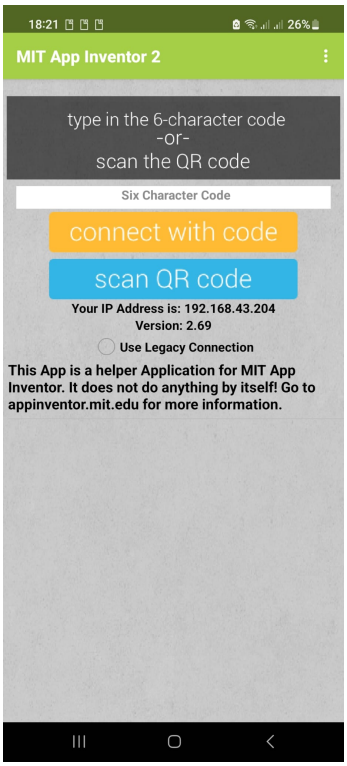
En APP INVENTOR



Y sale un código y un QR asociado al código



Abrimos la APP MIT AI2 COMPANION y metemos el código anterior (o lo escaneamos con el QR)



En APP INVENTOR verás que sale una barra de progreso enviando tu APP a tu móvil. Cuando termina automáticamente lo ejecuta.

A jugar...

https://www.youtube.com/embed/ZS5d-xrDVcA