Prácticas con otros elementos del kit

Copyright 2026

Actividad 12 Puerto de expansión I2C. La LCD de 2x16

Página extraída de Federico Coca Notas sobre ESP32 STEAMakers CC-BY-SA

Enunciado

Utilizaremos las comunicaciones I2C para mostrar textos en la pantalla LCD I2C incluida en el kit.

Teoría

Las comunicaciones I2C

Dado que la placa TdR STEAM dispone de un puerto de expansión I2C (del inglés Inter-Integrated Circuit = Circuito inter-integrado) vamos a explicar un poco en que consiste este sistema de conexionado.

Bus conocido por las siglas I2C, IIC o I²C, es un bus serie de datos desarrollado en 1982 por Philips Semiconductors (hoy NXP Semiconductors, parte de Qualcomm). Se utiliza principalmente internamente para la comunicación entre diferentes partes de un circuito, por ejemplo, entre un controlador y circuitos periféricos integrados. Posteriormente fue adoptado progresivamente por otros fabricantes hasta convertirse en un estándar del mercado con miles de circuitos integrados de diferentes fabricantes.

I2C también se denomina TWI (Two Wired Interface) únicamente por motivos de licencia, denominación introducida por Atmel. No obstante, la patente caducó en 2006, por lo que actualmente no hay restricción sobre el uso del término I2C.

El bus I2C requiere únicamente dos cables o lineas de señal para su funcionamiento, uno para la señal de reloj (SCL, Serial Clock) y otro para el envío de datos (SDA, Serial Data). Ambas líneas precisan resistencias de pull-up hacia Vcc. Cualquier dispositivo conectado a estas líneas es de drenador o colector abierto (Open Collector), lo cual en combinación con las resistencias pull-up, crea un circuito Wired-AND. En la imagen siguiente vemos el diagrama básico de conexionado del bus con algunos ejemplos de dispositivos. La señal de reloj siempre es generada por el circuito que actúa como Master.

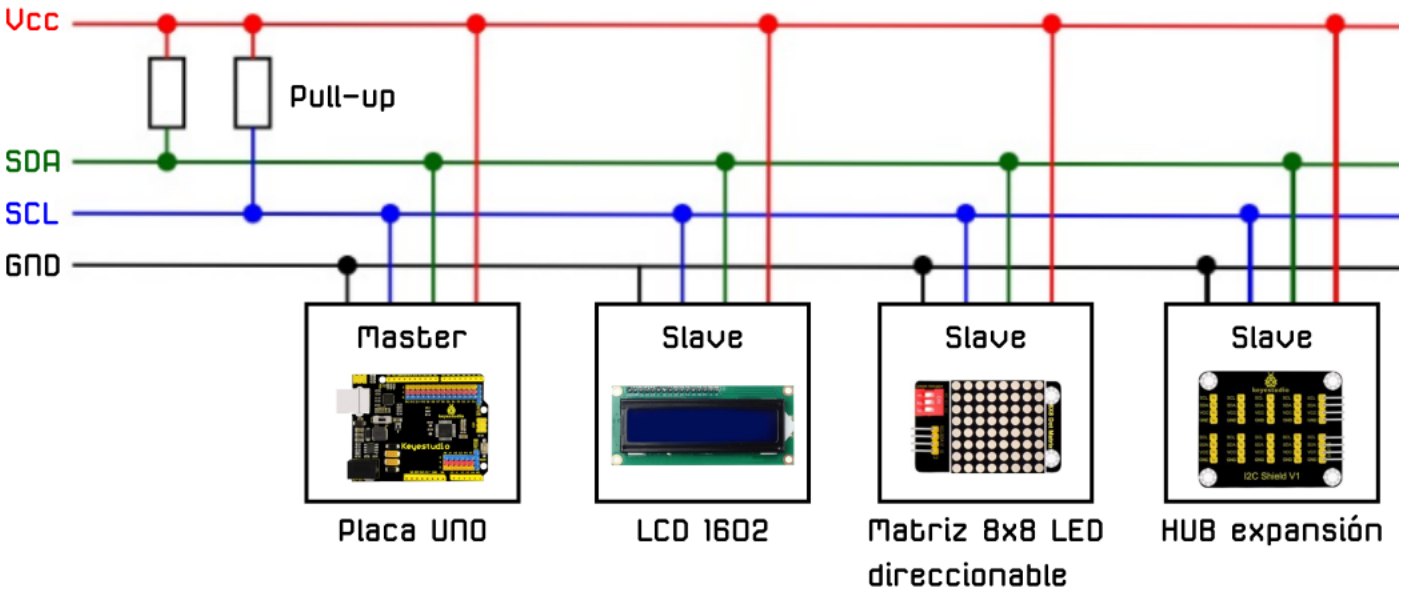


Imagen Federico Coca [Notas sobre ESP32 STEAMakers](#) CC-BY-SA

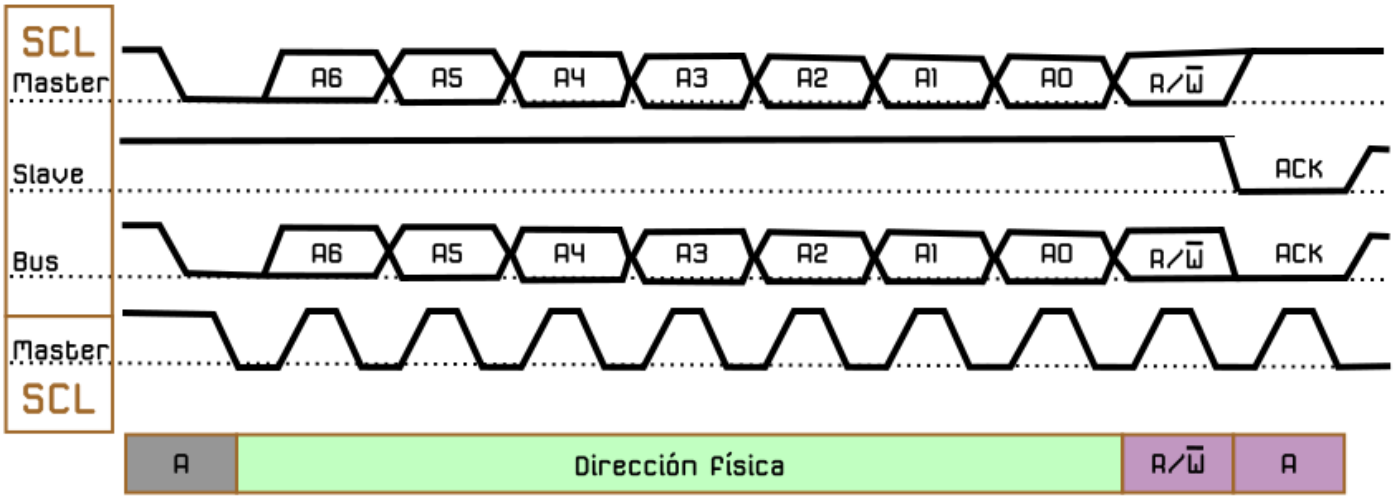
Para ser reconocido en el bus, cada dispositivo dispone de una dirección física, que se emplea para acceder a cada uno de ellos de forma individual. Esta dirección puede ser fijada por hardware, en cuyo caso se pueden modificar los últimos 3 bits mediante “jumpers” o microinterruptores (ejemplo de la matriz de 8x8), o por software.

En general, cada dispositivo conectado al bus debe tener una dirección única. Si tenemos varios dispositivos similares tendremos que cambiar la dirección o, en caso de no ser posible, implementar un bus secundario.

El bus I2C tiene una arquitectura de tipo master-slave, lo que indica que el master es el encargado de controlar al resto de dispositivos tipo slave con los que se comunica y que se comunican con el, teniendo siempre el marte prioridad absoluta. El dispositivo master es el que inicia la comunicación con los slaves. Los slaves no pueden iniciar la comunicación (el master tiene que preguntarles), ni hablar entre si directamente.

El bus I2C debe ser por lo tanto síncrono, es decir debe existir una señal de reloj que controle las comunicaciones. Es el master el que proporciona la señal de reloj, que mantiene sincronizados a todos los dispositivos del bus. De esta forma, se elimina la necesidad de que cada dispositivo tenga su propio reloj, de tener que acordar una velocidad de transmisión y mecanismos para mantener la transmisión sincronizada como en UART o SPI. En la imagen vemos un cronograma ejemplo del funcionamiento del sistema.

Copyright 2026



Cronograma trabajo bus I2C *Imagen Federico Coca Notas sobre ESP32 STEAMakers CC-BY-SA*

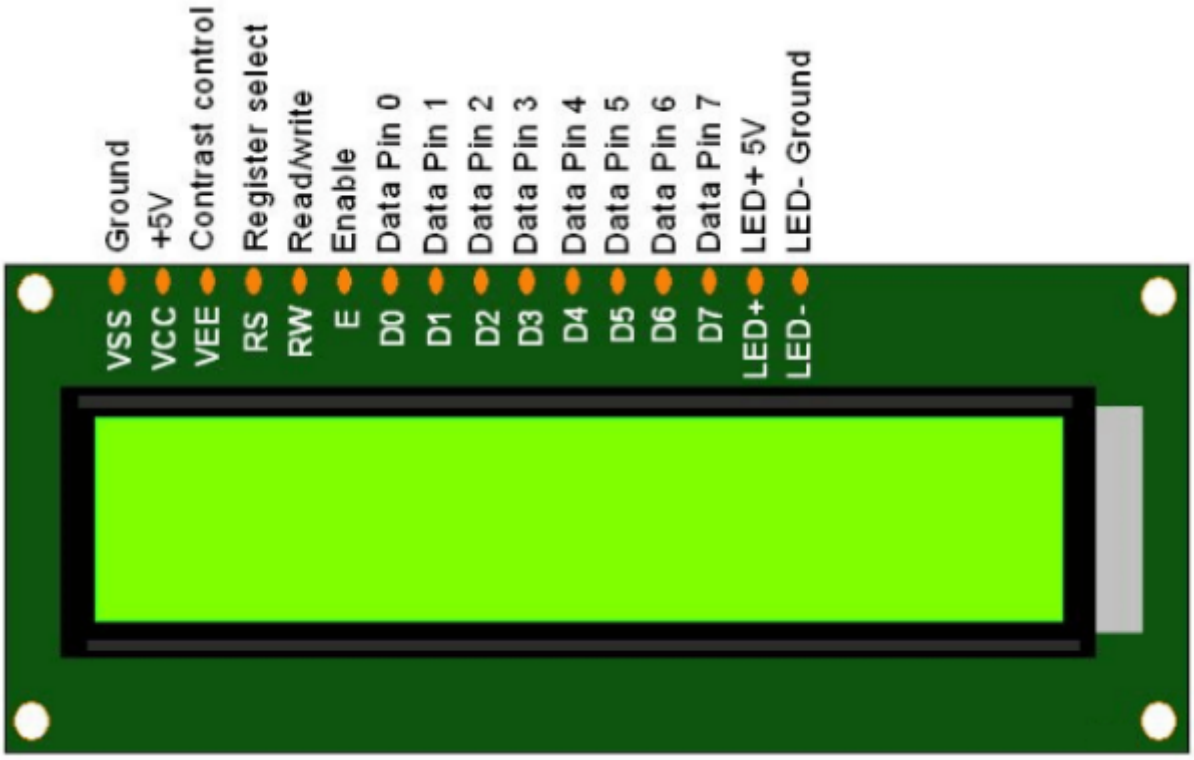
El protocolo de comunicación I2C sigue la siguiente secuencia:

- Primero, el master genera la señal de reloj del bus (SCL).
- Se inicia la comunicación por orden del master al establecer la condición de START, que se produce cuando SDA pasa de uno a cero y se mantiene en cero durante un tiempo.
- El master direcciona a los slaves.
- El master indica si se va a leer o escribir.
- El slave direccionado responde con una señal de conformidad ACK (acknowledge).
- Se transmite los datos byte a byte desde SDA al receptor. Por cada pulso desde SCL se transmite un bit de información.
- El destinatario de la información responde con una señal de conformidad ACK.
- Se acaba la comunicación cuando el master establece la condición de STOP, que se produce cuando SDA, por orden del master pasa de cero a uno y se mantiene en uno durante un tiempo.

Son muchos los dispositivos I2C que se pueden direccionar por este bus I2C, siendo lo más común en los dispositivos para I2C que utilicen direcciones de 7 bits, aunque existen dispositivos de 10 bits, pero es un caso raro. Una dirección de 7 bits implica que se pueden poner hasta 128 (27) dispositivos sobre un bus I2C. Hemos visto que las direcciones son de 8 bits y esto es porque el bit extra de los 7 de la dirección lo emplea el master para informar al slave si va a leer o escribir. Si el bit de lectura/escritura es cero, el dispositivo master está escribiendo en el slave. Si el bit es 1 el master está leyendo desde el slave. La dirección de 7 bit se coloca en los 7 bits más significativos del byte y el bit de lectura/escritura es el bit menos significativo.

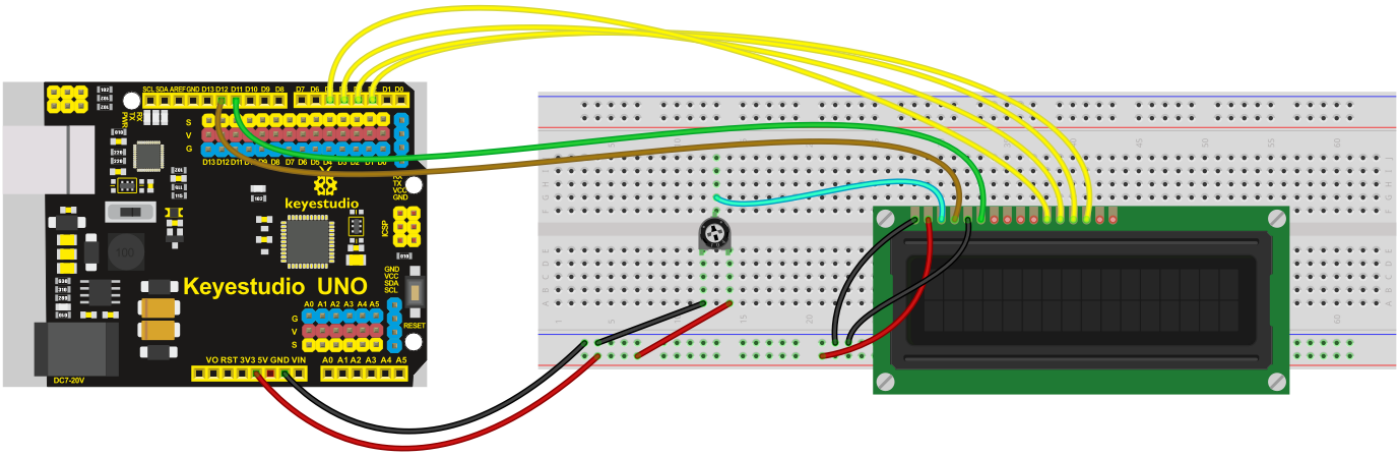
La LCD 1602

Una pantalla LCD (del ingés, Liquid Cristal Display) de 2 líneas de 16 caracteres tiene el aspecto y la distribución de pines que vemos en la imagen siguiente.



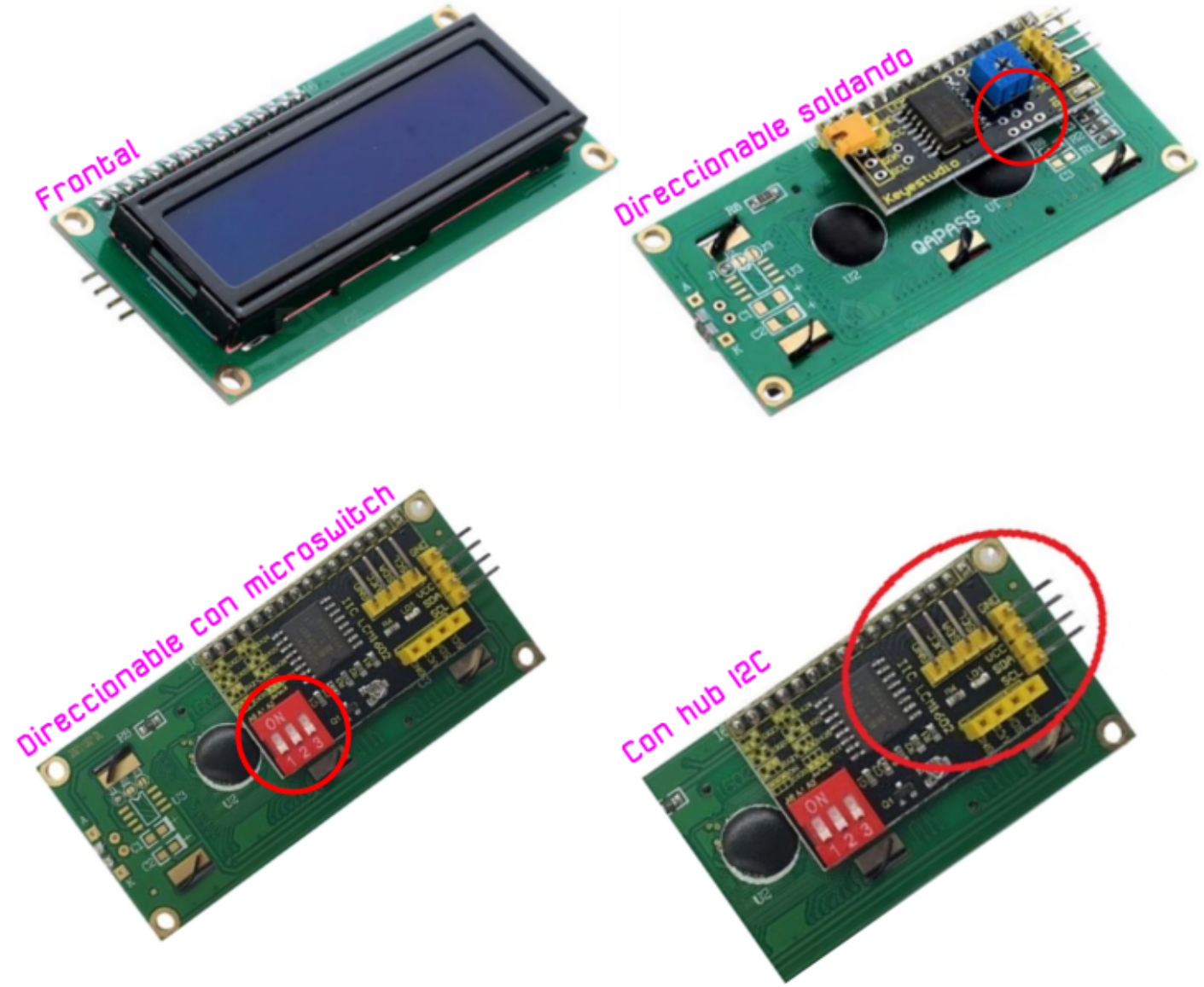
LCD 2x16 *Imagen Federico Coca [Notas sobre ESP32 STEAMakers] CC-BY-SA*

Es evidente que deberíamos utilizar bastantes patillas de nuestra placa tipo UNO (es el formato que tiene la ESP32 STEAMakers) para su control. En la imagen siguiente se muestra el conexionado mínimo necesario en una pantalla de este tipo: 4 bits para datos y dos señales de control En (Enable) y Rs (Register select). La conexión RW la ponemos a GND. Además se debe añadir una resistencia ajustable o un potenciómetro para regular el contraste de la pantalla.



Conexionado mínimo LCD 2x16 *Imagen Federico Coca [Notas sobre ESP32 STEAMakers](#) CC-BY-SA*

Es preferible utilizar alguna de las que tienen el aspecto que vemos en la imagen siguiente:



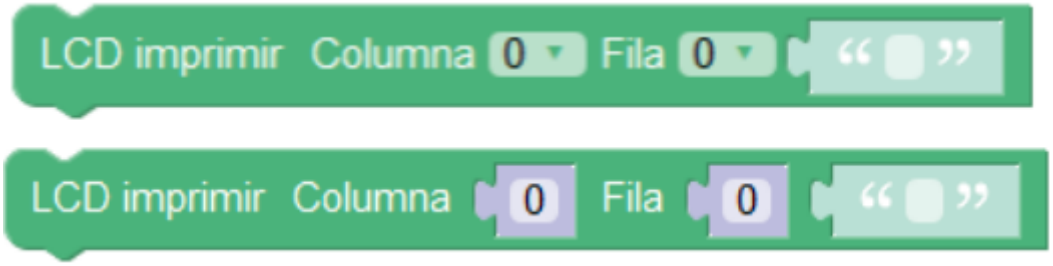
LCD 2x16 con módulo I2C *Imagen Federico Coca [Notas sobre ESP32 STEAMakers](#) CC-BY-SA*

En realidad el conjunto que vemos en la imagen anterior no es mas que una LCD 2x16 a la que se ha incorporado un módulo adaptador de LCD a I2C. Este modulo está especialmente diseñado para poder soldarlo directamente a la LCD y convertirla así en un dispositivo I2C que además ya lleva incorporado el potenciómetro de ajuste de contraste en alguno de los casos.

Este tipo de pantalla requiere cuatro cables para establecer las conexiones, dos cables SDA (datos) y SCL (reloj) para el bus de comunicaciones I2C y los dos cables de alimentación VCC y GND.

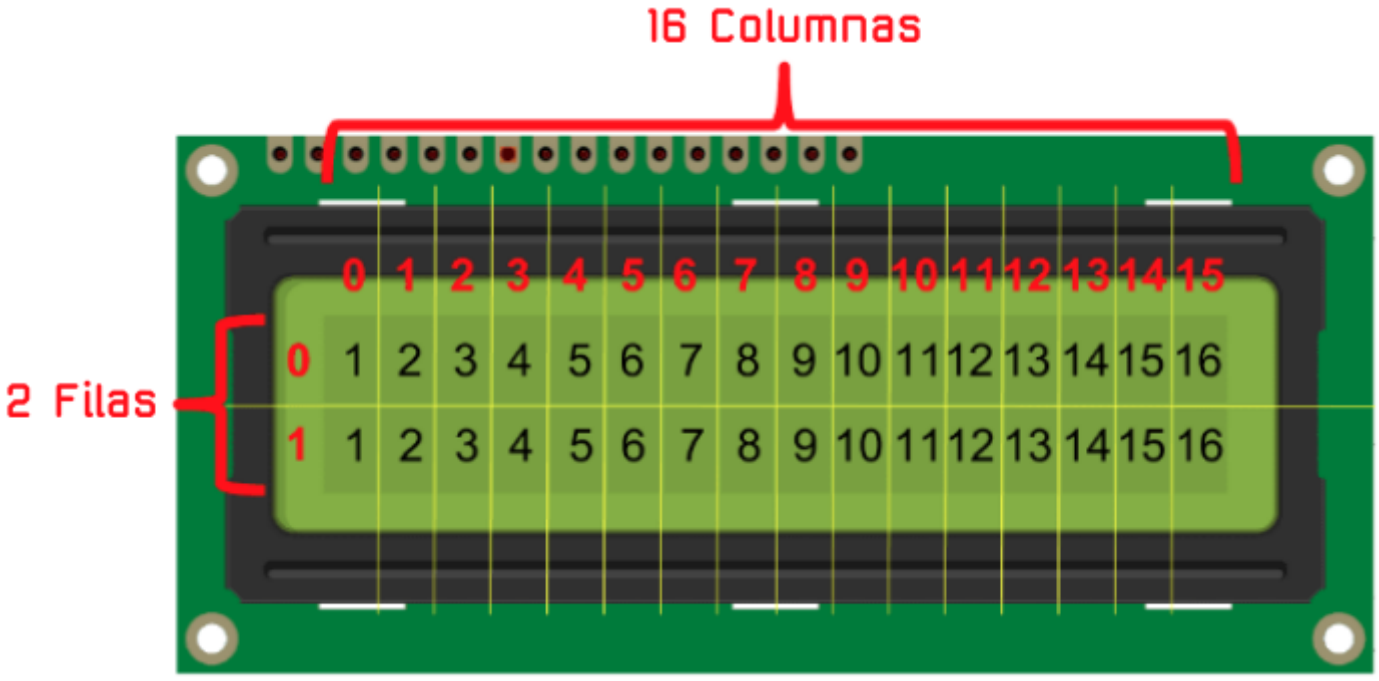
La dirección I2C por defecto de este tipo de módulos puede ser 0x3F o en otros casos 0x27, e incluso hay modelos en los que se puede cambiar. Para un correcto control es muy importante identificar correctamente la dirección I2C de nuestro modulo pues de otra forma nuestro programa no funcionará correctamente.

Una LCD 1602 I2C es muy sencilla de controlar a partir de los bloques que nos proporciona ArduinoBlocks. En la imagen siguiente ponemos como ejemplo los bloques para imprimir un texto o variable en un par fila-columna determinado.



Bloques para imprimir en un par fila-columna *Imagen Federico Coca* *Notas sobre ESP32 STEAMakers* *CC-BY-SA*

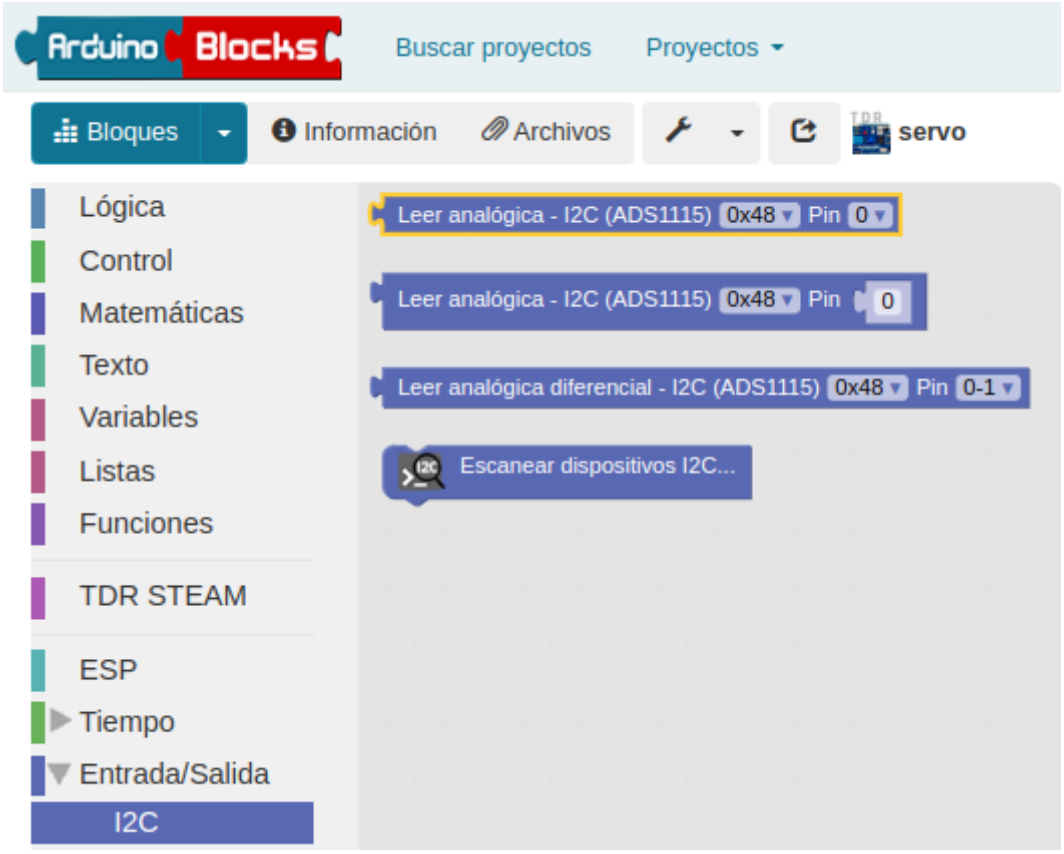
En la siguiente figura se muestra el sistema de distribución de filas y columnas.



Sistema de coordenadas en una LCD 1602 *Imagen Federico Coca* *Notas sobre ESP32 STEAMakers* *CC-BY-SA*

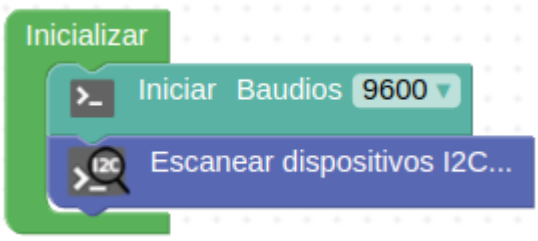
Escanear dispositivos I2C

Si no conocemos la dirección especifica de nuestro módulo podemos utilizar un pequeño programa que llamaremos Escaner-I2C y que se encargará de identificar la dirección I2C y todos los dispositivos I2C conectados a nuestra placa. Debemos crear un proyecto, en esta ocasión, de tipo "Arduino UNO" para tener disponible el menú I2C que nos de acceso al bloque "Escanear dispositivos I2C..." tal y como vemos en la imagen siguiente:



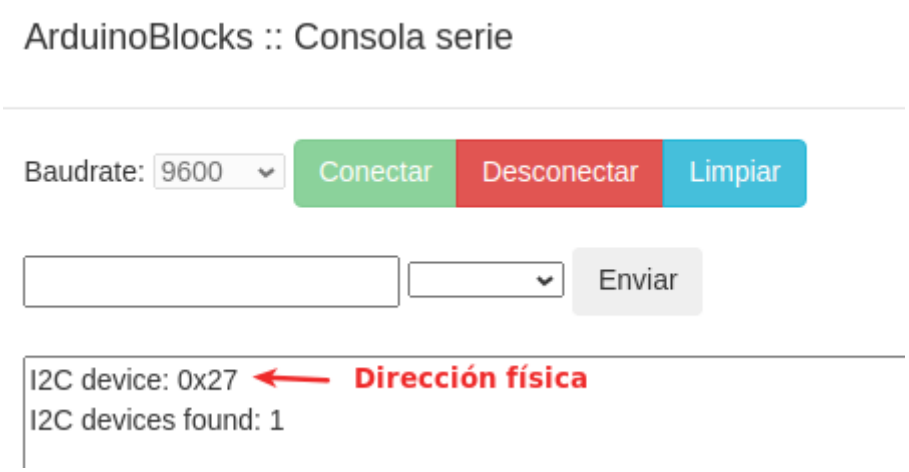
Menu I2C *Imagen Federico Coca Notas sobre ESP32 STEAMakers CC-BY-SA*

Un sencillo programa como el de la imagen siguiente nos permite ver los dispositivos I2C conectados y su dirección física asociada. El programa está disponible como Escaner-I2C.



Programa Escaner-I2C *Imagen Federico Coca Notas sobre ESP32 STEAMakers CC-BY-SA*

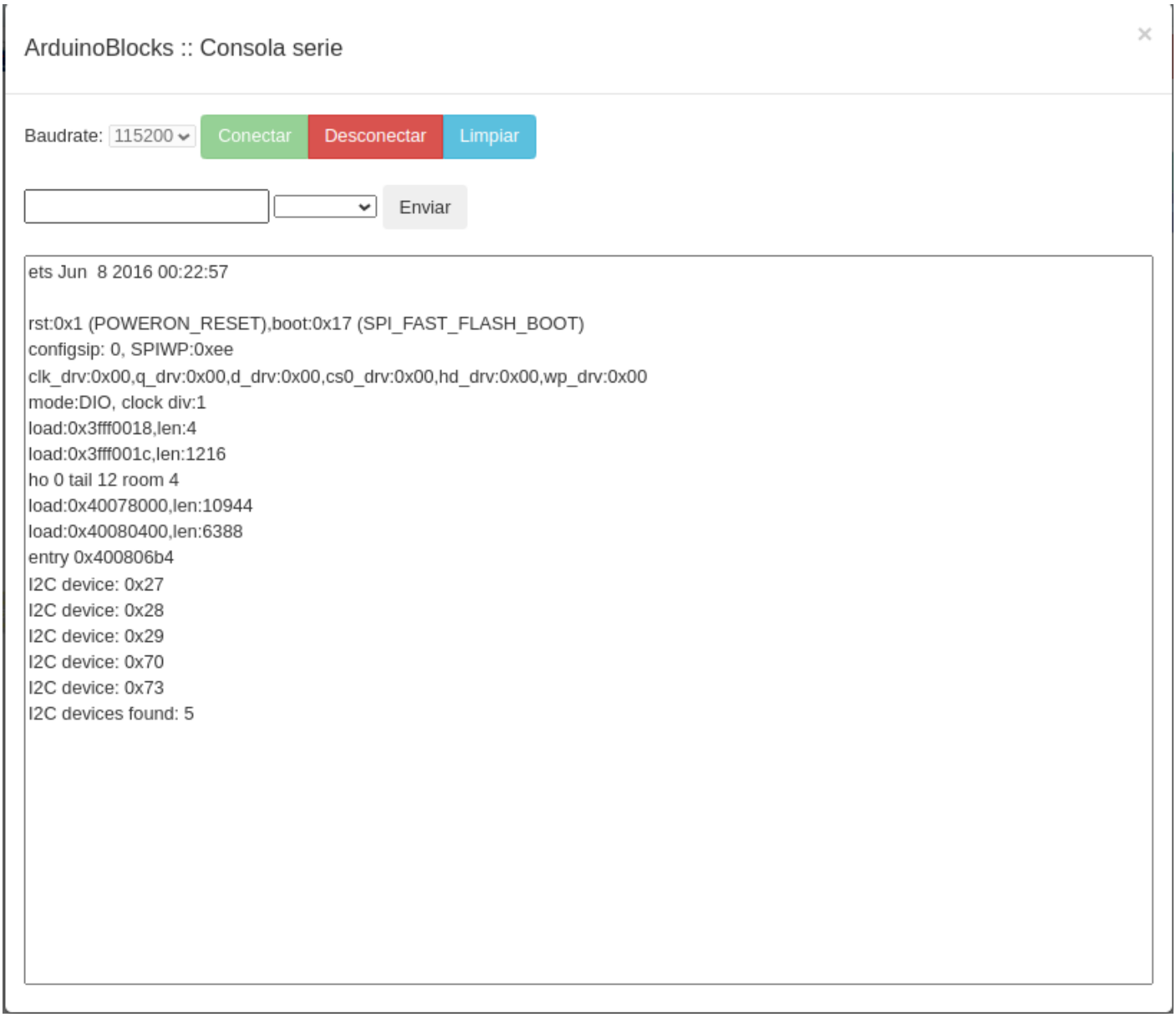
Si conectamos la consola el resultado de tener la placa TdR STEAM con una LCD I2C conectada debe ser similar al de la imagen siguiente:



Consola para el programa Escaner-I2C *Imagen Federico Coca [Notas sobre ESP32 STEAMakers] CC-BY-SA*

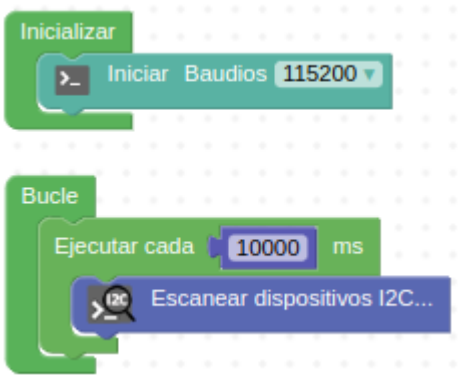
> **IMPORTANTE** En el caso de la placa ESP32 STEAMakers no ocurre como en las tipo Arduino, que cuando conectamos la consola se efectúa un reset automático de la placa por lo que es muy posible que el programa de escaneo nos muestre una consola sin resultado alguno o en blanco. Simplemente, con la consola ya conectada, **hacemos un reset físico en la placa** y el problema quedará solucionado.

En la figura siguiente tenemos el resultado que se muestra con el programa anterior cargado, conectar la consola y hacer reset. En este caso vemos los cinco dispositivos conectados a través de un hub y la placa TdR STEAM.



Consola para el programa Escaner-I2C con 5 dispositivos *Imagen Federico Coca Notas sobre ESP32 STEAMakers CC-BY-SA*

Otra opción para hacer un escáner I2C que no necesita el reset es el programa que vemos en la figura siguiente y que lleva al bucle la orden de escaneo que se va a ejecutar cada 10 segundos, aunque este tiempo puede ser otro cualquiera.



Programa Escaner-I2C en el bucle con 5 dispositivos *Imagen Federico Coca Notas sobre ESP32 STEAMakers CC-BY-SA*

En la figura siguiente vemos el resultado en la consola y como coincide con el anterior.

Consola para el programa Escaner-I2C en el bucle con 5 dispositivos

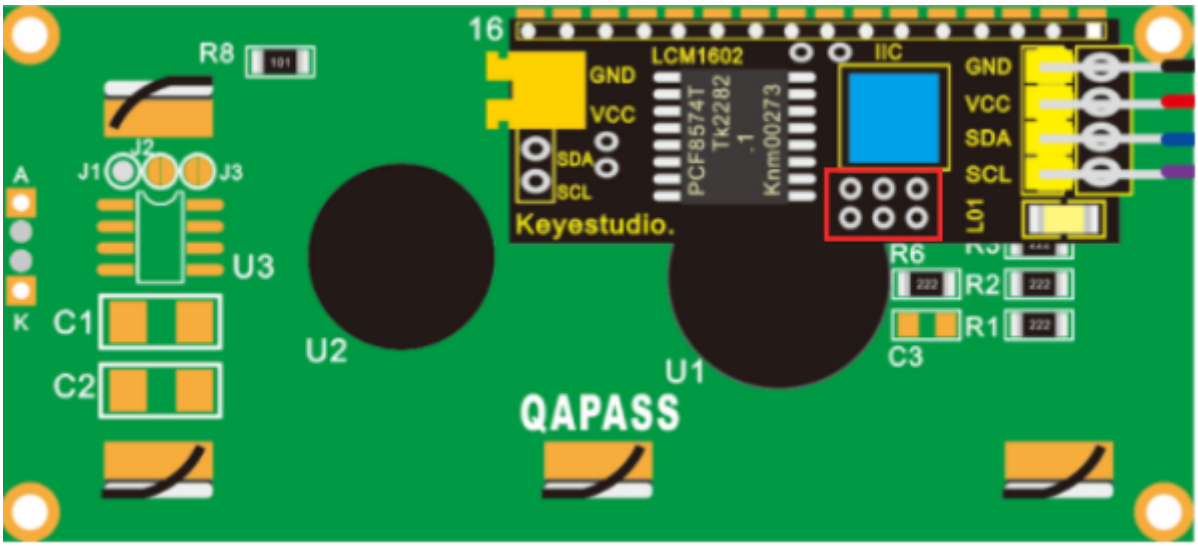
Imagen Federico Coca Notas sobre ESP32 STEAMakers CC-BY-SA

Cambiar la dirección física del dispositivo I2C

Esta tarea nos va a resultar especialmente útil si disponemos de LCDs I2C con una dirección fijada de fábrica y queremos conectar varias de ellas en nuestro proyecto.

LCD SIN micro interruptores

La parte posterior de la LCD 1602 de Keyestudio tiene el aspecto de la imagen siguiente. Nos fijaremos especialmente en los tres grupos de agujeros enmarcados en rojo. Aunque en este caso no vienen nombrados los vamos a denominar A0, A1 y A2 de izquierda a derecha.

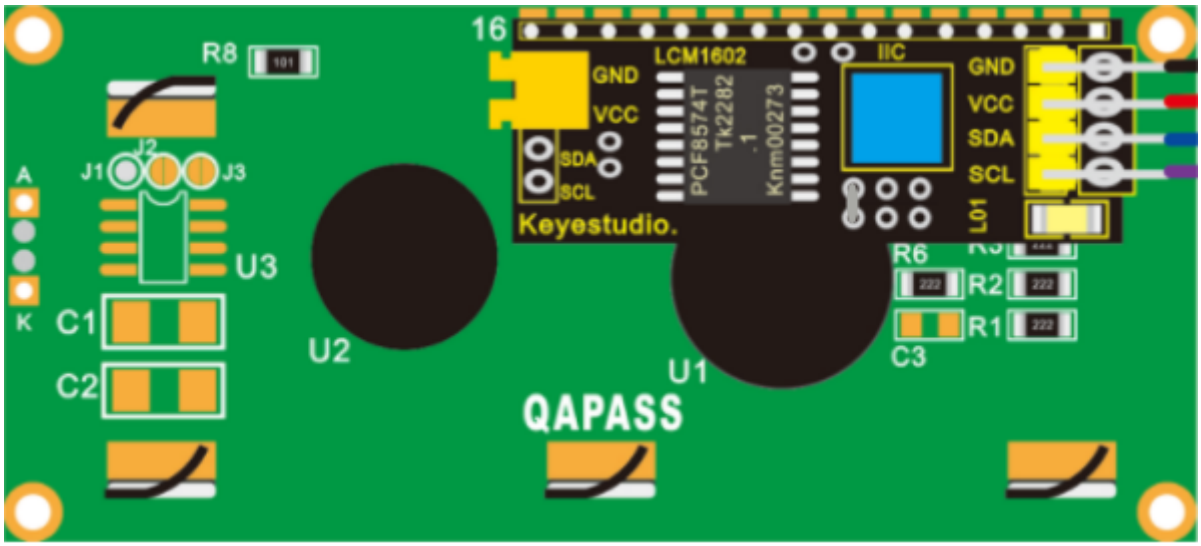


Pads configuración dirección física en la parte posterior LCD I2C Imagen Federico Coca [Notas sobre ESP32 STEAMakers](#) CC-BY-SA

La dirección por defecto de fábrica en este caso es la 0x27, pero se puede cambiar alterando la situación de conexionado de estos agujeros entre si según la tabla siguiente:

A2	A1	A0	Dirección
0	0	0	0x27
0	0	1	0x26
0	1	0	0x25
0	1	1	0x24
1	0	0	0x23
1	0	1	0x22
1	1	0	0x21
1	1	1	0x20

Para establecer los unos de la tabla anterior basta con cortocircuitar los dos pads correspondientes. En la imagen siguiente se ha establecido la dirección física como 0x26.



Dirección 0x26 *Imagen Federico Coca [Notas sobre ESP32 STEAMakers] CC-BY-SA*

LCD CON micro interruptores

La parte posterior de la LCD 1602 de Keyestudio con micro interruptores para cambiar su dirección física tiene el aspecto de la imagen siguiente. Nos fijaremos especialmente en el recuadro azul donde están los microinterruptores que permiten cambiar el estado del bit y encima de los mismo está la información de la dirección física que se asigna a cada combinación. La tabla es exactamente la misma que en el caso anterior.

Configuración dirección física en la parte posterior LCD I2C con microinterruptores

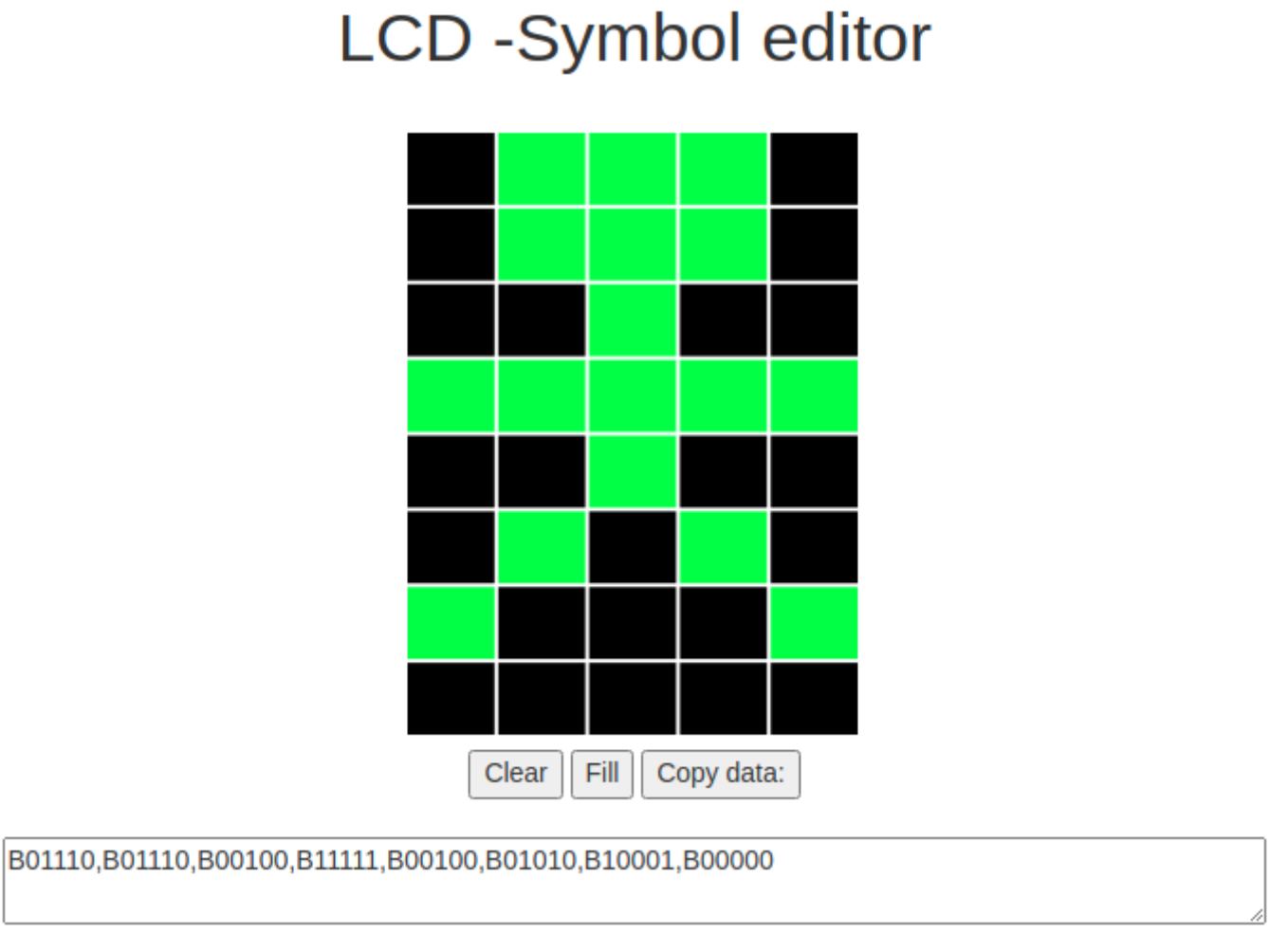
Imagen Federico Coca [Notas sobre ESP32 STEAMakers] CC-BY-SA

Definición de símbolos en la LCD

Dentro de los bloques del menú Visualización -> Pantalla LCD está el de "definir símbolo", que permite definir uno de los 8 símbolos personalizables que puede almacenar el micro de la pantalla LCD. El símbolo se define por un mapa de bits (unos y ceros indicando cada píxel del símbolo). Los

símbolos tienen una resolución de 5x8 píxeles (blanco o negro).

En ArduinoBlocks disponemos de una herramienta que nos ayuda a definir nuestros propios símbolos y podemos acceder a ella desde herramientas o haciendo clic derecho sobre el bloque, desplegándose en cualquier caso un editor muy sencillo de usar y que vemos con un ejemplo en la imagen en la siguiente:



Ejemplo de símbolo creado con el editor *Imagen Federico Coca [Notas sobre ESP32 STEAMakers](#) CC-BY-SA*

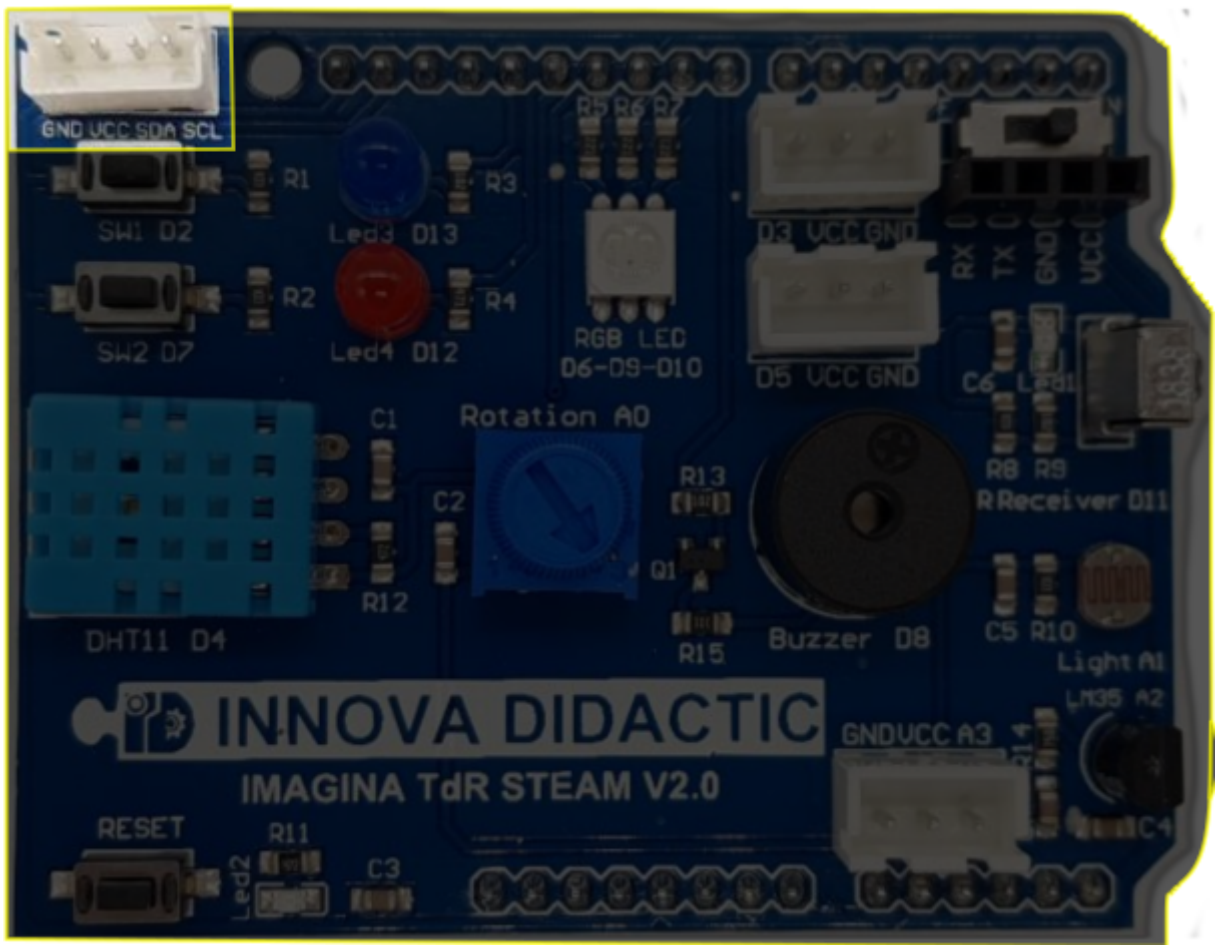
Para tener el símbolo disponible simplemente copiamos la cadena generada en el lugar correspondiente del bloque.

En la TdR STEAM

En la placa TdR STEAM tenemos el conector I2C con los mismo pines de conexión que la LCD y debemos respetar el orden de conexionado indicado en la siguiente tabla:

TdR STEAM	LCD 1602	Color cable
GND	GND	Negro
Vcc	Vcc	Rojo
SDA	SDA	Amarillo
SCL	SCL	Blanco

La ubicación del conector I2C en la TdR STEAM lo vemos en la imagen siguiente:

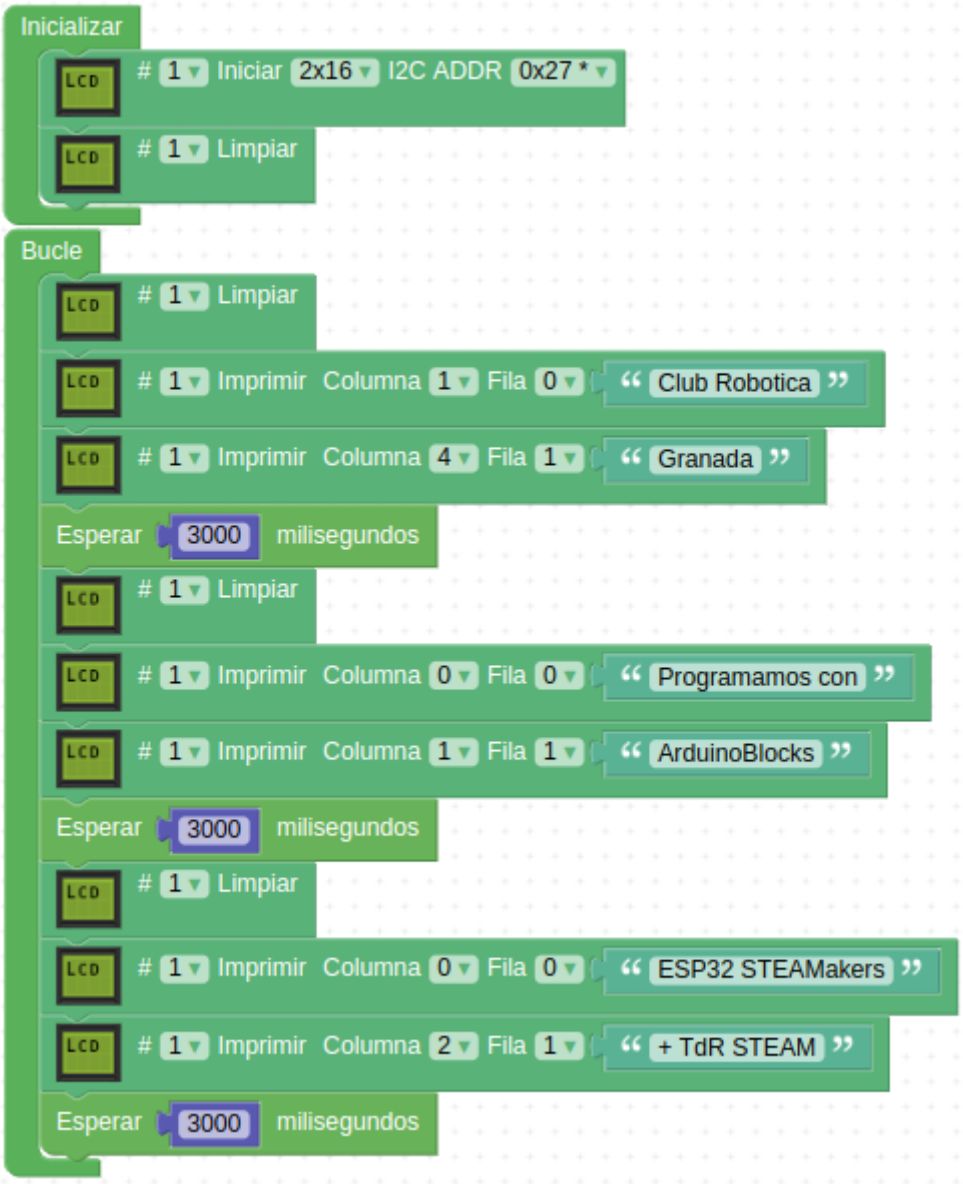


Conector I2C en la TdR STEAM *Imagen Federico Coca* *Notas sobre ESP32 STEAMakers* *CC-BY-SA*

Recordemos que en el caso de la LCD con micro interruptores o LCD direccionable dispone de un pequeño hub I2C que permite conectar dos dispositivos I2C mas. El término direccionable no me parece una denominación muy correcta dado que el otro tipo también es direccionable, aunque tengamos que soldar.

Programando la actividad

Como primer ejemplo de uso vamos a mostrar una determinada información en la pantalla LCD, en concreto vamos a mostrar una serie de seis textos en ambas filas que se irán alternando cada tres segundos en la pantalla. El programa de la imagen siguiente lo tenemos disponible en ESP32-SM-Actividad-inicial.

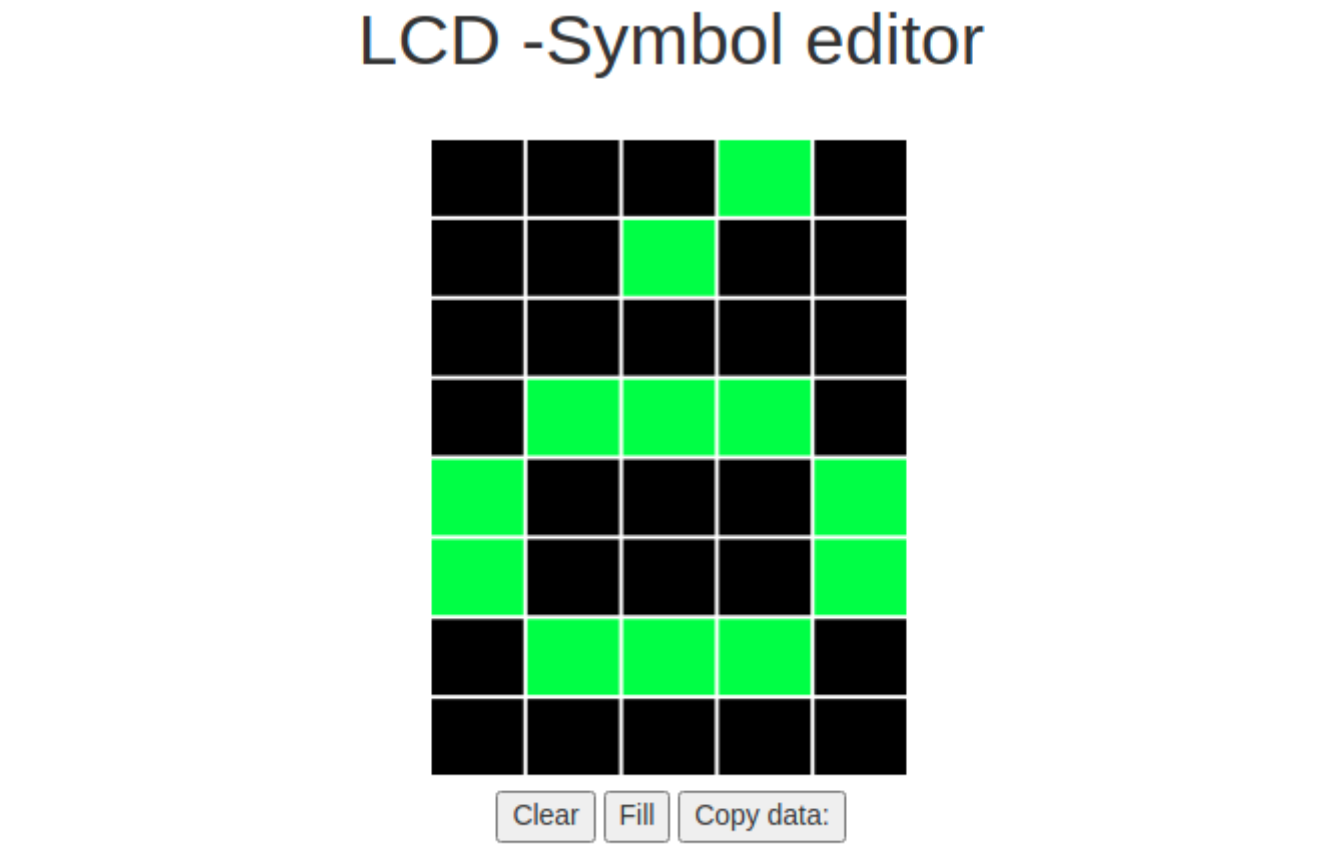


Actividad inicial *Imagen Federico Coca Notas sobre ESP32 STEAMakers CC-BY-SA*

Retos de ampliación

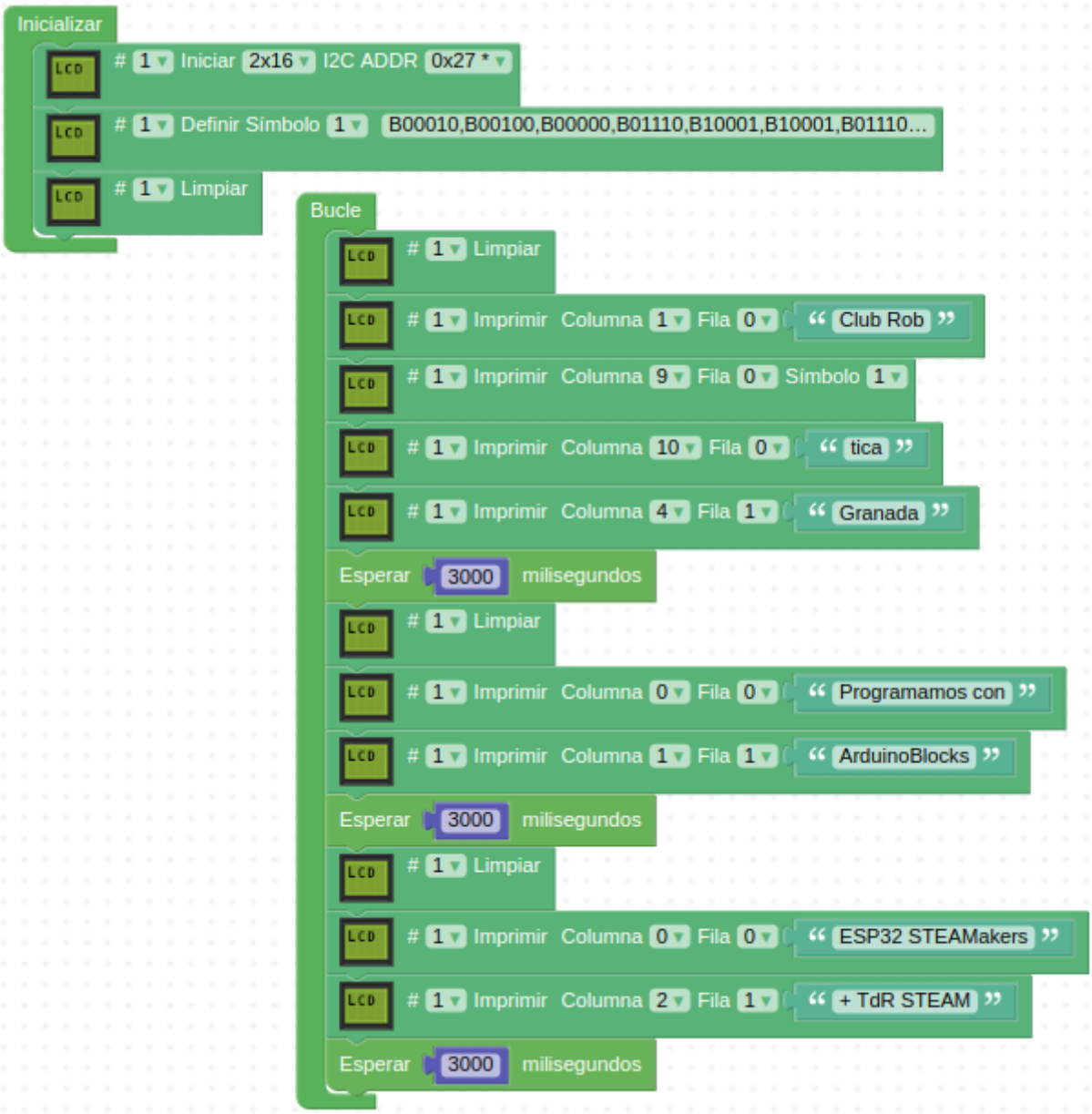
LCD.R1. Corregir el error ortográfico en la palabra "*Robotica*" de la actividad anterior. Para ello tenemos que definir nuestro propio símbolo de "ó" (o acentuada).

Definimos el símbolo requerido tal y como vemos en la imagen siguiente:



Símbolo 'o' acentuada *Imagen Federico Coca* *Notas sobre ESP32 STEAMakers* *CC-BY-SA*

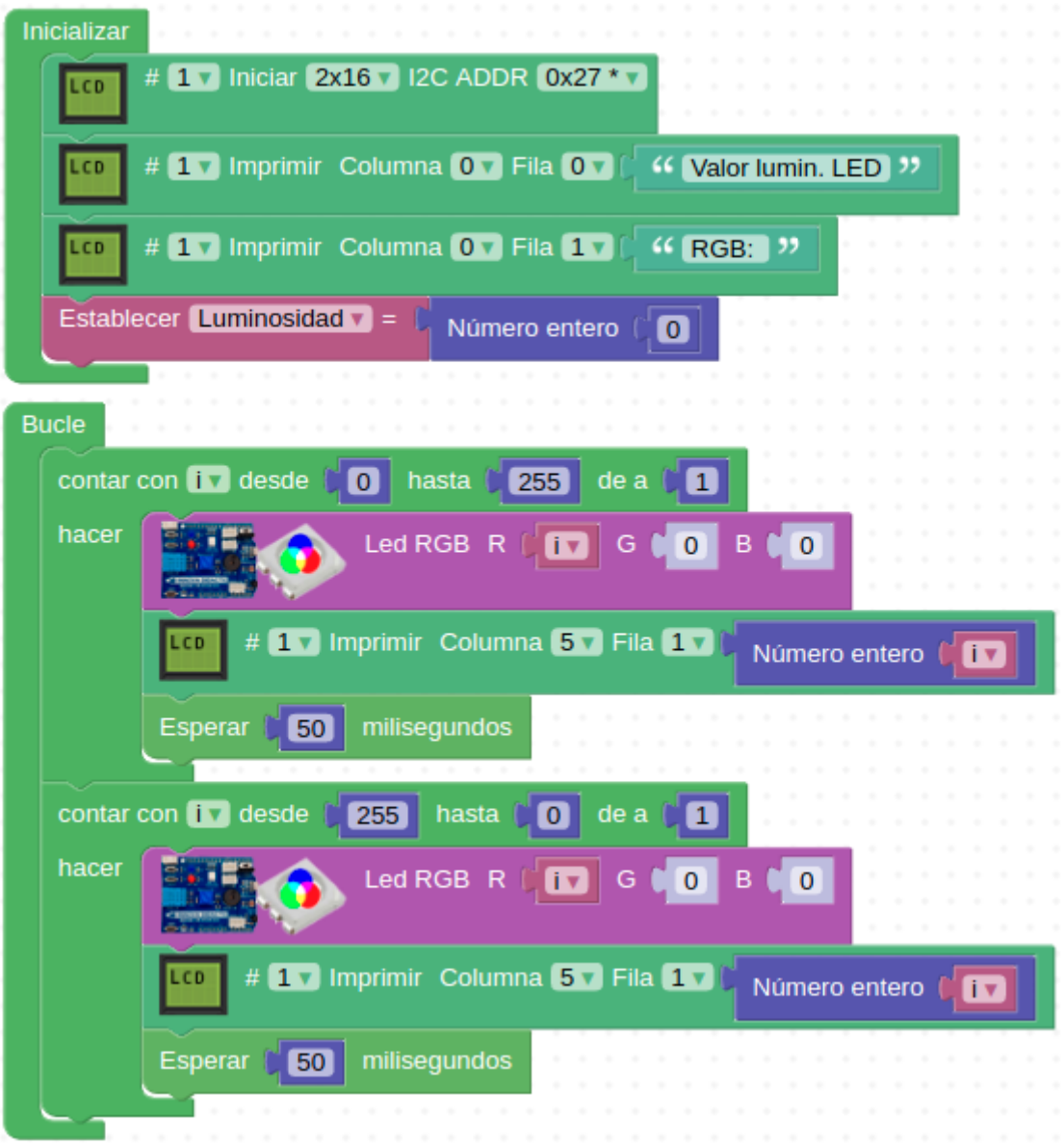
El programa resuelto es el de la figura siguiente:



Solución al reto 1 de la actividad LCD *Imagen Federico Coca Notas sobre ESP32 STEAMakers CC-BY-SA*

LCD.R2. Realizar un control de iluminación del LED RGB mediante PWM y mostrar por pantalla el valor de la iluminación en cada instante. Se realizará el control desde el mínimo de iluminación a máximo y cuando se alcance el máximo bajamos otra vez a mínimo.

El programa resuelto es el de la figura siguiente:



Solución al reto 2 de la actividad LCD *Imagen Federico Coca [Notas sobre ESP32 STEAMakers](#) CC-BY-SA*

Se ha resuelto con un contador ascendente y otro descendente sobre el LED rojo. Se pueden probar diferentes opciones de LEDs o incluso combinación de ellas. También se anima al lector a comprobar que ocurre si quitamos la opción "Número entero" y ponemos directamente la variable del bucle y buscar la explicación.

Actividad 13 Sensor de sonido o micrófono

Actividad 13 Visualización del nivel sonoro

Vamos a hacer este programa:

vernivelsonoro.abp

https://www.youtube.com/embed/u3ipwFAjf9Y

> Página extraída de Federico Coca Notas sobre ESP32 STEAMakers CC-BY-SA

Actividad 13 bis Enunciado

Realizaremos pruebas básicas de detección de nivel de sonido añadiendo esta opción externa al pin analógico A3 disponible en el conector correspondiente.

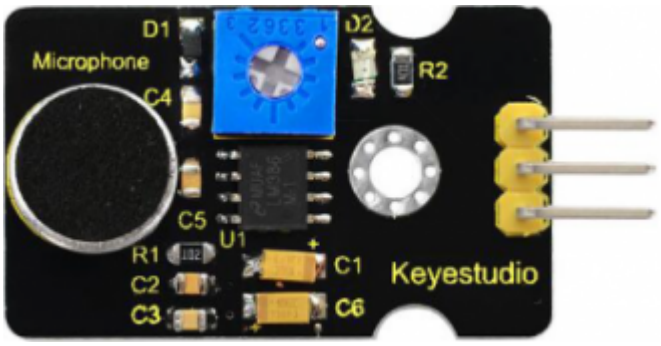
Teoría

Un micrófono es un transductor (dispositivo que convierte energía de una forma a otra) que convierte la energía sonora en señales eléctricas. Micrófonos hay una amplia diversidad tanto en formas como tamaños. Dependiendo de la aplicación, un micrófono puede utilizar diferentes tecnologías para convertir sonidos en señales eléctricas.

Para el caso de aplicaciones con placas tipo Arduino suelen usarse sensores basados en el micrófono de condensador electret que es un condensador de placas paralelas y trabaja como una capacitancia variable. Se forma con una placa fija (placa trasera) y una movible (diafragma) con

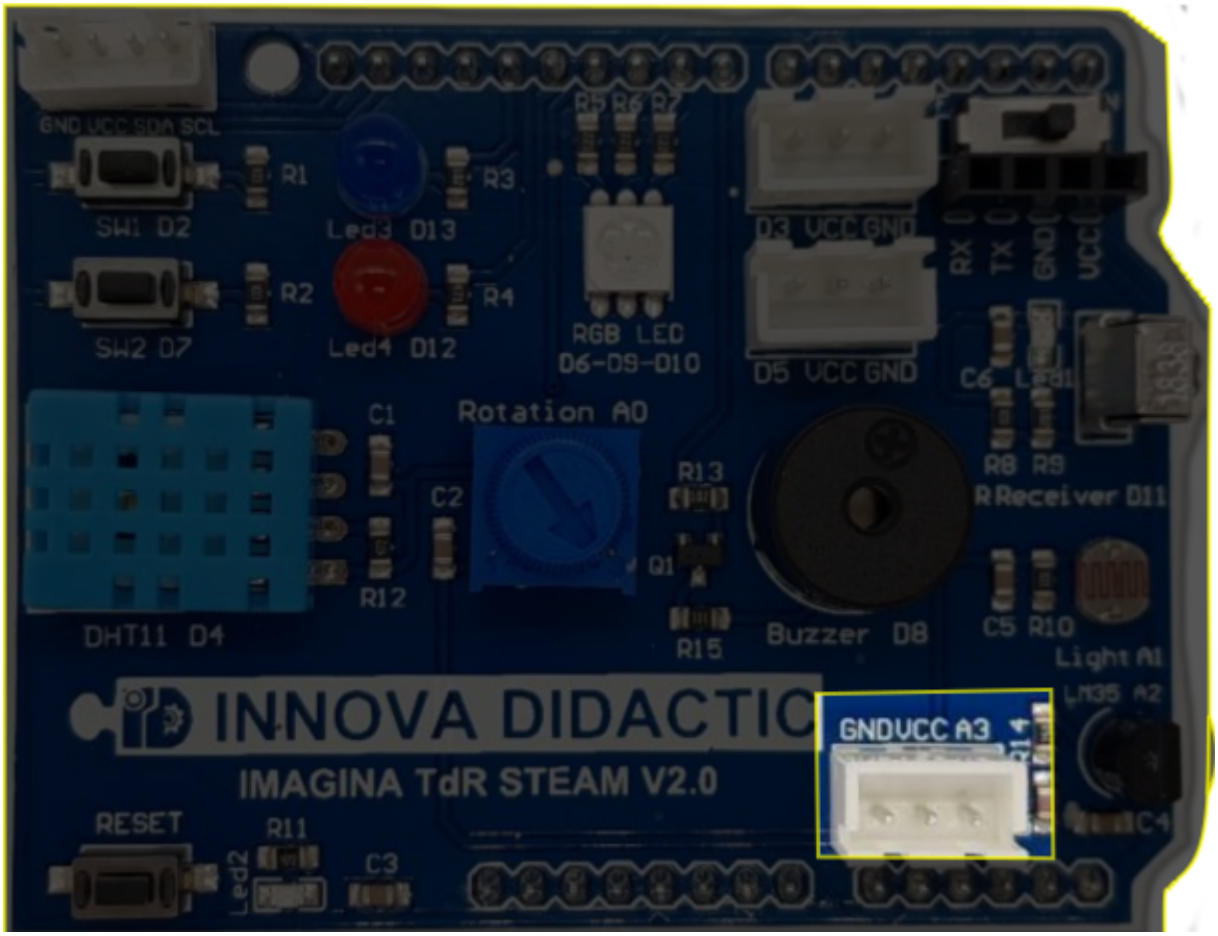
una pequeña separación entre ellas. Cuando el sonido golpea al diafragma este se mueve cambiando así la capacitancia entre las placas.

El sensor microfónico de keyestudio se utiliza normalmente para detectar el nivel de ruido en el ambiente. El pin S del mismo es una salida analógica, es decir, una tensión de salida en tiempo real del micrófono. El sensor viene con un potenciómetro que permite ajustar la ganancia de la señal dentro de un determinado rango. Su aspecto es el de la imagen siguiente:



Sensor de sonido con micrófono KS0035 con potenciómetro *Imagen Federico Coca* *Notas sobre ESP32 STEAMakers* *CC-BY-SA*

En la TdR STEAM



El conector micrófono externo en A3 *Imagen Federico Coca Notas sobre ESP32 STEAMakers CC-BY-SA*

Programando la actividad

Como programa sencillo inicial vamos simplemente a enviar a la consola el nivel de sonido que detecte nuestro micrófono. El programa de la imagen siguiente esta disponible como ESP32-SM-micro-inicial.

Actividad inicial con el micro *Imagen Federico Coca [Notas sobre ESP32 STEAMakers] CC-BY-SA*

Retos de ampliación

Micro.R1. Modificar el programa de la actividad inicial para que muestre los datos por la LCD.

Micro.R2. Partiendo del programa del reto 2 de la actividad LCD (LCD.R2) modificarlo para que muestre los datos del micrófono tanto al final de la cuenta ascendente como de la descendente.

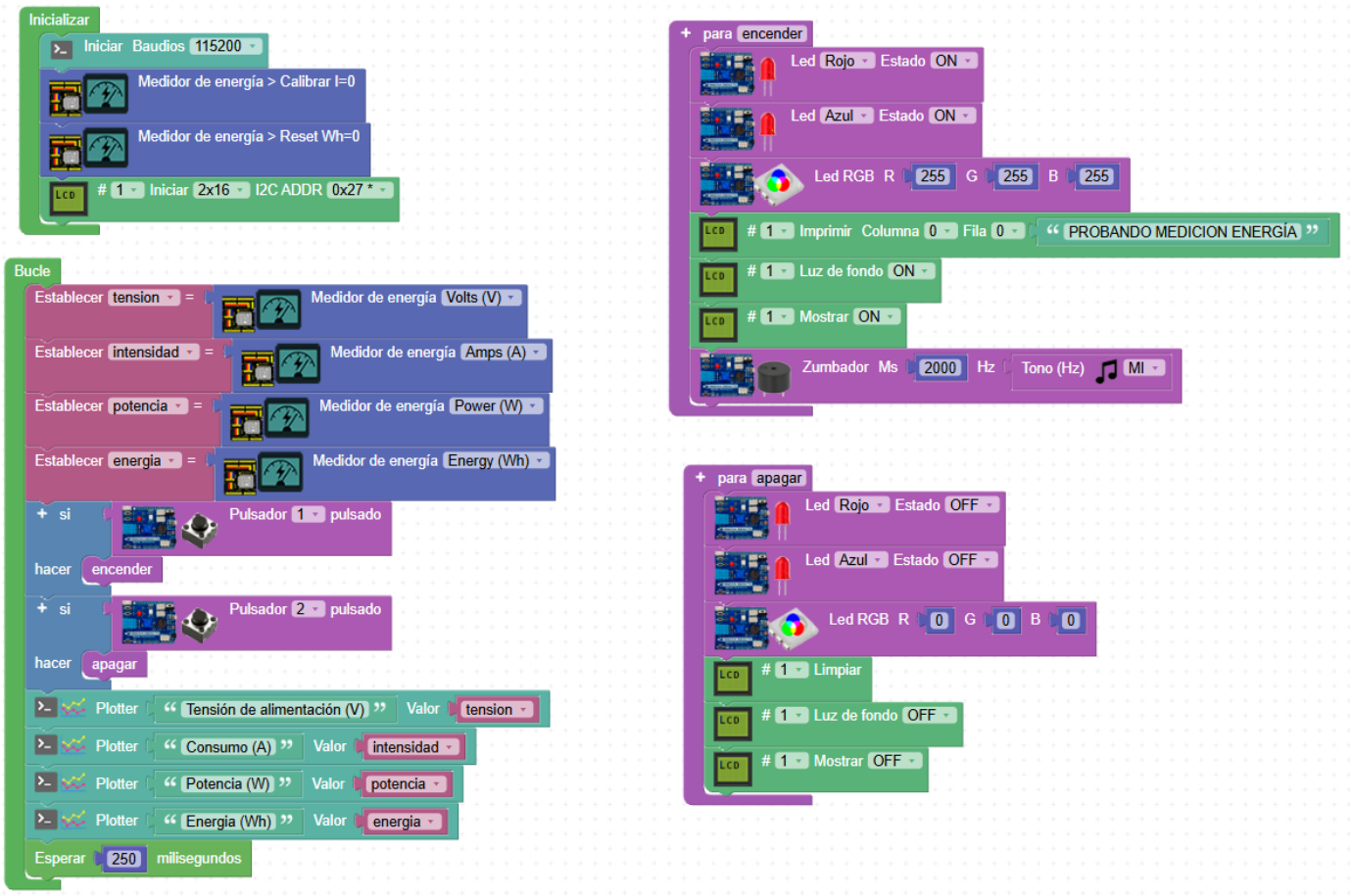
Copyright 2026

Actividad 14 Sensores internos de ESP32

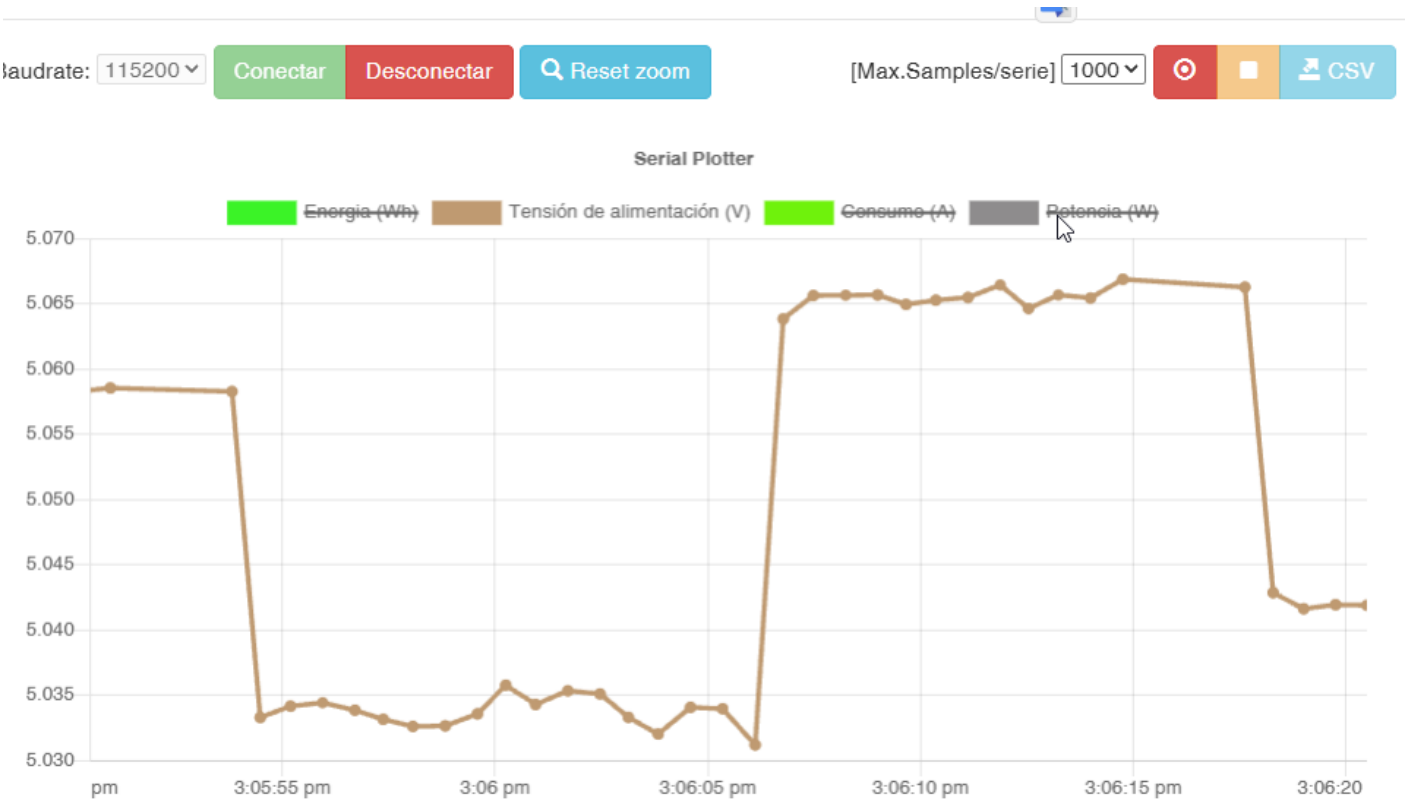
Ya vimos en SENSORES ESP32 que esta placa tiene internamente unos sensores que podemos utilizar

Nuestra propuesta para experimentar es con los sensores de potencia. Vamos a ver este experimento consistente en encender y apagar diferentes elementos de TDR STEAM Imagina y ver las variaciones:

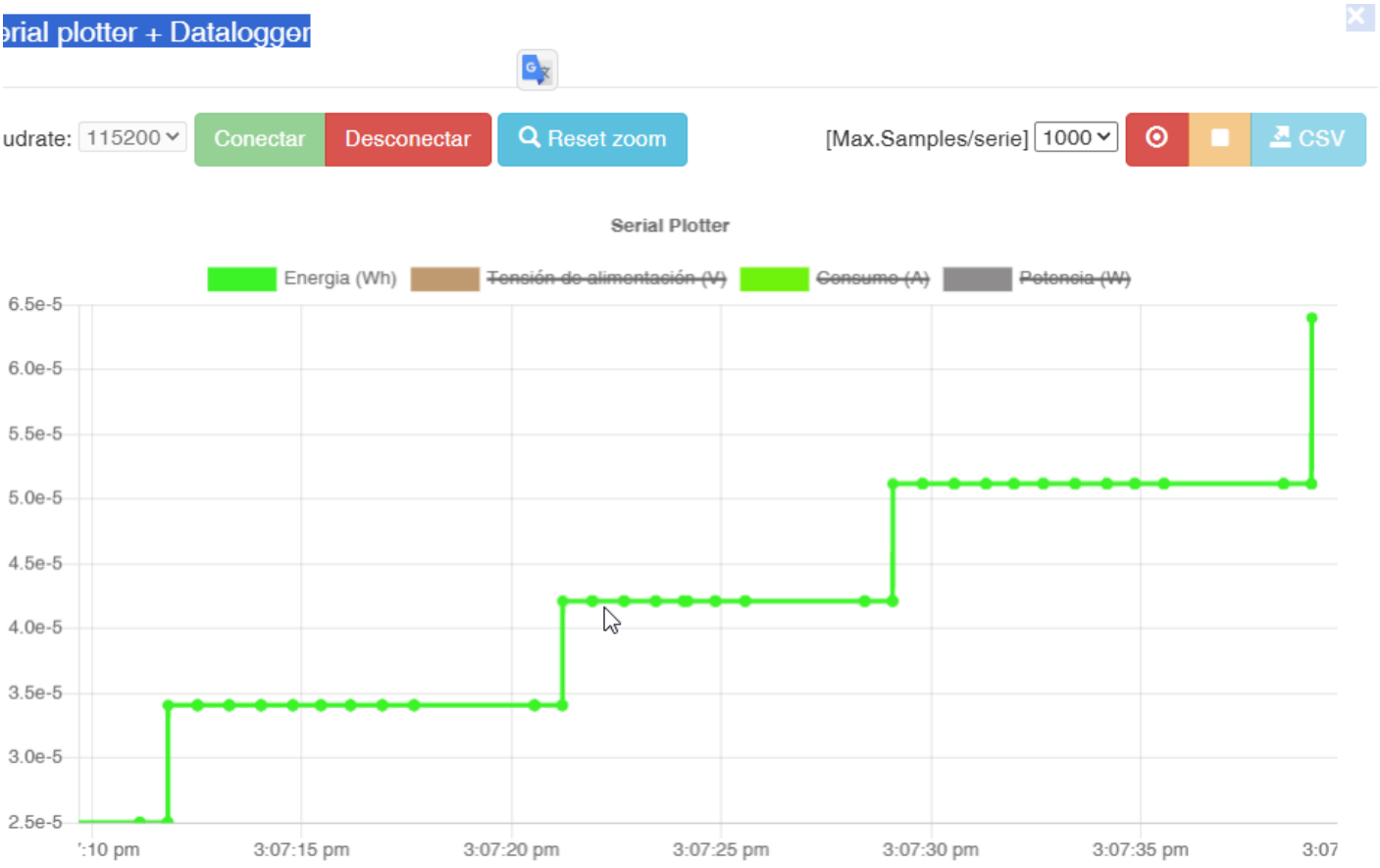
experimentoenergia.abp



Se puede observar las variaciones de tensión cuando se apaga y se enciende



Y también se puede ver como el consumo va a aumentando cada ver que encendemos y apagamos



Copyright 2026

¿No tendría que subir linealmente? Pues por lo visto el consumo en los cambios es mucho mayor y los sensores lo detectan y en los estados estacionarios o los sensores no lo detectan

Página extraída de Federico Coca Notas sobre ESP32 STEAMakers CC-BY-SA

A1. Leer valor de tensión de alimentación

Un programa para verificar la tensión de alimentación de la placa ESP32 Plus STEAMakers lo vemos en la figura siguiente:

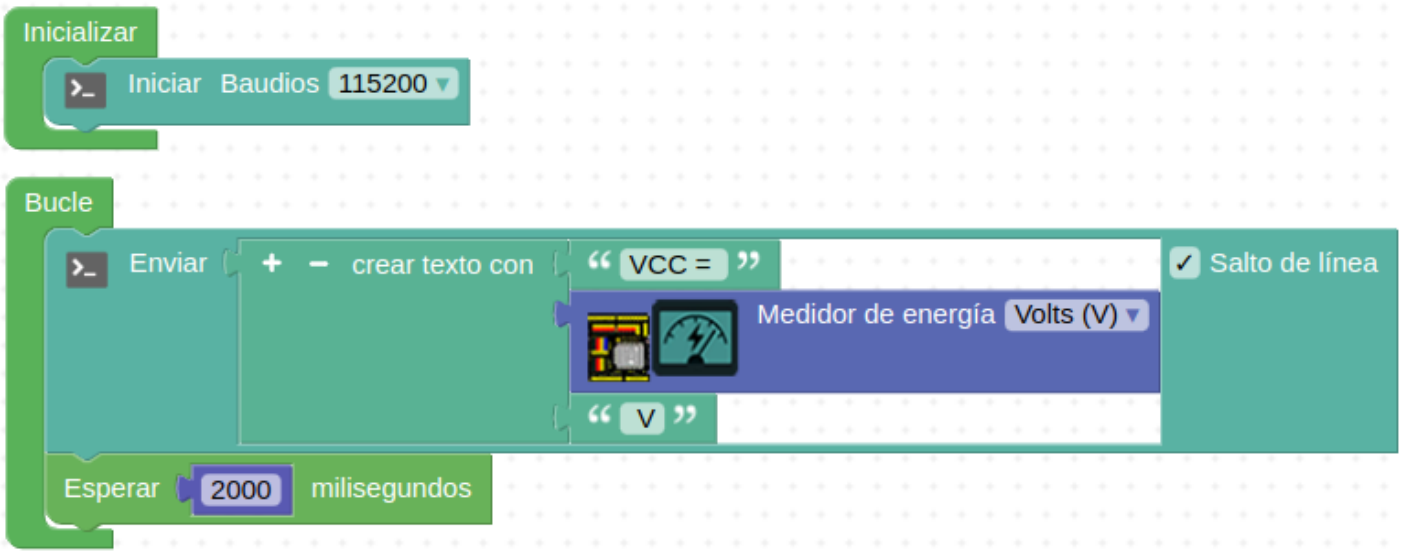


Imagen Federico Coca Notas sobre ESP32 STEAMakers CC-BY-SA

Medir la tensión de alimentación

Que nos arroja un resultado en la consola como el que vemos en la figura siguiente:

Imagen Federico Coca [Notas sobre ESP32 STEAMakers](#) CC-BY-SA

A.2. Graficar todos los parámetros del 'Medidor de energía'

El programa SP-medidor-energia de la figura siguiente realiza la captura en variables de los cuatro parámetros disponibles y los envía al Serial Plotter. Para modificar el valor de los mismos se han creado las funciones encender y apagar que encienden y apagan los diodos LEDs de la TdR STEAM y de la tira de 8 LEDs **(NO PROPORCIONADO POR CATEDU)**

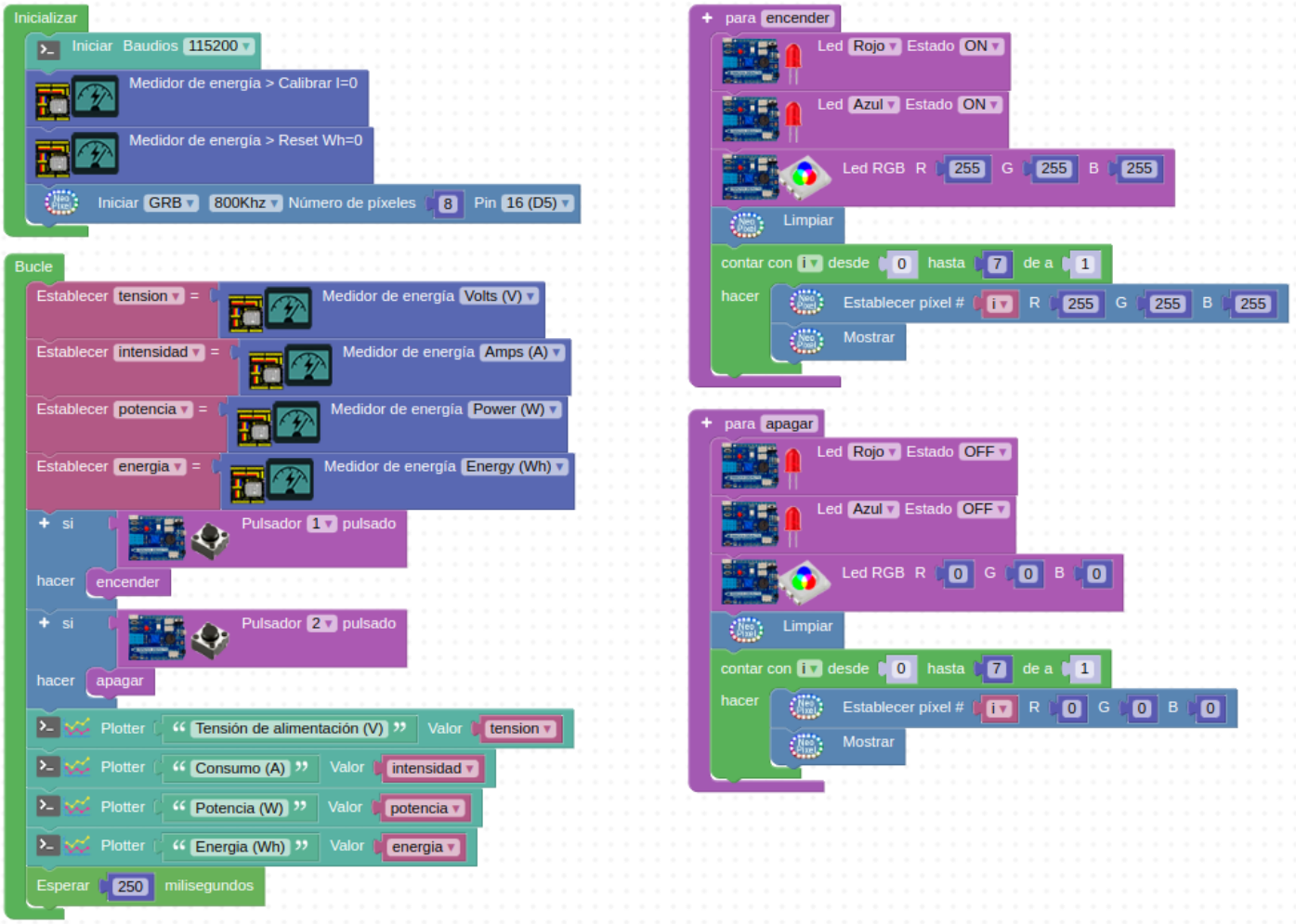


Imagen Federico Coca [Notas sobre ESP32 STEAMakers] CC-BY-SA

Programa Serial Plotter

En la figura siguiente vemos el resultado de forma gráfica.

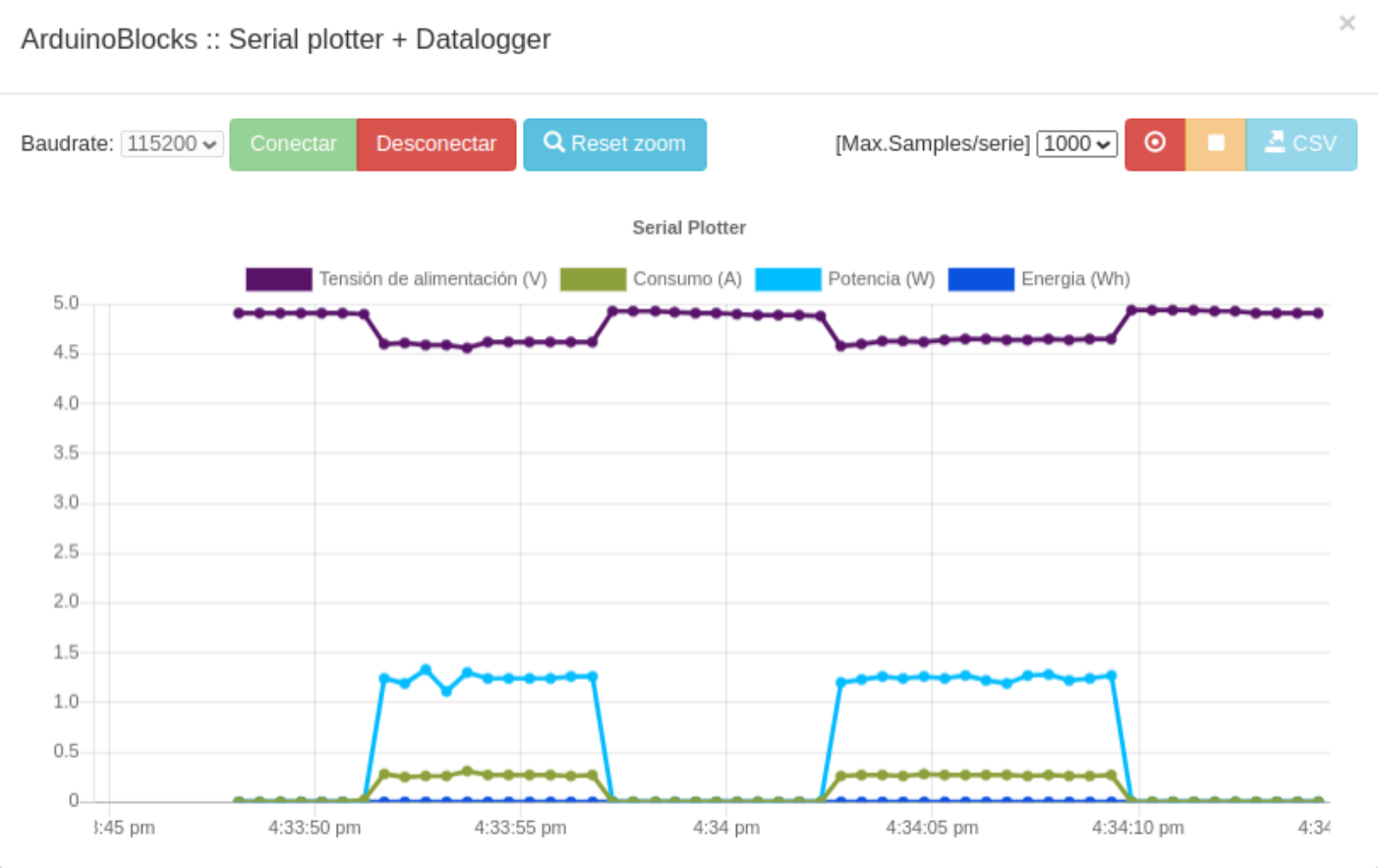


Imagen Federico Coca [Notas sobre ESP32 STEAMakers](#) CC-BY-SA

A.3. Sensores internos

Un programa como el de la figura siguiente nos va a mostrar por consola los valores leídos por los sensores internos.

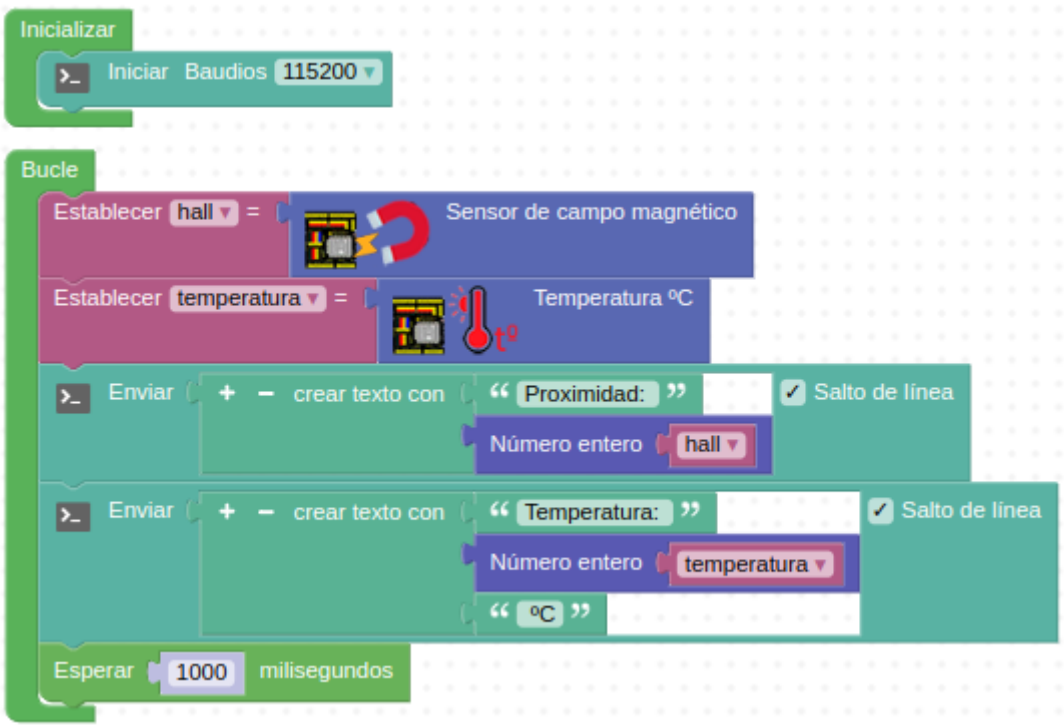


Imagen Federico Coca [Notas sobre ESP32 STEAMakers](#) CC-BY-SA

Sensores internos

El resultado en consola lo vemos en la figura siguiente:

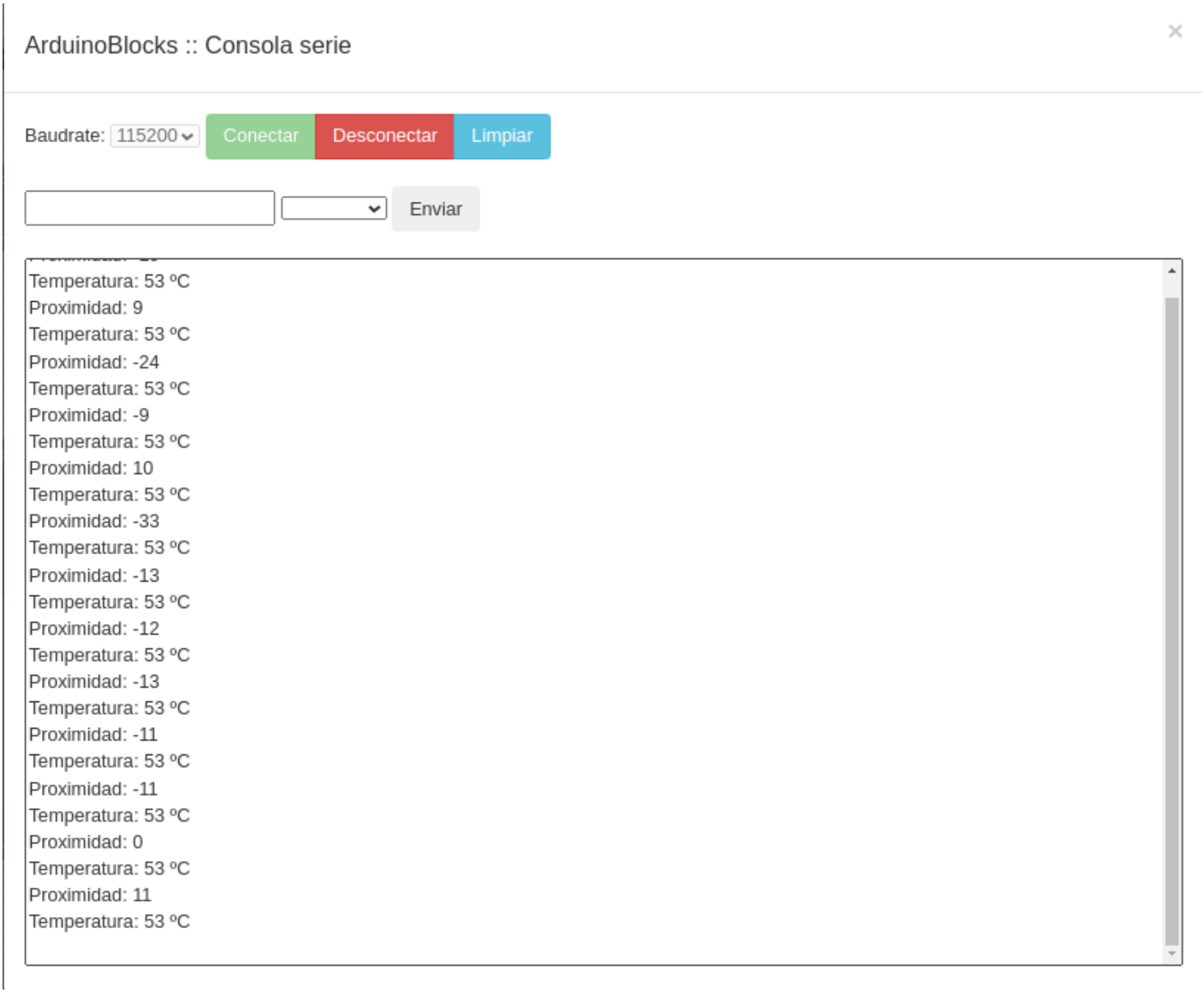


Imagen Federico Coca [Notas sobre ESP32 STEAMakers] CC-BY-SA

Retos

R1. Realizar la actividad A2 mostrando los datos por consola.
R2. Realizar la actividad A2 mostrando los datos en la LCD.
R3. Hacer una captura de datos en formato CSV de alguno de los parámetros del 'Medidor de energía'.

Avanzado: Multitarea, interrupciones, memoria Flash y tarjeta SD

Multitarea

Las placas convencionales y ordenadores antiguos que sólo tienen un núcleo en el microprocesador, sólo pueden hacer una tarea. Se puede hacer por software que el microprocesador realice varias tareas simplemente dando tiempo a cada una de ellas e ir saltando de una a otra, es decir, no es una programación en paralelo, y si el micro no es potente, acaba por saturarse.

ESP32 Plus Steammaker tiene un microprocesador pero tiene dos núcleos. Tiene capacidad para hacer algunas tareas en paralelo.

Como estos conceptos pertenecen a un perfil más avanzado, se escapa de los propósitos de este curso. Si se quiere trabajar con ellos recomendamos el siguiente Descarga desde el drive del autro tutorial Manual de Actvidades ESP32 SteamMaker 2022_Junio

Aquí están las páginas y los retos para más información:

Interrupciones

Una interrupción es un evento (por ejemplo pulsar un pulsador) y hace que el micro deje lo que estaba haciendo y salte a realizar una tarea concreta. En la placa TDR STEAM hay dos pulsadores, en D3 y en D5 que pueden realizar interrupciones al ESP32.

Aquí están los retos y las páginas en

- Descarga desde libros.catedu..es Manual Actividades ESP32 SteaMakers 2022_Junio.pdf
- Descarga desde el drive del autro tutorial Manual de Actvidades ESP32 SteamMaker 2022_Junio

Memoria Flash

El ESP32 Plus STEMaker tiene una memoria Flash que puede mantener los datos incluso después de apagar la placa. Su capacidad es muy limitada y no se almacenan eternamente. Descarga desde el drive del autro tutorial Manual de Actvidades ESP32 SteamMaker 2022_Junio

Tarjeta SD

El ESP32 Plus STEAMaker tiene un zócalo para insertar tarjetas microSD. Tiene que estar formateada a FAT32 y admite hasta 2 Teras con archivos máximos de 2G.

Aquí están los retos y las páginas en Descarga desde el drive del autro tutorial Manual de Actvidades ESP32 SteamMaker 2022_Junio

Copyright 2026