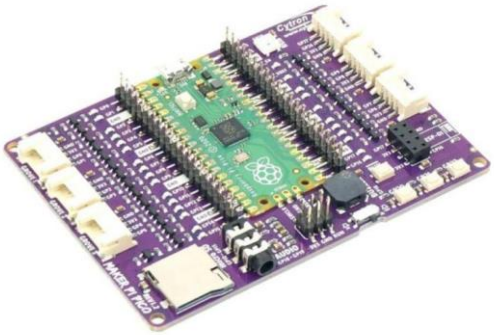
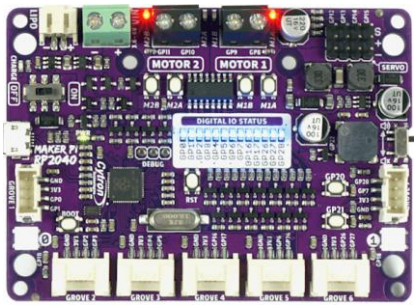
PROGRAMACIÓN

de MAKER PICO PI Cytron con MicroBlocks





Prof: José Manuel Ruiz Gutiérrez

Enero 2023

Índice

1. Introducción a este trabajo

La aparición de la tarjeta Raspberry Pi Pico en todas sus versiones marca un punto de inflexión en la oferta de *Plataformas Hardware Educativas* orientadas al estudio y desarrollo de los microcontroladores en el ámbito educativo y en el mundo Maker.

La firma Cytron ha presentado al mercado una serie de diseños basados en esta nueva versión de Raspberry Pi que ha despertado el interés de la comunidad de usuarios de estas plataformas. Los desarrollos mas importantes disponibles en esta oferta son básicamente dos: Tarjeta MAKER PI PICO y Tarjeta MAKER PI RP2040.

En este libro se van a usar ambas tarjetas para desarrollar los mas de 60 ejemplos que se abordan como aplicaciones básicas basadas en los procesadores Raspberry Pi PICO.

Para la programación de las plataformas basadas en este nuevo procesador se pueden seleccionar diversas herramientas basadas en los estándares MicroPython y CircuitPython así como el IDE Arduino. Por otra parte, existe una herramienta de programación de libre difusión basada en el estándar Snap! Denominada MicroBlocks que ofrece unas potentes posibilidades para iniciarse en la programación grafica muy adecuadas para usuarios que no controlen aun el lenguaje de programación Python o C.

Usando MicroBlocks vamos a realizar los ejemplos de programación a lo largo de este trabajo.

Recomiendo el uso de este libro especialmente a los estudiantes de *Educación Secundaria, Formación Profesional y Bachillerato* y todos cuantos quieran iniciarse en el uso de este poderoso microcontrolador.

Para finalizar quiero agradecer a Cytron el haberme brindado la posibilidad de manejar su documentación, usar sus imágenes y disponer de las tarjetas. Especialmente mi agradecimiento a **Cheryl Ng**. *Head of rero EDUteam*. **Cytron Technologies Malasia**

Profesor: **José Manuel Ruiz Gutiérrez**

Enero 2023

Introduction to this work

The appearance of the Raspberry Pi Pico card in all its versions marks a turning point in the offer of Educational Hardware Platforms aimed at the study and development of microcontrollers in the educational field and in the Maker world.

The Cytron firm has presented to the market a series of designs based on this new version of Raspberry Pi that has aroused the interest of the community of users of these platforms. The most important developments available in this offer are basically two: MAKER PI PICO Card and MAKER PI RP2040 Card.

In this book, both cards will be used to develop the more than 60 examples that are addressed as basic applications based on the Raspberry Pi PICO processors.

For the programming of the platforms based on this new processor, various tools based on the MicroPython and CircuitPython standards can be selected, as well as the Arduino IDE. On the other hand, there is a free broadcast programming tool based on the Snap! Called MicroBlocks, it offers powerful possibilities to get started in graphic programming, very suitable for users who do not yet control the Python or C programming language.

Using MicroBlocks we are going to carry out the programming examples throughout this work.

I recommend the use of this book especially to students of Secondary Education, Vocational Training and Baccalaureate and all those who want to start using this powerful microcontroller.

Finally, I want to thank Cytron for having given me the possibility of managing their documentation, using their images and having the cards. Especially my thanks to **Cheryl Ng**. *Head of rero EDUteam*. Cytron Technologies **Malaysia.**

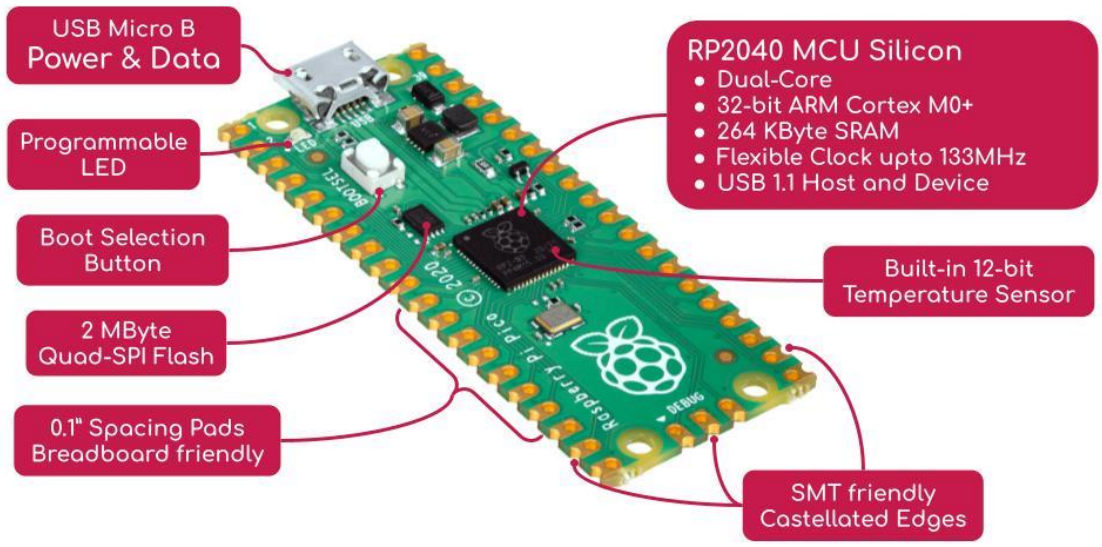
Professor: **José Manuel Ruiz Gutiérrez**

January 2023

2. Microcontrolador Raspberry Pi Pico

Descripción

La 1ª placa MCU de Raspberry Pi - Pico



Lanzada el 21 de enero de 2021, la Raspberry Pi Pico es la 1ª placa de desarrollo de microcontroladores de la Fundación Raspberry Pi. También se basa en el 1er Microcontrolador IC / Silicon - RP2040, diseñado y producido por ingenieros del equipo de Raspberry Pi también.

2.1. ¿Qué es Raspberry Pi Pico?

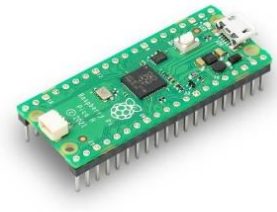
Una placa de desarrollo de microcontroladores

¿Cuáles son las diferencias entre la Raspberry Pi 4 Modelo B y esta Raspberry Pi Pico? Bueno, básicamente, la Raspberry Pi 4 Modelo B es una placa base o una computadora de placa única que puede usar para jugar, trabajar, grabar datos, navegar por Internet, ver películas, como un reproductor multimedia y muchos más. Raspberry Pi Pico no está diseñado para reemplazar a la Raspberry Pi 4 Modelo B (o placa similar), es más para proyectos de computación física donde controla cualquier cosa, desde pequeños componentes electrónicos, LED, motores; Leer información de sensores o comunicarse con otros microcontroladores.

Probablemente ya tenemos muchos microcontroladores en nuestra casa. Por ejemplo, una lavadora es controlada por un microcontrolador; Lo más probable es que nuestro reloj también lo sea; También hay uno en un microondas. Por supuesto, todos estos microcontroladores ya tienen sus programas para las aplicaciones.

Esta Raspberry Pi Pico es una placa de microcontrolador que puede realizar computación física y se puede reprogramar fácilmente a través de una conexión USB.

Se dispone de dos modelos de Raspberry Pi Pico



Raspberry Pi Pico con cabezales **PRE-SOLDADOS**

Raspberry Pi Pico sin cabecera, SMD amigable

2.2. Programación con MicroPython

Como Python es el lenguaje de programación oficial de Raspberry Pi OS, MicroPython es elegido para ser uno de los lenguajes de programación de Raspberry Pi Pico. MicroPython es una implementación eficiente y eficiente del lenguaje de programación Python 3 que incluye un pequeño subconjunto de la biblioteca estándar de Python y está optimizada para ejecutarse en microcontroladores y en entornos restringidos.

La programación de la carga de MicroPython en Raspberry Pi Pico es fácil. Simplemente conecte el Pico a cualquier computadora (incluido Raspberry Pi SBC) a través de USB, luego **arrastre y suelte** el archivo en él. ¡Sí! ¡Así de fácil! Y Raspberry Pi Foundation ha reunido un archivo UF2 descargable para permitirle instalar MicroPython más fácilmente. Visite la página de introducción de Raspberry Pi para descargar el archivo necesario. O puede obtener una copia impresa de "Comience con MicroPython en Raspberry Pi Pico" y comience su viaje de creación digital.

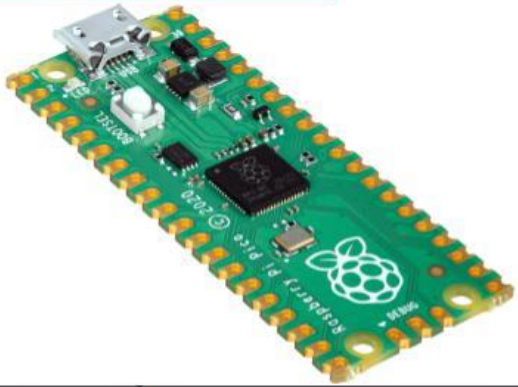
2.3. Trabaje sin problemas con la computadora Raspberry Pi

El sistema operativo oficial, Raspberry Pi OS viene preinstalado con Thonny Python IDE que está listo para que comience a escribir código MicroPython para Pico. Si está utilizando otro sistema operativo (Windows, macOS u otra distribución de Linux), visite https://thonny.org/ para descargar el IDE e instalarlo.

En la siguiente pagina se muestra tabla con las características del procesador Raspberry utilizando otro sistema operativo (Windows, macOS u otra distribución de Linux), visite https://thonny.org/ para descargar el IDE e instalarlo.

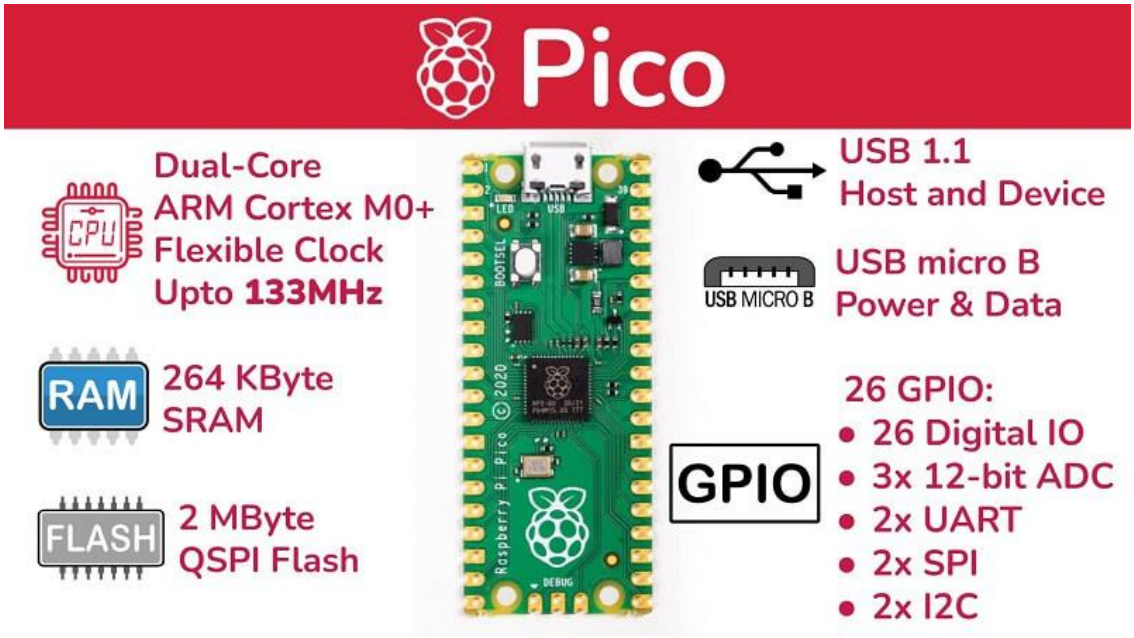
En la siguiente página se muestra tabla con las características del procesador Raspberry Pi Pico

Raspberry Pi Pico



Features/Specs	Raspberry Pi Pico
Release Date	**21st January 2021**
Microcontroller	**RP2040**
Cores	**Dual-Core**
Core Architecture	32-bit ARM **Cortex-M0+**
CPU Clock	Flexible clock upto **133MHz**
RAM Size	**264 KByte SRAM**
Flash Size	**2 MByte Q-SPI Flash**
Programming Language	**MicroPython, C, C++**
Board Power Input	**5VDC** via USB Micro B
Alternative Board Power	**2 - 5VDC** via VSYS Pin (Pin 39)
MCU Voltage	**3.3VDC**
GPIO Voltage	**3.3VDC**
USB Interface	USB 1.1 **Device and Host**
Program Loading	USB Micro B, **USB Mass Storage**
GPIO	**26 x Digital** Input/Output (Total)
ADC	**3 x 12-bit** 500ksps
Temperature Sensor	**Built-in, 12-bit**
UART	**2 x UART**
I^2C	**2 x I^2C**
SPI	**2 x SPI**
PWM	**16 x PWM**
Timer	1 x Timer with 4 x Alarm
Real Time Counter	1 x Real Time Counter
PIO	2 x Programmable High Speed IO
On Board LED	1 x Programmable LED (GP25)
On Board Button	1 x BOOTSEL Button
Breakout of PCB	40 x 0.1" (100mil) Standard Header Pads 40 x 0.1" (100) **Castellations Edge** (SMT Friendly)
Debug Port	3-pin ARM Serial Wire Debug (SWD) Port

2.4. Lo más destacado del procesador:

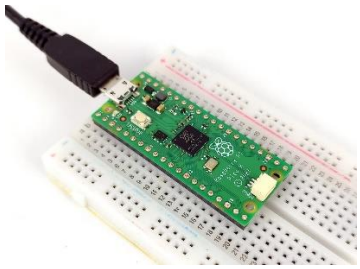


La 1ª plataforma MCU por la Fundación Raspberry Pi

¡1ª plataforma de microcontrolador con el primer silicio MCU diseñado internamente, **RP2040**! Sin embargo, el equipo de Raspberry Pi nunca deja de sorprendernos con su trabajo. Esta pequeña placa viene con un procesador de **doble núcleo de** 32 bits, ARM Cortex M0 + es un superconjunto optimizado del Cortex-M0. El Cortex-M0+ tiene compatibilidad completa del conjunto de instrucciones con el Cortex-M0, lo que permite el uso del mismo compilador y herramientas de depuración. La tubería Cortex-M0+ se redujo de 3 a 2 etapas, lo que reduce el uso de energía. Además del Cortex M0+ de doble núcleo, Raspberry Pi Pico viene con una velocidad de reloj reconfigurable, con el PLL (Phase-Locked Loop) en chip, que permite que el MCU se cronometre a una velocidad máxima de **133MHz**, por supuesto, es necesaria una configuración.

Compacto y listo para ser montado en el producto

Raspberry Pi Pico no solo es súper asequible, sino que también está listo para integrarse en cualquier producto listo para usar. Si elige la versión sin cabezales presoldados, está listo para SMT (Surface Mount Technology). Raspberry Pi Pico se extiende a 40 pines 21x51 DIP (paquete dual en línea), PCB de 1 mm de grosor con pines de orificio pasante de 0.1 "(100 mil). Los pines se extienden aún más al borde de la PCB con una placa de circuito almenada. Esto permite que se suelde a otra placa de PCB sin la necesidad de pines de cabezal adicionales, lo que lo convierte en un producto terminado más pequeño y compacto. ¡Genial!

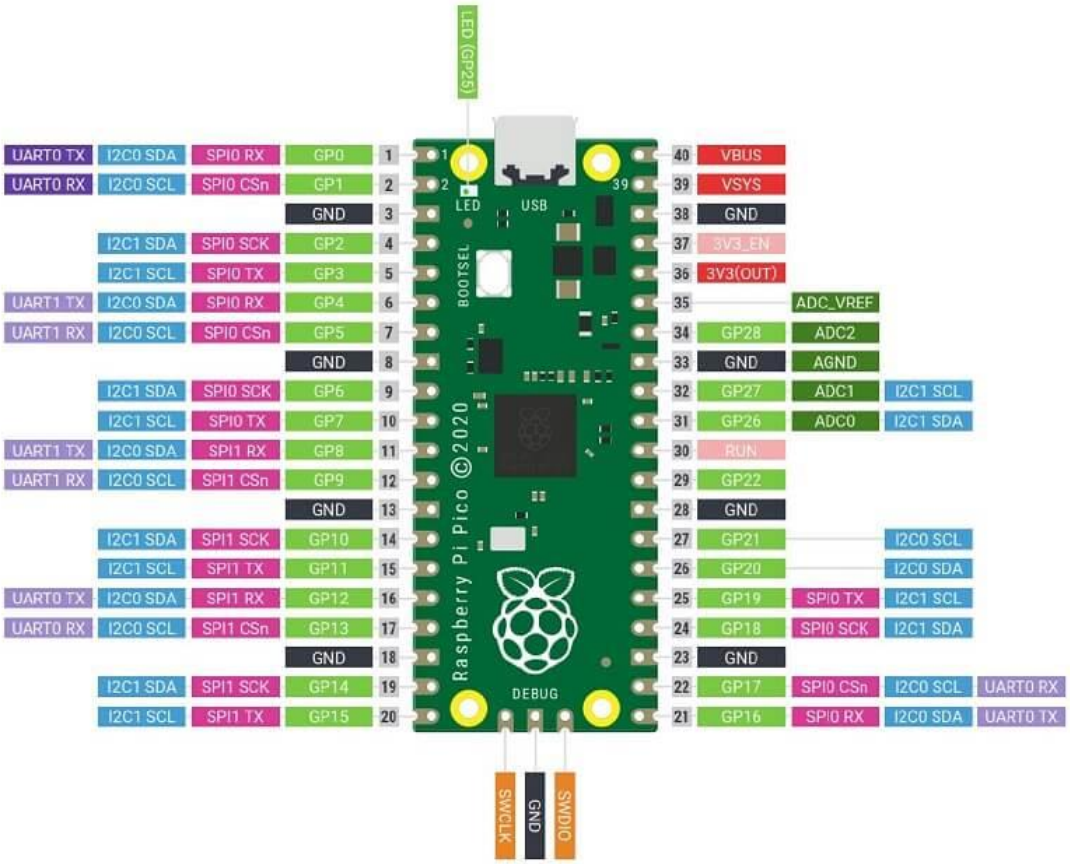


Mientras tanto, la versión con cabezales pre-soldados es amigable con la placa de pruebas, por lo que los estudiantes, fabricantes e ingenieros pueden usar la Raspberry Pi Pico en una placa de prueba o cualquier placa PCB estándar para desarrollo o creación de prototipos.

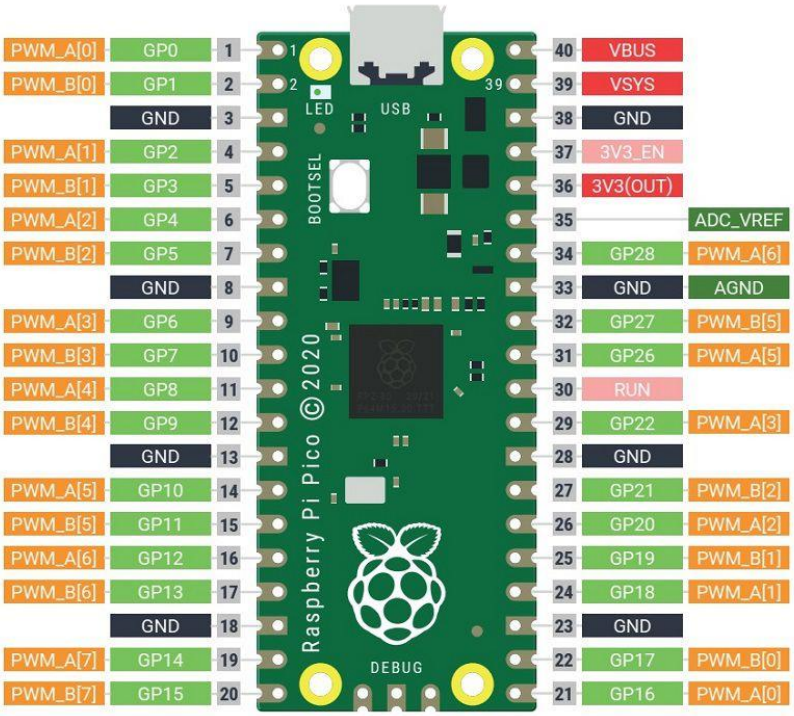
USB Micro B para alimentación y datos

Raspberry Pi Pico incorpora el famoso y comúnmente utilizado receptor USB Micro B tanto para alimentación como para datos. Simplemente obtenga un cable USB Micro B que normalmente viene con un teléfono Andriod o un banco de energía para alimentarlo y cargar el programa en él. No se necesita ningún adaptador USB a serie adicional. ¡Pulcro!

MCU rico en periféricos



GPIO, ADC, UART, SPI, I2C de Raspberry Pi Pico

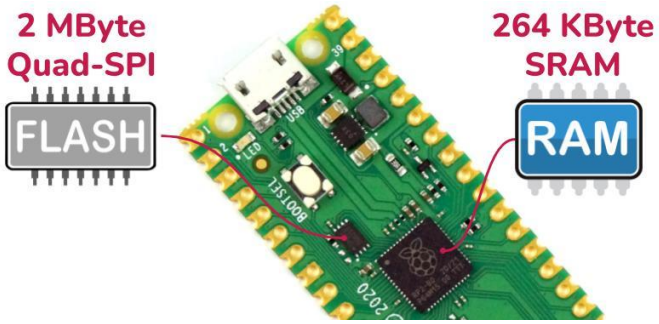


PWM Pins de Raspberry Pi Pico

Con 26 GPIO (3.3V) desglosados para aplicaciones, tiene más pines GPIO que Arduino UNO, Arduino NNO o incluso Arduino MKR Zero. Entre estos 26 GPIO, 3 se pueden configurar como ADC de 12 bits con 500ksps (kilo de muestra por segundo), 2 x UART, 2 x SPI, 2 x I^2C y hasta 16 x PWM pin. Internamente, también viene con 1 x temporizador con 4 alarmas y 1 x contador en tiempo real. Sin olvidar los periféricos duales de E/S programables (PIO) que son E/S de alta velocidad flexibles y programables por el usuario. Puede emular interfaces como tarjetas SD y VGA.

Nota: El Raspberry Pi Pico GPIO se ejecuta en **3.3VDC.** El voltaje máximo que los pines de E/S pueden tolerar es de 3.3V. La aplicación de voltajes superiores a 3.3V a cualquier pin de E/S podría dañar la placa.

Gran tamaño de RAM y flash

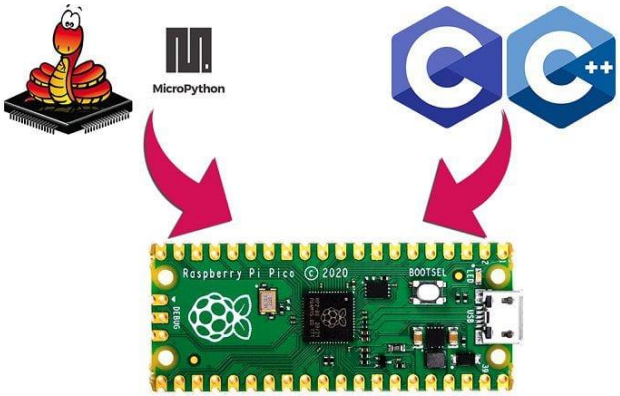


Con **2 MByte** de QSPI Flash externo y **264 KByte** de SRAM en **Raspberry Pi Pico**, nunca le preguntará por no tener suficiente memoria :) Además, el gran tamaño de RAM y Flash también permite que **Raspberry Pi Pico** sea compatible con lenguajes de programación superiores como **MicroPython** o incluso **Javascript**.

Método de carga del programa de arrastrar y soltar

Con el receptor USB Micro B listo como conexión física a un ordenador y el USB 1.1 PHY en el RP2040 (MCU), la Raspberry Pi Pico ofrece un método de carga de programas simple y directo. Es como copiar archivos de una unidad a otra. ¡El Pico aparece como almacenamiento masivo USB cuando se conecta a la computadora a través del puerto USB! ¡Se convierte en una unidad USB! Escriba el código y arrastre el archivo a esa unidad USB. Después de que el archivo se haya copiado por completo, Pico se reiniciará y ejecutará el programa :) Fácil, ¿verdad?

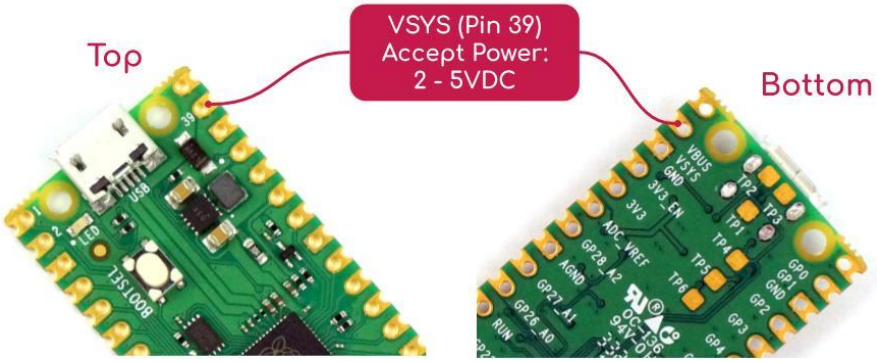
Soporta MicroPython, C y C++



Python es uno de los lenguajes de programación más famosos y poderosos de la actualidad. Se está utilizando en muchas aplicaciones de alto nivel, como IA (inteligencia artificial), DL (aprendizaje profundo) y desarrollo web e Internet, y más. Python se utiliza con éxito en miles de aplicaciones empresariales del mundo real en todo el mundo, incluidos muchos sistemas grandes y de misión crítica. MicroPython es una implementación eficiente y eficiente del lenguaje de programación Python 3 que incluye un pequeño subconjunto de la biblioteca estándar de Python y está optimizada para ejecutarse en microcontroladores y en entornos restringidos. Te encantará.

Además de MicroPython, Raspberry Pi Pico también es compatible con C y C ++ Programming Langauge. Consulte el SDK de C/C++ para obtener más información. Todos estos lenguajes de programación se cargan en Raspberry Pi Pico a través de USB Mass Storage que permite el método simple de arrastrar y soltar (como copiar un archivo a otra unidad).

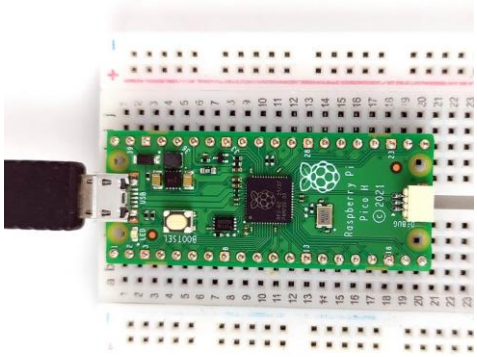
Entrada de energía flexible



El receptor USB Micro B es la entrada de alimentación principal para que la Raspberry Pi Pico "funcione", simplemente conecte el cable USB a cualquier puerto USB y suministrará la energía necesaria para que el MCU ejecute el programa. En el caso de que no desee utilizar el puerto USB, por ejemplo, un producto alimentado por batería o personalizado; No se preocupe, Raspberry Pi Pico viene con una fuente de alimentación de modo de conmutación (SMPS) flexible a bordo que es capaz de aceptar **entradas de 2 a 5VDC** y la convierte en una fuente estable de 3.3V para que funcione el MCU RP2040.

¡Simplemente increíble! El pin es **VSYS** (Pin 39). Con un amplio rango de voltaje, Raspberry Pi Pico puede ser alimentado por USB, 2 x pilas AA, 2 pilas NiMH AA, ¡1 x 18650 Li-ion o 1 x pila LiPo de celda!

Raspberry Pi Pico se inserta en una placa de pruebas, hay dos filas de orificios de puente a ambos lados de la PCB para la creación de prototipos.



2.5. ¿Qué necesitará para comenzar?

Dado que Raspberry Pi Pico es la 1ª plataforma MCU de la Fundación Raspberry Pi, todos son nuevos en ella. ¡No temas! Es una placa de microcontrolador bastante amigable para principiantes. Aquí están nuestras recomendaciones para el principiante, fabricante e ingeniero:

1. Obtenga el Raspberry Pi Pico con cabezales pre-soldados si desea una experiencia sin problemas para comenzar.
2. Un cable USB Micro B, al menos necesitará este cable para alimentar y cargar el programa en Raspberry Pi Pico.
3. Breadboard 8.5x5.5cm (400 agujeros), puede ayudar a sostener la Raspberry Pi Pico y puede extender el GPIO para la creación de prototipos.
4. Official Get Started with MicroPython on Raspberry Pi Pico-Color Printed, una guía completa de Raspberry Pi para comenzar con esta nueva plataforma MCU.
5. Maker Pi Pico es una placa de desarrollo para principiantes que extendió todo el GPIO de Pico a cabezales, conectores Grove, ranura para tarjetas MicroSD, zócalo ESP-01 (WiFi), indicadores LED, botones integrados, LED RGB (Neo-Pixel).
6. Raspberry Pi Pico Basic Kit sin Pico, un kit electrónico basado en la guía oficial, **sin el Pico y Serial LCD y RGB LED**
7. Raspberry Pi Pico Basic Kit - con Pico, un kit electrónico completo basado en la guía oficial, **sin el LCD serie y el LED RGB**

Funciones

- **1ª placa de desarrollo de microcontroladores** de la Fundación Raspberry Pi
- **1st Silicon (IC), MCU RP2040** diseñado desde cero por ingenieros de Raspberry Pi Foundation
- **Procesador ARM Cortex M0+ de doble núcleo y 32 bits**

- Reloj flexible, **configurable máximo a 133MHz**
- Preparado con receptor USB Micro B para alimentación y datos
- **Soporte USB 1.1 host y dispositivo**
- Conectado al puerto USB y aparecerá como **almacenamiento masivo USB** de forma predeterminada, no se necesita ningún controlador
- Soporta **CircuitPython, MicroPython, C y C++,** lenguaje **de programación Arduino IDE**
- Método de carga del programa de arrastrar y soltar, al igual que mover archivos en el Explorador de Windows
- Viene en PCB de 40 pines de 1 mm de grosor estilo 'DIP' 21x51 con **pines de orificio pasante de 0.1 ", amigable con la placa de pruebas**
- Con las **castellaciones de borde** PCB, está listo para ser montado en otra PCB sin un pin de cabezal adicional, compatible con SMD
- Dos opciones:
    - Viene con pines de cabezal PRE-SOLDADOS, amigable con la placa de pruebas
    - Viene sin cabecera, compatible con SMD
- Rico periférico:
    - Salida extendida 26 **E/S multifunción de 3,3 V de uso general (GPIO)**
    - 23 GPIO son solo digitales
    - **3 ADC de 12 bits capaz de 500Ksps**, convertidor analógico a digital
    - 2 x UART (Receptor/transmisor asíncrono universal)
    - 2 x SPI (interfaz periférica serie)
    - 2 x I2C (Inter IC)
    - **16 x PWM** (modulación de ancho de pulso)
    - 1 x temporizador con 4 alarmas
    - 1 x contador en tiempo real
    - **2 x E/S programables (PIO)** que pueden emular interfaces de alta velocidad como tarjeta SD o VGA
    - Sensor de temperatura ADC de 12 bits integrado
- Puerto de depuración de cable serie (SWD) ARM de 3 pines
- **LED programable integrado, GP25**
- Arquitectura de fuente de alimentación simple pero altamente flexible
    - Admite alimentación USB, fuente externa (2 - 5VDC) o incluso energía de la batería.
- SDK completo, ejemplos de software y documentación
- Oficialmente de la Fundación Raspberry Pi
- Compatible con cualquier ordenador con puerto USB, Windows, macOS, Linux
- Funciona sin problemas con Raspberry Pi 4 Model B, o Raspberry Pi 400 y Raspberry Pi OS
- Dimensiones: 51mm x 21mm x 1mm

Recursos

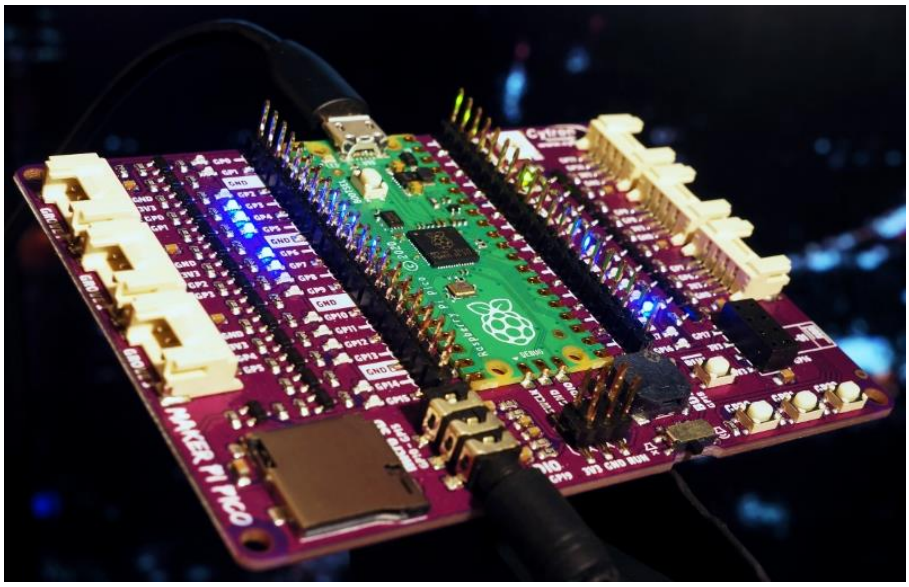
- Introducción a Raspberry Pi Pico (URL)
- Introducción a Raspberry Pi Pico (pdf), Desarrollo de C/C++ con Pico y otras placas de microcontroladores basadas en RP2040
- Hoja de datos de Raspberry Pi Pico (pdf), una placa de microcontrolador basada en RP2040

- SDK de Pico Python (pdf), Un entorno de MicroPython para el microcontrolador RP2040
- SDK de Pico C/C++ (pdf), Bibliotecas y herramientas para el desarrollo de C/C++ en el microcontrolador RP2040
- RP2040 Hoja de datos (pdf), Un microcontrolador por Raspberry Pi
- Un nuevo retador en la plataforma MCU: Raspberry Pi Pico, un artículo sobre Raspberry Pi Pico y RP2040
- Raspberry Pi Pico Vs Arduino UNO R3, un artículo para comparar las especificaciones de RPi Pico y Arduino UNO R3

3. Tarjetas de Cytron con el procesador Raspberry Pi Pico

Tarjeta Maker Pi Pico

Es un producto certificado **Powered by Raspberry Pi**.

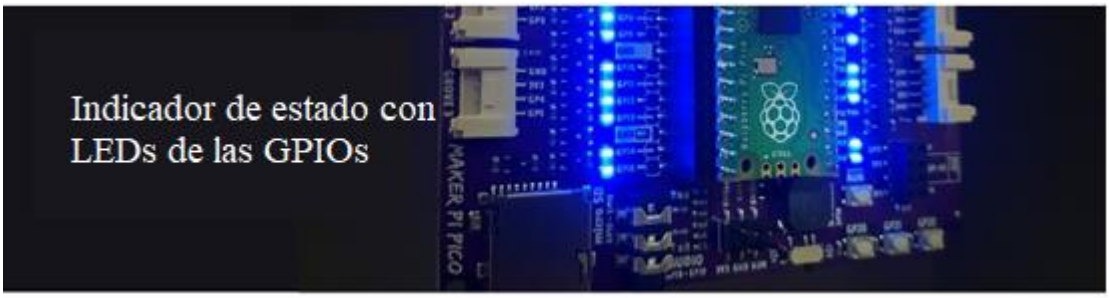


Crédito de la foto: Kevin J. Walters

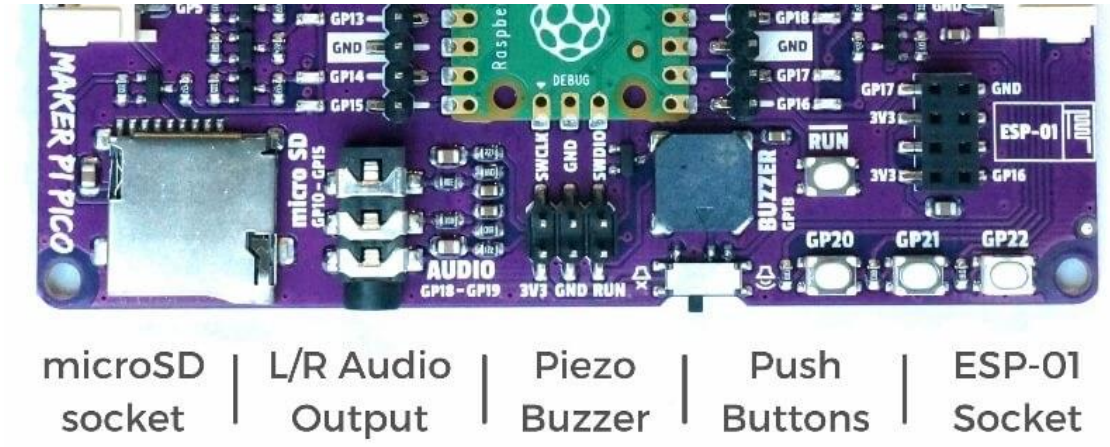
Maker Pi Pico y Maker Pi Pico Base incorporan el botón de reinicio más buscado para Raspberry Pi Pico o Pico W **y** le da acceso a todos los pines GPIO en dos cabezales de pines de 20 vías, con etiquetas claras. Cada GPIO se combina con un indicador LED para realizar pruebas de código y solucionar problemas de manera conveniente. La capa inferior de esta placa incluso viene con un diagrama de pines completo que muestra la función de cada pin.

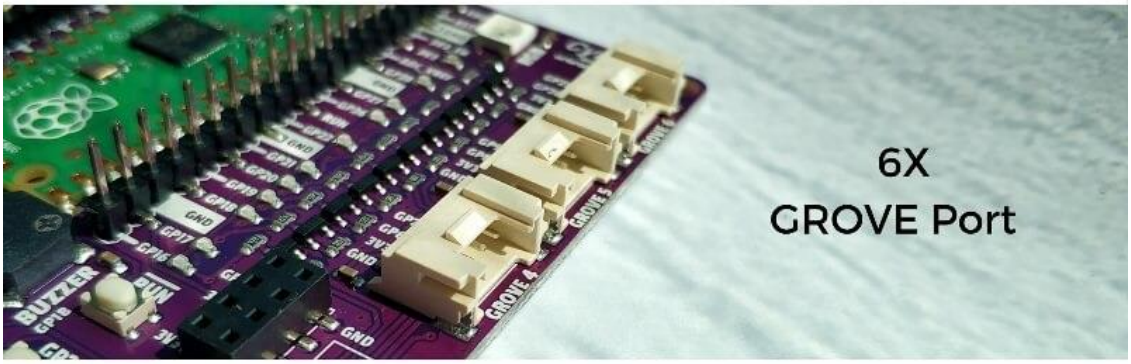
Nota Para los fabricantes internacionales, consulte a nuestros **socios internacionales** que llevan Maker Pi Pico Base.

Sus características más destacables

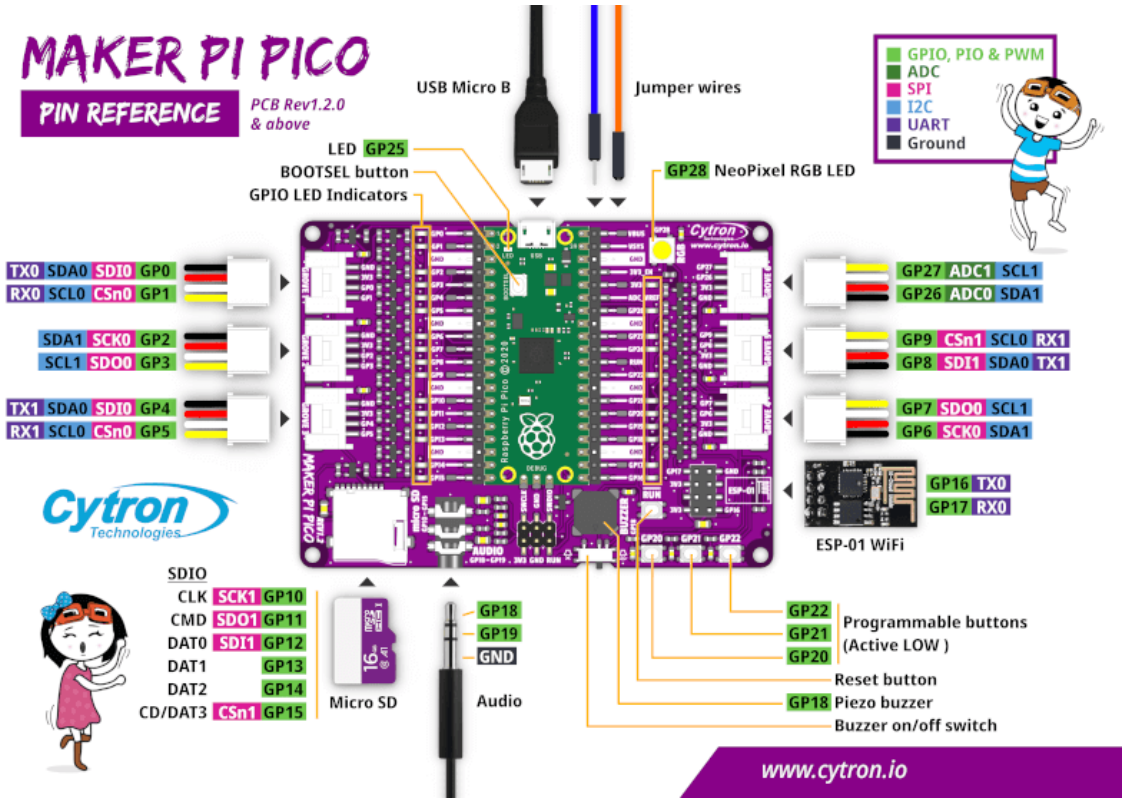


Detalle de algunas de las partes mas importantes de la tarjeta. Los conectores son compatibles con el estándar Grove





Es una placa compacta pero repleta de funciones diseñada para sentarse en el corazón de sus proyectos de STEM y robótica. Simplemente use los cables de puente y / o cables Grove para conectar cualquier sensor y módulos de salida para ampliar su capacidad. ¡No se requiere soldadura!



Maker Pi Pico se puede programar con **CircuitPython**, **MicroPython** y C / **C++.** Conéctese a cualquier computadora (incluida Raspberry Pi SBC) a través de USB, luego **arrastre y suelte** el archivo para cargar el programa. ¡Es tan fácil como copiar un archivo en su memoria USB!

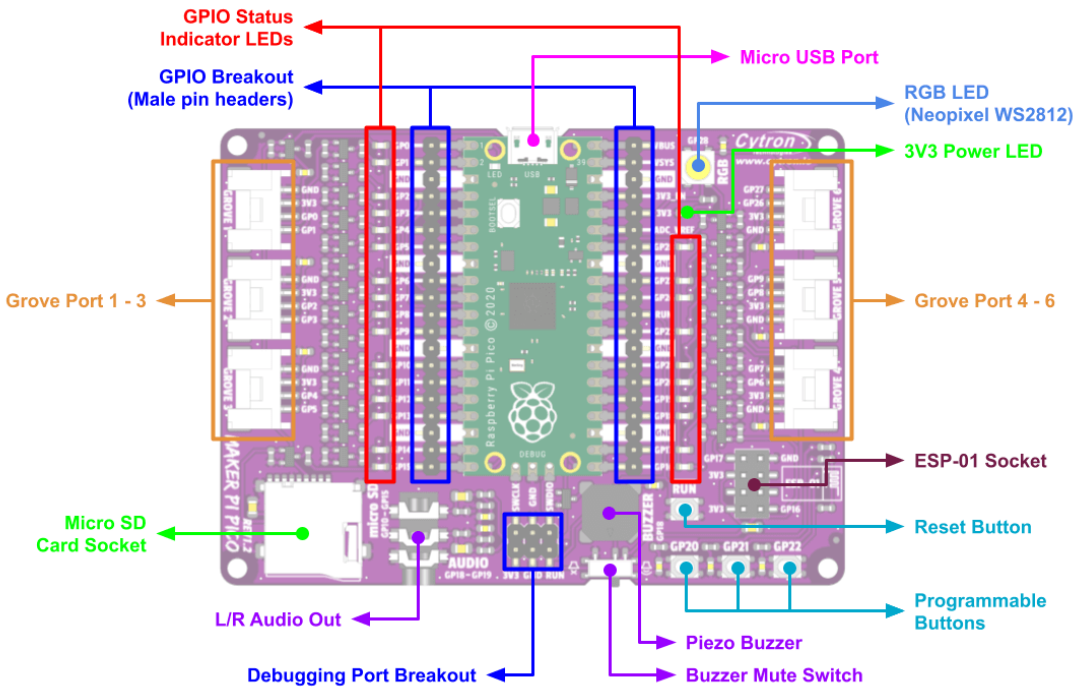
¿Qué es Raspberry Pi Pico W?

Es la Raspberry Pi Pico con :) inalámbrica incorporada Lo mejor de todo es que tiene la misma asignación de 40 pines, dimensión y PCB de borde almenado que la Raspberry Pi Pico. ¡La Raspberry Pi Pico W es un reemplazo directo de Raspberry Pi Pico si necesita actualizar su producto a Wireless o IoT! Por supuesto, Raspberry Pi Pico W se puede utilizar para proyectos de computación física donde controla cualquier cosa, desde pequeños componentes electrónicos, LED y motores; para leer información de sensores o comunicarse con otros microcontroladores.

Raspberry Pi Pico W

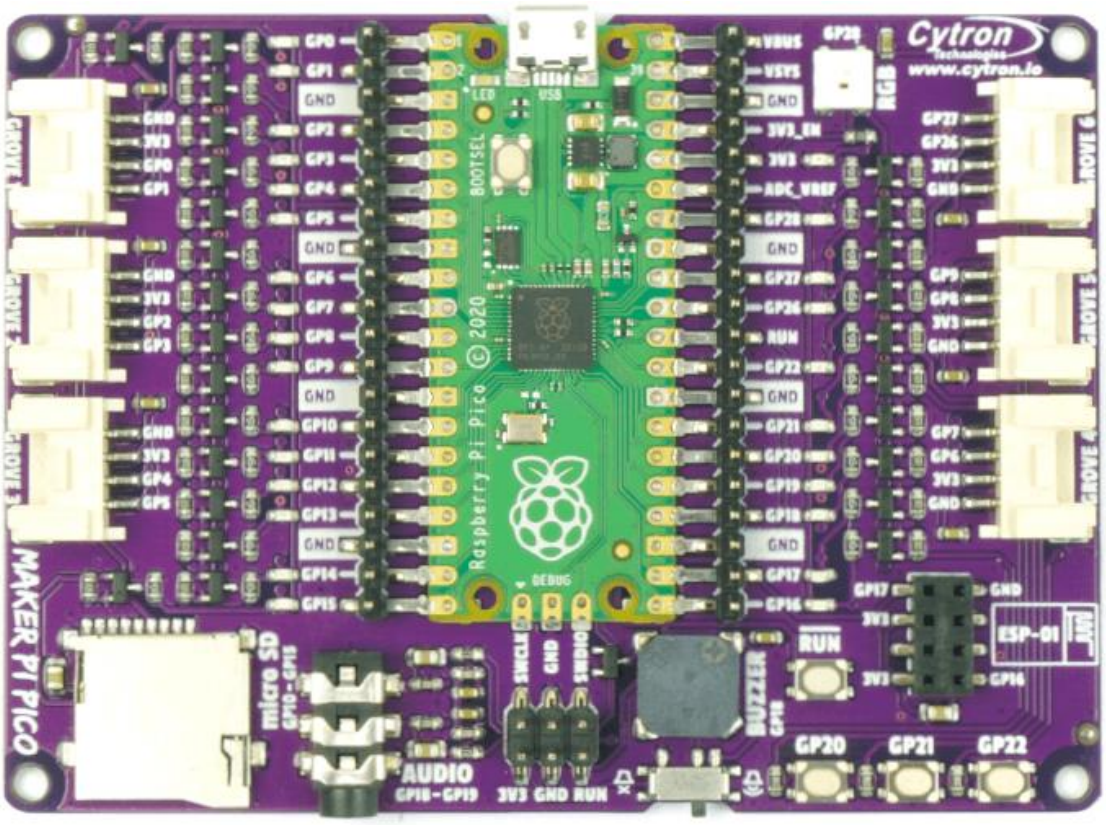
Features/Specs	Raspberry Pi Pico W
Release Date	**30th June 2022**
Microcontroller	**RP2040**
Cores	**Dual-Core**
Core Architecture	32-bit ARM **Cortex-M0+**
CPU Clock	Flexible clock upto **133MHz**
RAM Size	**264 KByte SRAM**
Flash Size	**2 MByte Q-SPI Flash**
Wireless	**IEEE 802.11 b/n/g (2.4GHz), Bluetooth 5.2**
Antenna	On board PCB Antenna
Programming Language	**MicroPython, CircuitPython, C, C++**
Board Power Input	**5VDC** via USB Micro B
Alternative Board Power	**2 - 5VDC** via VSYS Pin (Pin 39)
MCU Voltage	**3.3VDC**
GPIO Voltage	**3.3VDC**
USB Interface	USB 1.1 **Device and Host**
Program Loading	USB Micro B, **USB Mass Storage**
GPIO	**26 x Digital** Input/Output (Total)
ADC	**3 x 12-bit** 500ksps
Temperature Sensor	**Built-in, 12-bit**
UART	**2 x UART**
I^2C	**2 x I^2C**
SPI	**2 x SPI**
PWM	**16 x PWM**
Timer	1 x Timer with 4 x Alarm
Real Time Counter	1 x Real Time Counter
PIO	2 x Programmable High Speed IO
On Board LED	1 x Programmable LED (GP25)
On Board Button	1 x BOOTSEL Button
Breakout of PCB	40 x 0.1" (100mil) Standard Header Pads 40 x 0.1" (100) **Castellations Edge** (SMT Friendly)
Debug Port	3-pin ARM Serial Wire Debug (SWD) Port

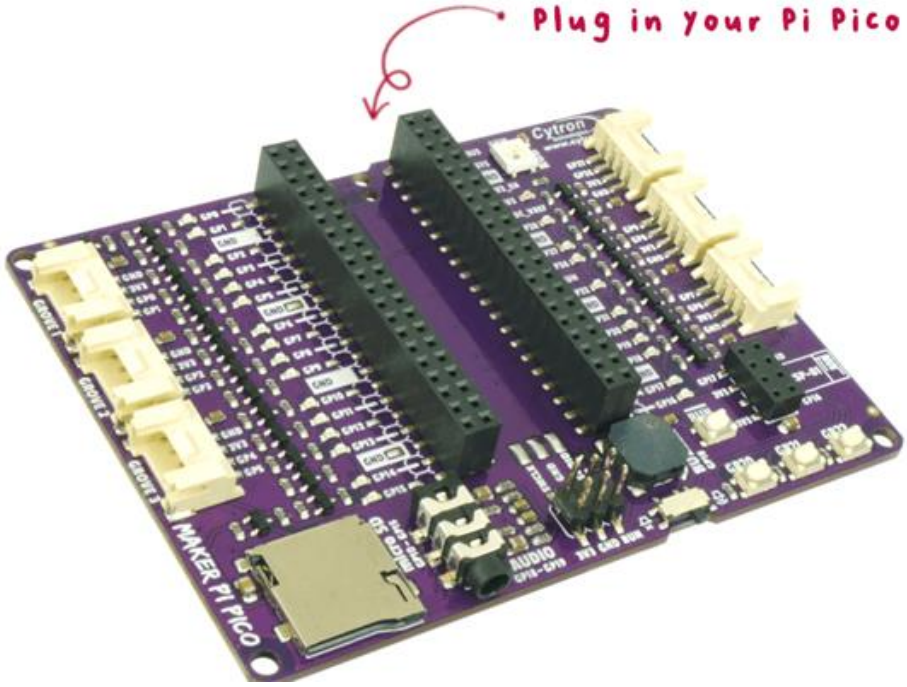
Diseño de la placa:



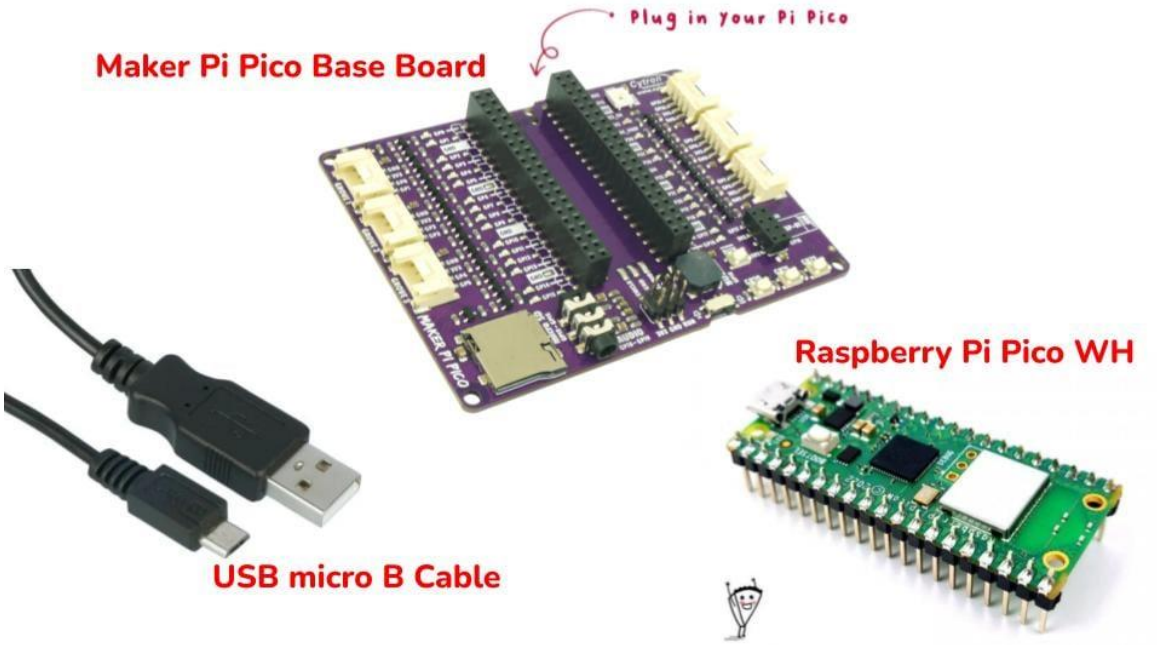
Características más destacables

- Trabaje fuera de la caja. ¡Sin soldadura!
- Acceso a todos los pines de Raspberry Pi Pico o Pico W en dos cabezales de pines de 20 vías
- Indicadores LED en todos los pines GPIO
- 3x pulsador programable (GP20-22)
- 1x LED RGB - NeoPixel (GP28)
- 1x zumbador piezoeléctrico (GP18)
- 1x conector de audio estéreo de 3,5 mm (GP18-19)
- 1x ranura para tarjeta Micro SD (GP10-15)
- 1x zócalo ESP-01 (GP16-17)
- 6x puerto Grove





- **Maker** Pi Pico Base con **Raspberry Pi Pico WSH (inalámbrico y presoldado con cabezales)**
    - 1 x Maker Pi Pico Base (Raspberry Pi Pico H o Pico WSH **NO está presoldado** a bordo)
    - 1 x Raspberry Pi Pico WSH (cabezales presoldados)
    - 1 x cable USB micro B



Recursos

- Hoja de datos de Maker Pi Pico (pdf)
- Esquema de Maker Pi Pico Rev1.2.0
- Demostración de Maker Pi Pico y código de ejemplo
- CAD 3D
- Página oficial de Raspberry Pi Pico
- Primeros pasos con Raspberry Pi Pico
- Hoja de datos de Raspberry Pi Pico (pdf)
- Hoja de datos de Raspberry Pi Pico W (pdf)
- Hoja de datos de RP2040
- Raspberry Pi Pico Python SDK
- Raspberry Pi Pico C/C++ SDK

Tarjeta MAKER PI RP2040

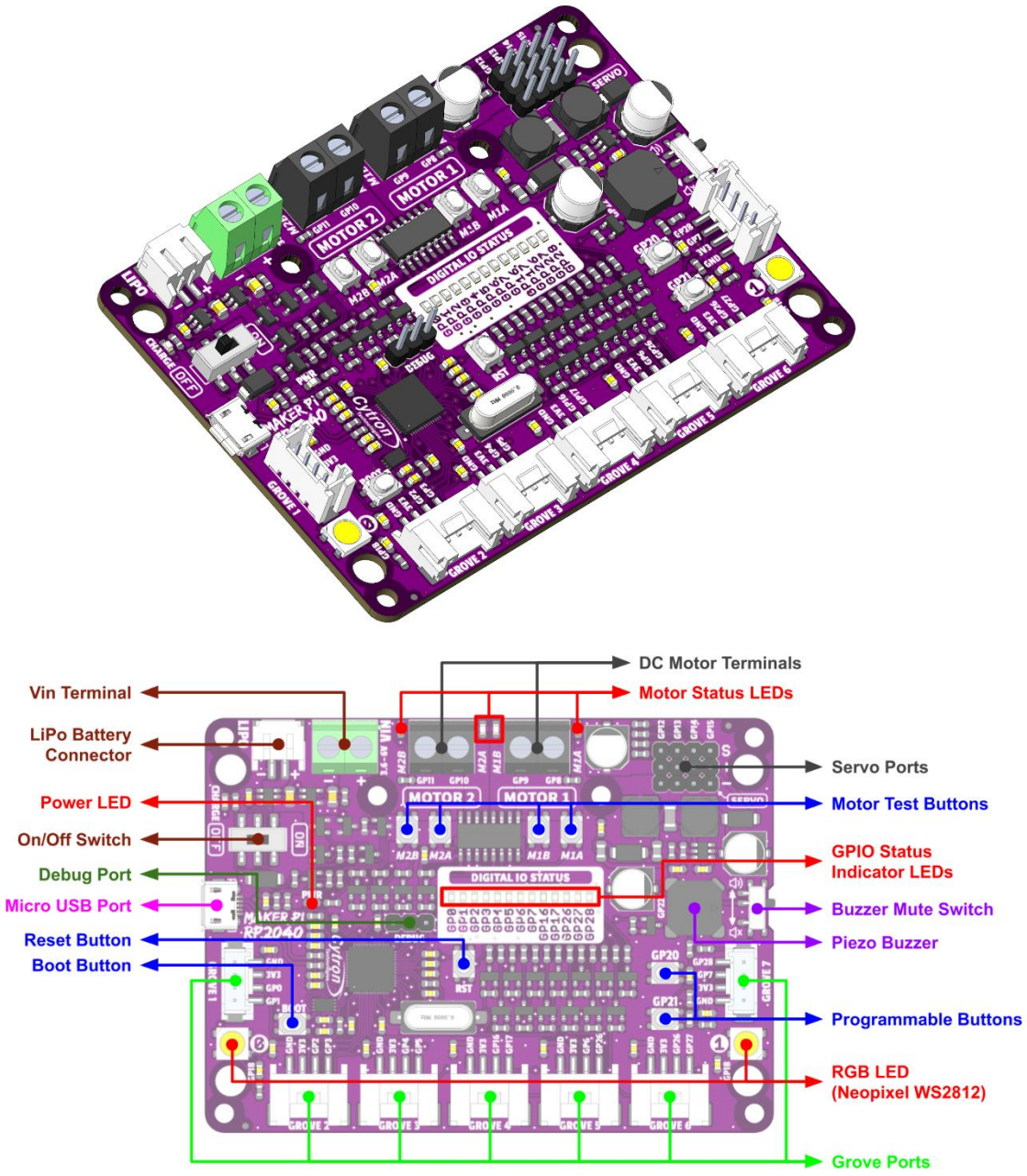


Figure 1: MAKER-PI-RP2040 Board Functions

Función	Descripción
Vin Terminal	Conecte a cualquier fuente de alimentación dentro de 3.6 - 6V.
Conector LiPo Bateria	Conector de batería LiPo Conectar a batería LiPo / Li-Ion de celda única. La batería es recargable a través de USB. * *La batería está protegida contra sobrecargas y descargas excesivas. Si la placa no se puede encender cuando la batería está conectada, cargue la batería para activar el circuito de protección de la batería.*

<table>
<tr><td>LED Encendico</td><td>Se enciende cuando se conecta</td></tr>
<tr><td>Interruptor On/Off</td><td>Enciende/apaga la alimentación.</td></tr>
<tr><td>Puerto depuración</td><td>Puerto de depuración del RP2040.</td></tr>
<tr><td>Puerto Micro USB</td><td>Utilizado para cargar programas desde la PC.
También se puede utilizar para encender la placa</td></tr>
<tr><td>Botón Reinicio</td><td>Presiónelo para reiniciar el RP2040.</td></tr>
<tr><td>Boton Para Modo Cargador</td><td>Mantenga presionado este botón mientras se reinicia el RP2040 para ingresar al modo de cargador de inicio. Se utiliza para cargar el firmware Micropython/Circuitpython o C/C++ personalizado.</td></tr>
<tr><td>Puertos Grove</td><td>Puertos Grove Conéctese a módulos Grove externos.
<table>
<tr><th>Grove Port</th><th>GPIO</th><th>PWM</th><th>SPI</th><th>I2C</th><th>UART</th><th>Analog</th></tr>
<tr><td rowspan="2">1</td><td>0</td><td>PWM0-A</td><td>SDI0</td><td>SDA0</td><td>TX0</td><td>-</td></tr>
<tr><td>1</td><td>PWM0-B</td><td>CSn0</td><td>SCL0</td><td>RX0</td><td>-</td></tr>
<tr><td rowspan="2">2</td><td>2</td><td>PWM1-A</td><td>SCK0</td><td>SDA1</td><td>-</td><td>-</td></tr>
<tr><td>3</td><td>PWM1-B</td><td>SDO0</td><td>SCL1</td><td>-</td><td>-</td></tr>
<tr><td rowspan="2">3</td><td>4</td><td>PWM2-A</td><td>SDI0</td><td>SDA0</td><td>TX1</td><td>-</td></tr>
<tr><td>5</td><td>PWM2-B</td><td>CSn0</td><td>SCL0</td><td>RX1</td><td>-</td></tr>
<tr><td rowspan="2">4</td><td>16</td><td>PWM0-A</td><td>SDI0</td><td>SDA0</td><td>TX0</td><td>-</td></tr>
<tr><td>17</td><td>PWM0-B</td><td>CSn0</td><td>SCL0</td><td>RX0</td><td>-</td></tr>
<tr><td rowspan="2">5</td><td>6</td><td>PWM3-A</td><td>SCK0</td><td>SDA1</td><td>-</td><td>-</td></tr>
<tr><td>26</td><td>PWM5-A</td><td>-</td><td>SDA1</td><td>-</td><td>ADC0</td></tr>
<tr><td rowspan="2">6</td><td>26</td><td>PWM5-A</td><td>-</td><td>SDA1</td><td>-</td><td>ADC0</td></tr>
<tr><td>27</td><td>PWM5-B</td><td>-</td><td>SCL1</td><td>-</td><td>ADC1</td></tr>
<tr><td rowspan="2">7</td><td>7</td><td>PWM3-B</td><td>SDO0</td><td>SCL1</td><td>-</td><td>-</td></tr>
<tr><td>28</td><td>PWM6-A</td><td>-</td><td>-</td><td>-</td><td>ADC2</td></tr>
</table></td></tr>
<tr><td>RGB LEDs (WS2812)</td><td>LED RGB WS2812B programable por el usuario. Conectado a GP18.</td></tr>
<tr><td>Botones programables</td><td>Accesibles desde el programa de usuario. Conectado a GP20 y GP21</td></tr>
<tr><td>Piezo Buzzer</td><td>Se puede utilizar para reproducir tonos o melodías. Conectado a GP22.</td></tr>
<tr><td>Interruptor Buzer OFF</td><td>Se utiliza para silenciar el zumbador piezoeléctrico.</td></tr>
<tr><td>GPIO Status LEDs</td><td>LED de estado de GPIO Indicadores LED para GPIO RP2040 en puertos Grove.
Encienda cuando el estado de GPIO sea alto.</td></tr>
<tr><td>Botones Test Motor</td><td>Presiónelos para probar la funcionalidad del controlador de motor.
El motor funcionará a toda velocidad.
● MxA: Adelante*
● MxB : Hacia atrás*</td></tr>
<tr><td>Puertos Servo</td><td>Conectores de puertos servo para 4 servomotores RC.
La señal está conectada a GP12, GP13, GP14 y GP15.
El voltaje V+ es igual al voltaje de la fuente de alimentación.</td></tr>
<tr><td>LEDs estado de motor</td><td>LED de estado del motor Se encienden cuando el motor está funcionando.
● MxA: Adelante*</td></tr>
</table>

	● MxB : Hacia atrás*
Terminales Motor DC	Terminales del motor de CC Conecte al terminal del motor. El voltaje del motor a toda velocidad es igual al voltaje de la fuente de alimentación. La dirección del motor depende de la polaridad. ● M1A: GP8 ● M2A: GP10 ● M1B: GP9 ● M2B: GP11

Table 1: MAKER-PI-RP2040 Board Functions

TABLA DE VERDAD DEL CONTROLADOR DE MOTOR

Entrada A (GP8 / GP10)	**Entrada B (GP9 / GP11)**	**Salida A (M1A / M2A)**	**Salida B (M1B / M2B)**	**Motor**
Bajo	**Bajo**	Bajo	Bajo	Freno
Alto	**Bajo**	Alto	Bajo	Adelante*
Bajo	**Alto**	Bajo	Alto	Atras*
Alto	**Alto**	Hi-Z (Open)	Hi-Z (Open)	Costa

Tabla 3: Tabla de verdad del controlador de motor

- *La dirección real del motor depende de la conexión del motor. Cambiar la conexión (MA y MB) invertirá la dirección.*

4. ¿Qué es MicroBlock?

MicroBlocks es un lenguaje de programación de bloques gratuito, similar a Scratch, para aprender computación física con placas de microcontroladores educativos como micro: bit, Adafruit Circuit Playground Express y muchos otros.

MicroBlocks permite a los principiantes completos comenzar rápidamente, desde niños de tan solo nueve años hasta adultos de todas las edades.

Sin embargo, **MicroBlocks** no es solo para principiantes. Se puede utilizar para aprender electrónica, experimentos científicos de instrumentos, automatizar su hogar y mucho más.

¿Qué hace que **MicroBlocks** sea diferente?

Si bien hay otros lenguajes de bloques para microcontroladores, lo que realmente distingue a **MicroBlocks** es su combinación de programación en vivo y operación autónoma. Otros lenguajes de bloques admiten una u otra de esas características, pero no ambas.

1. Programación en vivo

> **MicroBlocks** es un entorno en vivo. Haga clic en un bloque y se ejecuta inmediatamente, justo en el tablero. Prueba los comandos. Vea y grafique los valores del sensor en tiempo real. No más esperar a que el código se compile y descargue.

2. Operación autónoma

> **MicroBlocks** descarga tu código a medida que lo escribes. Cuando te guste lo que hace tu programa, simplemente desenchufa la placa y listo. Haz un juego, una aplicación de fitness o ropa iluminada que vaya a donde quiera que vayas.

3. Paralelismo

> ¿Desea mostrar una animación mientras controla un motor? ¡No hay problema! **MicroBlocks** le permite escribir scripts separados para cada tarea y ejecutarlos al mismo tiempo. Su código es más simple de escribir y más fácil de entender.

4. Portabilidad

> **MicroBlocks** se ejecuta en muchas placas diferentes, pero sus scripts son portátiles. Los botones, sensores y bloques de visualización se comportan de la misma manera en todas las placas con el hardware correspondiente. **MicroBlocks** incluso simula la pantalla LED micro:bit 5x5 en pantallas TFT.

5. Leer de Archivo

> Con **MicroBlocks**, la placa actúa como una tarjeta de memoria. No hay necesidad de archivos; Simplemente conecte una placa y los scripts volverán a aparecer. Entregue su tablero a un amigo para que pueda explorar su código. ¡Incluso podrían agregar una nueva característica genial!

Con **MicroBlocks**, la placa actúa como una tarjeta de memoria. No hay necesidad de archivos; Simplemente conecte una placa y los scripts volverán a aparecer. Entregue su tablero a un amigo para que pueda explorar su código. ¡Incluso podrían agregar una nueva característica genial!

¿Cómo funciona **MicroBlocks**?

El sistema **MicroBlocks** tiene tres componentes:

Un editor de bloques, que se ejecuta en un equipo host durante el desarrollo de código

Máquina virtual, que se ejecuta en el microcontrolador y ejecuta el código del usuario

Un sistema de comunicación que actualiza el código en la pizarra a medida que el usuario edita scripts.

El editor de bloques permite al usuario crear y editar código basado en bloques. También administra bibliotecas de **MicroBlocks** que proporcionan funcionalidad adicional. Algunas bibliotecas admiten sensores o dispositivos de salida como servos y NeoPixels. Otros proporcionan API para trabajar con texto, listas y música. Las bibliotecas están escritas en **MicroBlocks** y pueden ser exploradas, modificadas y ampliadas por los usuarios.

Al igual que MicroPython, el código de **MicroBlocks** se compila en códigos de bytes que son ejecutados por una máquina virtual que se ejecuta en el microcontrolador. Los códigos de bytes son instrucciones de bajo nivel similares al código máquina, pero independientes de cualquier arquitectura de procesador en particular. Este diseño facilita que **MicroBlocks** admita muchos microcontroladores de 32 bits diferentes. De hecho, la máquina virtual **MicroBlocks** no está restringida a microcontroladores; también se ejecuta en computadoras Linux como Raspberry Pi.

Si desea obtener más información sobre los códigos de bytes, consulte la página de la máquina virtual en el wiki. Si habilita los "bloques avanzados", puede usar el menú contextual de un script para ver las instrucciones y los códigos de bytes generados por sus propios scripts. Esa es una excelente manera de obtener una comprensión más profunda de cómo las computadoras ejecutan código.

El sistema de comunicación envía los códigos de bytes de los scripts a la máquina virtual y actualiza esos códigos de bytes a medida que el usuario edita los scripts. Dado que los scripts se vuelven a compilar y se envían a la máquina virtual de forma incremental, el código está listo para funcionar inmediatamente. Eso facilita las pruebas y mejoras del código.

El sistema de comunicación también envía comandos para iniciar scripts y procesa mensajes del microcontrolador que indican cuándo los scripts se detienen o devuelven resultados. Eso permite al editor proporcionar retroalimentación gráfica sobre lo que está sucediendo en el microcontrolador.

Una parte clave del aprendizaje sobre los sensores es ver cómo responden en tiempo real. Por ejemplo, ¿cómo cambia la aceleración a medida que lanzas y atrapas el micro:bit? El sistema de comunicación permite que los valores del sensor y el resultado del cálculo se muestren en una pequeña "burbuja de conversación". También admite la representación

gráfica de datos del sensor en tiempo real. La representación gráfica es una herramienta poderosa para construir intuición sobre procesos físicos y eléctricos en tiempo real.

¡Comenzar con **MicroBlocks** es fácil!

Sólo tienes que seguir estos sencillos pasos.

Necesitará una computadora (¡no un dispositivo móvil!) con un puerto USB, un cable USB y una placa. Puede ejecutar **MicroBlocks** en un navegador Chrome o Edge, o descargar una aplicación independiente para Chromebook, Windows, MacOS o Linux.

Paso 1

Configuración del equipo

Instalación de Windows

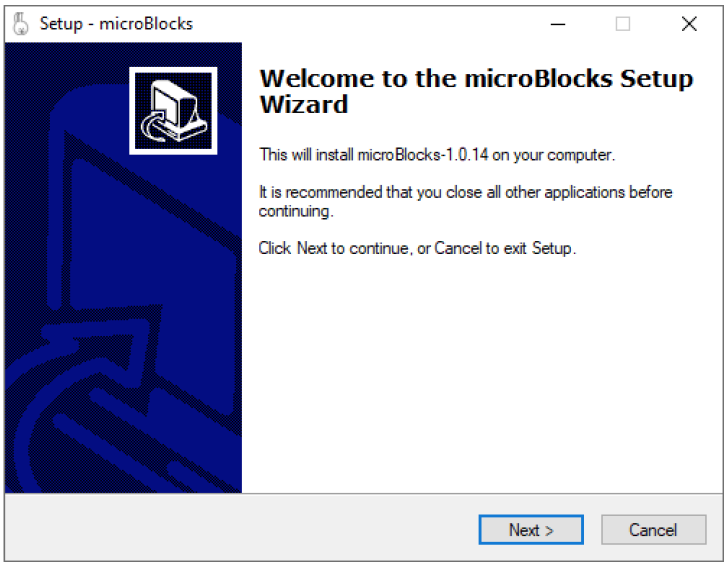
Ve a la página Descargar y haga clic en el botón Descargar.

En Chrome o Edge, se te advertirá que el archivo descargado podría dañar su computadora. No preste atención a este mensaje y siga con la descarga

A continuación, recibirá otra advertencia. Haga clic en **Mostrar más** y **Conservar de todos modos**.

Abra el archivo guardado, **configuración de MicroBlocks.exe**.

Después de una advertencia más, se abrirá el instalador.



Haga clic en las pantallas del instalador para instalar la aplicación.

El instalador lanzará **MicroBlocks** cuando haya terminado. También agregará un icono de acceso directo al escritorio que puede usar para ejecutar **MicroBlocks** más tarde.

Paso 2

Configuración de la placa

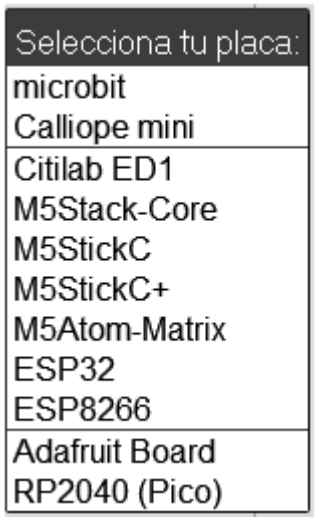
Cómo configurar tu placa para **MicroBlocks**.

La configuración de la placa es similar para **Citilab ED1**, M5 Stack, M5 StickC, **M5 StickC +**, **M5 Atom Matrix**, y otras placas populares basadas en ESP32 y **ESP8266** tales como como el **NodeMCU** y **Wemos D1 Mini**.

Conecte la placa al equipo.

En el menú **MicroBlocks** (icono de engranaje), seleccione **actualizar firmware a bordo**.

Seleccione su tipo de tablero en el menú.



Si está ejecutando **MicroBlocks** en el navegador, se le pedirá que Seleccione la placa si aún no está conectada.

Si no hay una tarjeta conectada aparecerá el siguiente aviso.

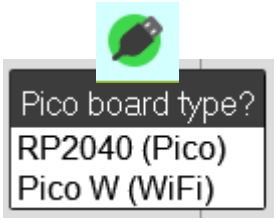
Conete el cable USB mientras mantiene presionado el botón BOOTSEL blanco e intente nuevamente.

Una vez que arrancamos de la manera indicada anteriormente y volvemos a solicitar la "actualización del firmware", seguidamente aprecerá una venta en donde seleccionaremos el tipo de Raspberry Pi Pico que tenemos en nuestra tarjeta.

Si todo va bien aparecerá el icono del USB de la siguiente forma

Paso 3

¡Empieza a programar!



Haga clic en el icono del conejito para iniciar la aplicación **MicroBlocks**.

Conecte la placa. Si ha instalado el firmware de **MicroBlocks** (consulte Configuración de la placa) debería aparecer un círculo verde detrás del icono USB.

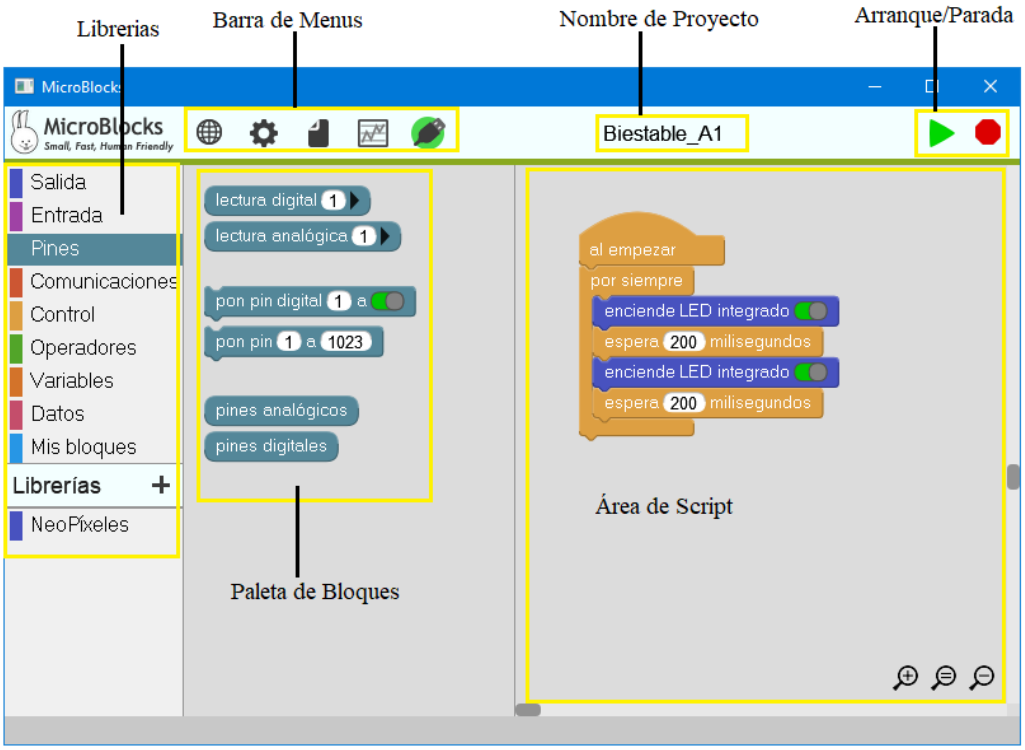
Para una introducción rápida al uso de **MicroBlocks**, mira este video (micro: bit) o este (Circuit Playground Express o Bluefruit). También puede explorar la guía del usuario y el manual de referencia de bloques.

5. Guía del usuario de **MicroBlocks**

*Cómo utilizar el editor de **MicroBlocks**. Material recogido de la página Web del Proyecto **MicroBlocks***

5.1. Visión general

La siguiente imagen muestra las principales áreas del editor de **MicroBlocks**.



Librerías de Bloques

Esta área contiene todas las categorías de bloques que se utilizan para programar en **MicroBlocks**. Se agrupan en nueve grupos codificados por colores. A medida que se seleccionan las categorías, los bloques correspondientes de esa categoría se enumerarán en el área Paleta de bloques junto a ella.

Para obtener una descripción detallada de todos los bloques, consulte las **Categorías de bloques** en la sección **Manual de referencia** de la WIKI.

Barra de menús

Proporciona tres menús del sistema, así como los iconos Gráfico y Conecta. Estos se describen en detalle a continuación.

Nombre del proyecto

Muestra el nombre del proyecto en el área de scripting.

Botones de inicio / parada

Estos consisten en dos iconos, Inicio y Parada, que se utilizan para controlar la ejecución de los programas **MicroBlocks**. El botón Inicio simula el encendido de un micro dispositivo ejecutando todos los bloques controlados por eventos.

Lista de librerías

El contenido de esta área refleja las diversas librerías que se cargan en función de los requisitos de los scripts de usuario y dispositivo micro. Podría ir desde totalmente vacío hasta con varios bloques. Para obtener una descripción detallada, consulte Librerías en la documentación del programa en la página Web.

Paleta de bloques

A medida que se realizan selecciones en el área Categorías de bloques, los bloques con esas funcionalidades específicas aparecen en esta área. Desde aquí, los bloques se arrastran y colocan en el área de script para escribir programas.

Área de script

Esta es el área principal del entorno **MicroBlocks** donde se producen todas las actividades de codificación. Los bloques se arrastran y se colocan en esta área para crear scripts de usuario y bloques personalizados (funciones).

5.2. Trabajar con scripts

Dado que el área de scripts es la parte más utilizada del editor, es útil mencionar las operaciones y las convenciones de uso particulares de esta área.

Operaciones de bloque

Los bloques que se colocan en el área de scripting se pueden mover libremente con el mouse.

Además, hay varias operaciones que uno puede realizar en los bloques y scripts colocados aquí.

Arrastrar y soltar

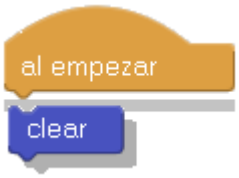
Como se mencionó anteriormente, el movimiento de arrastrar y soltar de los bloques es demasiado familiar para casi todos los que han pasado algún tiempo con entornos de desarrollo basados en bloques (¡Scratch, SNAP!, MakeCode, etc.).

Lo que es diferente y muy agradable en **MicroBlocks** es que un bloque en el que se hace clic obtiene una bonita **sombra paralela** que lo resalta contra el fondo. Este **efecto 3D** hace que sea fácil percibir que el bloque objetivo está de hecho en

movimiento, **flotando** sobre la superficie del área de scripting, incluso por encima de otros bloques de script.

Unión de bloque

A medida que uno o grupos de bloques que se mueven con el ratón, se acercan a otros bloques cercanos, aparecerá una **línea blanca horizontal**, designando un punto de ajuste.



Si uno suelta el mouse en ese momento, los bloques debajo del mouse se **ajustarán** a los bloques al otro lado de la línea blanca.

Bloquear menú de clic derecho

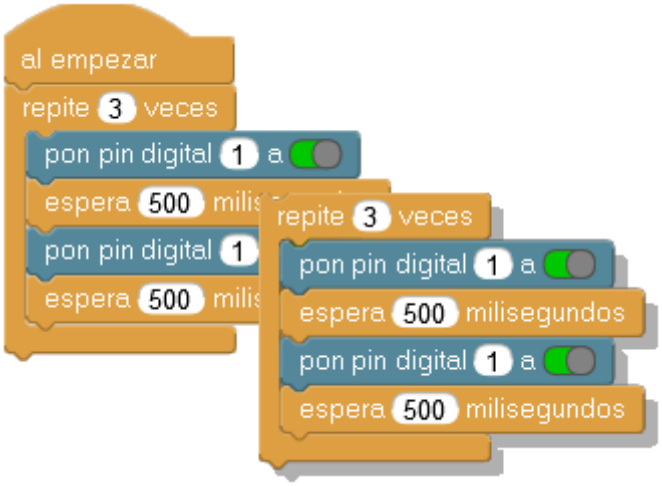
Al hacer clic con el botón derecho en cualquier bloque dentro de un grupo de bloques (script) se mostrará un **menú de nivel de bloque**.

duplica
duplica hasta abajo
extrae bloque
help
copia al portapapeles copy to clipboard as URL
guarda la imagen del programa guarda la imagen del programa con su resultado
muestra instrucciones muestra los bytes compilados
find uses of this block
elimina bloque

Duplicado

Al seleccionarlo, se hará una copia del bloque único en el que se coloca el mouse. Ciertos bloques (C-Blocks, IF-Blocks) pueden contener un montón de otros bloques en su interior.

Cuando se aplica duplicado a uno de estos bloques, se duplicará todo el grupo.



Duplicar todo

Esta opción es similar al duplicado, con la diferencia de que duplica todos los bloques por debajo del seleccionado.

Según el ejemplo que hemos estado usando, al hacer clic en el bloque **de repetición** y seleccionar duplicar todo se duplicará el bloque de repetición y todos los bloques debajo de él.

Una vez más, el punto a recordar es que, si los "bloques de abajo" están dentro de un C-Block, entonces duplicar todo está restringido al C-Block.

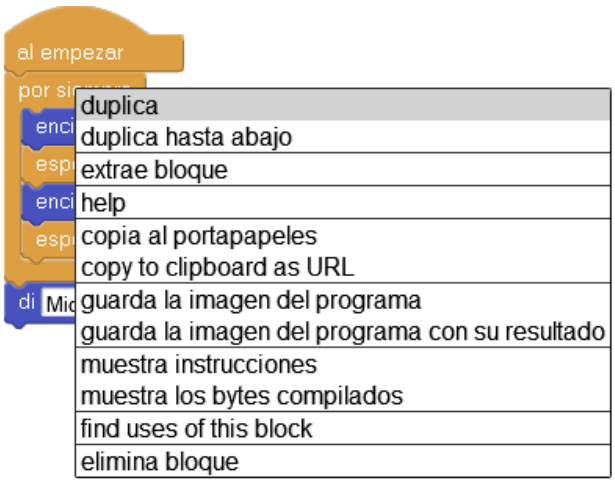
El segundo ejemplo muestra lo que sucede cuando se hace clic en el primer bloque dentro del bloque de repetición.

Solo se duplican los bloques dentro del bloque de repetición.

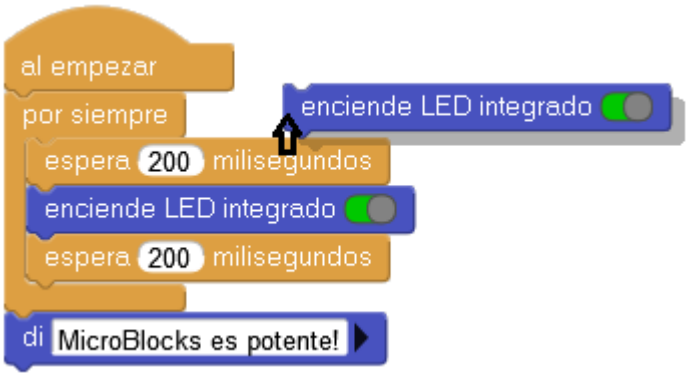
Extraer Bloque

Una de las trampas de la programación de bloques es que es bastante engorroso manipular bloques individuales que están en medio de un grupo de bloques.

MicroBlocks es uno de los pocos IDE que maneja esto con mucha gracia, al proporcionar un medio para **extraer** un bloque de un grupo de bloques.



Cuando se hace clic, el bloque de destino (o grupo de bloques) se extraerá y se adjuntará al puntero del mouse para moverlo y colocarlo en otro lugar.



Ayuda

El bloque de ayuda proporciona un enlace a la página WIKI que describe el bloque de destino. La página web se abre en una nueva pestaña y se posiciona automáticamente en el bloque seleccionado.

Copiar en el portapapeles

Esta es una operación muy útil que coloca una copia de los conjuntos de bloques seleccionados en el portapapeles.

Estos se pueden pegar no solo dentro del mismo proyecto, sino también entre diferentes proyectos.

Incluso es posible copiar/pegar entre diferentes instancias del editor de **MicroBlocks** (versión del navegador o versión independiente).

Esta operación tiene un par relacionado, **copiar todos los scripts en el portapapeles, al** que se accede cuando se hace clic en el fondo del área de scripting. Esa copia todos los scripts del proyecto.

Copiar al Portapapeles como URL

Esta es una característica que se utiliza principalmente para incorporar programas **MicroBlocks** en composiciones de páginas web, ya que el formato está codificado en URL. Aquí hay un ejemplo del programa utilizado anteriormente:

Y este es el mismo programa guardado en formato URL:

https://**MicroBlocks**.fun/run/**MicroBlocks**.html#scripts=GP_Scripts_script_454_96_{whenStarted setUserLED true }

Guardar imagen de script

Otra operación útil para hacer una copia de imagen del script o bloque seleccionado. La imagen se guardará en formato PNG.

Esta operación tiene un par relacionado, **guarde una imagen de todos los scripts**, al que se accede cuando se hace clic en el fondo del área de scripting. Ese hace una imagen de todos los guiones del proyecto.

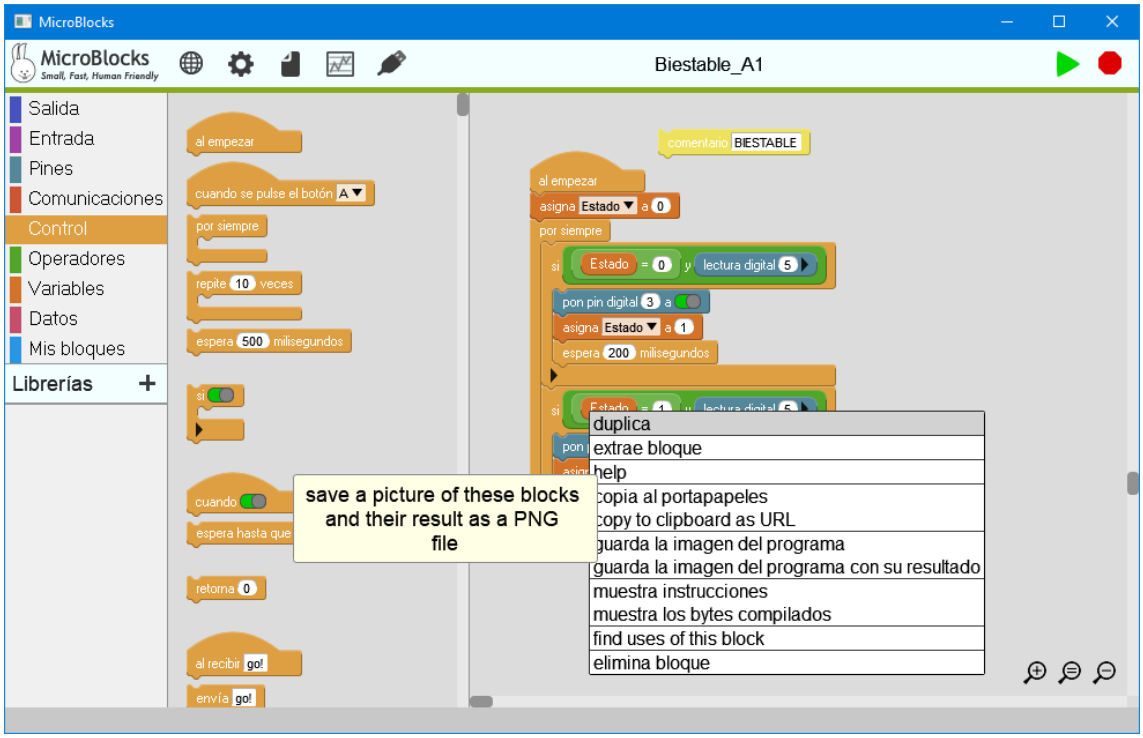
El tamaño de la imagen guardada se controla mediante la opción **establecer el tamaño del script exportado** en el menú que se muestra cuando se hace clic con el botón derecho en el área IDE.

El siguiente paso es donde se muestra un cuadro de diálogo de archivo del sistema y el usuario puede designar la ubicación de guardado en su PC.

Guardar imagen de secuencia de comandos con el resultado

Verá esta operación solo si está seleccionada la **opción Mostrar bloques avanzados** en el menú **MicroBlocks**.

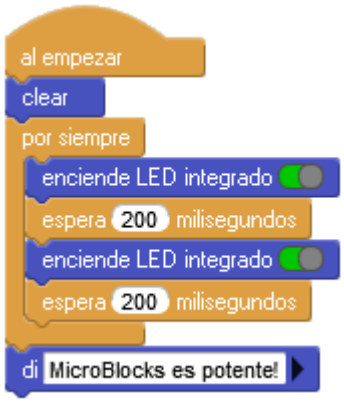
Al igual que con el otro bloque "guardar imagen de script", este hace una copia de imagen del script o bloque seleccionado. Sin embargo, antes de que la imagen sea instantánea, se ejecuta el contenido del bloque / script y el resultado, si se muestra, se incluye en la instantánea. Muchos de los ejemplos proporcionados en el Manual de referencia se realizan con esta opción.



Aquí hay una secuencia de imágenes que demuestran esta operación.

Se muestra un cuadro de diálogo de archivo del sistema y el usuario puede designar la ubicación para guardar en su PC.

NOTA: Dado que esta operación espera hasta que finaliza la ejecución del script antes de tomar la instantánea, NO es posible tomar fotografías de los scripts en los que están activos bucles largos o interminables.

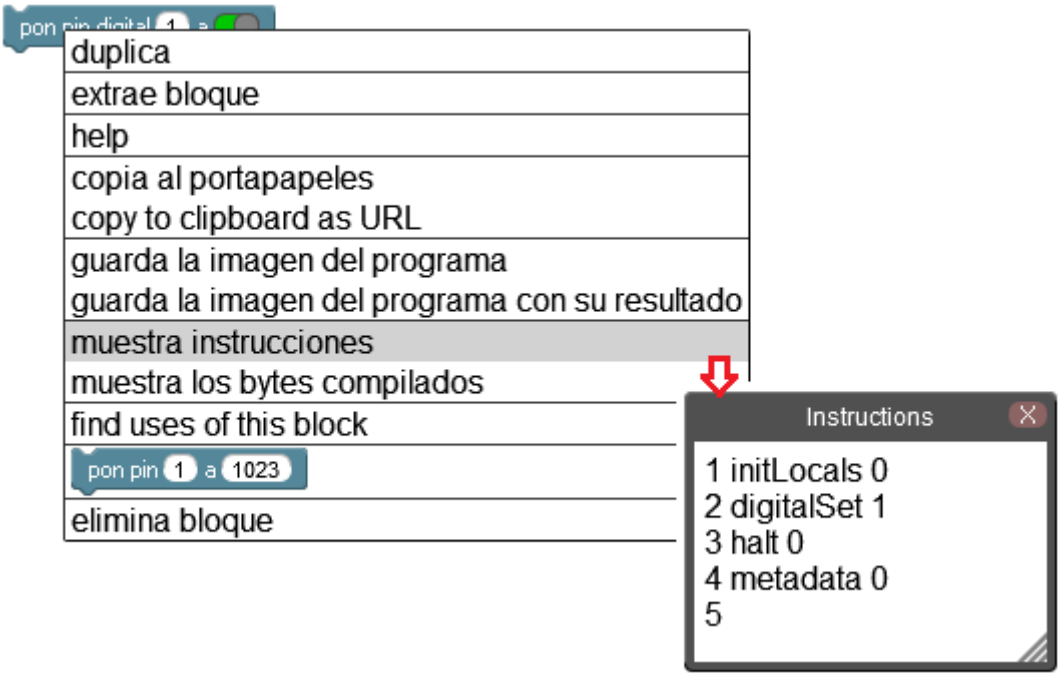


Debido a que hay un **bucle eterno antes de la visualización** de decir, el script nunca termina, el mensaje de decir nunca se muestra y la imagen de guardado del script con la operación de resultado nunca logra guardar la imagen.

RECUERDA: Ambas operaciones de guardar descritas anteriormente funcionan en el nivel de bloque individual, así como en una secuencia de bloques o scripts completos.

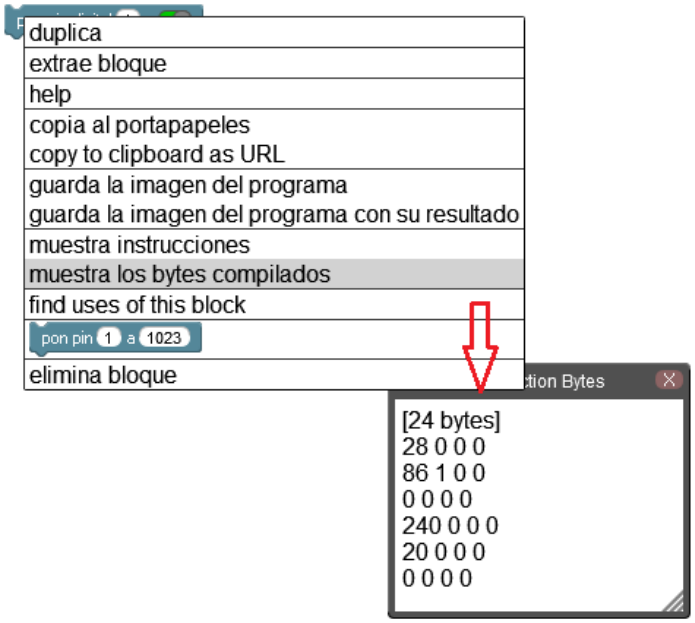
Mostrar Instrucciones

Verá esta operación solo si se selecciona **mostrar bloques avanzados** en el menú **MicroBlocks**.



Mostrar bytes compilados usados en este bloque

Verá esta operación solo si se selecciona mostrar bloques avanzados en el menú **MicroBlocks**.



La parte más importante de esta información es la primera línea, donde se muestra el número de bytes compilados. esta información es importante, porque los scripts de **MicroBlocks** tienen una limitación de tamaño debido al tamaño limitado de la memoria del microcontrolador. El tamaño máximo de bytes compilados para un script no debe superar los 1000 bytes.

Encontrar usos de este bloque

Esta operación enumera todas las apariciones del bloque seleccionado dentro de todo el script, incluidos los bloques personalizados y las bibliotecas en uso.
Cuando se hace clic en los elementos de la lista de resultados que muestran las ocurrencias, se mostrará la sección específica del programa que contiene el bloque en particular y el bloque parpadeará varias veces y luego se detendrá.

Mostrar definiciones de bloques

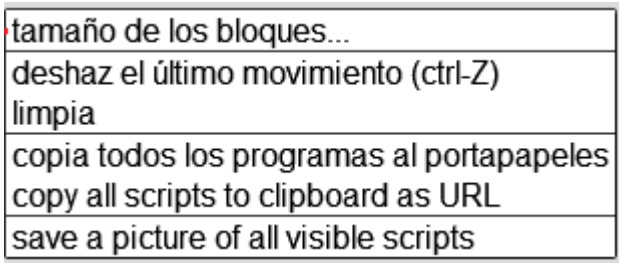
Este bloque se utiliza para mostrar el contenido de los bloques personalizados escritos por los usuarios y de cualquiera de los bloques de biblioteca escritos en **MicroBlocks**. Algunos de los bloques de la biblioteca se implementan utilizando código escrito en un nivel inferior. Estos no son posibles de mostrar y cuando se hace clic en ellos, no se mostrará la opción para mostrar la definición de bloque.

Borrar Bloque

Elimina el bloque o script de destino.

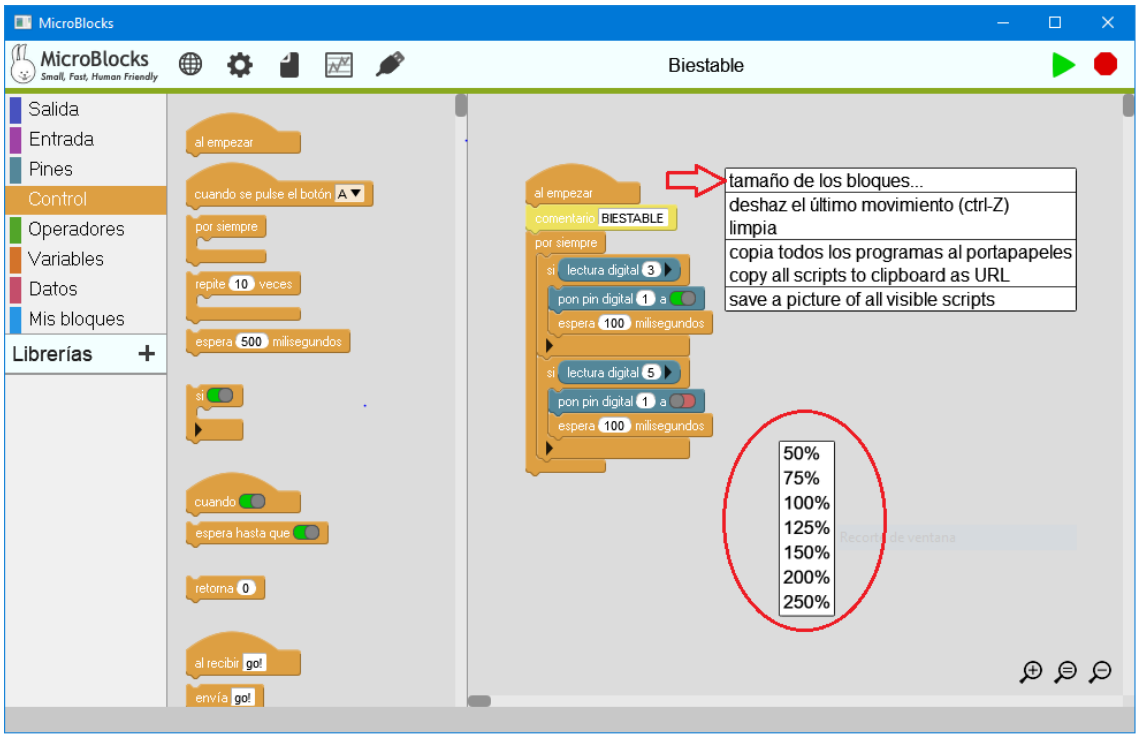
Menú contextual del área de secuencias de comandos

El menú final que un usuario puede encontrar mientras trabaja en el área de secuencias de comandos es el que se muestra cuando se hace clic con el botón derecho en la superficie.



Configurar tamaño de bloque

Cuando se selecciona **establecer tamaño de bloque**, se mostrará un menú de tamaños de bloque.



Los tamaños de los bloques se ajustarán en función de la selección realizada por el usuario. Aquí están los resultados de las selecciones 100% y 250%.

Recupera lo borrado (ctrl-Z)

Invierte el último movimiento de bloque completado. Las ediciones (copiar, eliminar y pegar) no cuentan como movimientos. Los movimientos están básicamente restringidos a los cambios de ubicación de bloques realizados con un clic del mouse.

De hecho, si no se realizaron movimientos de bloque o secuencia de comandos cuando se hizo clic con el botón derecho en el área de la secuencia de comandos, la opción de anular no se muestra en el menú.

Borrar

Organiza los segmentos del guión y los bloques sueltos en la pantalla en una disposición de arriba hacia abajo y de izquierda a derecha en el lado izquierdo del área de guión. Los bloques se organizan en múltiples columnas determinadas por el ancho del área IDE. ¡Esta operación no se puede deshacer!

Copiar todos los guiones al portapapeles

Este es el par complementario de la operación de copiar al portapapeles. Mientras que el otro solo copió el bloque de destino o la secuencia de comandos, este copia todos los guiones en el área de secuencias de comandos.

No copia definiciones de bloques personalizados. Pero copia cualquier biblioteca cargada.

Copiar todos los scripts al portapapeles como URL

Esta es una característica que se utiliza principalmente para incorporar programas de **MicroBlocks** en composiciones de páginas web, ya que el formato está codificado en URL. Aquí hay un ejemplo del programa utilizado anteriormente:

Y este es el mismo programa guardado en formato URL:
https://**MicroBlocks**.fun/run/**MicroBlocks**.html#scripts=GP Scripts script 454 96 { whenStarted setUserLED true }

Si bien esta muestra de guardado de URL es de un programa muy pequeño (solo dos bloques) y la versión de URL es solo de una línea, un programa real tendrá un tamaño considerablemente mayor y su versión de URL también será de una longitud considerable.

Guardar una imagen de todos los guiones

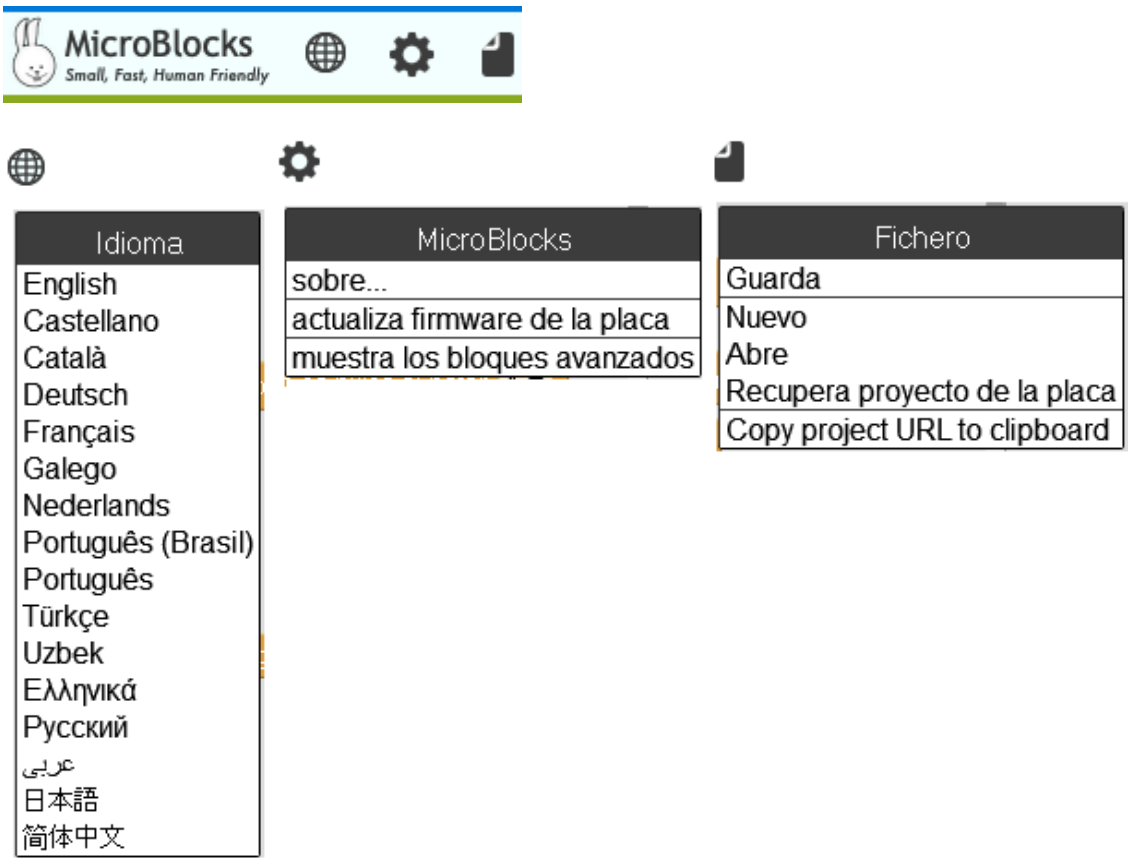
Esta es la versión complementaria de la imagen guardada de la operación de secuencia de comandos. En lugar de un solo bloque o secuencia de comandos, este creará una imagen de todas las secuencias de comandos en el área de secuencias de comandos en formato PNG.

Establecer la escala del script exportado

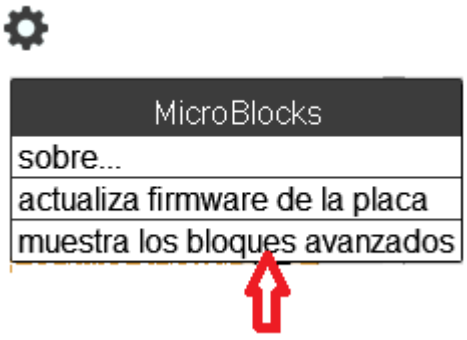
Además de los controles de desplazamiento del mouse y de la ventana, las teclas de flecha del teclado brindan soporte de desplazamiento mientras se trabaja en esta área.

5.3. Barra de Menú

Los primeros tres íconos presentan menús desplegables cuando se les hace clic.



Al seleccionar Mostrar bloques avanzados, aparecen algunos bloques adicionales en la paleta. También habilita funciones adicionales y opciones de menú que necesitan los usuarios más avanzados.



Trazar Gráfico y conectar

Trazar gráfico abre una ventana de gráfico utilizada por el bloque de gráfico para trazar datos. (Para obtener más información, consulte la descripción del bloque gráfico en el manual de referencia).

Conectar se utiliza para mostrar y conectarse a los puertos USB del sistema que tienen micro dispositivos conectados a ellos.

Arrancar/Parar

En el extremo derecho de la barra del menú principal se encuentran los dos íconos Iniciar y Detener, que se utilizan para controlar la ejecución del programa.

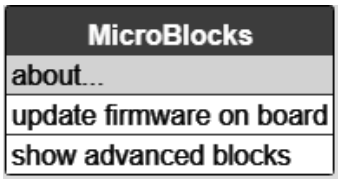
5.4. Descripciones del menú

Las siguientes secciones con PESTAÑAS describen en detalle todo el contenido y la funcionalidad de los menús.

Idioma

Este elemento del menú permite al usuario seleccionar uno de los muchos Idiomas a los que está traducido el editor de **MicroBlocks**. Una vez establecido, todos los menús, mensajes y bloques de código se presentará en ese idioma.

La opción Personalizada... en la parte inferior del menú se usa al crear traducciones para el redactor Permite al traductor probar el recién creado archivo de traducción seleccionándolo del menú del sistema y aplicándolo.



MicroBlocks

about...

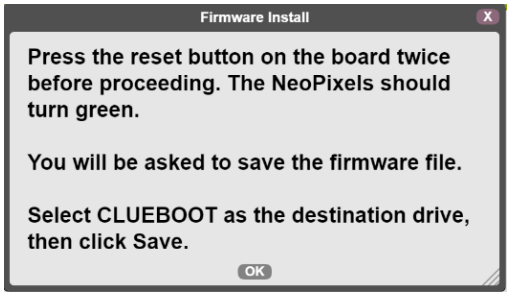
Muestra información sobre el editor de **MicroBlocks** y las versiones de Firmware.

Actualizar el firmware de la tarjeta

Este elemento permite al usuario cargar la última versión del firmware en el micro dispositivo conectado. Dependiendo de las condiciones, se puede presentar un menú de selección de dispositivos para que el usuario elija. Después de la selección, se mostrarán diferentes mensajes solicitando más acciones.

MicroBlocks es compatible con muchas placas de microcontroladores. Consulte la sección Tarjetas para obtener información detallada.

Aquí hay algunos mensajes de muestra para el dispositivo Adafruit CLUE:



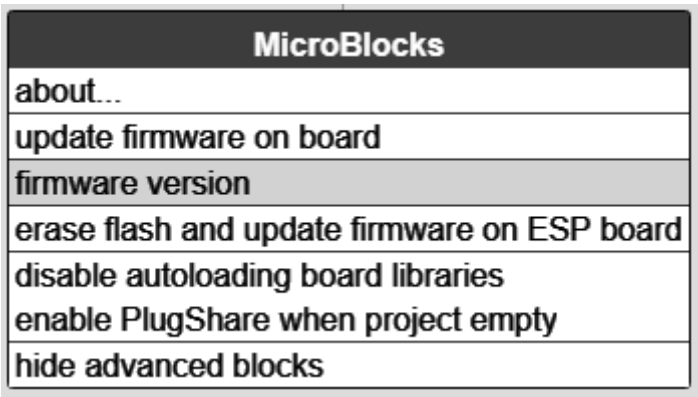
A medida que avanza la actualización del firmware, el icono verde de USB estará apagado. Una vez completada la actualización, se debe restablecer la conectividad entre el micro dispositivo y el puerto USB. Consulte la sección USB.

Mostrar bloques avanzados

Cuando se selecciona, se habilitarán las opciones avanzadas en los menús, así como los bloques avanzados en las Categorías de bloque.

Una vez habilitado, el elemento del menú cambiará para **ocultar los bloques avanzados**.

MicroBloques Avanzado

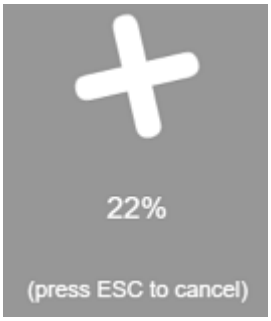


Versión de firmware

Muestra la versión de firmware de la máquina virtual para el micro dispositivo.

Borrar la memoria flash y actualizar firmware en placa ESP

Los tableros ESP son una categoría especial de tableros de **Espressive Corp**. que cuentan con capacidad WIFI. Tienen diferentes especificaciones técnicas sobre cómo se pueden actualizar. Este elemento borra totalmente el contenido de la memoria de la placa y carga el firmware más reciente.
Se muestra un menú de selección de dispositivos para elegir. Una vez realizada la selección, comienza la actualización del firmware. Una pantalla gráfica informa del progreso.



Cuando se complete la actualización, es posible que se le pida que restablezca la conectividad USB. Consulte la sección USB.
Deshabilitar la carga automática en la tarjeta de las bibliotecas

Una de las sutilezas del editor de **MicroBlocks** es que siempre trata de simplificar las cosas para los usuarios. Dado que cada microdispositivo tiene diferentes características, funciones y capacidades, **MicroBlocks** intenta automáticamente complementar las funcionalidades básicas cargando varias bibliotecas.

Aquí, como ejemplo, hay dos bibliotecas cargadas para el dispositivo micro: bit:

Cuando se selecciona, este elemento deshabilita esta funcionalidad; y depende del usuario cargar las bibliotecas requeridas.

Habilitar PlugShare cuando el proyecto esté vacío

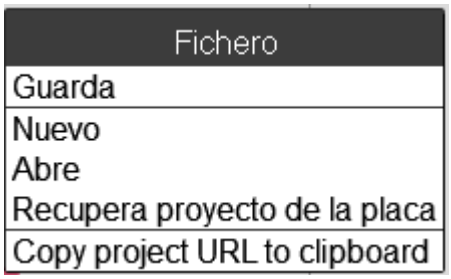
Otra característica SUPER del editor de **MicroBlocks** es que cargará automáticamente un proyecto desde el micro dispositivo adjunto, si no hay otro proyecto cargado en el editor.

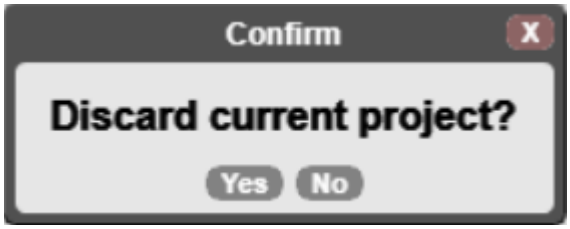
Para habilitar esta función, todo lo que hay que hacer es seleccionar este elemento en el menú. Una vez hecho esto, el editor leerá el proyecto desde el dispositivo y luego procederá a cargarlo en el editor.

Aquí hay pantallas de un dispositivo micro: bit, cargando automáticamente un proyecto:

Esta función permite a los usuarios compartir proyectos entre ellos simplemente conectando sus dispositivos a la PC.

Fichero

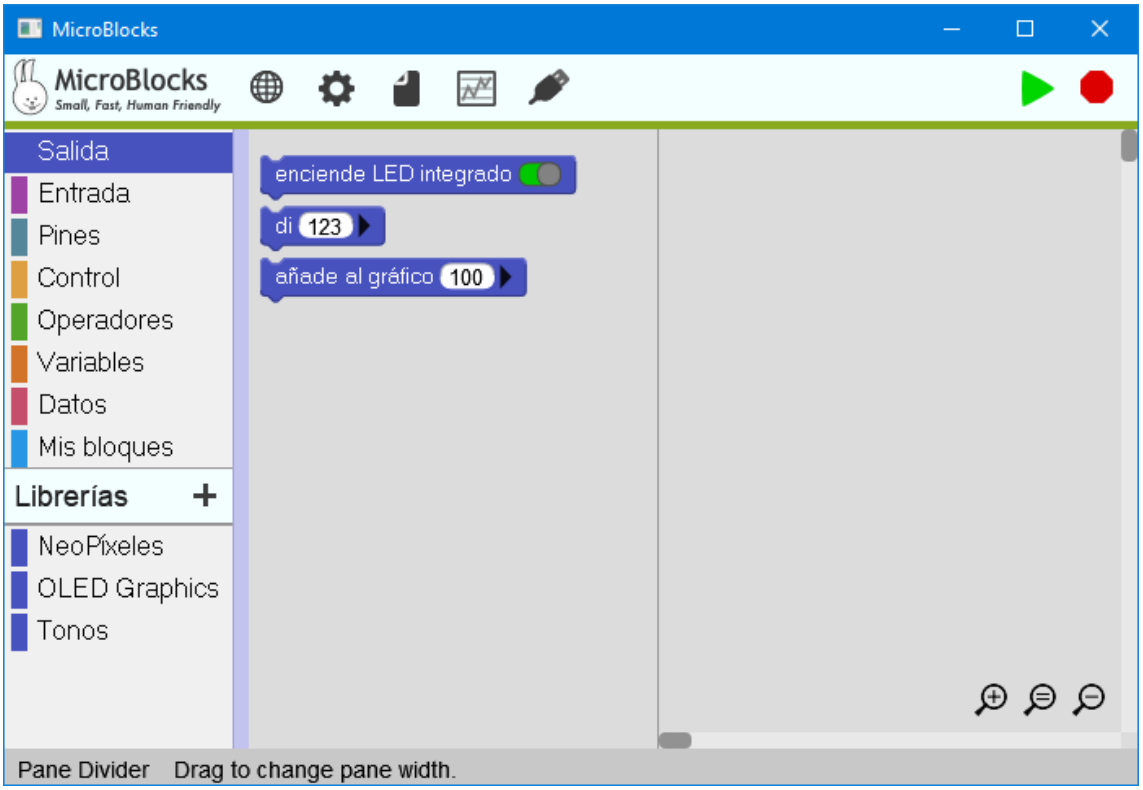




Nuevo

Si hay un programa cargado en el área de trabajo, mostrará un mensaje solicitando confirmación para borrar el proyecto activo.

Tras la confirmación positiva, comienza una nueva área de secuencias de comandos limpia como se muestra a continuación:



El área de secuencias de comandos se borra de los bloques de secuencias de comandos anteriores. Todas las bibliotecas añadidas por el usuario se borran. Según el tipo de micro dispositivo y la configuración de la opción **habilitar la carga automática de las librerías de la tarjeta**, se cargan ciertas bibliotecas.

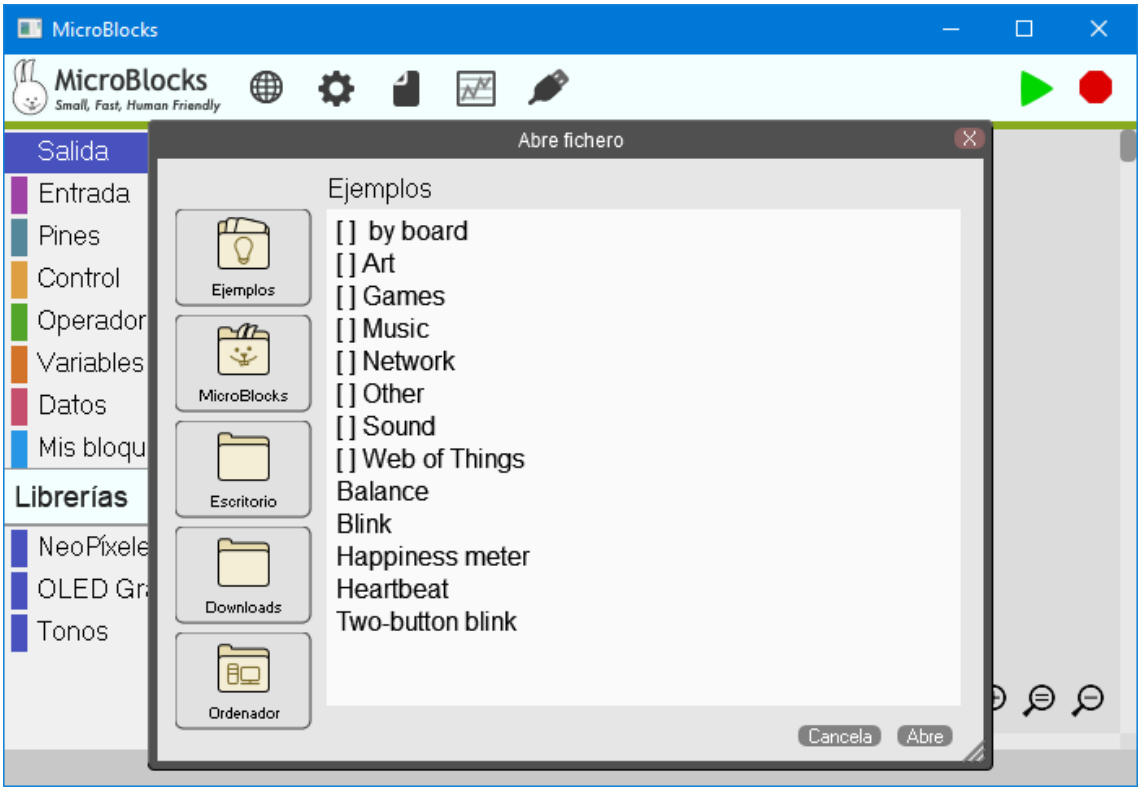
Si el ícono de conexión no es verde y habilita **PlugShare cuando se selecciona el proyecto vacío**, tan pronto como se establezca la conexión USB, **MicroBlocks** leerá y cargará el proyecto desde el micro dispositivo conectado.

Abrir

Así es como los proyectos previamente guardados en la PC del usuario se cargan en el área de trabajo. Además de los proyectos propios del usuario, se pueden cargar otros proyectos de muestra en varias categorías.

Si hay un proyecto activo en el área de secuencias de comandos, se muestra un mensaje de confirmación. Ver Archivo Nuevo arriba.

Se presenta un diálogo de apertura de archivo:

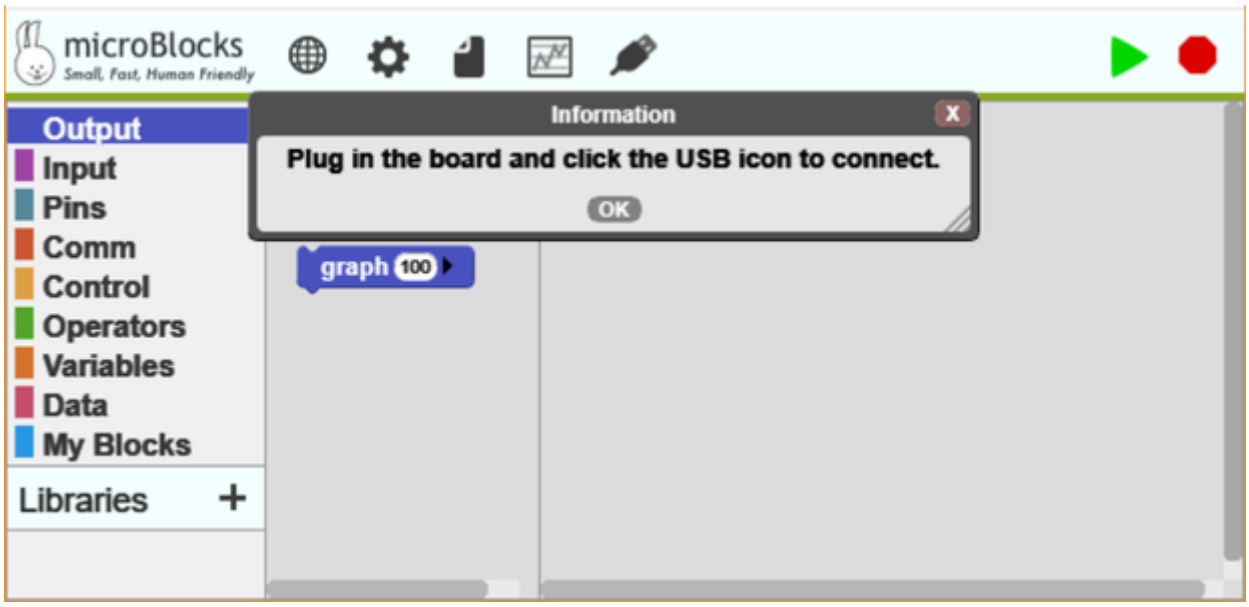


La primera exhibición se centra en la categoría **Ejemplos**, donde se presentan muchos proyectos de **MicroBlocks** para diferentes micro dispositivos y opciones de hardware.

Se puede acceder a los directorios de la PC del usuario cambiando a la selección de la pestaña **Computadora** a la izquierda. Esto presentará el cuadro de diálogo estándar para abrir archivos del sistema, donde se pueden seleccionar proyectos guardados previamente.

Abrir el contenido de la tarjeta

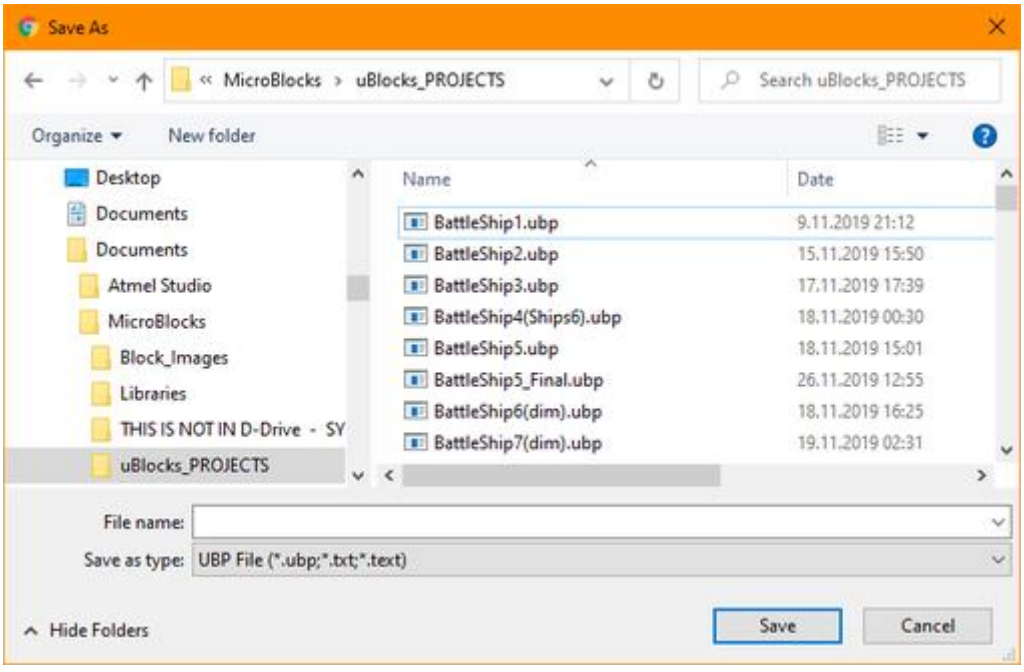
Si el **ícono de conexión** no es verde y **habilita PlugShare cuando se selecciona proyecto vacío**, entonces esta opción será visible en el menú de archivo. No será visible si la conectividad USB está establecida y el ícono de conexión es verde. Si hay un proyecto activo en el área de secuencias de comandos, se muestra un mensaje de confirmación. Ver Archivo Nuevo arriba. Suponiendo que se cumplan las condiciones anteriores, tan pronto como se seleccione el elemento, aparecerá un mensaje para conectar el dispositivo:



Cuando se establece la conexión USB, **MicroBlocks** leerá y cargará el proyecto desde el micro dispositivo conectado.

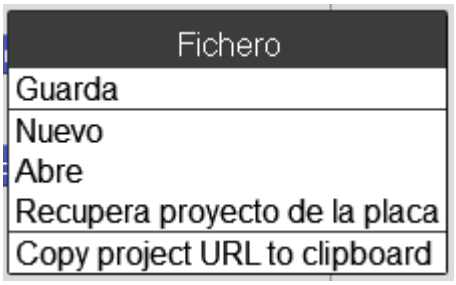
Guardar

Una vez que un proyecto está listo, debe guardarse en la PC del usuario.
Al seleccionar GUARDAR, se mostrará un cuadro de diálogo de archivo del sistema, donde el usuario puede dar un nombre al programa y guardarlo en su PC.



No hay una ubicación u opción preconfigurada para guardar archivos en **MicroBlocks**.
La designación de las ubicaciones de los archivos en la PC es responsabilidad del usuario.

Fichero Avanzado



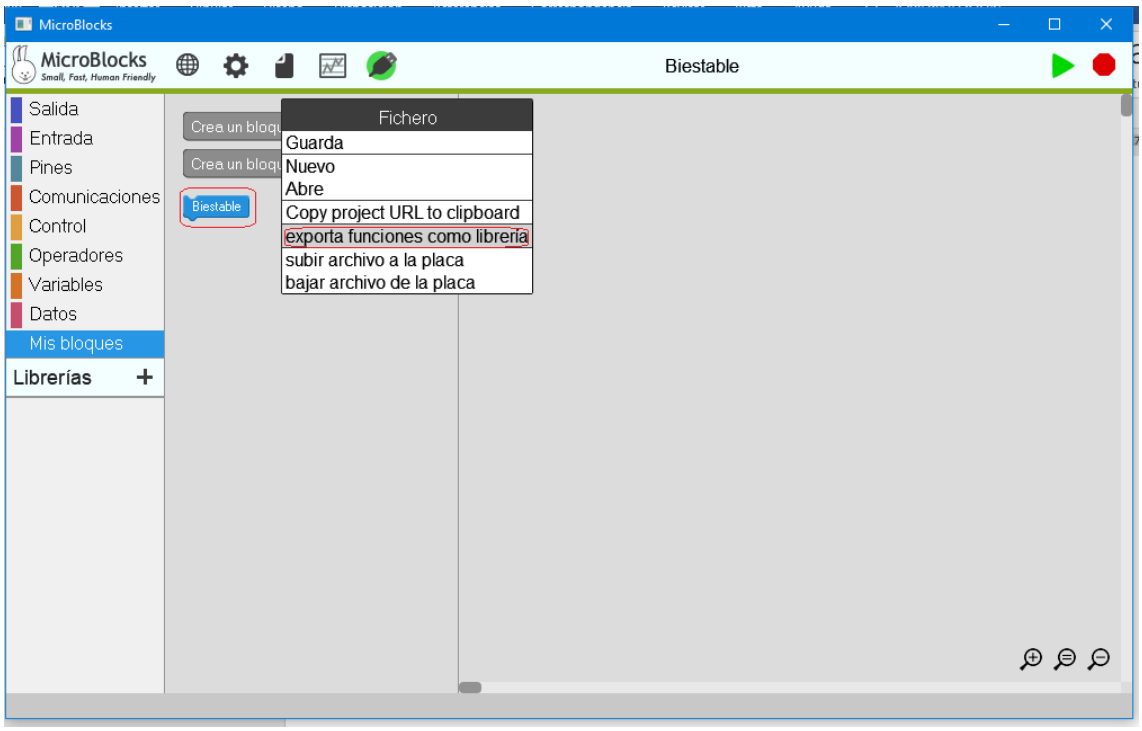
Exportar funciones como librería

Cuando se crean bloques personalizados en un proyecto, normalmente se guardarán como parte del archivo de proyecto del usuario con una extensión UBP.

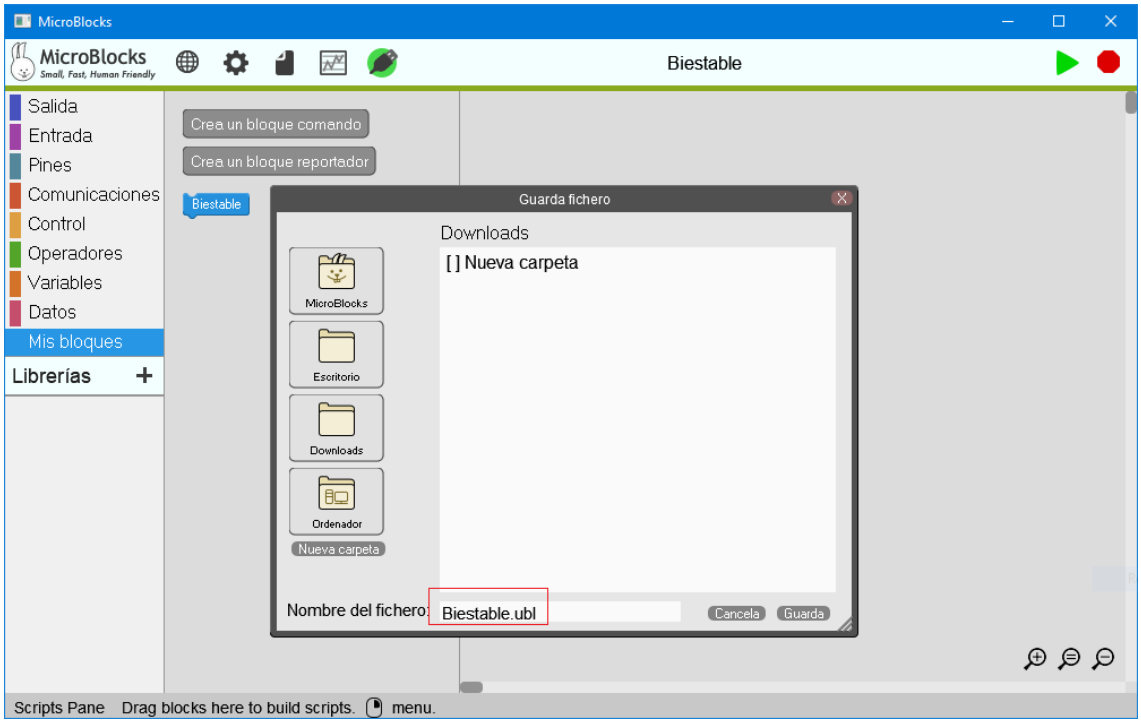
Sin embargo, si desea crear una biblioteca a partir de los bloques personalizados y guardarlos y cargarlos de forma independiente como código de biblioteca (con una extensión UBL), será necesario exportarlos desde el script y guardarlos en un archivo separado.

Esta selección hace que eso suceda.

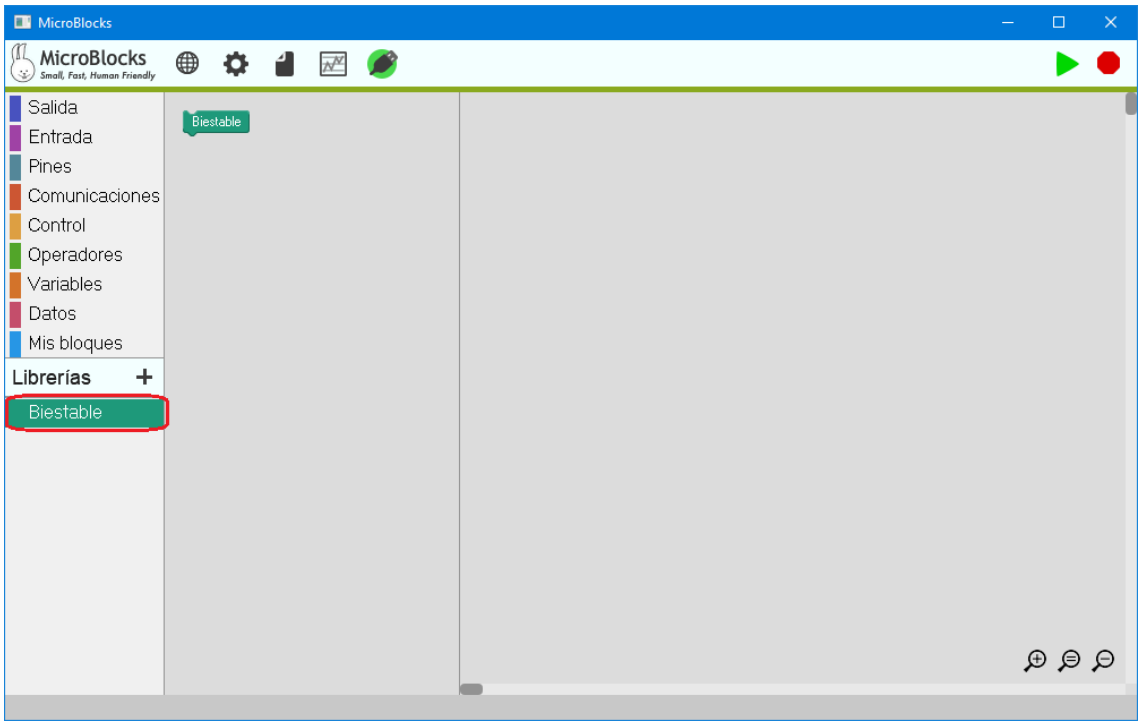
Mostrará una ventana de diálogo, donde se puede asignar y guardar el nombre de la biblioteca. El nuevo archivo tendrá una extensión UBL. La función que queremos exportar se llama "**Biestable**"



Guardamos la función como librería.

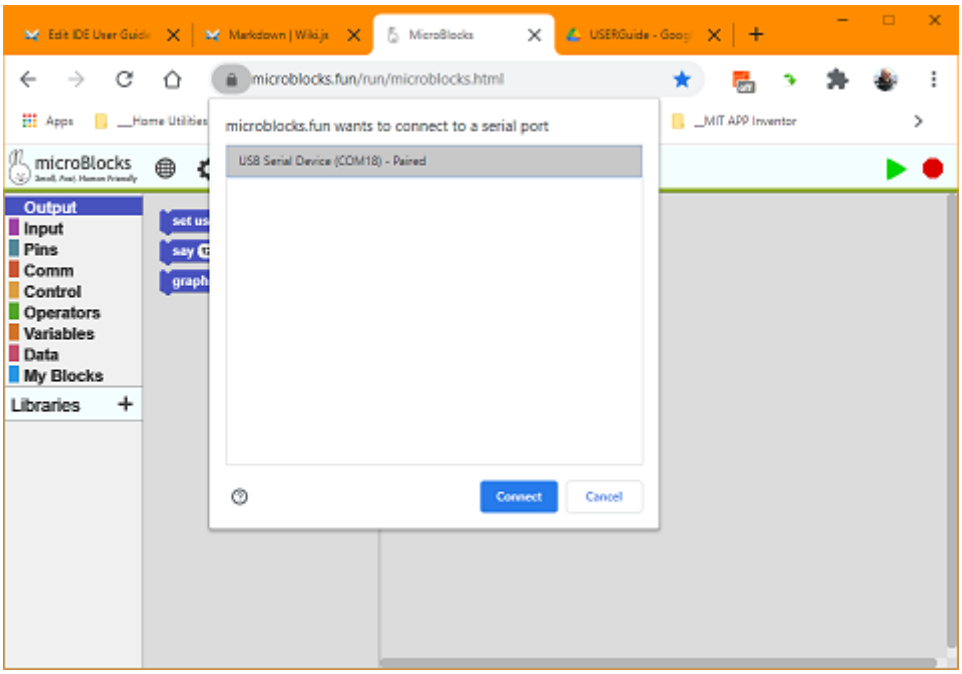


Para cargar la librería que hemos creado basta con pulsar el **Librerías +** y aparecerá la ventana de la figura en la que buscamos nuestra librería "**Biestable.ubl**"



Para obtener más información específica sobre **cómo crear una biblioteca**, consulte el elemento del menú WIKI: Creación de una nueva biblioteca.

Conectar



Como se mencionó anteriormente, el **icono Conectar** se utiliza para mostrar y conectarse a los puertos USB del sistema que tienen micro dispositivos conectados a ellos.

Cuando se hace clic, se muestra un menú de puerto USB del sistema, con una lista de todos los dispositivos conectados a USB en la PC. Al seleccionar una entrada y hacer clic en **Conectar**, se establecerá una conexión con el micro dispositivo correspondiente. Tras una conexión exitosa, el icono USB cambiará a uno con un fondo circular verde: .

Es importante prestar atención al estado de este icono. **MicroBlocks** tiene una gran cantidad de procesos internos automatizados para facilitar la vida del usuario cuando se trabaja en el editor. La actualización y sincronización automática del código del proyecto en el dispositivo conectado es una de ellas. Esto solo es posible cuando el icono está en modo verde.

De vez en cuando, por varias razones, la conexión al micro dispositivo puede cortarse. En estos momentos, el icono no mostrará el círculo verde. Si encuentra esta condición, la edición de los programas debe detenerse y se debe investigar la razón de la desconexión.

Es aconsejable no desarrollar o solucionar problemas de programas cuando se corta la conexión al micro dispositivo conectado, indicado por la ausencia del círculo verde alrededor del icono USB.

Empezar 

MicroBlocks siempre está en vivo; el usuario puede hacer clic en bloques o scripts individuales para ejecutarlos sin hacer clic en el botón Inicio.

Lo principal que hace el botón Inicio es simular el encendido del dispositivo iniciando todos los scripts con **cuando se inician** y todos los scripts **de cuándo <condición>**.

Parar 

El icono STOP detiene la ejecución del proyecto. Todas las variables están desasignadas. Todos los scripts con **cuándo se inicia** y **cuándo <condición>** se detienen.

El proyecto cargado en su micro dispositivo estará intacto. De hecho, puede desconectarlo, encenderlo desde una fuente externa y ejecutar su proyecto aparte del editor.

Glosario

Frase	Definición
Bloques personalizados	Un conjunto de scripts desarrollados por el usuario para lograr una funcionalidad específica. También se puede denominar función personalizada.
Guión	Pila única de bloques.
Biblioteca	Scripts de propósito específico proporcionados con **MicroBlocks** para manejar varias características y funcionalidades de micro dispositivos, así como para mejorar las capacidades estándar de **MicroBlocks**.
Proyecto	Una combinación de scripts, bloques personalizados y bibliotecas.

Documentación sobre **MicroBlocks:** Home | **MicroBlocks** Wiki

6. Prácticas de programación con **MicroBlocks**

Índice de Practicas

6.1. Seguidor de Entrada/Salida Digitales

Objetivo

Manejar una entrada digital y una salida digital programando distintas formas de ejecución. Usaremos un Botón y un LED

Funcionamiento:

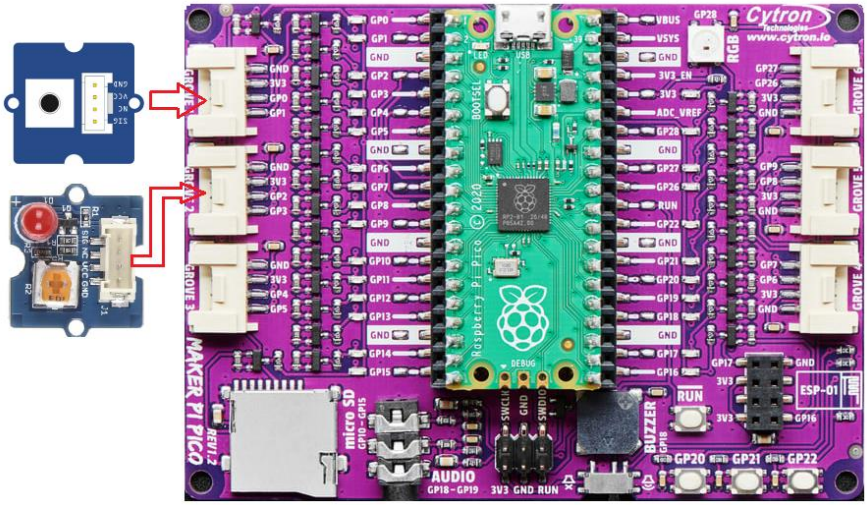
Usaremos el bloque “cuando..”

- Cuando pulsamos el botón conectado en el GP1 se activa el LED conectado en el GP3
- Cuando soltamos el botón conectado en el GP1 se apaga el LED conectado en el GP3

Entradas salidas:

- GP1 Botón Entrada Digital
- GP3 LED Salida Digital

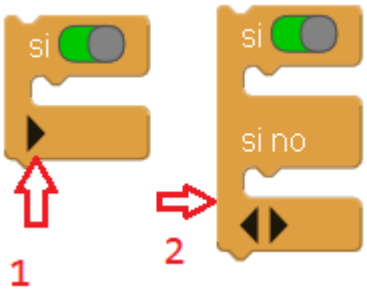
Esquema de montaje:



Programa:

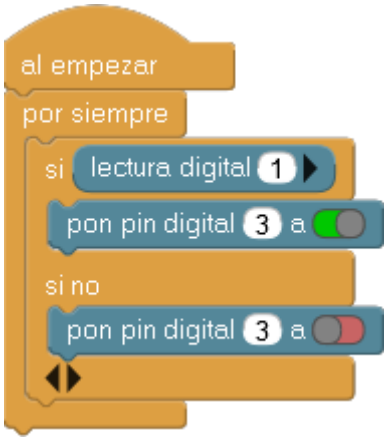
Actividades de Ampliación

1. Realizar el mismo ejemplo, pero usando un bloque condicional:



Solución

En este caso seleccionamos el bloque "**si**" y ponemos en la parte de "**condición**" el valor digital leído en el pin GP3 de la tarjeta

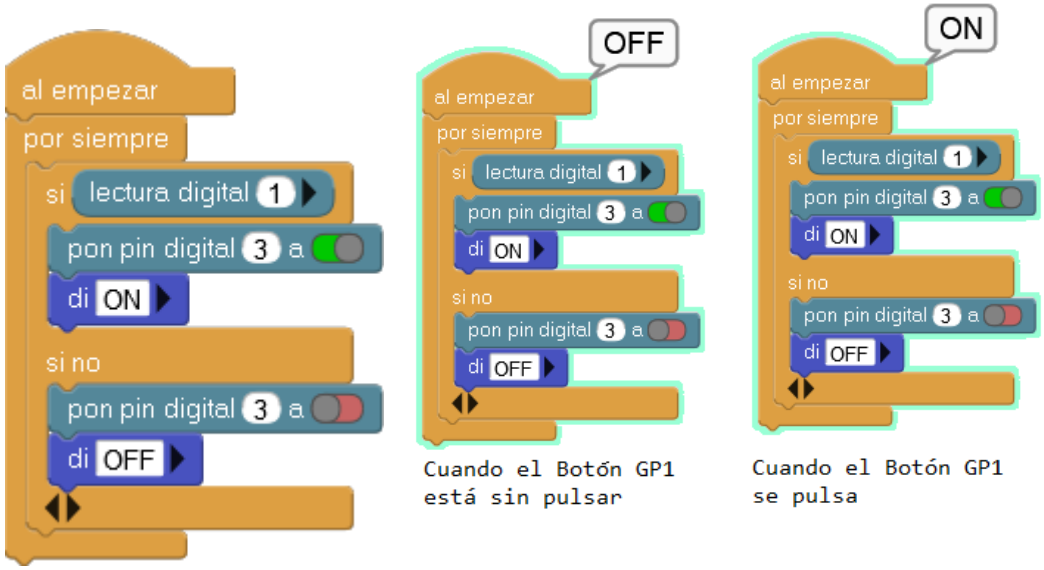


2. Realizar el mismo ejemplo mostrando el estado del LED en la misma pantalla de trabajo de **MicroBlocks**. Utiliza el bloque señalado

Solución

Para esta acción hacemos uso del bloque que se indica en la imagen anterior, y con ello podemos comprobar que mientas que se ejecuta el programa en modo Online se muestra el texto en la parte superior del bloque.

Vemos en la siguiente imagen el aspecto de la aplicación cuando esta funcionando.



6.2. Blink LED

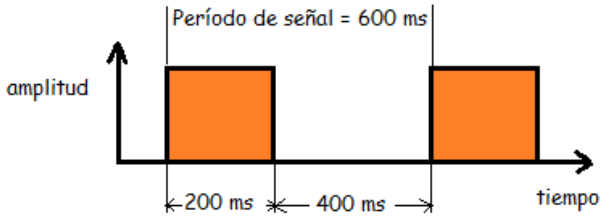
Objetivo

Queremos manejar una salida digital en la que pondremos un LED y sobre el queremos actuar realizando un intermitente.

Funcionamiento:

Con este programa vamos a hacer que el GP7 de la tarjeta reciba señales digitales de encendido y apagado de acuerdo a la siguiente trama temporal.

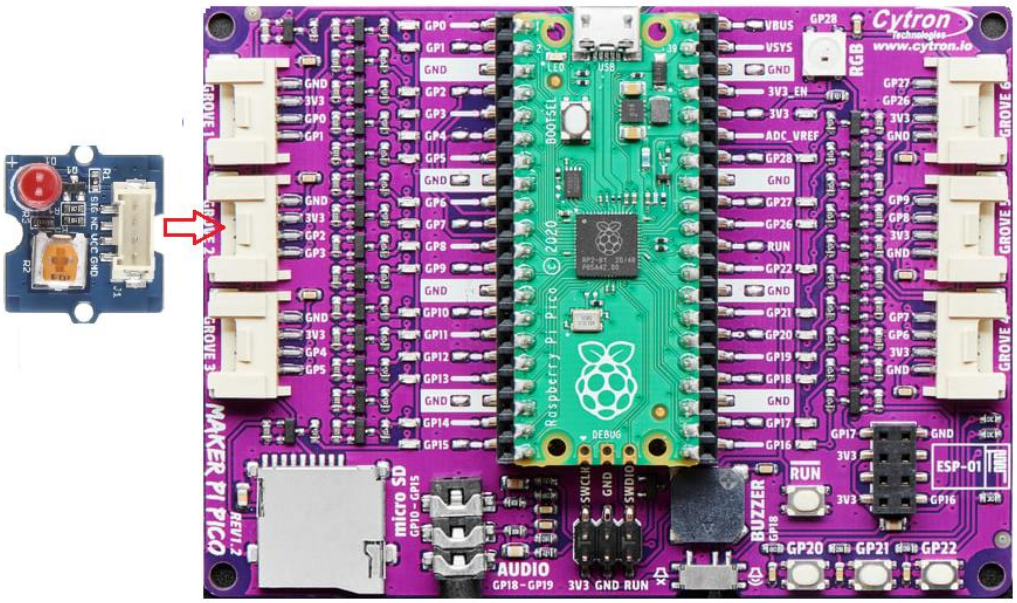
El LED estar encendido 200 ms y estará apagado 400 ms



Entradas salidas:

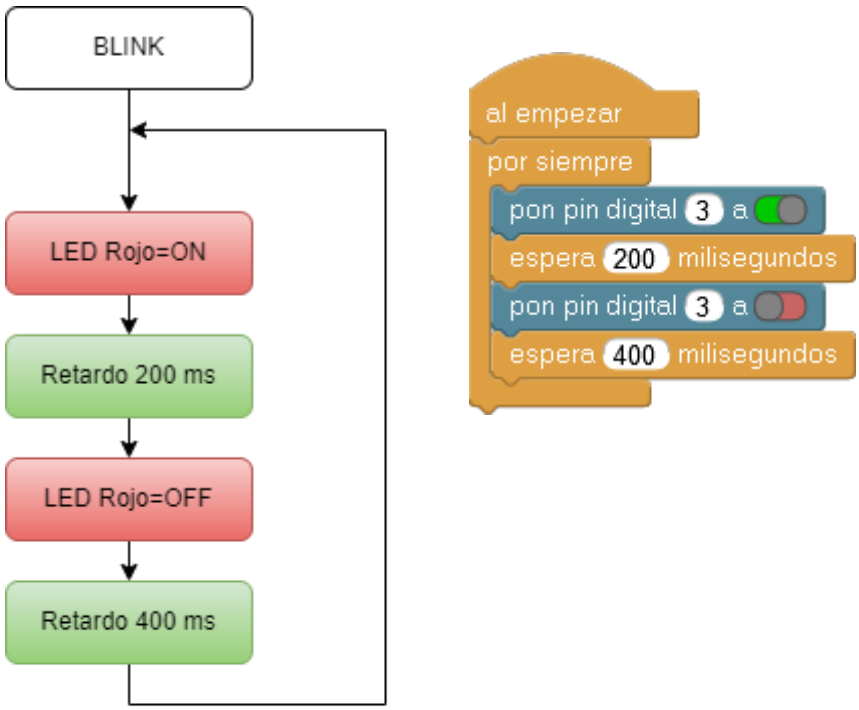
- GP3 Salida digital con LED Rojo

Esquema de montaje:



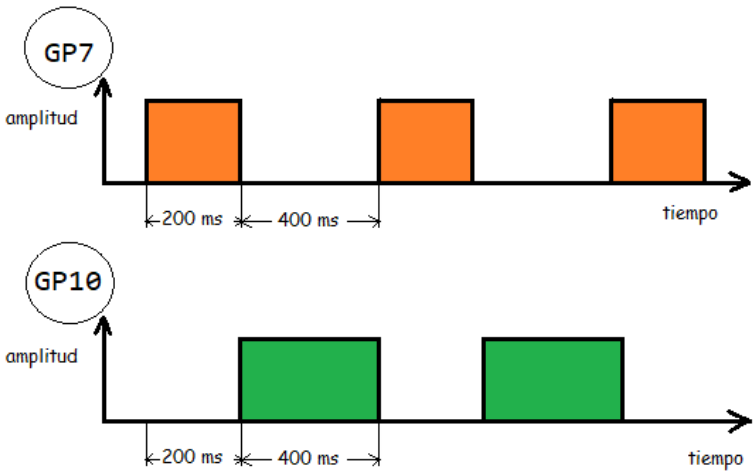
Programa:

En la imagen vemos el organigrama de este algoritmo. Para implementar el algoritmo bastará con colocar dentro del bucle "**por siempre**" el bloque de activación y desactivación de la salida **GP3** con los retardos correspondientes "**espera …milisegundos**"



Actividades de Ampliación

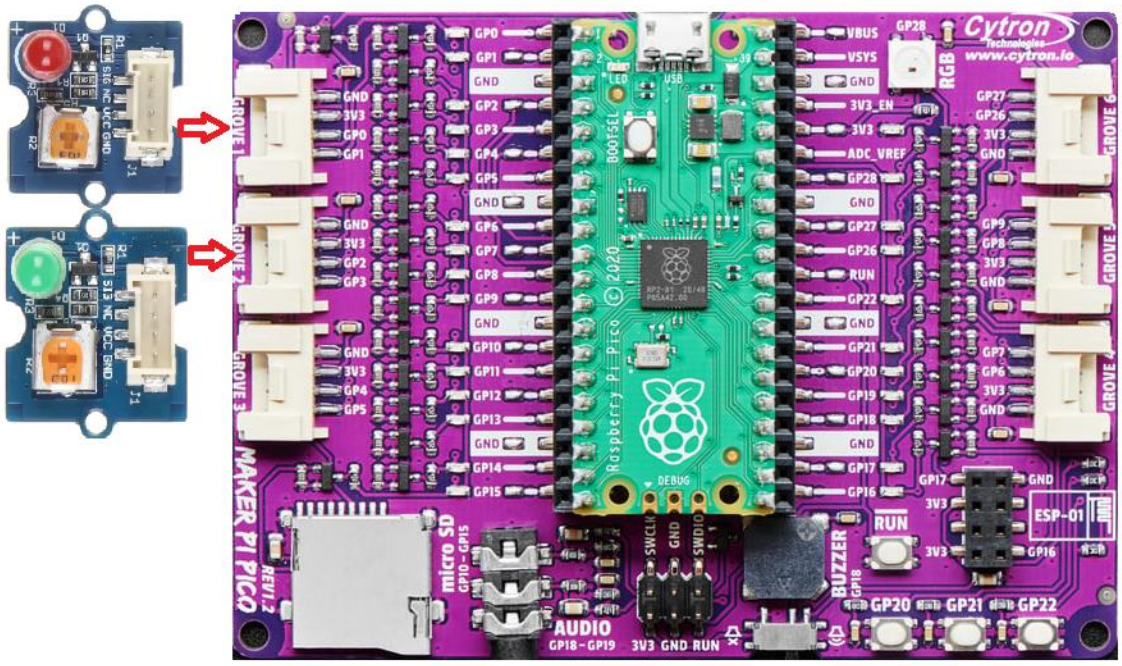
1. DOBLE BLINK: Usando dos salidas GP1 y GP3 queremos que el Blink se adapte a este diagrama de tiempo.



Solución

En este caso se pretende que de manera continua los leds se alternen en el encendido y apagado de acuerdo a la secuencia siguiente:

- GP1 activado GP3 Desactivado - Retardo de 200 ms
- GP1 desactivado Gp3 Activado - Retardo de 200 ms



6.3. Blink con tiempo variable

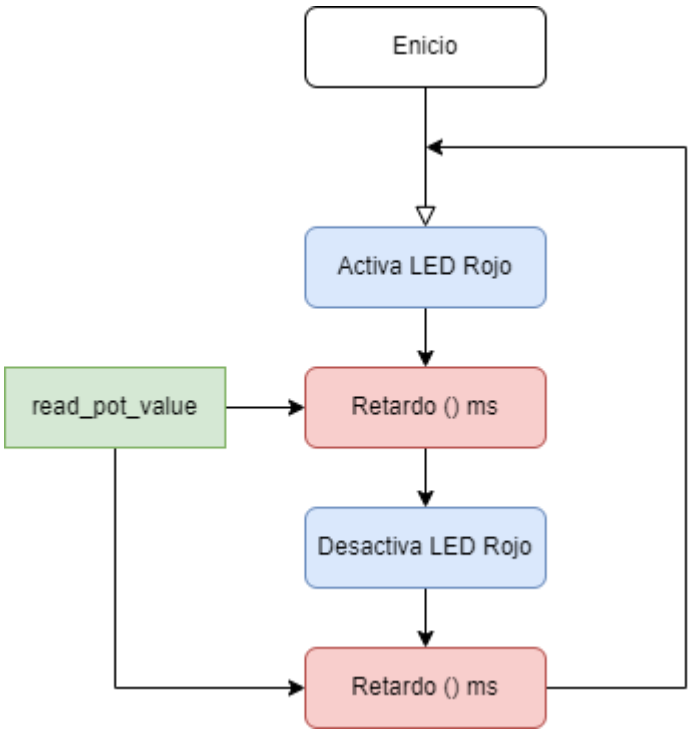
Objetivo

Realizar un intermitente pudiendo modificar el tiempo de encendido/apagado de un LED Rojo mediante un potenciómetro que conectaremos en el pin GP27 que es una entrada analógica de la tarjeta.

Funcionamiento:

Con este ejemplo se trata de realizar el encendido y apagado del **LED Rojo** pero pudiendo modificar el tiempo de entre los estados de encendido y apagado.

En nuestra práctica, por lo tanto, van a intervenir dos dispositivos: El Potenciómetro y el LED Rojo.

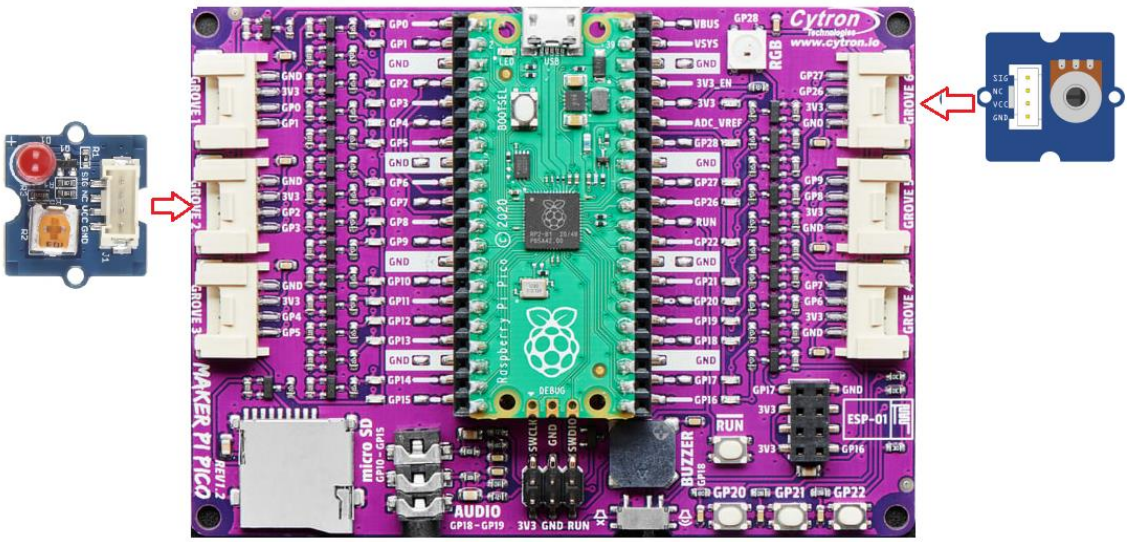


Al variar la lectura del valor del potenciómetro variaran los tiempos de retardo.

Entradas salidas:

- GP27 Potenciómetro
- GP3 LED Rojo

Esquema de montaje:

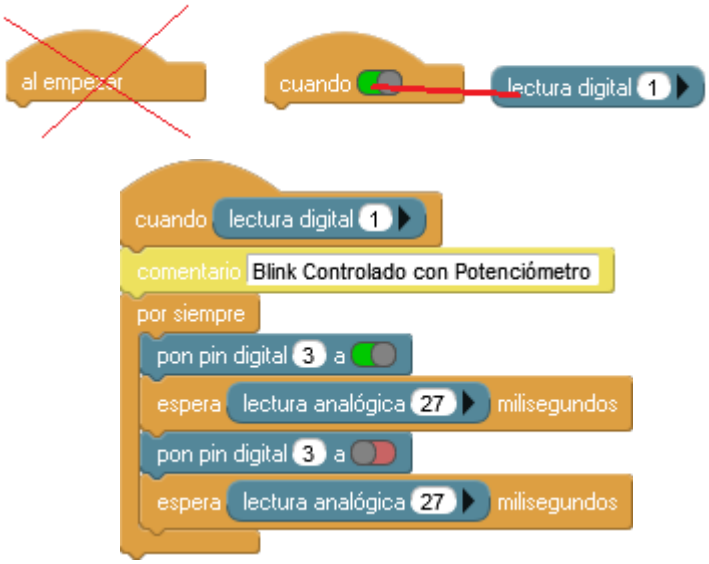


Programa:

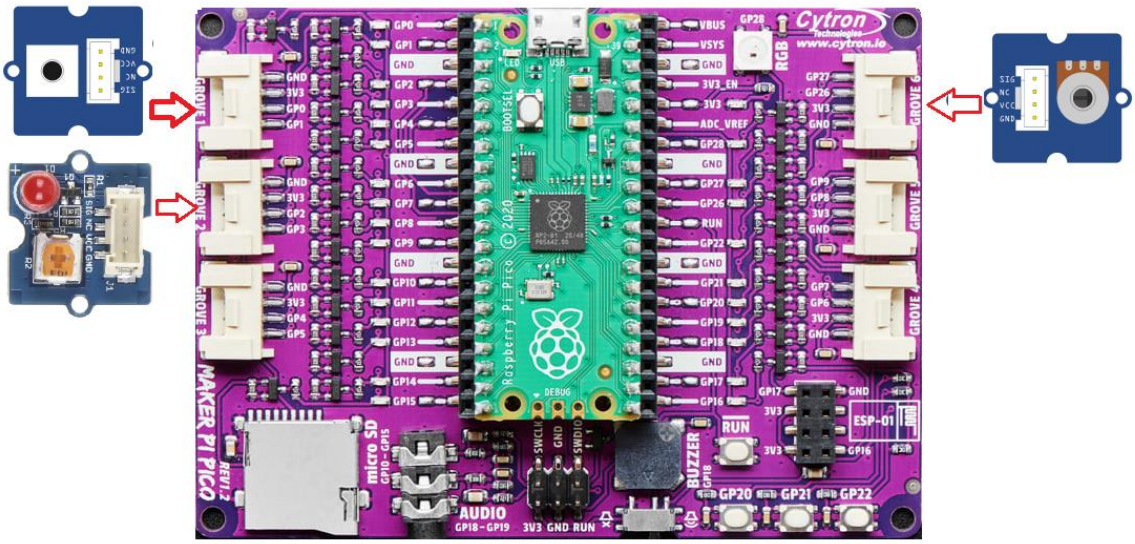
En la imagen anterior vemos el algoritmo en el que, a diferencia del anterior, el tiempo no es una constante sino el valor que recogemos del pin **GP27** "**lectura analógica…**"

Actividades de ampliación

1. Deseamos que la ejecución del algoritmo de intermitencia se lleva a cabo siempre que pulsemos un pulsador conectado al pin **GP1**.
   Esta sería la solución. Solo debemos cambiar el bloque de control "**al empezar**" por bloque "**cuando**" evento


al empezar
cuando
lectura digital 1
cuando
lectura digital 1
comentario
Blink Controlado con Potenciómetro
por siempre
pon pin digital 3 a
espera
lectura analógica 27
milisegundos
pon pin digital 3 a
espera
lectura analógica 27
milisegundos

Montaje


Cytron
MAKER PI PICO
Raspberry Pi Pico

6.4. Creación y uso de una FUNCIÓN

Objetivo

Con esta aplicación vamos experimentar con la creación de un **bloque de función** con la ayuda de la función "**Mis Bloques**" La función la usaremos para hacer una intermitencia en una salida digital (Blink).

Funcionamiento:

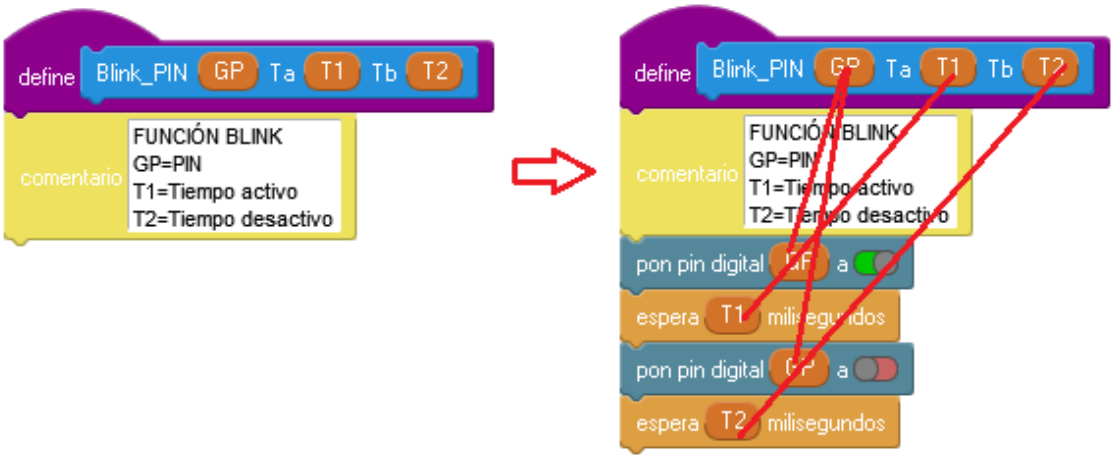
Nuestra función lo que hace es activar y desactivar una salida con unos tiempos, tanto de activación como de desactivación.

Los parámetros de nuestra función serán:

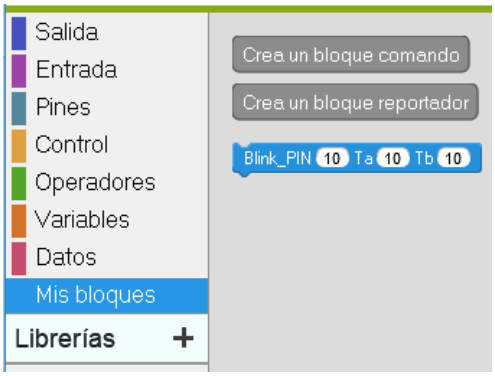
- PIN Numero de pin GP en el que conectaremos el dispositivo de salida (LED)
- Ta Tiempo durante el que permanecerá activa la salida
- Tb Tiempo durante el que permanecerá desactivada la salida

A la hora de construir la función colocaremos estas variables en el comienzo

Seguidamente completamos los bloques de la función y de ese modo queda definida esta función. Los parámetros PIN, Ta y Tb se incrustan en las correspondientes cajas de los bloques de función.

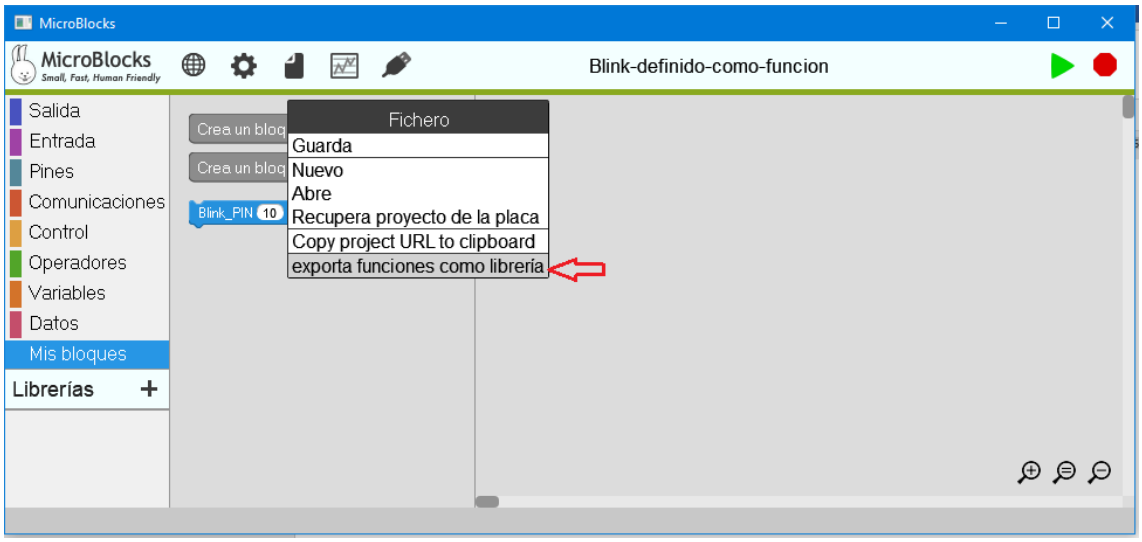


Una vez creado el bloque de función miramos a la zona de librerías y allí encontraremos nuestra función.



Este bloque podríamos convertirlo en una nueva librería que podríamos usar en otras aplicaciones. Para ello bastaría guardarlo en forma de librería.

Una vez creado este nuevo bloque de función procedernos a construir la aplicación del programa que usara esta función.



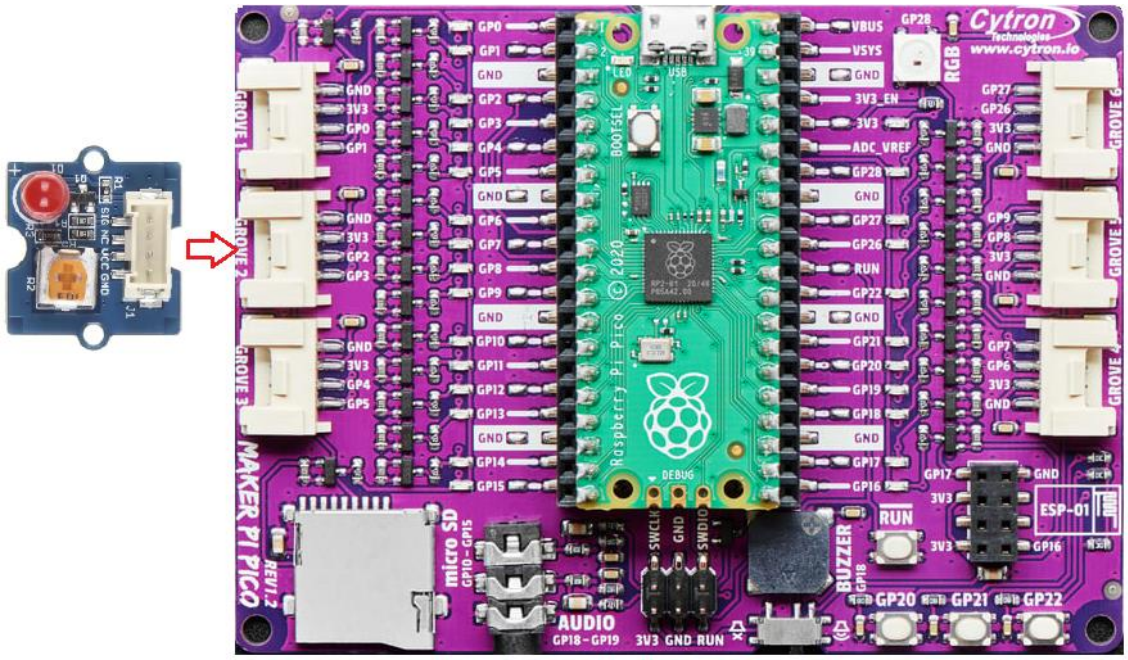
Nuestro programa solo tendrá que invocar a la función que hemos creado y así construiremos una aplicación Blink con el uso de un bloque de función creado por el "usuario"

Como queremos que la activación del Blink se realice cuando pulsemos un pulsador conectado en el pin GP1

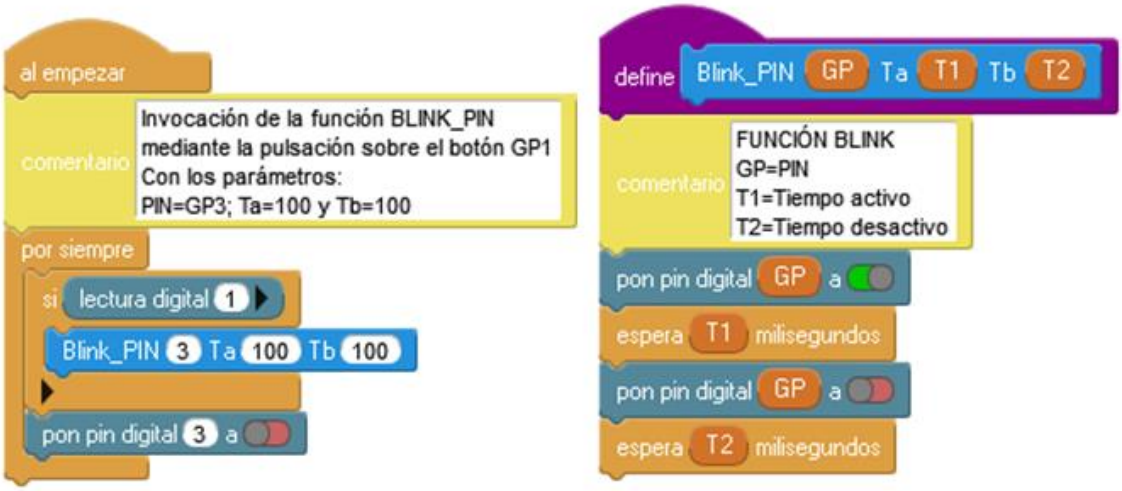
Entradas salidas:

- GP3 Salida de LED

Esquema de montaje:



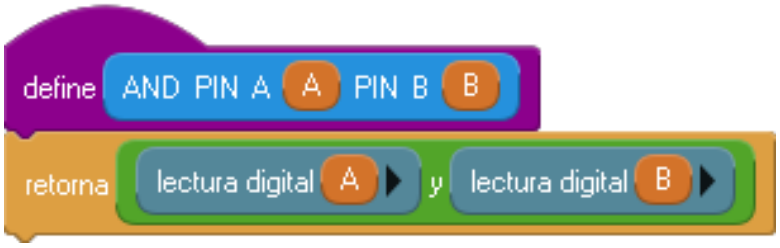
Programa:

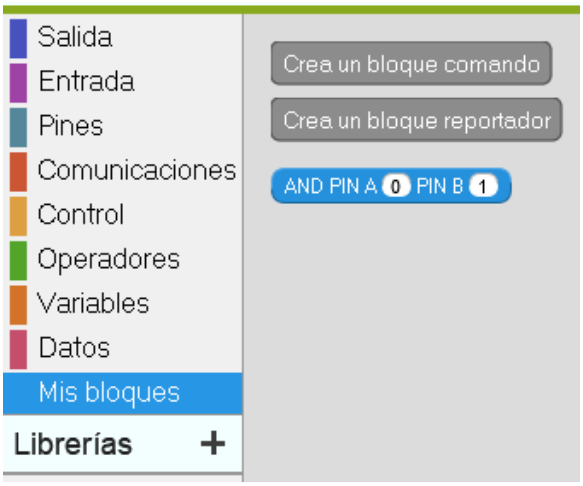


Actividades de ampliación

1. Crea un bloque de usuario que realice la función AND de dos entradas digitales
   Solución

En nuestro ejemplo hemos designado para probarlo los pines 1 y 3 de entrada y el pin 5 como salida





6.5. Semáforo Básico

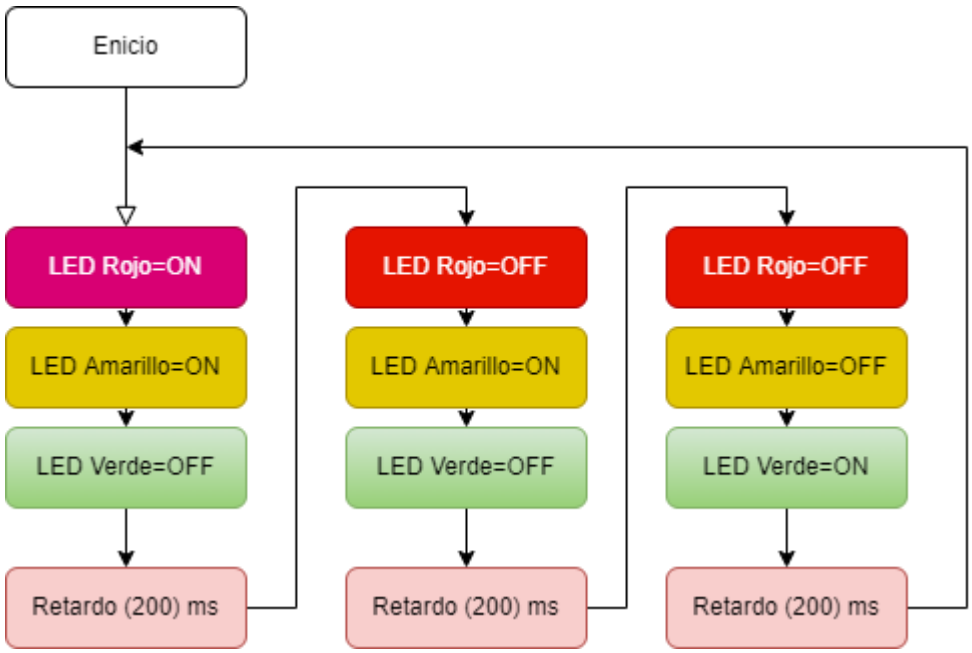
Objetivo

Con esta práctica vamos a trabajar con la unidad "Semáforo" de nuestro kit. Realizaremos programas para controlar esta unidad.

Funcionamiento:

En este caso se trata de que cada uno de los LEDs del semáforo conectados a los Pines GP0, GP1 y GP2 se enciendan y apaguen de manera secuencial de acuerdo a la siguiente tabla.

ESTADO	Rojo	Amarillo	Verde	Tiempo
1	ON	OFF	OFF	200 MS
2	OFF	ON	OFF	200 MS
3	OFF	OFF	ON	200 MS

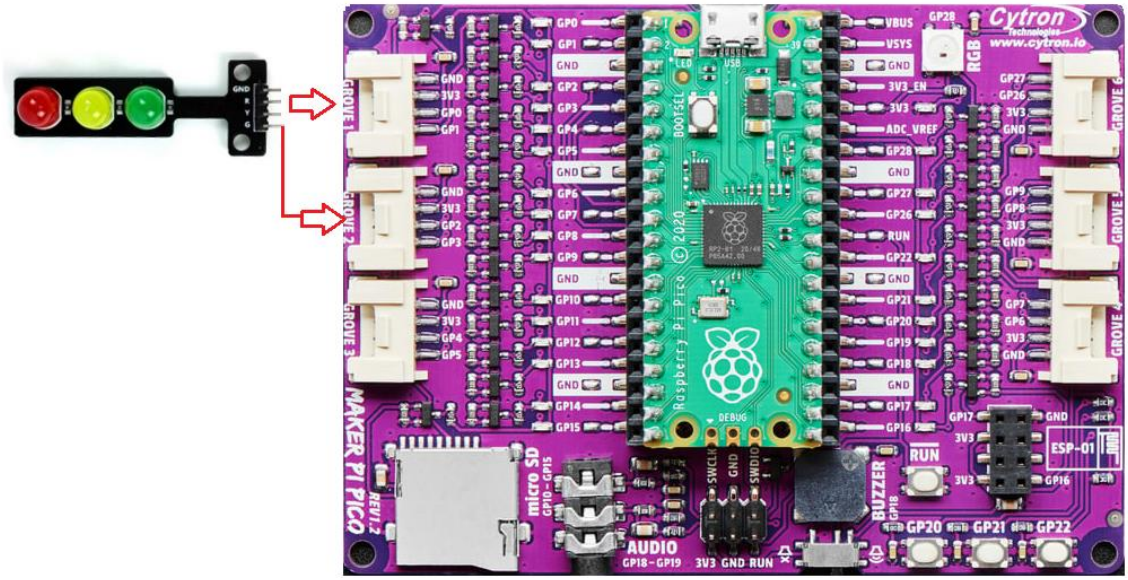


Deseamos visualizar los distintos estados de los LEDs del semáforo por lo que usaremos el bloque de función "**di..**"

Entradas salidas:

- GP0 LED Rojo
- GP1 LED Amarillo
- GP2 LED Verde

Esquema de montaje:



Programa:

Actividades de Ampliación

1. Realizar cambios en los tiempos de encendido de cada LED

2. Realizar una modificación en la que al pulsar un Pulsador P se rompa la secuencia del semáforo y se ponga en estado de Amarillo intermitente. Designar el GP20 como P (se encuentra en la tarjeta).
   SUGERENCIAS: En este caso vamos a usar la opción de crear bloques propios con la pestaña “Mis bloques” podemos hacerlo.

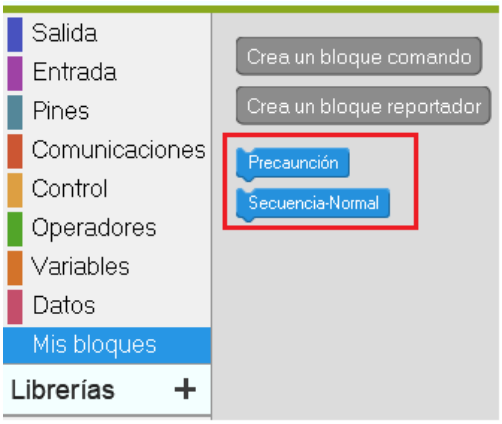
   Crearemos dos bloques: Secuencia-Normal y Precaución.

   - **El Bloque Secuencia -Normal** contiene todas las acciones para seguir la secuencia normal del semáforo que hemos visto en primer lugar.
   - **El Bloque Precaución** contendrá las acciones de encendido y apagado solo del LED Amarillo

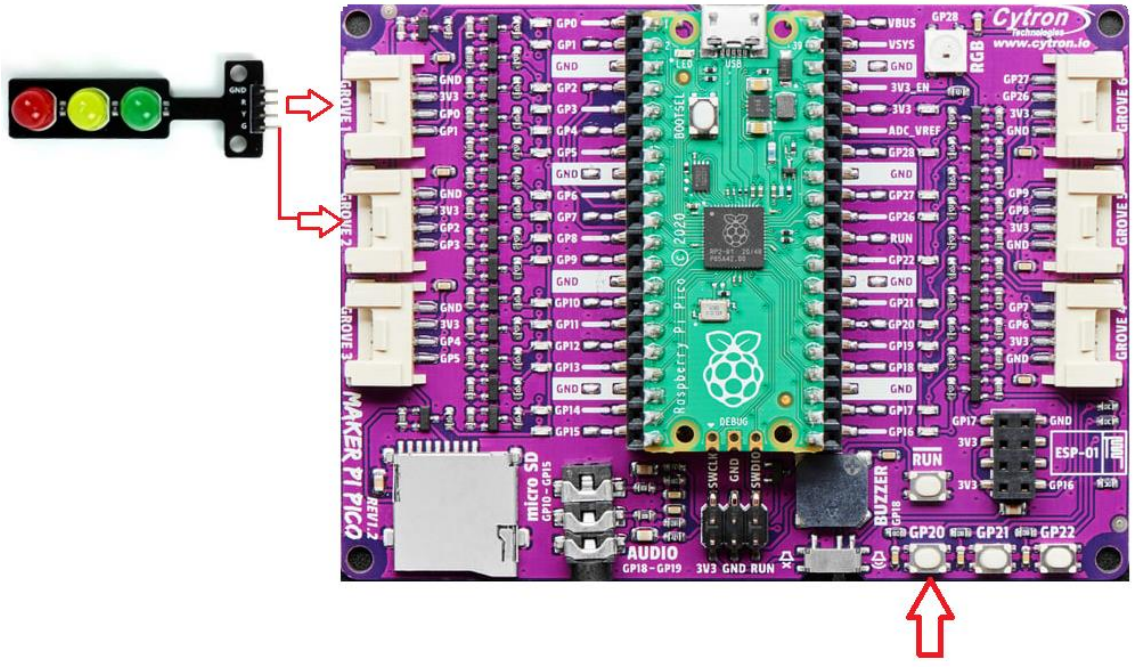
   En el programa principal lo que haremos se colocar un condicional “SI ENTONCES” cuya condición será el estado del Botón conectado al GP20 de la placa con el que gobernamos la ejecución de las dos secuencias vistas

   - Si el pin GP20=1 Se activara la función Secuencia-Normal
   - Si el pin GP20=0 Se activara la función Precaución

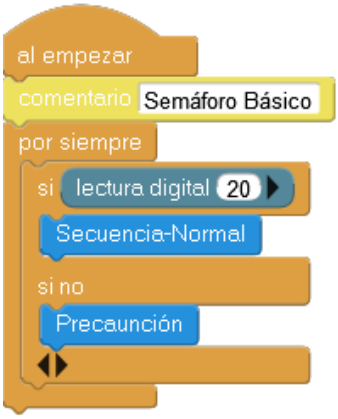
   Téngase en cuenta que en reposo (sin pulsar el botón) el pin GP20 está en valor 1

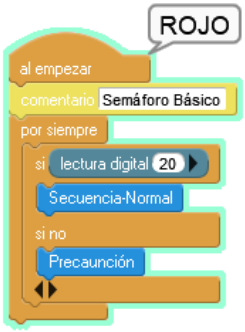


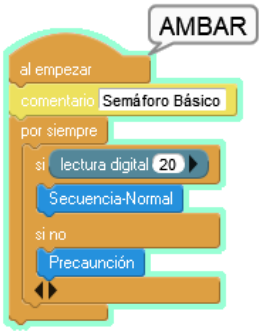
Montaje



Este será el programa







6.6. Monoestables

Objetivo

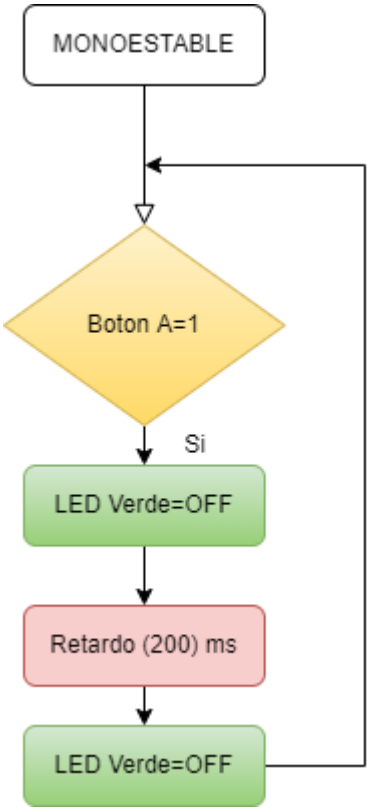
Estudiar el comportamiento de un Monoestable o Temporizador

Funcionamiento:

Una aplicación muy usada en los dispositivos de control es realizar “temporizaciones”. Con este ejemplo vamos a implementar un sistema que realizará las funciones del llamado “monoestable”.

Usaremos el botón de entrada A y la salida con un LED Verde.

Realizaremos la aplicación usando una instrucción de retardo que será exactamente el valor en ms durante el que la salida permanecerá activa.

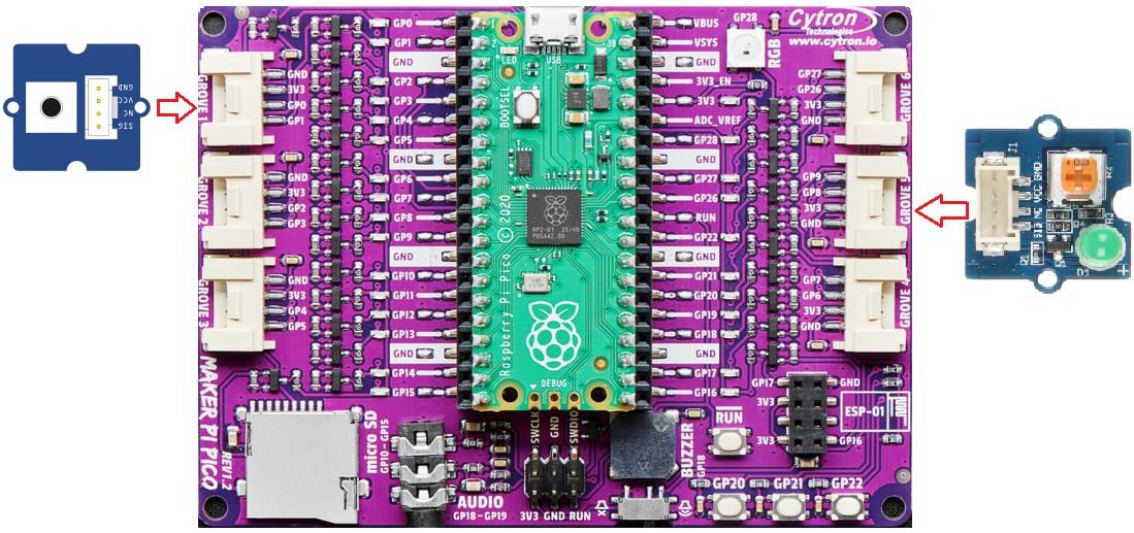


Se utilizará dentro de un bucle “por siempre” una instrucción de tipo condicional que evaluará en todo momento el estado del pin en el que tenemos colocado el botón.

Entradas salidas:

- GP1 Botón de activación. Botón A
- GP7 Led de salida. Led Verde

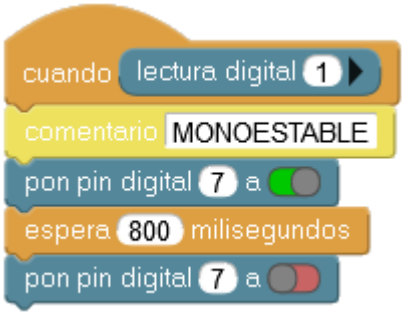
Esquema de montaje:



Programa:

Actividades Propuestas

1. Realizar la aplicación usando el bloque de activación "**cuando..**" de la figura
   Solución



2. Realizar una secuencia de activación en la salida LED de cuatro estados, de acuerdo a la siguiente tabla cuando se pulse el Botón A colocado en el GP1. El tiempo está en ms.

	Estado 1º	Estado 2º	Estado 3º	Estado 4º
T.Activado	4000	3000	1000	500
T.Desactivado	2000	1000	5000	100

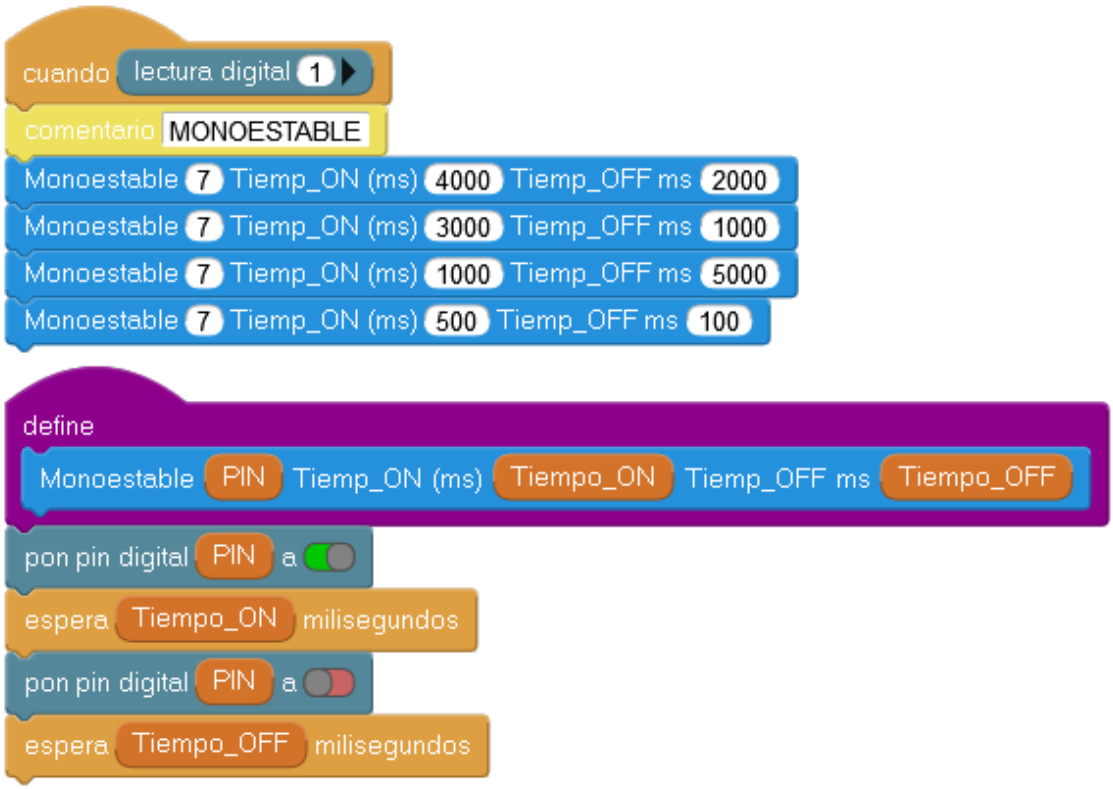
Solución

Crearemos un nuevo bloque de función que implementará la activación de un monoestable. La función tendrá tres parámetros: PIN el número de pin en el que colocaremos el LED de salida y los tiempos ON y OFF. Como es una secuencia lo que hacemos es colocar ambos tiempos Tiempo_ON y Tiempo_OFF.

3. Realizar el monoestable usando una función que llamaremos “Monoestable” y que tendrá los siguientes parámetros: **PIN**, **Tiempo_ON** ms y **Temp_OFF** ms.

   Usaremos el pin **GP1** para disparar una cadena de cuatro monoestables cuya salida será el pin **GP7**

   Solución.



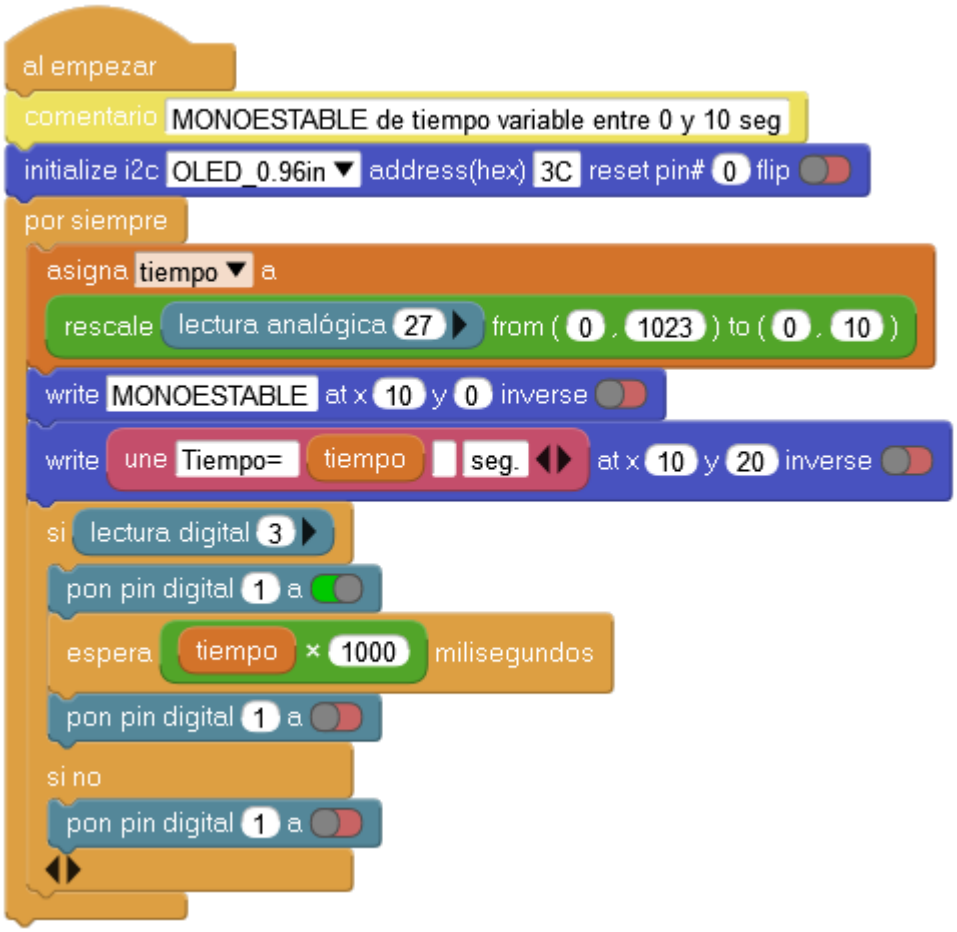
4. Realizar una aplicación en la que el tiempo del monoestable podamos ponerlo variable en un margen entre 0 y 10 seg. Usaremos un display OLED en donde se escriba el valor del tiempo que seleccionaremos mediante un potenciómetro conectado en el GP27

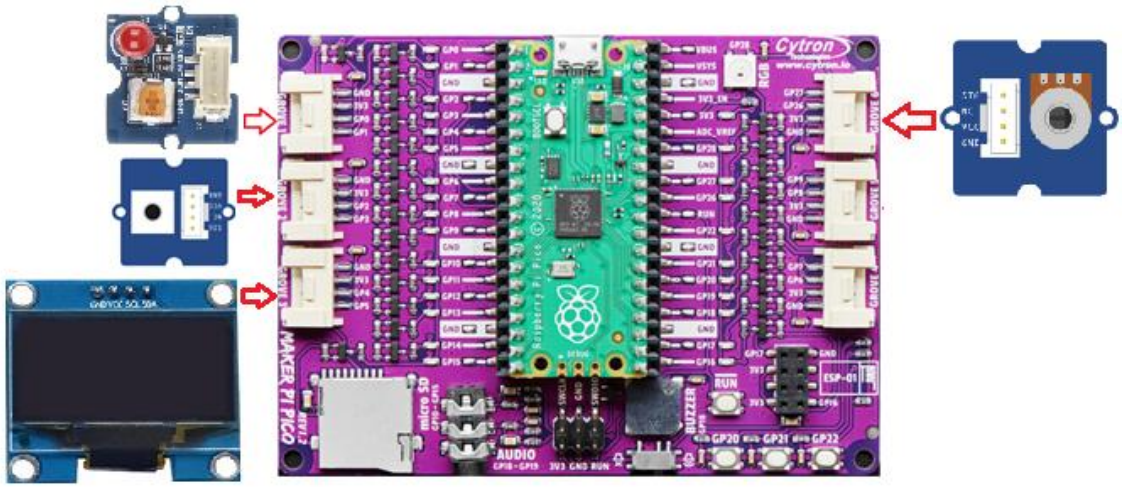
   Solución.

   Definiremos una variable llamada “**Tiempo**” que asociaremos al valor leído del potenciómetro previamente escalado al rango 0 a10.

   El aspecto de la pantalla OLED será el de la siguiente figura.

El botón de disparo del monoestable se conectara en el pin GP3 y el LED de salida del monoestable se colocará en el pin GP1




Cytron
MAKER PI PICO
RUN
BUZZER
AUDIO
micro SD
ESP-01
GROVE

6.7. Biestable

Objetivo

Vamos a realizar la simulación de un Biestable con nuestra tarjeta, usando dos Botones y un LED

Funcionamiento:

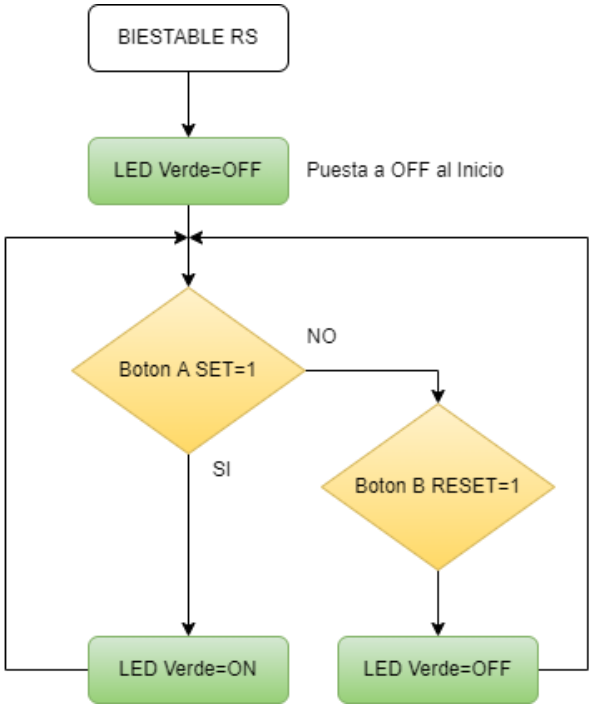
Un biestable sabemos que es un elemento básico de "memoria". Vamos a implementar uno del tipo RS usando los botones A y B y una salida.

- Botón A SET (puesta a "1")
- Botón B RESET (puesta a "0")
- Salida LED Verde

La tabla de verdad de este dispositivo es la siguiente:

Entradas S	R	Q_t	Salida Q_{t+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	x
1	1	1	x

S=SET, R=RESET, Qt=Estado anterior, Qt+1=Salida



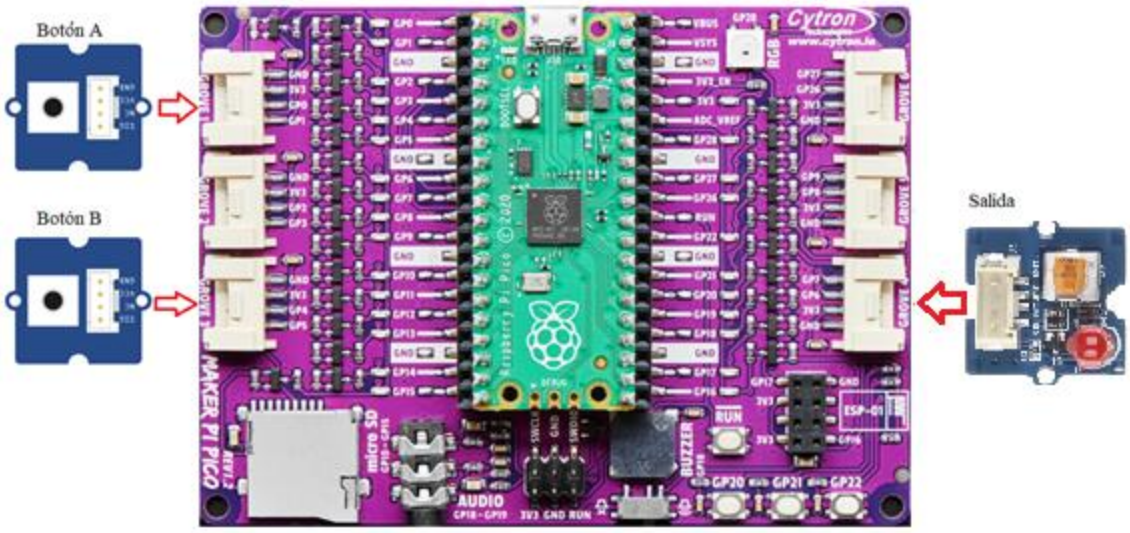
Entradas salidas:

Usaremos los siguientes elementos

Conectaremos los elementos en los siguientes terminales

- Entrada GP1 Pin Set. Botón A
- Entrada GP5 Pin Reset Botón B
- Salida GP7 Pin Salida LED verde

Esquema de montaje:



Programa:

Nuestro programa es muy sencillo, dentro de un bucle "**por siempre**" colocaremos dos bloques condicionales que testean el estado de las entradas **SET** y **RESET** de nuestro biestable que se corresponden, como sabemos, como las entradas digitales **GP3** y **GP5.**

Se han colocado bloques de **"espera.."** con el fin de evitar los efectos de rebote en los pulsadores que pueden falsear la activación.

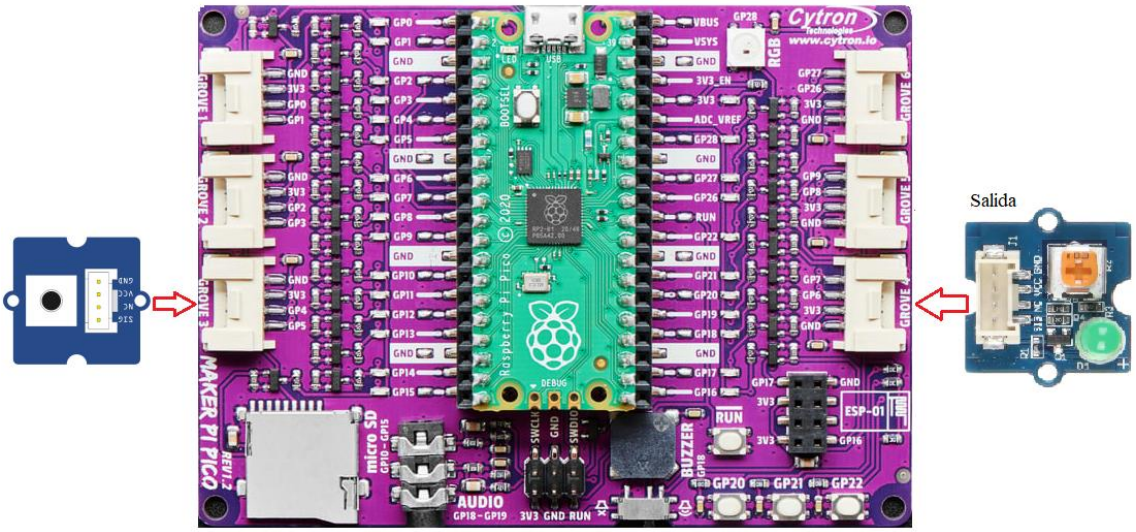
Este es el modelo más básico para configurar un biestable tipo RS

Actividades Propuestas

1. Realiza el mismo ejercicio, pero usando un solo Botón de activación del biestable. De tal forma que, si pulsamos el Botón, estando apagado el LED (salida "0"), se activará el LED (salida "1") y viceversa, si pulsamos el Botón y este encendido el LED (salida "1") se desactivará el LED (salida "0"). El botón se colocará en el

GP5 y la salida LED en GP7. Se propone crear una variable que podemos nombrar como "**Estado**" y que almacenará el estado anterior del Biestable.

Montaje



2. Realizar el mismo programa anterior pero esta vez queremos que se utilice la pantalla OLED para mostrar el estado del biestable. Usaremos

   - GP1 Para la salida LED
   - GP3 Para el Botón
   - GP4 Para SDA del display OLED
   - GP5 Para SCL del display OLED

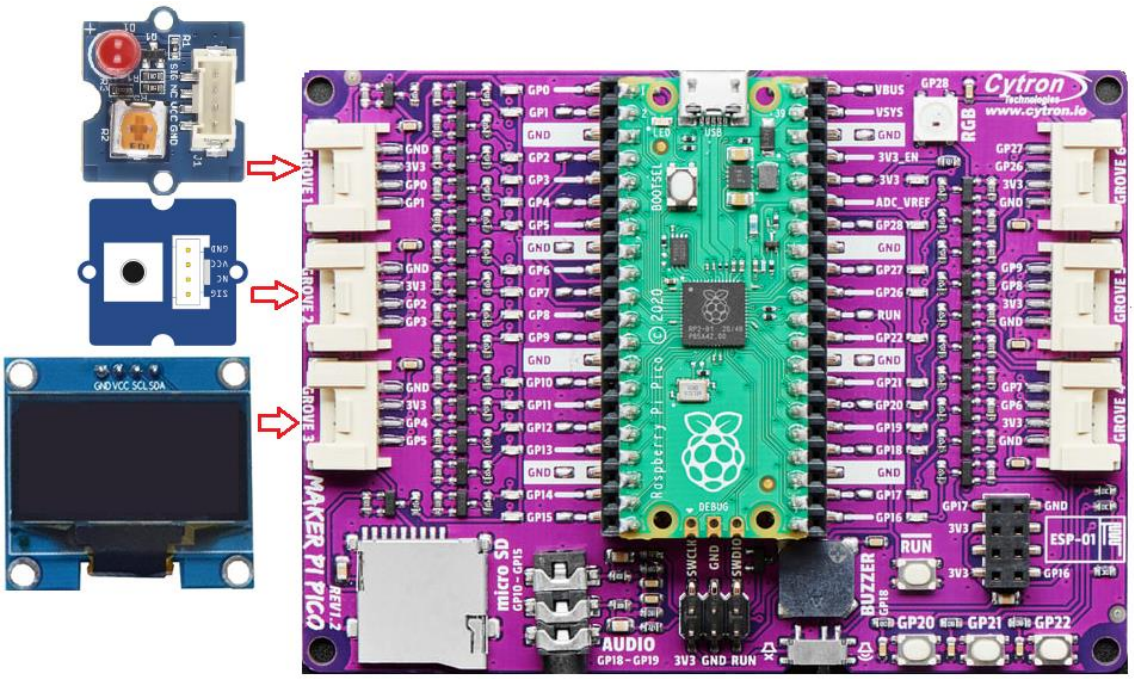
   La pantalla mostrara los textos que se indican a continuación y si la salida se activa se mostrara un rectángulo blanco

   - Si la salida esta apagada: Mostraremos en la pantalla APAGADO
   - Si la salida esta encendida: Mostraremos en la pantalla ENCENDIDO

El programa es como el anterior, pero se ha incorporado la inicialización del dispositivo OLED y los correspondientes mensajes para cada uno de los estados del biestable

```
al empezar
initialize i2c OLED_0.96in address(hex) 3C reset pin# 0 flip
write ESTADO at x 0 y 0 inverse
write APAGADO at x 55 y 20 inverse
asigna estado a 0
por siempre
  si lectura digital 3
    si estado = 0
      pon pin digital 1 a
      fill rectangle x 20 y 20 w 30 h 20 erase
      write ENCENDIDO at x 55 y 20 inverse
      asigna estado a 1
      espera 200 milisegundos
    si no
      pon pin digital 1 a
      fill rectangle x 20 y 20 w 30 h 20 erase
      write APAGADO at x 55 y 20 inverse
      asigna estado a 0
      espera 200 milisegundos
```

Montaje



6.8. Contador

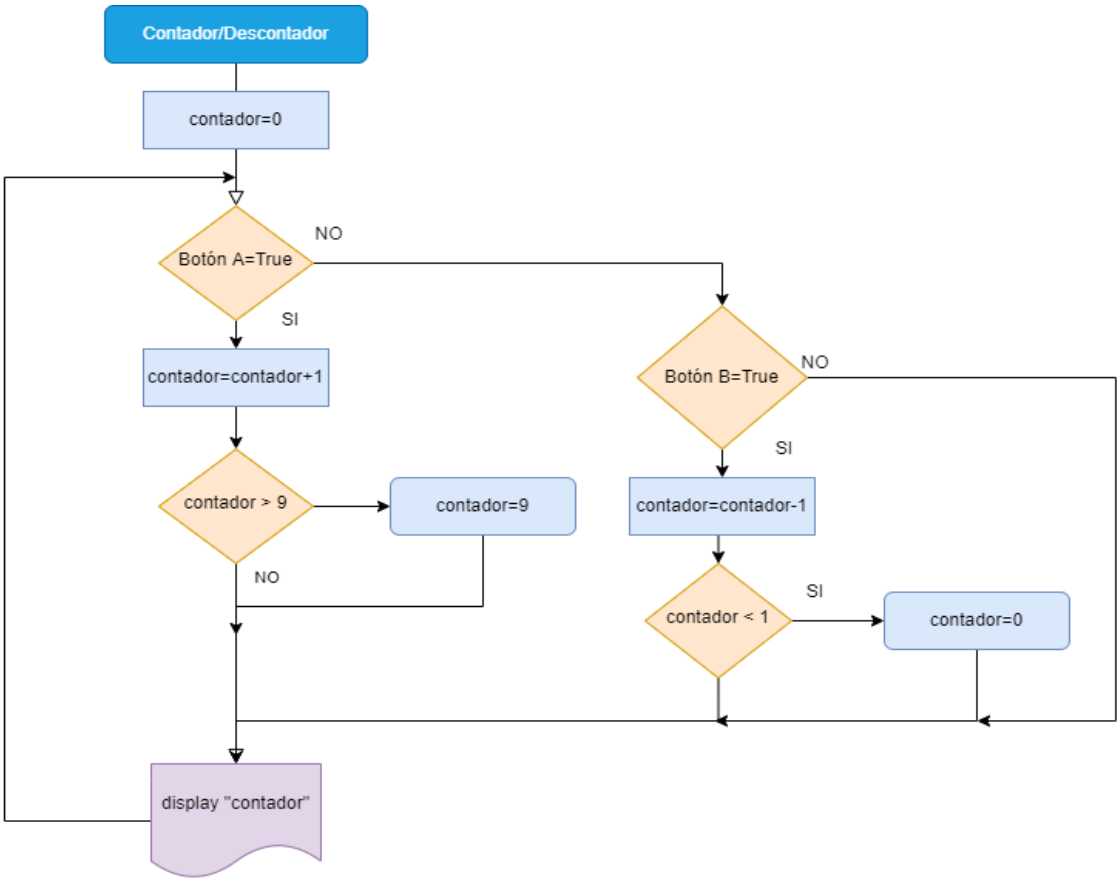
Objetivo

En esta práctica te proponemos que experimentes con un contador. Usaremos dos Botones para crear los pulsos de cuenta, ascendente y descendente, y mostraremos el valor de la cuenta en el display OLED.

Funcionamiento:

En esta práctica vamos a crear un contador de acuerdo a las siguientes condiciones:

- Con el Botón A haremos que el valor de la cuenta aumente: Contador+1
- Con el Botón B haremos que el valor de la cuenta disminuya: Contador-1
- El contador podrá contar hasta un máximo de 9 y no podrá contar números negativos
- La salida del contador se realizará en el display OLED



Entradas salidas:

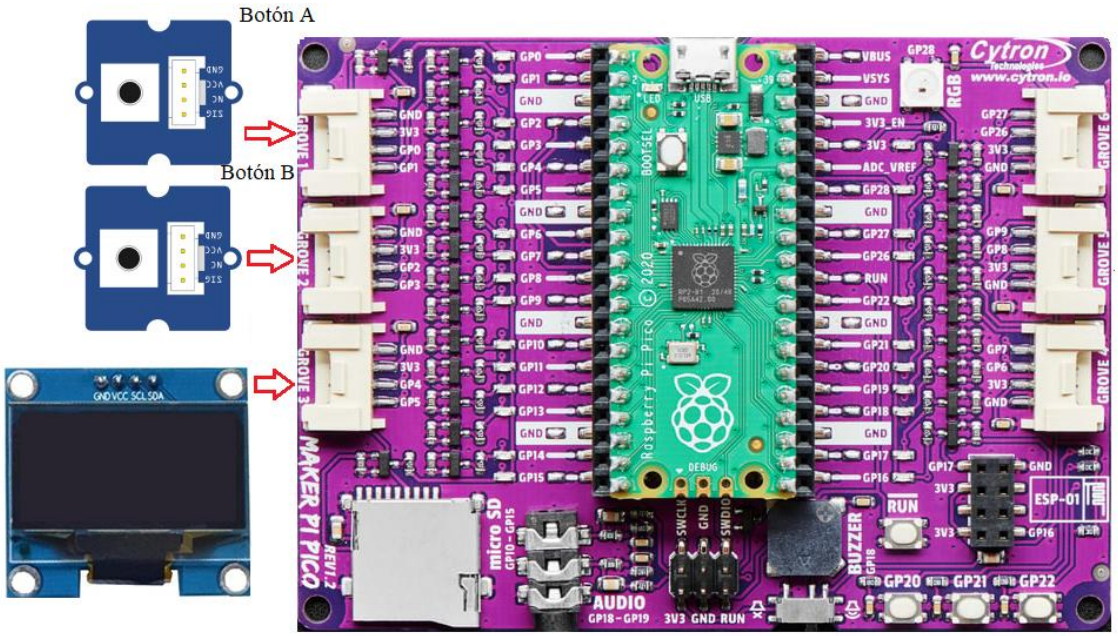
Usaemos dos dispositivos de tipo Botón

Botón A GP1

Botón B GP3

OLED Display 0.96
GP4 SDA y GP5 SCL

Esquema de montaje:



Programa:

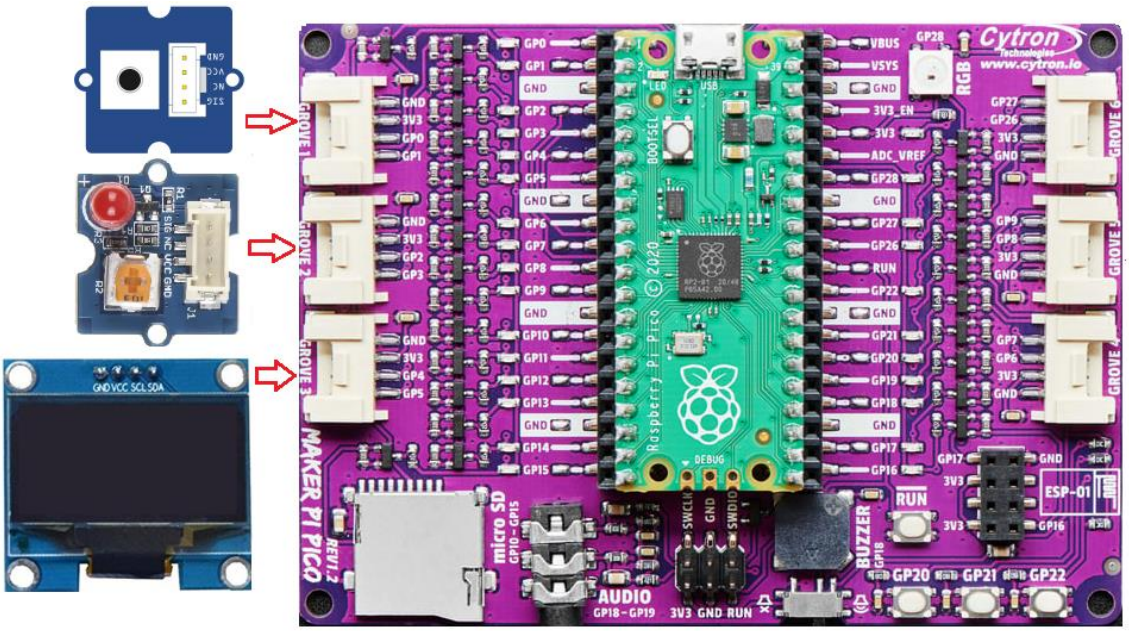
- Se trata de crear una variable que llamaremos "**contador**"
- Dentro del bucle del programa estableceremos dos condicionales principales que testearan la acción de pulsar cualquiera de los botones A y B.
- Dentro del condicional del botón A que es el que incrementa el valor del contador pondremos un condicional que vigilará el valor de la variable. Cuando contador sea mayor que 9 el valor se recoloca en 9 (contador no avanza)
- Dentro del condicional del botón B que es el que decrementa el valor del contador pondremos un condicional que vigilará el valor de la variable. Cuando contador sea menor que 1 el valor se recoloca en 0 (contador no avanza)

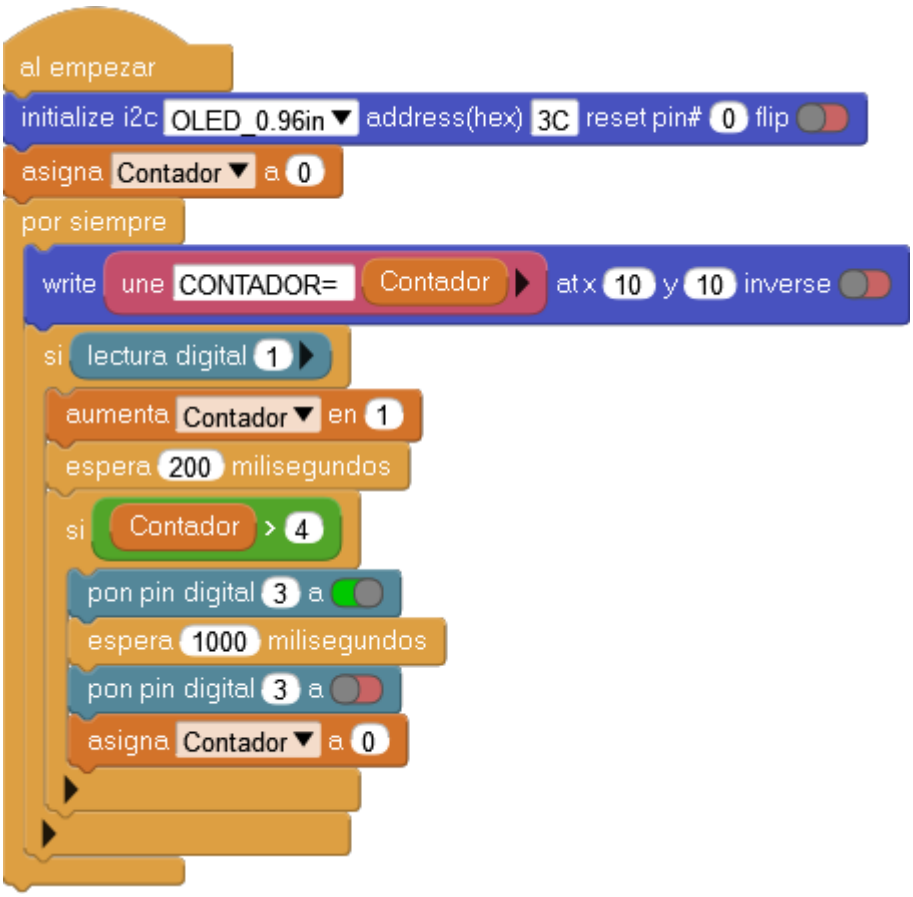
El display OLED quedará tal como se indica en la siguiente figura

Actividades de Ampliación

1. Crear un contador que cuente hasta 5 y que cada vez que llegue a este valor mande un impulso encendiendo un LED Rojo durante 1 segundo. Colocar el LED en GP3, el Pulsador en GP1 y el OLED en GP4 SDA y GP5 SCL

Solución




al empezar
initialize i2c OLED_0.96in address(hex) 3C reset pin# 0 flip
asigna Contador a 0
por siempre
write une CONTADOR= Contador at x 10 y 10 inverse
si lectura digital 1
aumenta Contador en 1
espera 200 milisegundos
si Contador > 4
pon pin digital 3 a
espera 1000 milisegundos
pon pin digital 3 a
asigna Contador a 0

6.9. Comparador básico

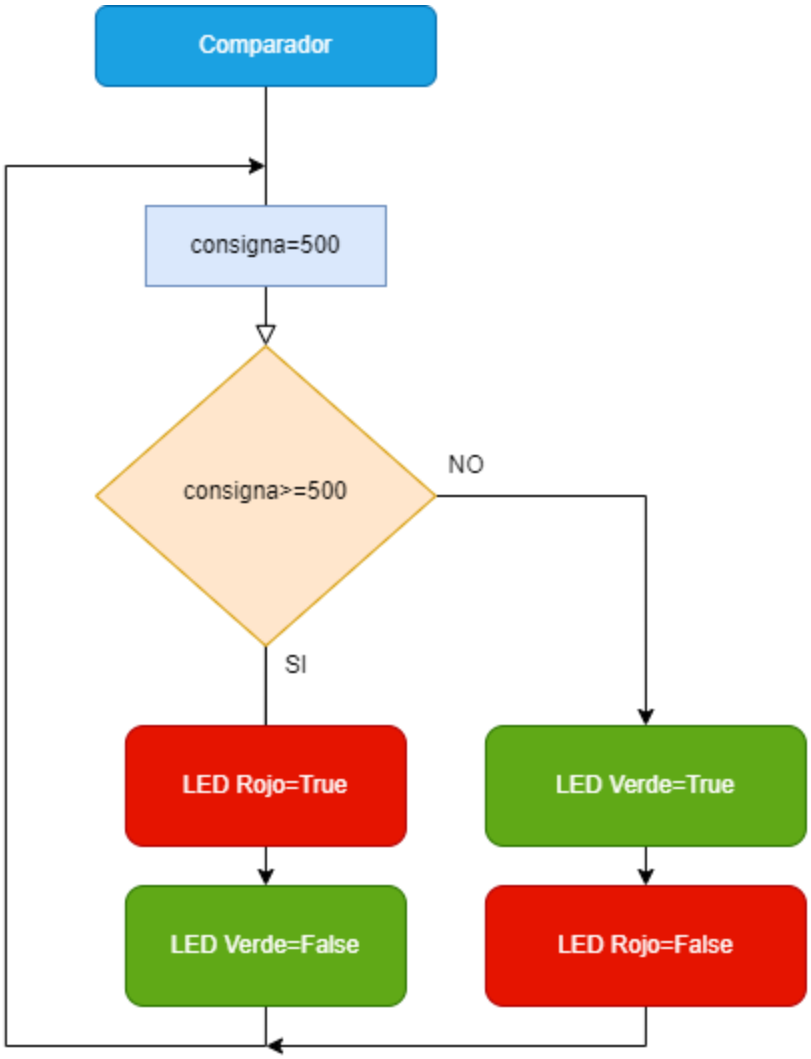
Objetivo

En esta práctica vamos a procesar la señal analógica de entrada del módulo Potenciómetro realizando la comparación de su valor con un valor constante "**consigna**" y en función del resultado de la comparación activaremos el **LED Rojo** o el **LED Verde**.

Funcionamiento: Condiciones de trabajo

- Valor de **consigna=500**
- Si el valor del **Potenciómetro >= consigna LED Rojo=1** y **LED Verde=0**
- En caso contrario **LED Rojo=0 y LED Verde=1**

En las imágenes anteriores vemos el organigrama que explica



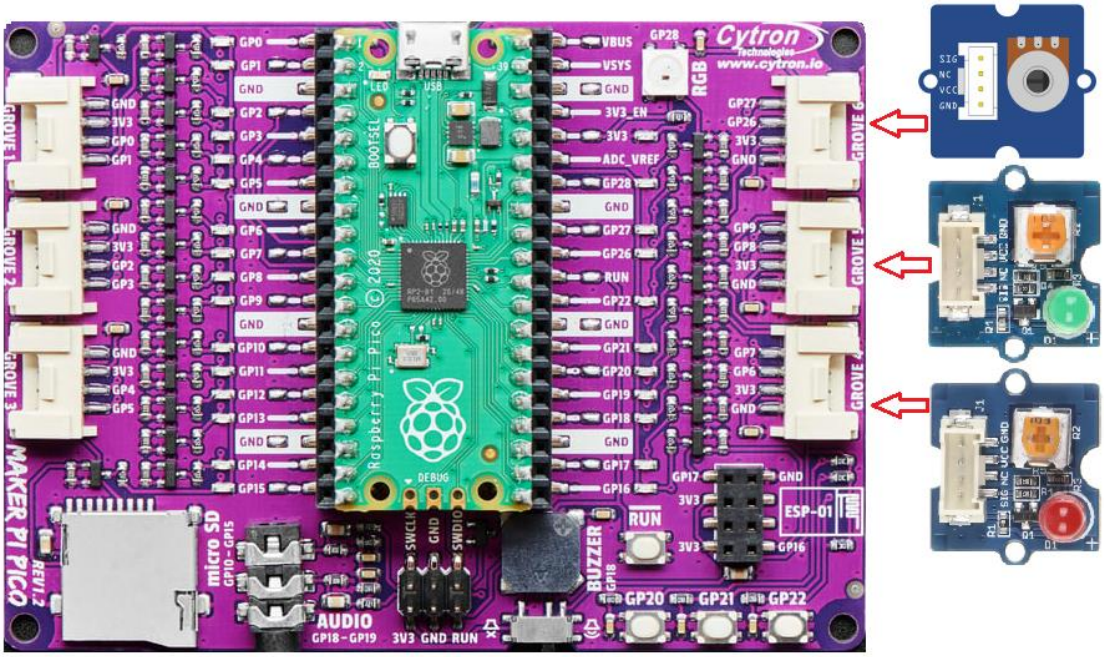
Entradas salidas:

- GP7 LED Rojo

- GP9 LED Verde
- GP27 Potenciómetro “Consigna”



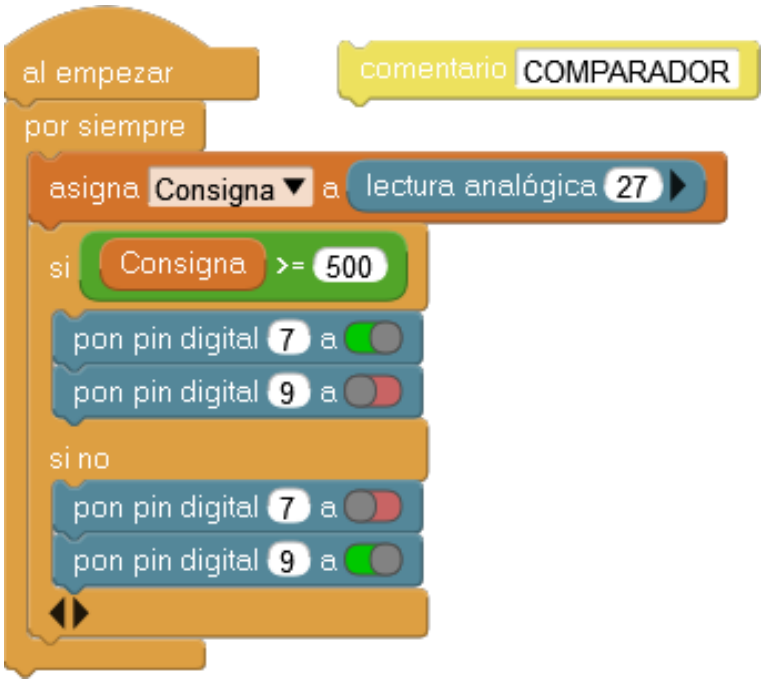
Esquema de montaje:



Programa:

Para realizar el programa debemos definir una variable que llamaremos “**consigna**” y que asociaremos a la lectura del canal analógico **GP27** en el que colocaremos un potenciómetro.

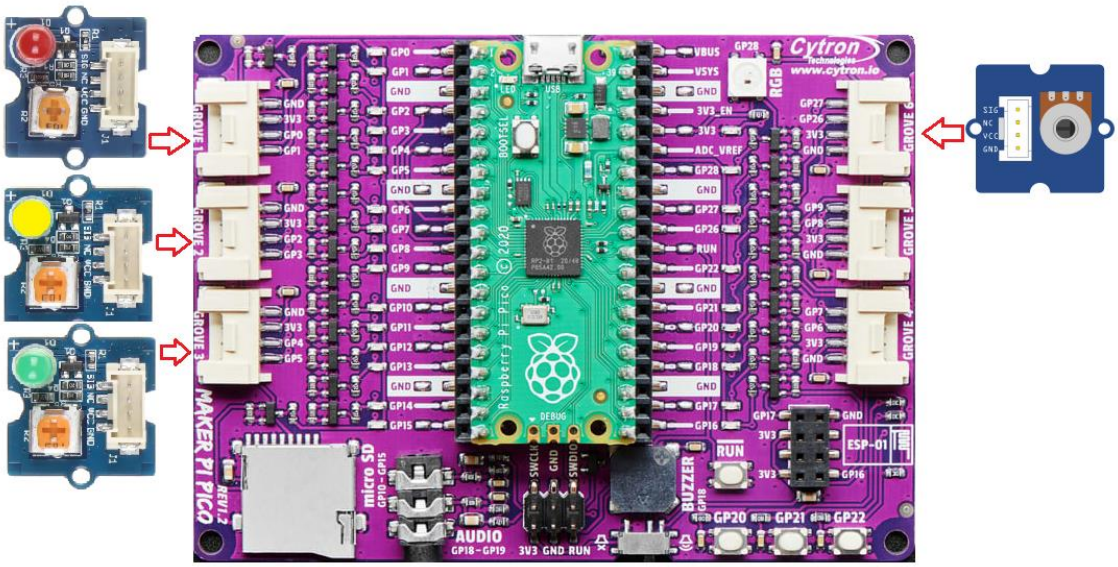
Mediante un condicional “**si si no**” testeamos si el valor de **consigna >=500** y si se cumple, de acuerdo con lo prescrito en nuestro programa activamos el LED Rojo y desactivamos el LED Verde. En caso contrario desactivamos el LED Rojo y Activamos el LED Verde.



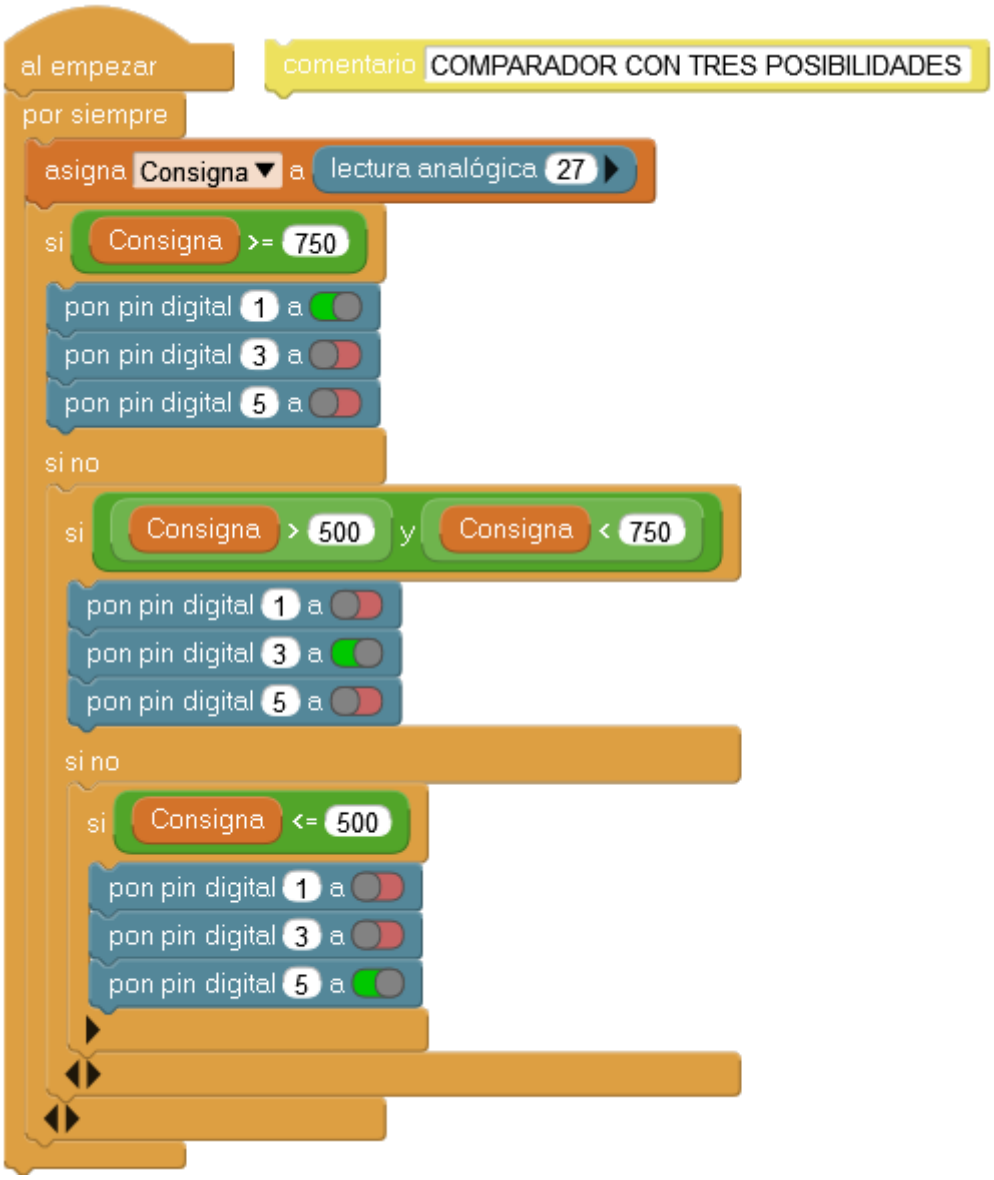
Actividades de Ampliación

1. Realiza los cambios necesarios, para establecer un nivel intermedio de visualización a través de un nuevo LED Amarillo. De acuerdo a la siguiente tabla:

Condición	**LED Rojo**	**LED Amarillo**	**LED Verde**
consigna>=750	Encendido	Apagado	Apagado
consigna < 750 Y consigna >500	Apagado	Encendido	Apagado
consigna <=500	Apagado	Apagado	Encendido



LED Rojo **GP1**; LED Amarillo **GP3**; LED Verde **GP5**; Potenciómetro **GP27**



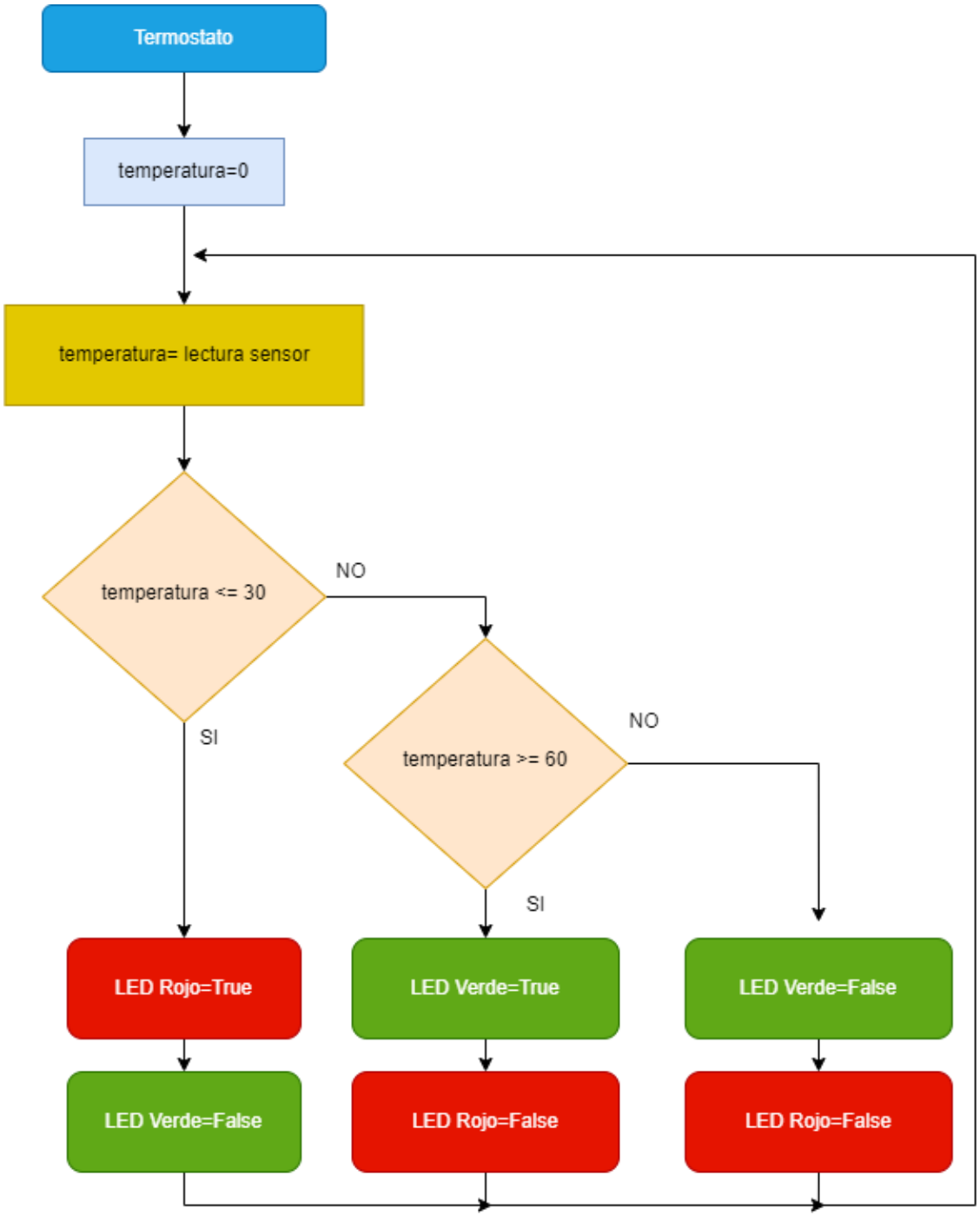
6.10. Termostato

Objetivo

Con la ayuda de un sensor de temperatura vamos a crear una aplicación en la que simularemos un sencillo termostato. Nuestra entrada será del sensor mencionado y colocaremos dos LEDs uno Verde y uno Rojo para monitorizar el estado del sistema.

Funcionamiento:

Con este ejemplo vamos a implementar la simulación de un sistema de control de un equipo de refrigeración/calefacción de tal manera que simulando la variación de la



temperatura con la unidad Potenciómetro podamos establecer el control de estos equipos. La salida que activa la refrigeración es un **LED Verde** que conectamos en el **GP1** de la tarjeta y un **LED Rojo** para la calefacción en el **GP3**

La señal que genera el potenciómetro la asociaremos con la variable "**temperatura**" previamente la mapearemos en el margen de 0 a 100 con la ayuda de la función

Las condiciones que se establecerán son las siguientes:

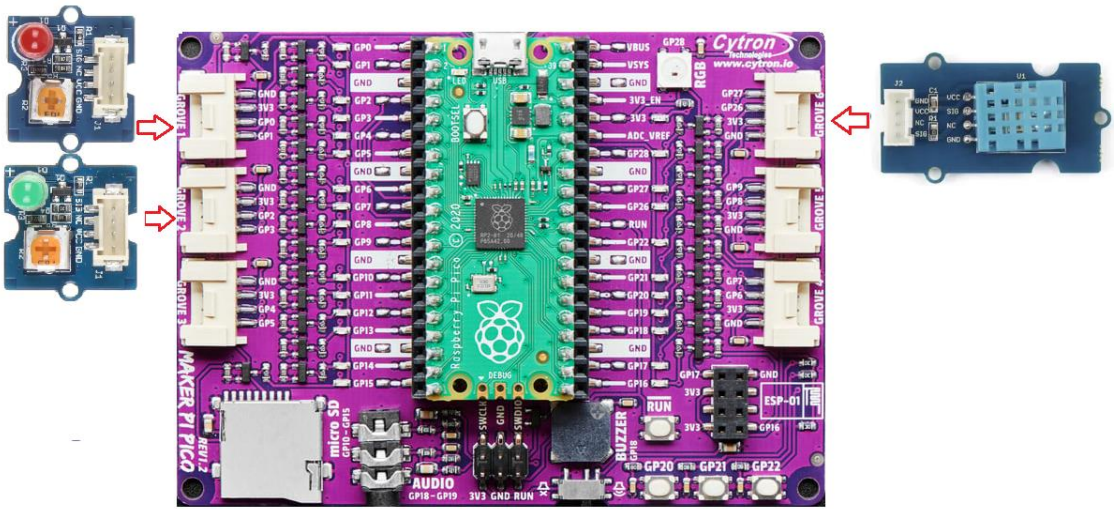
- Si **temperatura <= 30** Se activará la calefacción LED Rojo=TRUE
- Si la **30 < temperatura < 60** Permanecen desactivados refrigeración y calefacción RED Rojo=FALSE y LED Verde=FALSE
- Si la **temperatura >= 60** Activar la refrigeración LED Verde=TRUE

Entradas salidas:

- GP27 Sensor de temperatura
- GP1 Led Rojo
- GP3 Led Verde

Esquema de montaje:

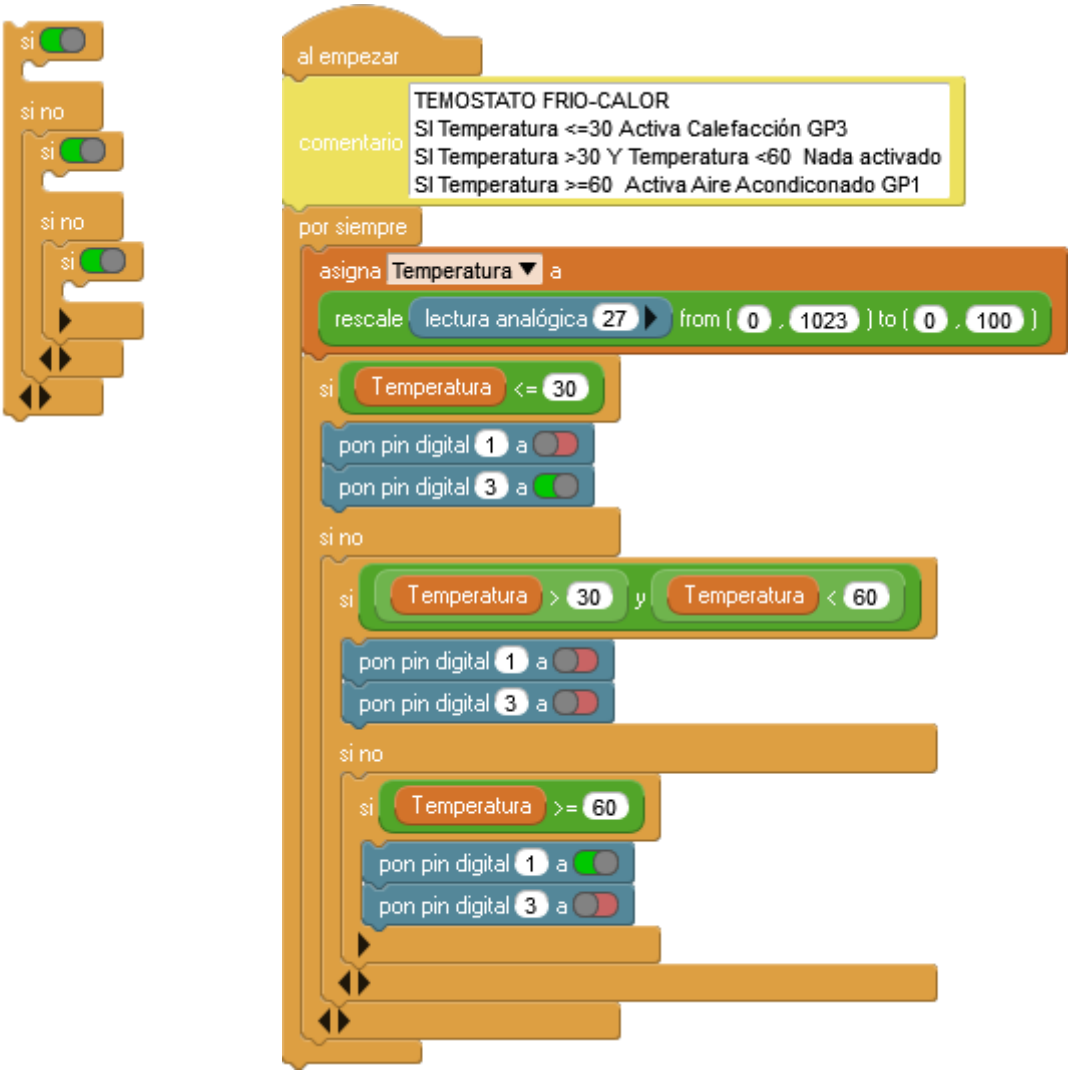
Para las pruebas se puede sustituir el sensor de temperatura por un potenciómetro



Programa:

Nuestro programa se centrará en el establecimiento de una serie de bloques de comparación del tipo "**si**" cuyas condiciones de disparo serán las que se establecen en la descripción del funcionamiento que hemos comentado anteriormente.

Lo primero será definir la variable "**Temperatura**" y su escalado al valor de 0 a 100 que será la primera instrucción del bloque "**por siempre**". Seguidamente se colocarán los condicionales en forma de "árbol", es decir colocando dentro de la parte "**si no**" del condicional el siguiente condicional, hasta cubrir las condiciones. Cada condicional, dentro de él, contendrá las ordenes de activación y desactivación de los pines correspondientes en los que hemos conectado los LEDs.



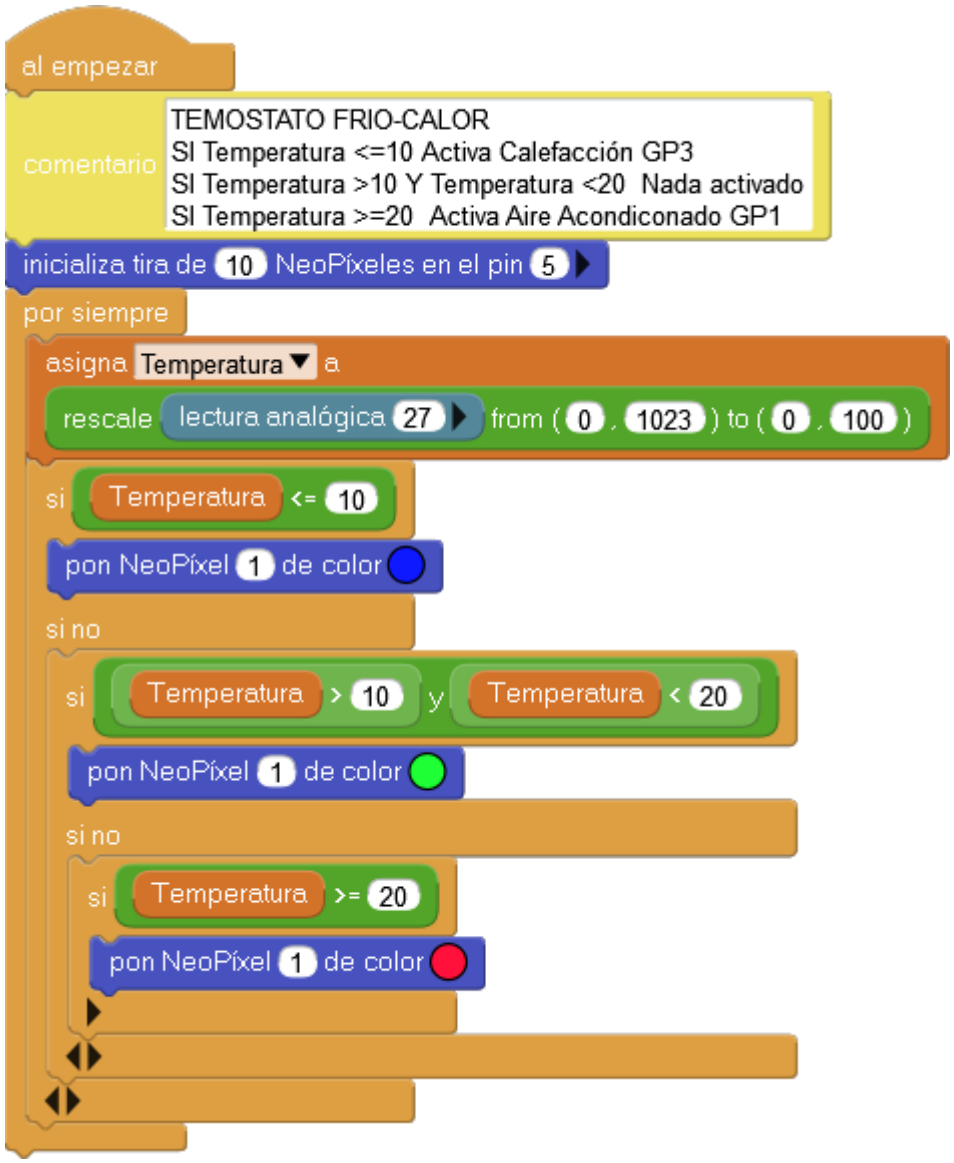
Actividades de ampliación:

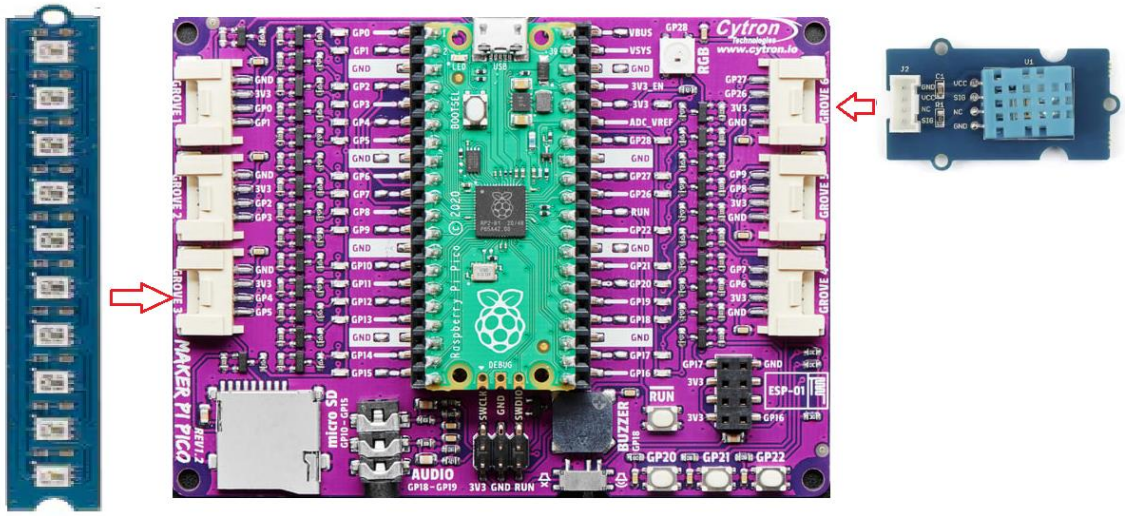
1. Crear la misma aplicación que acabamos de explicar, pero añadiendo una señalización haciendo uso de un módulo Neopixel RGB
   - LED Azul Temperatura <10
   - LED Verde 10<Temperatura<20
   - LED Rojo Temperatura>20

Barra LED RGB

Las condiciones que se establecerán son las siguientes:

- Si **temperatura <= 10** Se activara la calefacción **LED Rojo=TRUE**
- Si la **10 < temperatura < 20** Permanecen desactivados refrigeración y calefacción **RED Rojo=FALSE** y **LED Verde=FALSE**
- Si la **temperatura >= 20** Activar la refrigeración **LED Verde=TRUE**




GROVE 1
GROVE 2
GROVE 3
GROVE 4
GROVE 5
GROVE 6
MAKER PI PICO
REV1.2
Cytron
www.cytron.io
Raspberry Pi Pico
© 2020
BOOTSEL
DEBUG
RUN
RGB
BUZZER
AUDIO
micro SD
ESP-01
GP20
GP21
GP22

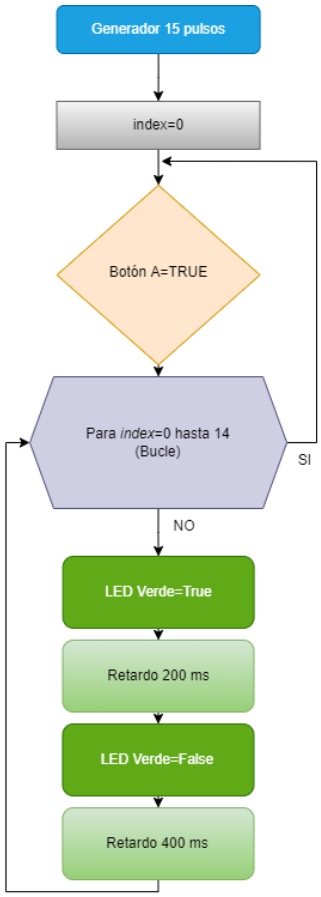
6.11. Generador de impulsos en el pin GP3

Objetivo

Queremos realizar una aplicación en la que al pulsar un botón se obtengan en una de las salidas un número determinado de impulsos.

Funcionamiento:

Al pulsar el **Botón A** colocado en el **GP1** queremos que se generen en el LED conectado al pin **GP3 15 impulsos**. Los impulsos serán de una duración: estado **ON 200 ms**, estado **OFF 400 ms.**



Queremos usar el display OLED para visualizar la cuenta de impulsos en sentido regresivo, es decir de 15 hasta 0.

Queremos que al empezar y durante la ejecución del ciclo de generación de pulsos se muestren unas pantallas parecidas a las que aparece en la siguiente imagen:

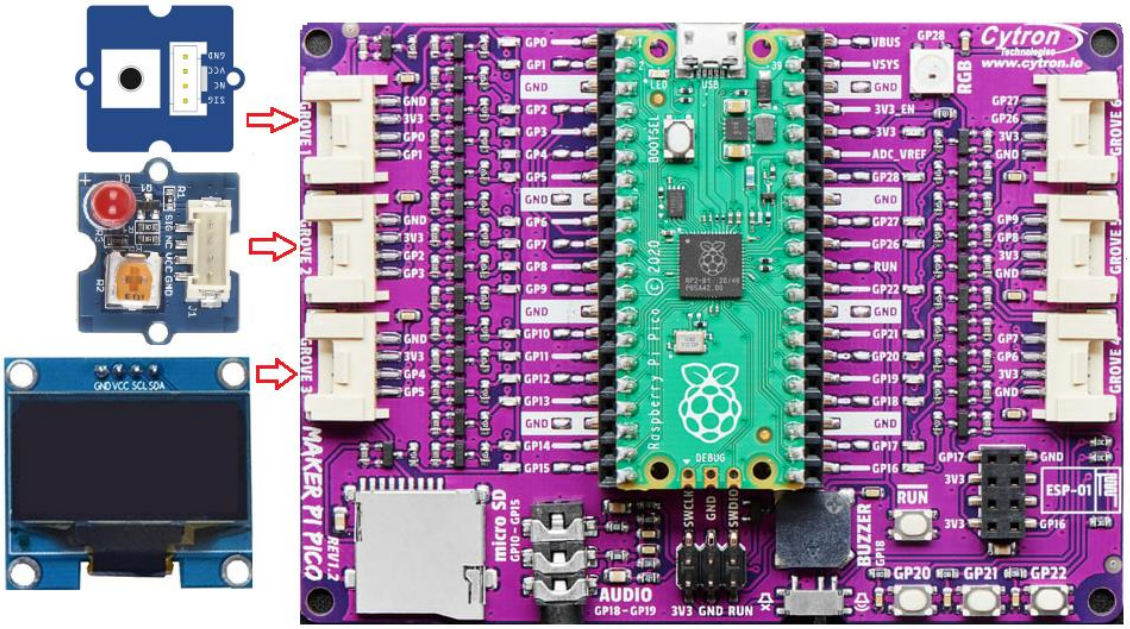
Pantalla de Inicio Pantalla de Ejecución

Entradas salidas:



- GP1 Botón
- GP3 Salida LED
- GP4 SDA del OLED
- GP5 SCL del OLED

Esquema de montaje:



Programa:

El programa va a tener dos fases principales con sus bloques de función asociados.

En primer lugar, el evento será:

Debemos definir dos variables: "**nºpulsos**" y "**pulso**". La primera indica el número de pulsos que queremos generar, en nuestro caso pondremos para probar 15.

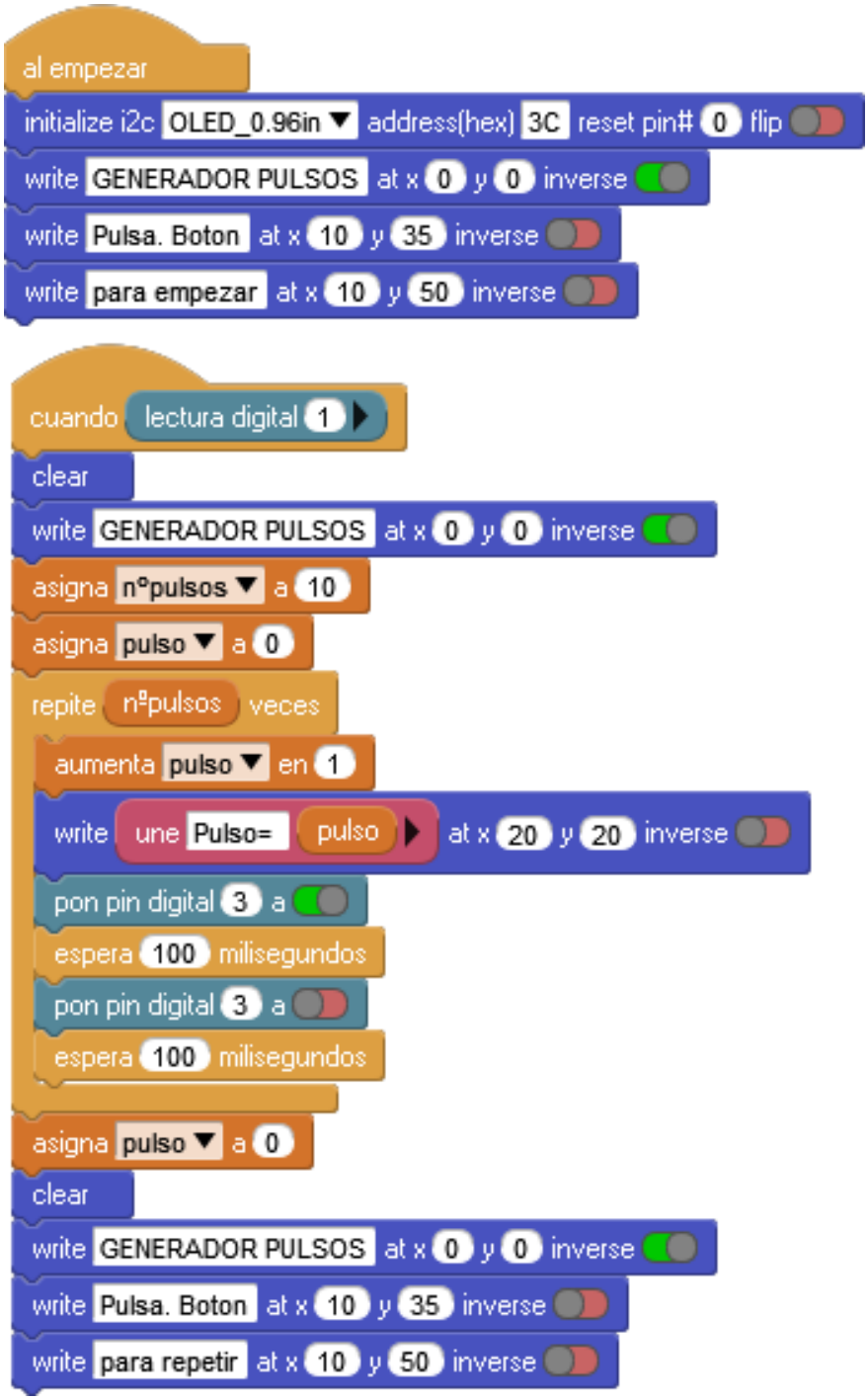
La segunda fase indica el número de pulso actual que se genera.

Al empezar: Se configura el display OLED como siempre hacemos y se ordena la escritura del texto de la pantalla de inicio

El siguiente evento será:

Al Pulsar en el botón conectado en la entrada **GP1**. En este caso se borra la pantalla y se escribe en ella el texto "*GENARADOR DE PULSOS*" y se asignan valores a las variables, *nºpulsos=15* y *pulso=0*.

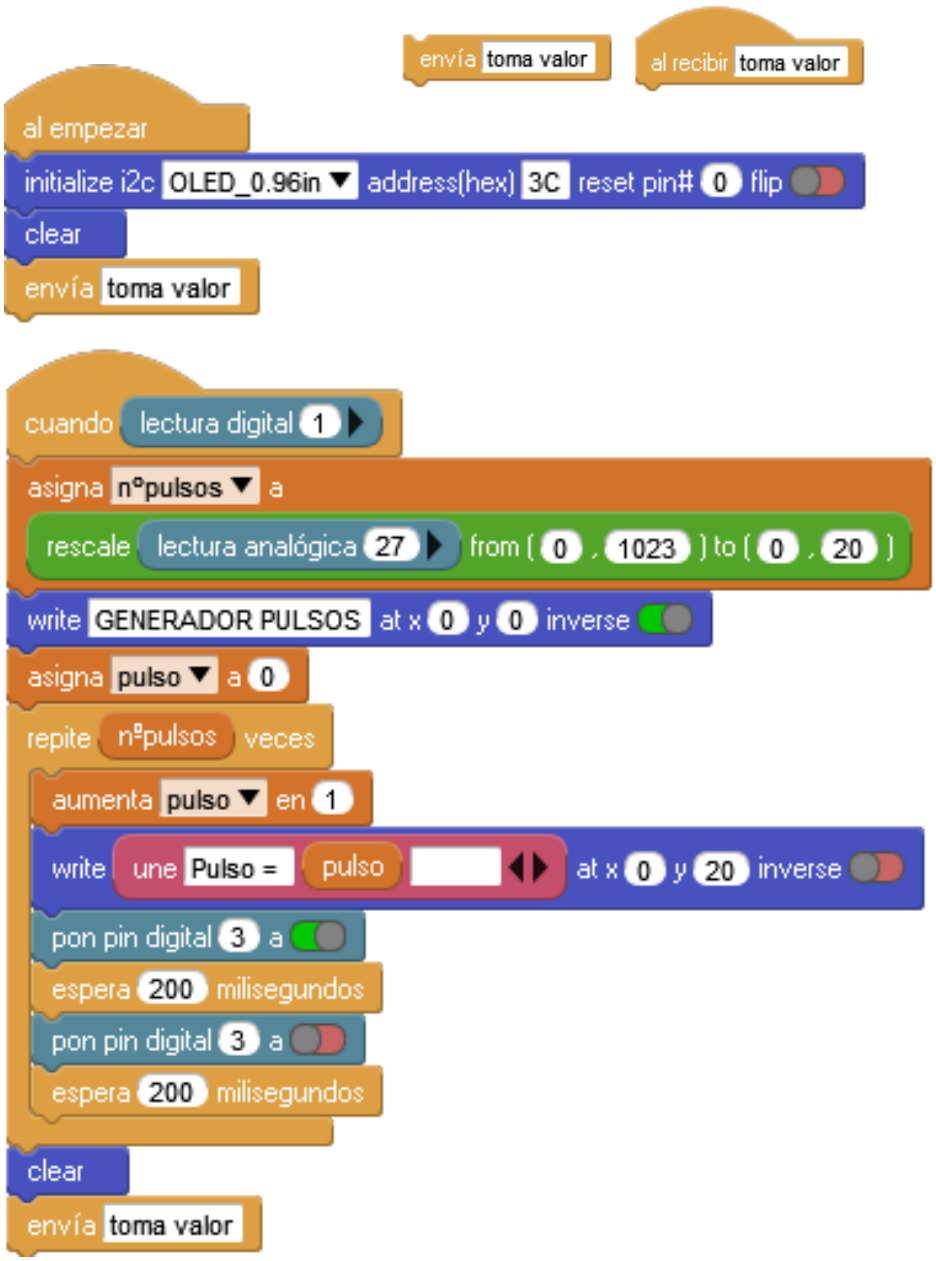
Seguidamente se define un bucle de repetición con el valor de numero de ciclos que indica la variable **nºpulsos**. Incrementamos en 1 el número de **pulso** y escribimos el valor en el display con la correspondiente orden "**write… inverse**". Seguidamente lo que hacemos es activar y desactivar la salida **GP5**, salida de pulsos con LED, con los tiempos de **activado =100 ms.** y **desactivado=100 ms**.

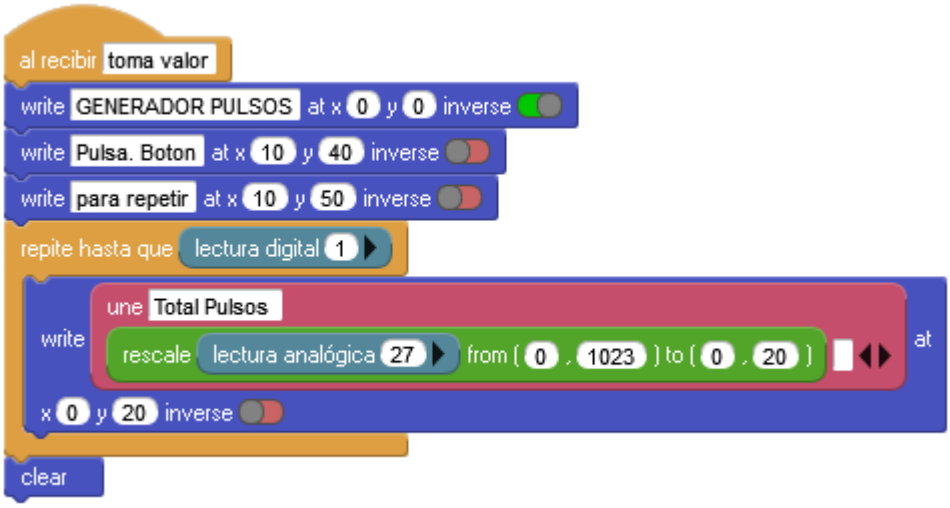


Terminado de ejecutar el bloque de repetición ponemos la variable **pulso=0** y borramos y escribimos en pantalla justo el texto de la pantalla de inicio, quedando a la espera de recibir una nueva orden de generación de pulsos.

Actividades de Ampliación:

1. Queremos realizar la misma aplicación, pero en esta propuesta deseamos que el número de pulsos a generar este comprendido entre 0 y 20 y que la selección de numero de pulsos se realice con la ayuda de un potenciómetro conectado en el pin **GP27**
   En este caso añadiremos un bloque de ejecución "**Toma valor**" que se activara mediante la instrucción de envío de mensaje "**envía** *toma valor*" y la instrucción "**al recibir** toma valor" ambas dentro de la librería "**Control**".



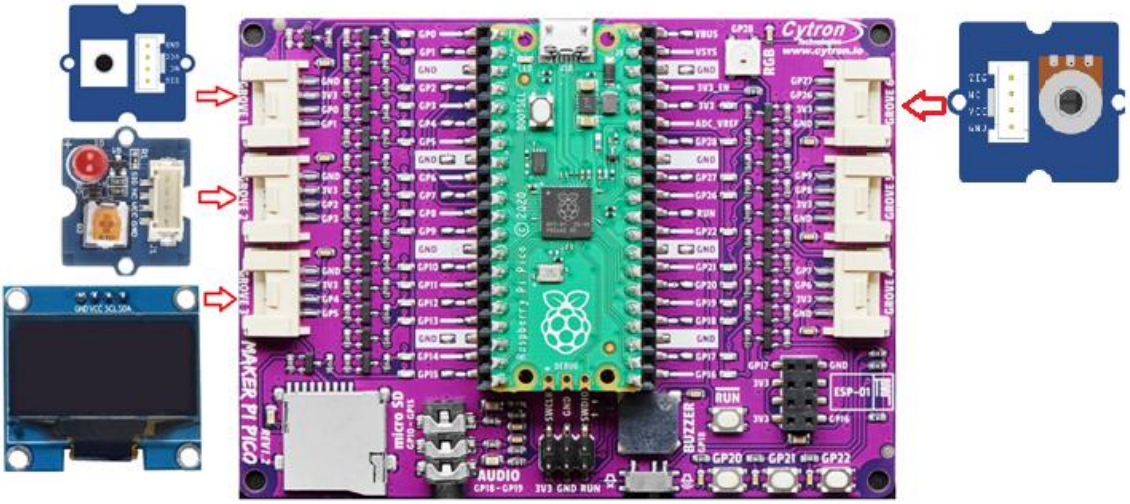


Para esta aplicación diseñaremos las pantallas siguientes

Pantalla de Inicio

Pantalla de Ejecución

Este sería el montaje



6.12. Medida del Nivel de sonido

Objetivo

Vamos a experimentar con un sensor de sonido con el que montaremos un dispositivo que detecte un determinado nivel de sonido y que se señalice el nivel de sonido con dos LEDs uno Rojo y uno Verde.

Funcionamiento:

Se trata de leer el sensor de sonido que este acoplado en el pin **GP27** y representar su valor en el display **OLED**. Como la medida del nivel es de varios dígitos queremos que la señal se "**mapee**" para que su variación sea de **0 a 9**.

Por otra parte, queremos monitorizar el estado del nivel de sonido de acuerdo a la siguiente condición:

- Si valor **sonido <=4** **LED Rojo=0** y **LED Verde=1**
- Si valor **sonido >4** **LED Rojo=1** y **LED Verde=0**

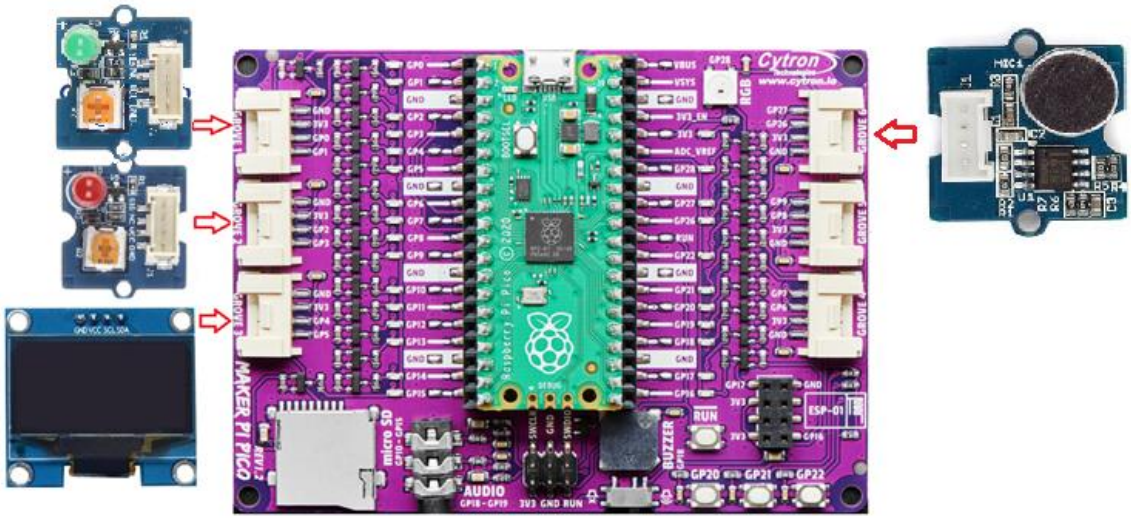
Entradas salidas:

Estos son elementos que usaremos para conectar a la tarjeta microcontroladora

Las conexiones se realizarán en los pines que se indican.

- GP1 LED Verde
- GP3 LED Rojo
- GP4 SDA del OLED
- GP5 SCL del OLED
- GP27 Sensor de Sonido

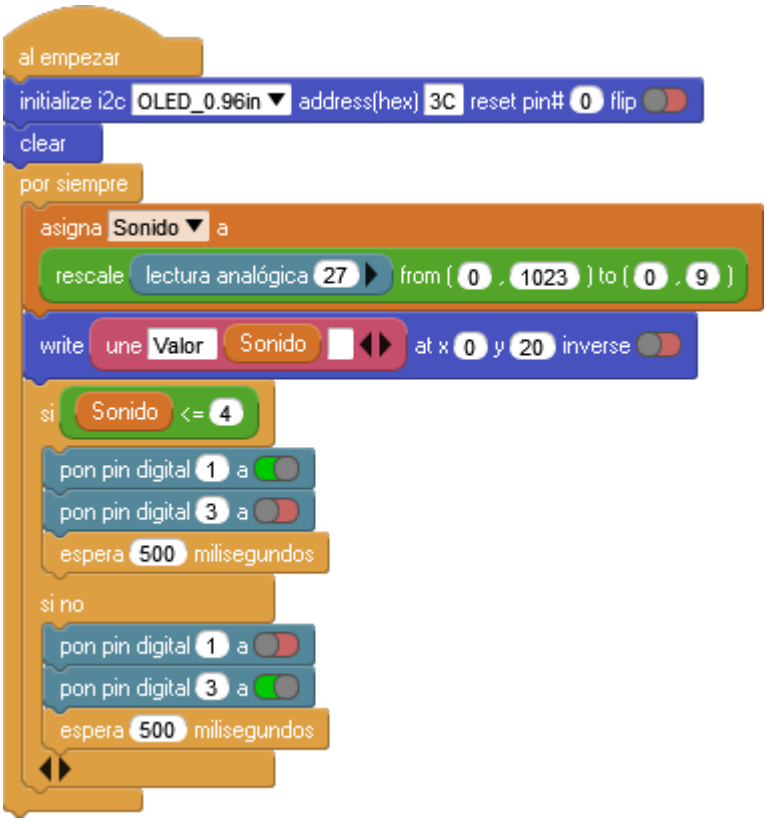
Esquema de montaje:



Programa:

El programa que vamos a crear empezará por la definición de la variable ”**sonido**” que recogerá el valor detectado por el sensor de sonido colocado en el **GP27** que se escalará mediante la instrucción “**rescala..**” llevando su rango a rango **0** a **9.** Seguidamente mostraremos el valor escalado en el display **OLED**.

A continuación, establecemos un condicional cuya condición será que el **sonido <=4**. Cumpliéndose la condición se activará el **LED Verde** colocado en el pin **GP1** y manteniendo apagado el **LED Rojo** conectado en el **GP3**, lo que significará que se mantiene el nivel sonoro en el rango bajo admitido. En caso contrario, cuando sonido >4 se activará el LED Rojo y se apagará el LED Verde, lo que significará que hemos sobrepasado e nivel admisible.



6.13. Interpretación de una melodía al pulsar el Botón A

Objetivo

Usaremos la salida de un Buzer para reproducir sonidos cuando se pulsa un Botón

Funcionamiento:

Utilizaremos la librería tonos y generaremos una secuencia de tonos mediante la función

Toca la nota dada en la octava dada durante el número dado de milisegundos. Los nombres de las notas son A-G (mayúsculas o minúsculas) opcionalmente seguidos de # para un sostenido o _ para un bemol. El valor de la Octava será un numero entre 0 y 5

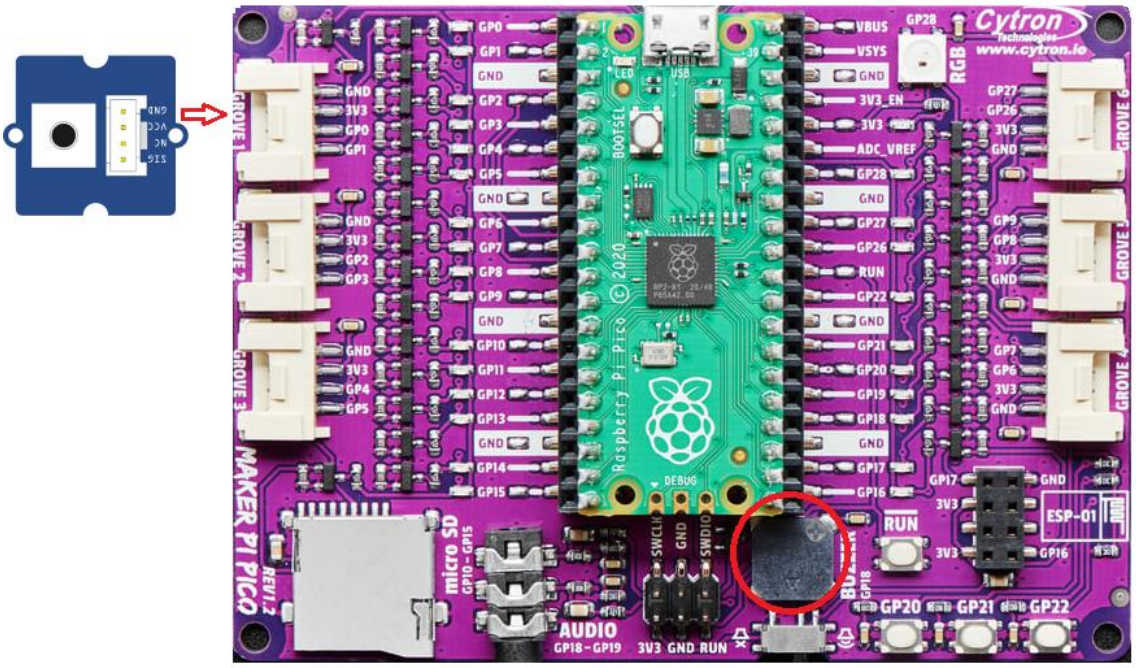
Entradas salidas:

Estos son los dispositivos que usaremos3

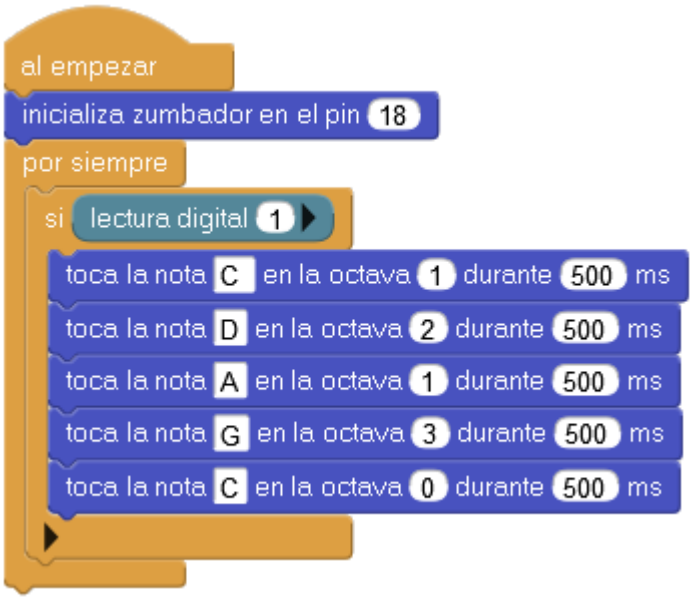
Las conexiones se realizarán en los pines siguientes

- GP1 Botón pulsador
- GP18 Buzer (zumbador)

Esquema de montaje:



Programa:



Actividades de Ampliación

1. Realizar un programa que ejecute en secuencia una lista de 10 notas cuando pulsemos un botón colocado en el pin GP1

Crearemos dos listas que contendrán:

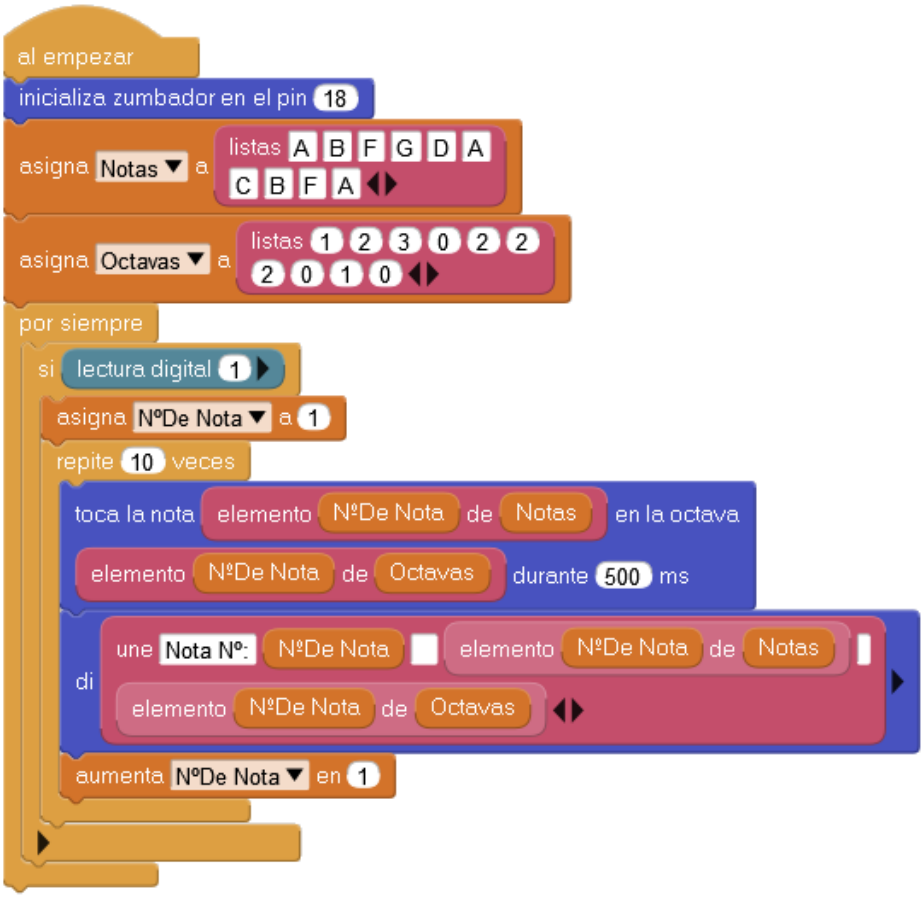
- **Notas** Las diez notas
- **Octavas** Las 0ctavas de cada una de las 10 notas

NºdeNota	Nota	Octava
1	A	1
2	B	2
3	F	3
4	G	0
5	D	2
6	A	2
7	C	2
8	B	0
9	F	1
10	A	0

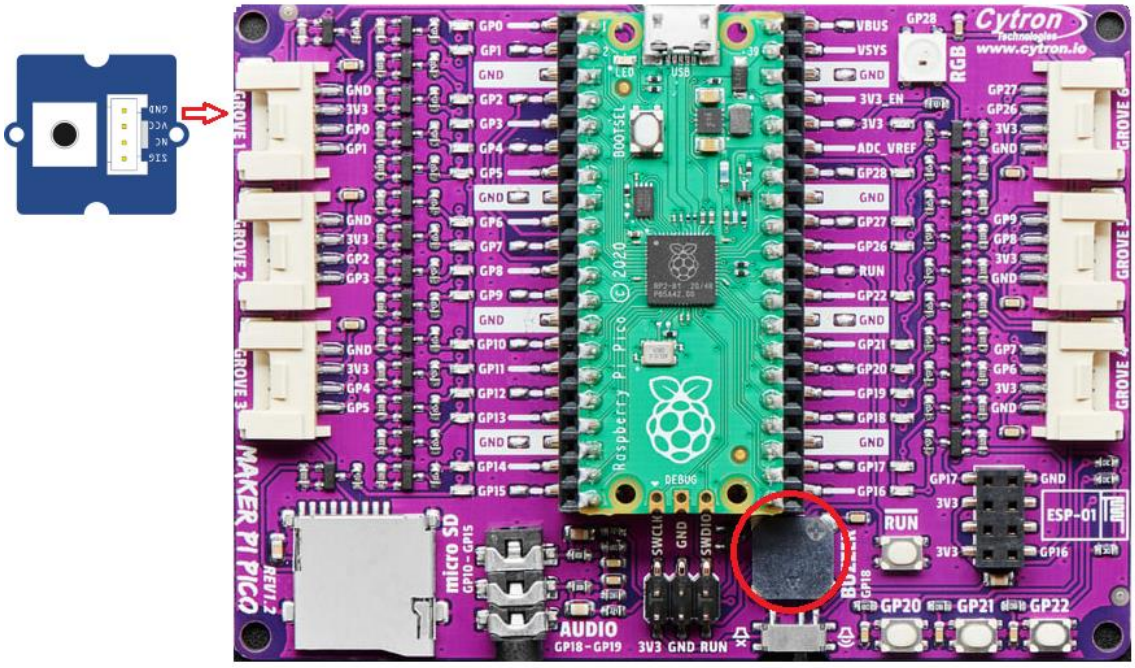
Crearemos una variable que será el **NºdeNota** y que nos servirá para establecer el puntero de recorrido de las dos listas Notas y Octavas.

Mostraremos en la pantalla de **MicroBlocks** la nota correspondiente junto a la octava. Esto lo hacemos con el bloque "**di**" de la librería "**Salida**"

La ejecución se lleva a cabo cuando a través del condicional "**si**.." cuando se cumpla la condición: Botón pulsado del pin **GP1**


al empezar
inicializa zumbador en el pin 18
asigna Notas a listas A B F G D A C B F A
asigna Octavas a listas 1 2 3 0 2 2 2 0 1 0
por siempre
si lectura digital 1
asigna NºDe Nota a 1
repite 10 veces
toca la nota elemento NºDe Nota de Notas en la octava elemento NºDe Nota de Octavas durante 500 ms
di une Nota Nº: NºDe Nota elemento NºDe Nota de Notas elemento NºDe Nota de Octavas
aumenta NºDe Nota en 1

Montaje


Cytron
MAKER PI PICO
Raspberry Pi Pico
AUDIO
micro SD
ESP-01
RGB
RUN

6.14. Control Neopixel

Objetivo

Vamos a controlar los 10 LEDs RGB de Neopixel que se conectara al **GP9** de la tarjeta.

Funcionamiento:

El control consistirá en encender de manera aleatoria cada LED y con color aleatorios los 10 leds de la unidad **Neopixel RGB** con una cadencia de **100 ms**

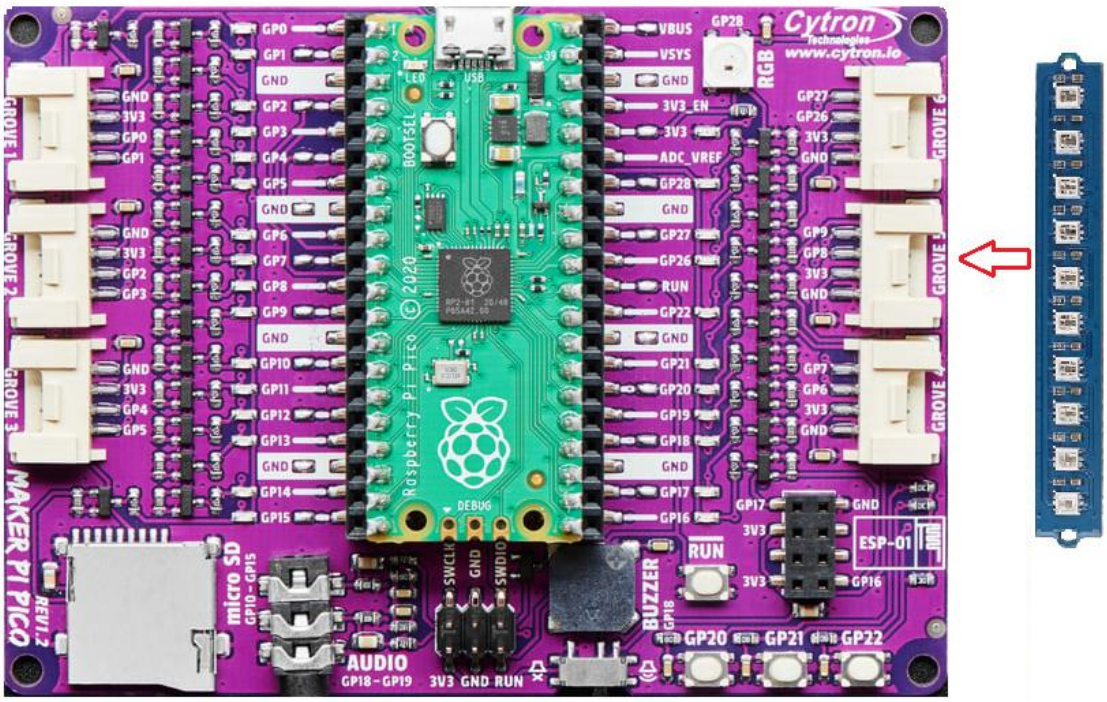
Para realizar esta aplicación no necesitamos la librería "**NeoPíxeles**" de **MicroBlocks**

- Importaremos la librería "**NeoPixeles**"
- Creamos la variable "**led**"
- Establecemos la inicialización del dispositivo **Neopixel RGB** 10 LEDs
- Establecer un bucle en el que la variable "led" adquiera un valor al azar entre 1 y 10 que será el número de LED que encendemos usando el bloque "**pon Neopixel … de color…"**
- El color del LED será al zar y para ello lo haremos usando el bloque "**color al azar**"
- Mostramos el número de LED encendido con la instrucción "**di**.."
- Se establece un retardo de 200 ms que será el tiempo en el que se mantendrá encendido el LED. Seguidamente se apaga el LED con la orden "**apaga Neopixels**"

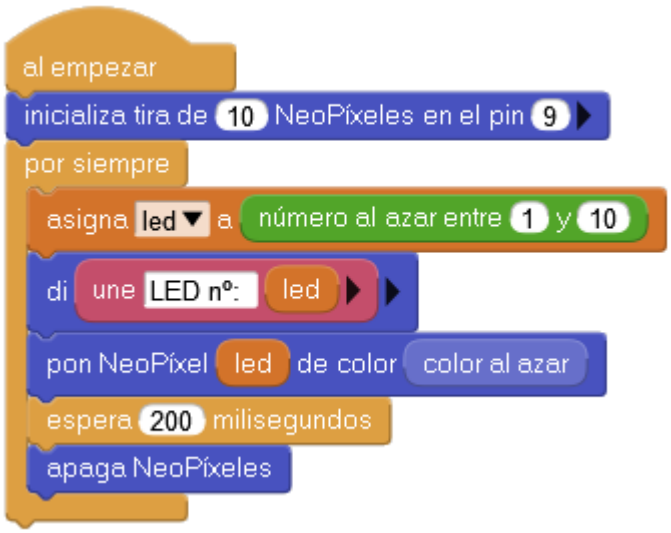
Entradas salidas:

- GP9 Conexión de salida para los LEDs RGB Neopixel

Esquema de montaje:



Programa:

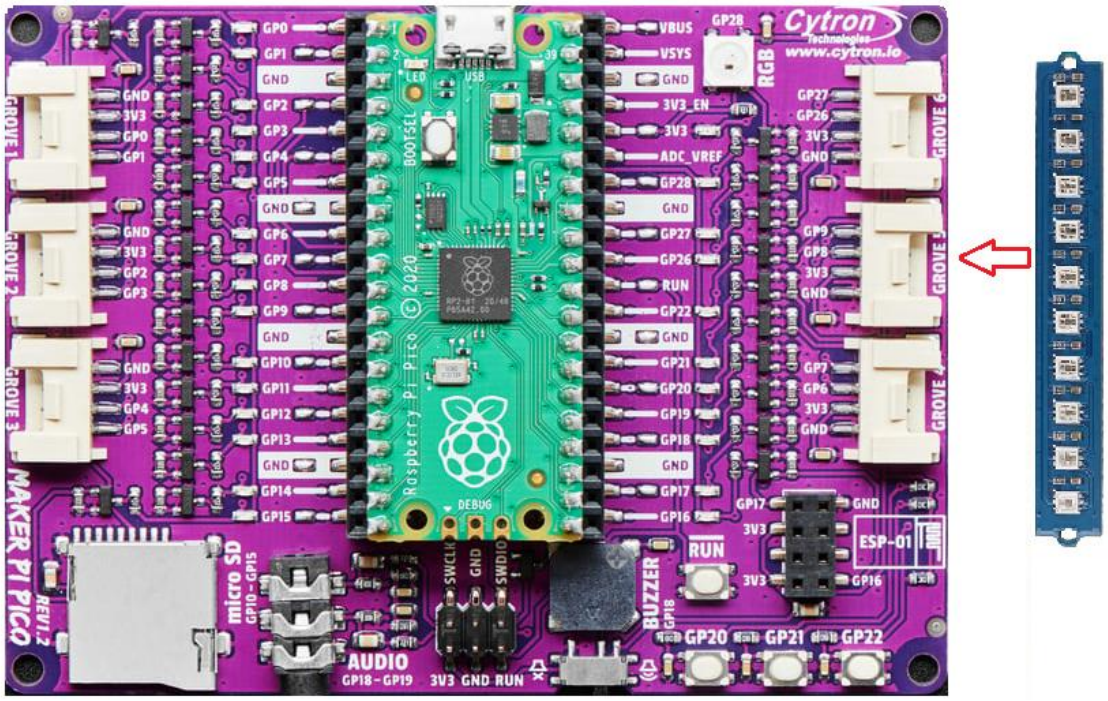


Actividades Propuestas

1. Realizar la aplicación usando **For Next**. Se trata de encender cada LED de manera secuencial con un color aleatorio.

La solución es la siguiente.

Montaje


Cytron
Technologies
www.cytron.io
MAKER PI PICO
REV1.2
Raspberry Pi Pico
© 2020
BOOTSEL
LED
USB
DEBUG
SWCLK
GND
SWDIO
RUN
BUZZER
GP18
micro SD
GP10 - GP15
AUDIO
GP18 - GP19
3V3 GND RUN
RGB
ESP-01
GP20 GP21 GP22
GROVE 1
GROVE 2
GROVE 3
GROVE 4
GROVE 5
GROVE 6

6.15. Semáforo con Neopixel

Objetivo

En esta ocasión vamos a realizar un semáforo usando los tres primeros LEDs de nuestra barra de LEDs Neopixel conectada en el **GP9** de la tarjeta.

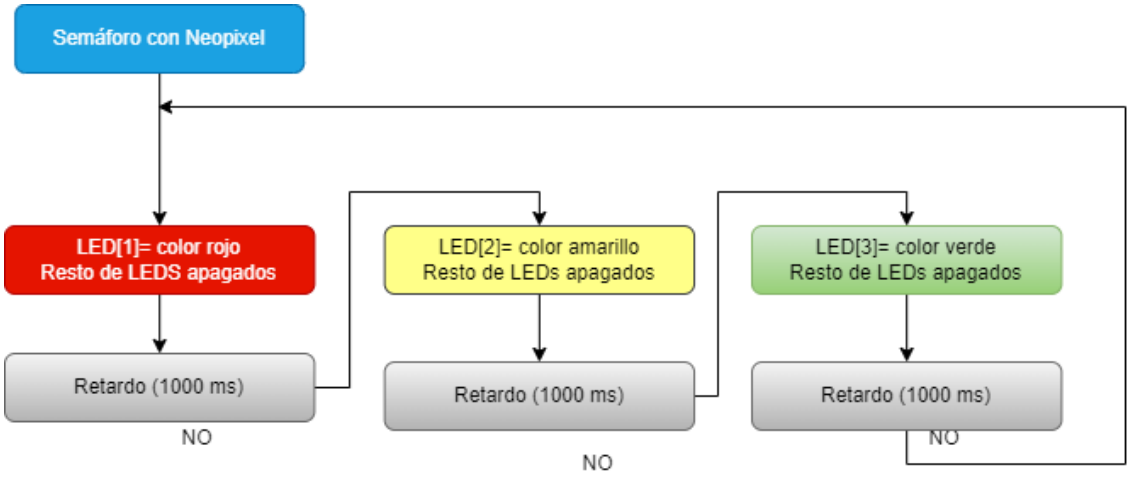
Funcionamiento:

La siguiente tabla de verdad nos indica el funcionamiento de nuestro semáforo:

Estado	LED [0]	LED [1]	LED [2]	Tiempo
Inicio	Todos apagados			
1	255,0,0 (rojo)			1000 ms
2		255,255,0 (amarillo)		1000 ms
3			0,255,0 (verde)	1000 ms

La combinación 0,0,0 es apagado

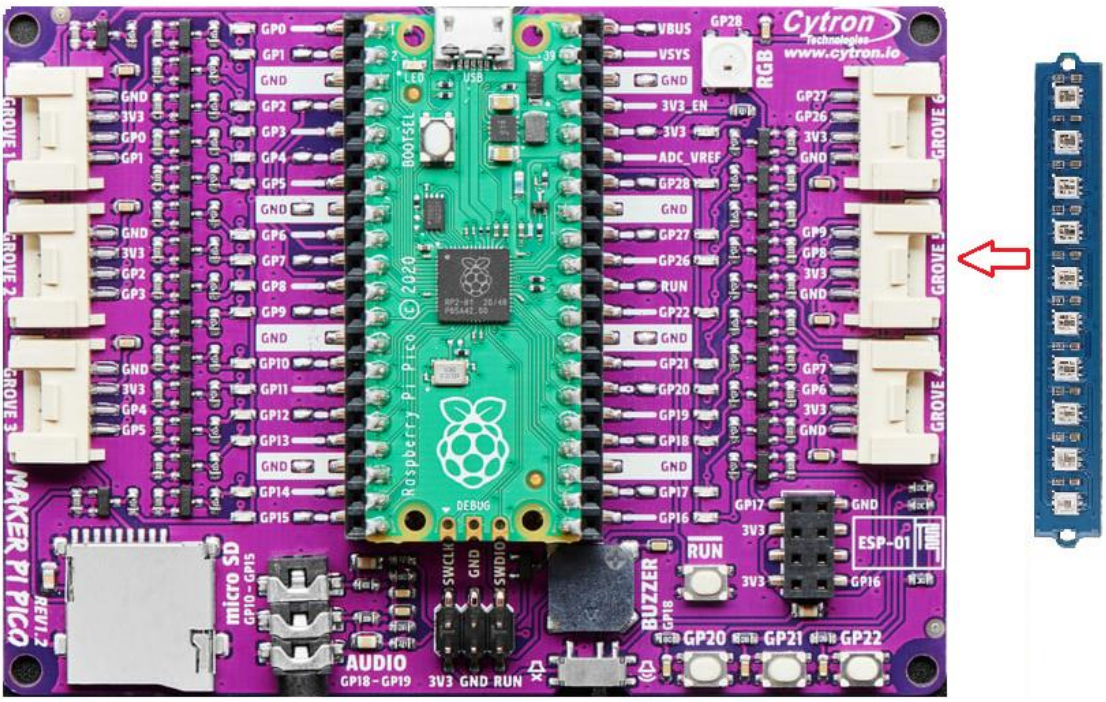
El algoritmo es el siguiente: Cada LED se invoca con la instrucción



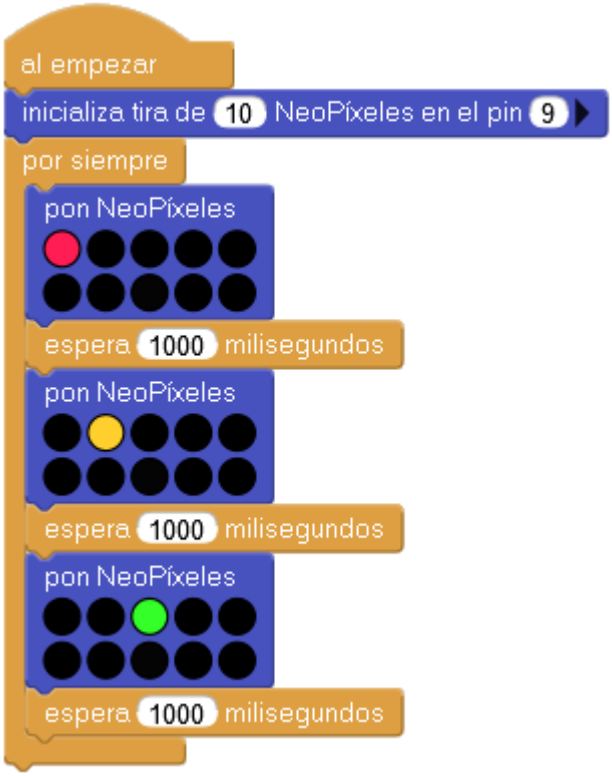
Entradas salidas:

- **GP9** Conexión unidad Neopixel

Esquema de montaje:



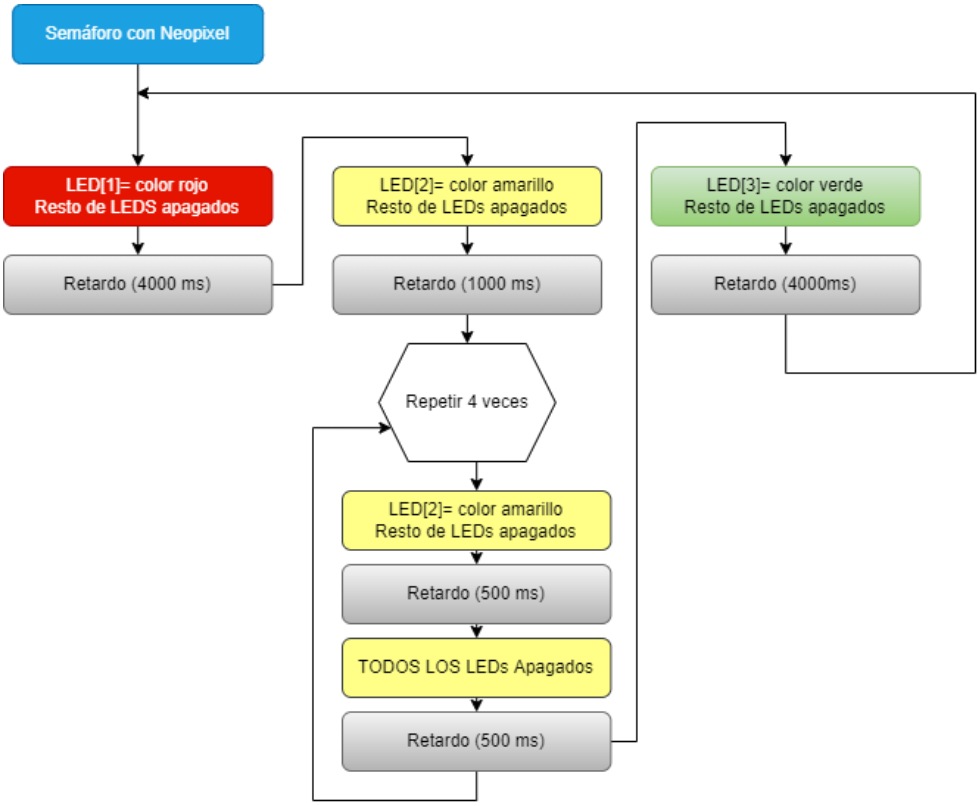
Programa:



Actividades Propuestas

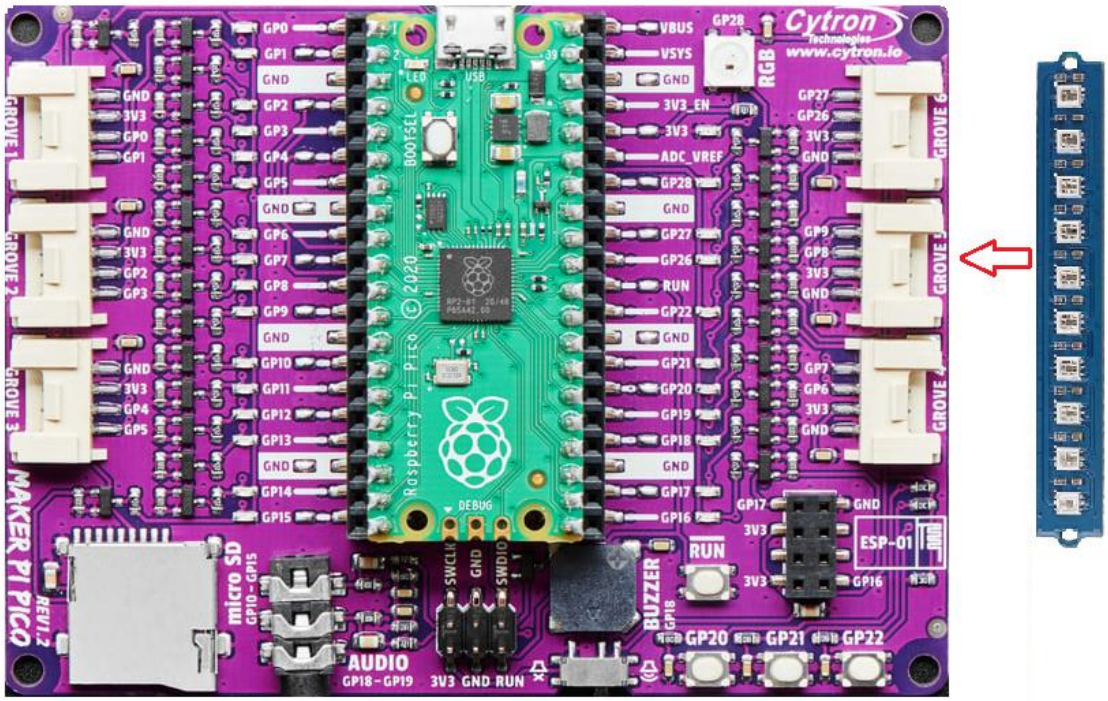
1. Se propone realizar un semáforo igual que el anterior, pero en este caso queremos que la lampara ámbar parpadee 4 veces.

La secuencia temporal será la siguiente:



- Lampara Roja LED1 4 seg.
- Lampara Ámbar LED2 1 seg + 4 intermitencias de o,5 seg.
- Lámpara Verde LED3 4 seg.

Montaje



Programa


al empezar
comentario Aplicacion con LED RGB Neopixel
inicializa tira de 10 NeoPíxeles en el pin 9
por siempre
pon NeoPíxeles
espera 4000 milisegundos
pon NeoPíxeles
espera 1000 milisegundos
repite 4 veces
pon NeoPíxeles
espera 500 milisegundos
pon NeoPíxeles
espera 500 milisegundos
pon NeoPíxeles
espera 4000 milisegundos

6.16. Alarma Básica.

Objetivo

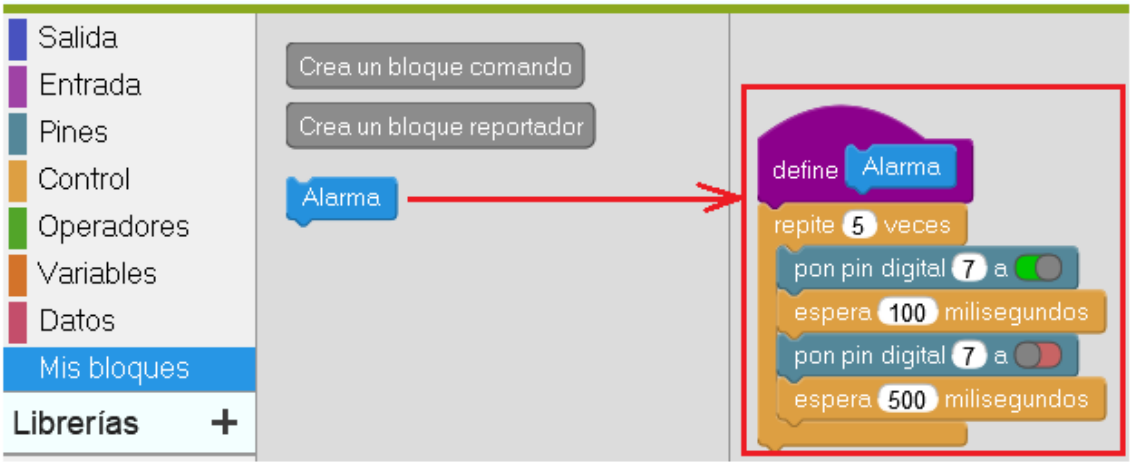
Con este ejemplo vamos a implementar un sencillo sistema de alarma luminosa que consistirá en el encendido y apagado de un LED 5 veces cuando se detecte un nivel digital en una entrada digital en la que conectaremos un pulsador.

Funcionamiento:

Para realizar nuestra aplicación vamos a recurrir a la opción de definir una función de usuario a la que denominaremos "**Alarma**". Esta función lo que hará será encender y apagar un led conectado al pin **GP7** de nuestra tarjeta. Estableceremos los tiempos de encendido y apagado de estos cinco pulsos considerando:

- T.encendido = 100 ms
- T.apagado = 500 ms

En el bucle principal del programa lo que estableceremos es un condicional basado en el estado del pin **GP9** en el que colocaremos un pulsador.



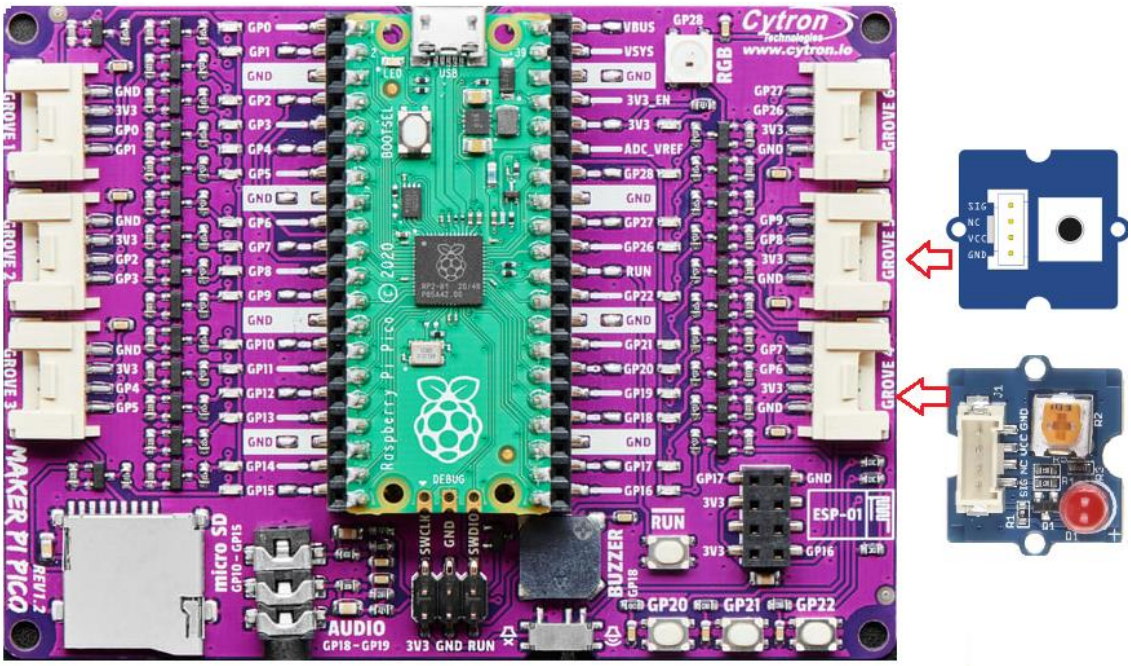
Entradas salidas:

Dispositivos que usaremos

Pines de conexión

- **GP7** Salida Digital LED
- **GP9** Entrada digital Botón

Esquema de montaje:



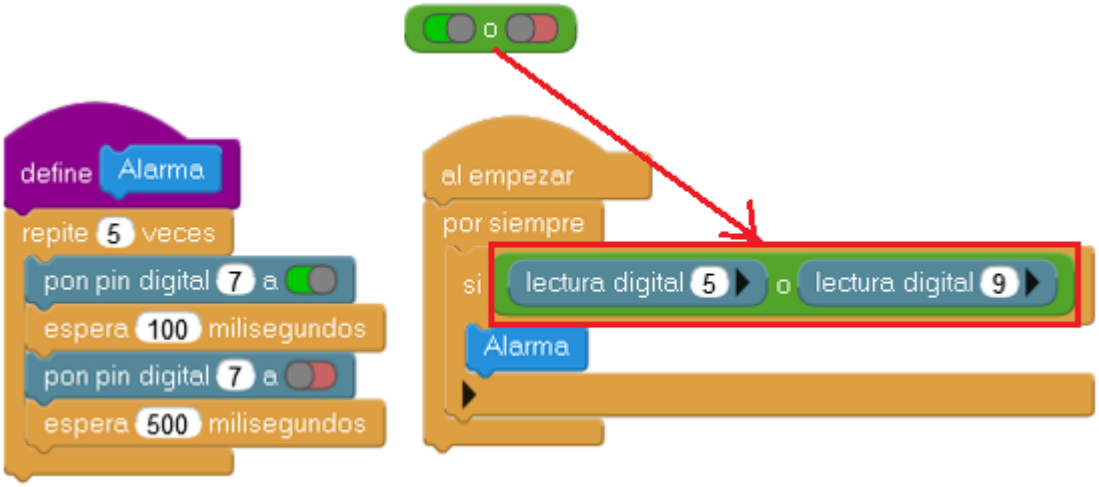
Programa:

Este sería el programa completo. El bucle principal "**por siempre**" incluye dentro de él un condicional que testea la señal digital del pin **GP9**, de tal manera que si el pulsador conectado al pin se pulsa se activa la función "**Alarma**"

Actividades de ampliación.

1. Queremos la alarma se pueda activar desde dos pulsadores distintos, **Botón A** (**GP9**) y **Botón B** (**GP5**). De tal manera que pulsando cualquiera de ellos se debe activar. Realizar el programa y probarlo.

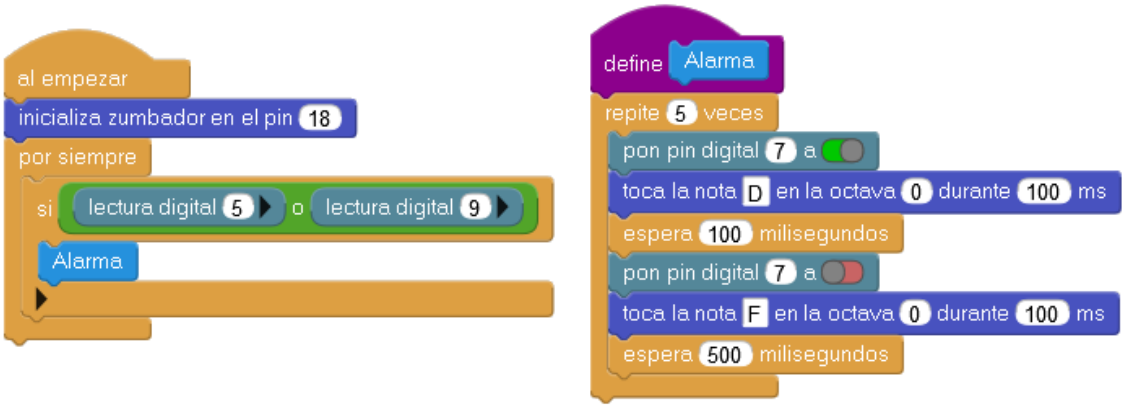
Esta sería la solución. Vemos que lo único que hemos cambiado es la incorporación en la condición de la función "**si**.." de una función lógica tipo **OR**



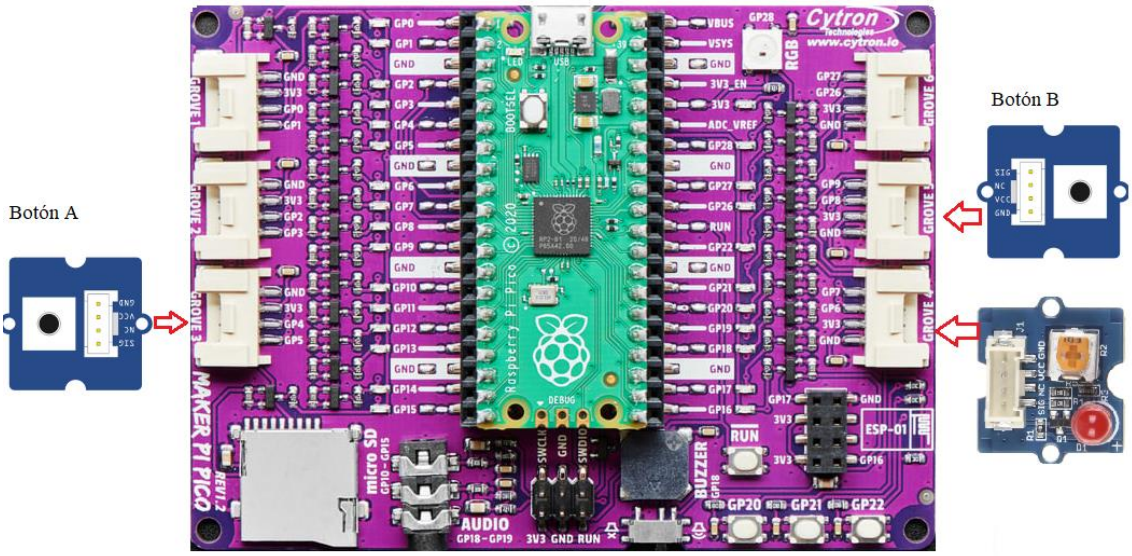
2. En este ejercicio de ampliación queremos que a la vez que se activa y desactiva una señal luminosa (LED) suenen dos tonos en el **Buzzer** de la tarjeta. Mantenemos la activación de la alarma mediante los dos botones **Botón A** y **Botón B** de la actividad anterior

   Lo que hacemos es modificar la función "**Alarma**" incluyendo en ella la generación de los tonos **D0** y **F0**. No olvidemos consignar en el bucle principal de la aplicación que el pin del Buzzer es el **GP18** (míralo en la tarjeta). La librería que tenemos que usar para la generación de los tonos es justamente la llamada "**Tonos**"

   Esta sería la aplicación completa



Montaje


Botón A
Botón B
MAKER PI PICO
Cytron
www.cytron.io
Raspberry Pi Pico © 2020
micro SD
AUDIO
BUZZER
RUN
ESP-01
RGB
GP20
GP21
GP22

6.17. Control de iluminación

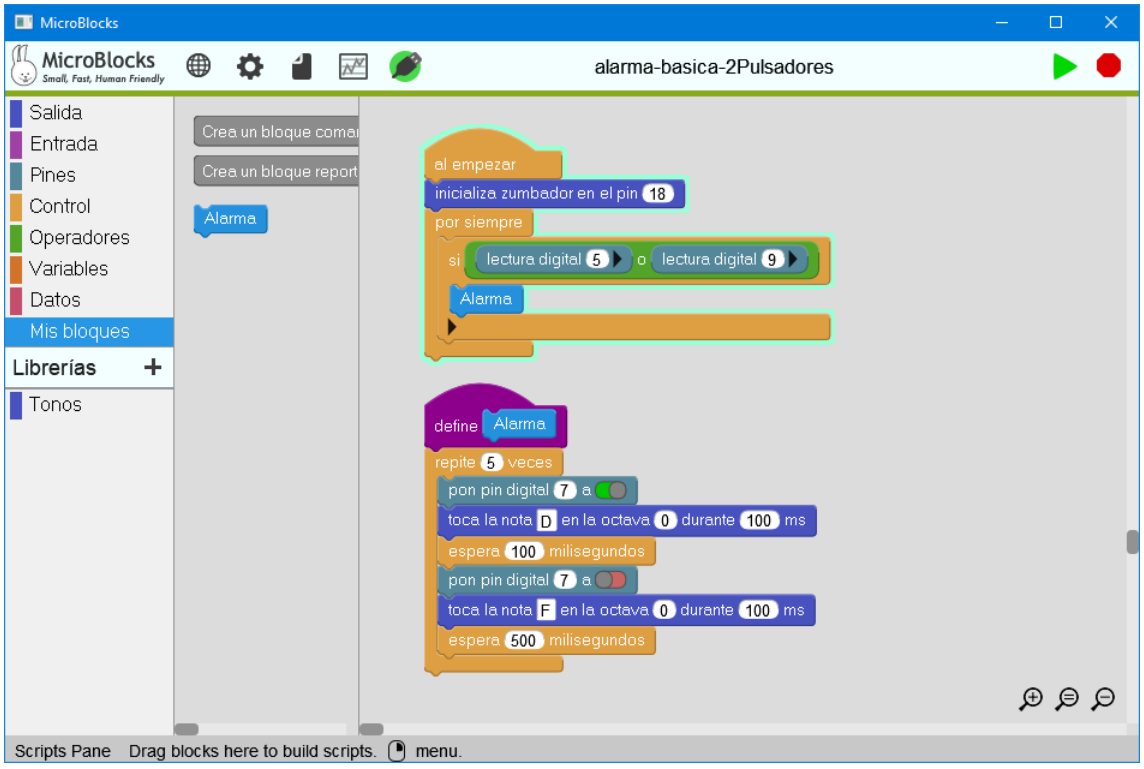
Objetivo

Queremos señalizar si la cantidad de luz en una sala es la adecuada mediante el color de un LED RGB. Para este ejemplo vamos a usar un sensor de Luz

Funcionamiento:

Crearemos una variable que nombraremos como **Luz** y que estará asociada al valor del sensor de Luz.

Tenemos que cargar la librería **Neopixel** que será con la que controlaremos el dispositivo **RGB**



Usaremos un sensor de luz cuyo valor analógico escalaremos al rango **0 a 100**. Lo conectaremos a la entrada analógica **GP27**

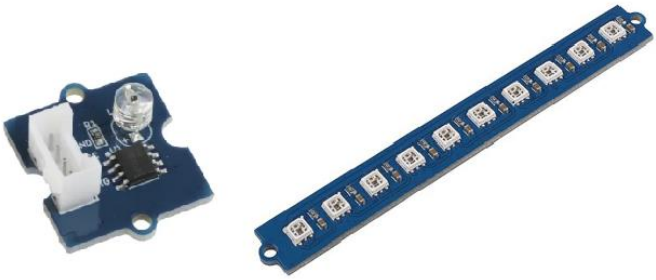
Se van a establecer tres niveles de luz que pondrán de un color determinado el LED RGB que conectaremos en el pin GP1

- Si **Luz <10** LED1 del dispositivo Neopixel RGB será de color **ROJO**
- Si **Luz >=10 Y Luz<40** la luz del LED1 será **AZUL**
- Si **Luz >=40** la luz del LED1 será **VERDE**

Queremos que en la pantalla de **MicroBlocks** sobre el bloque de inicio aparezca el valor de la variable Luz escalada.

Entradas salidas:

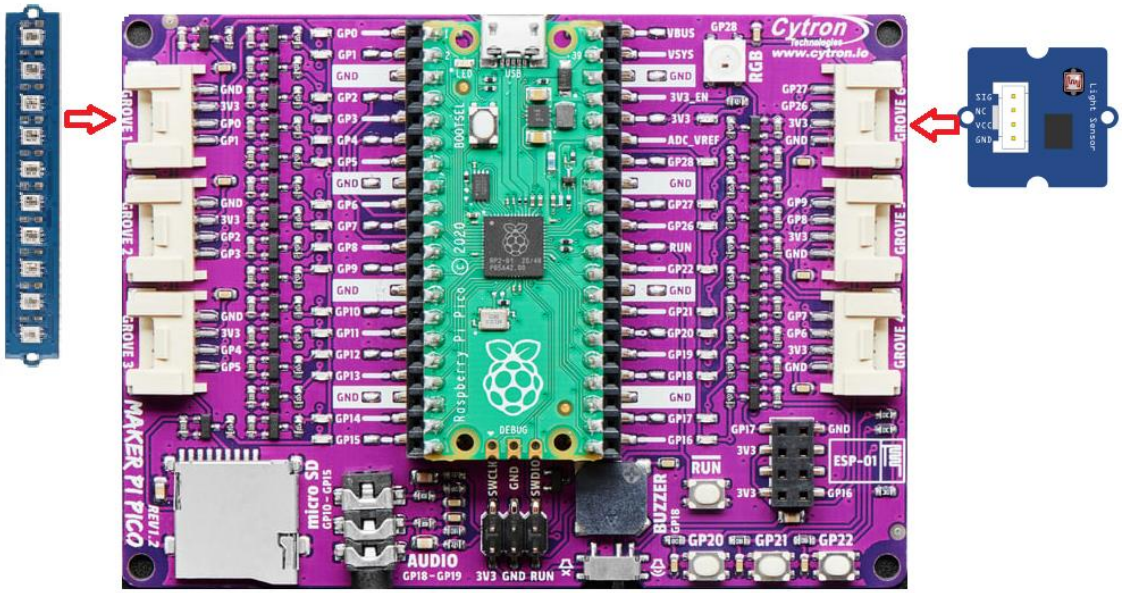
Los elementos usados serán.



Las conexiones son las siguientes

- **GP1** Dispositivo **Neopixel** RGB
- **GP27** Sensor de Luz

Esquema de montaje:



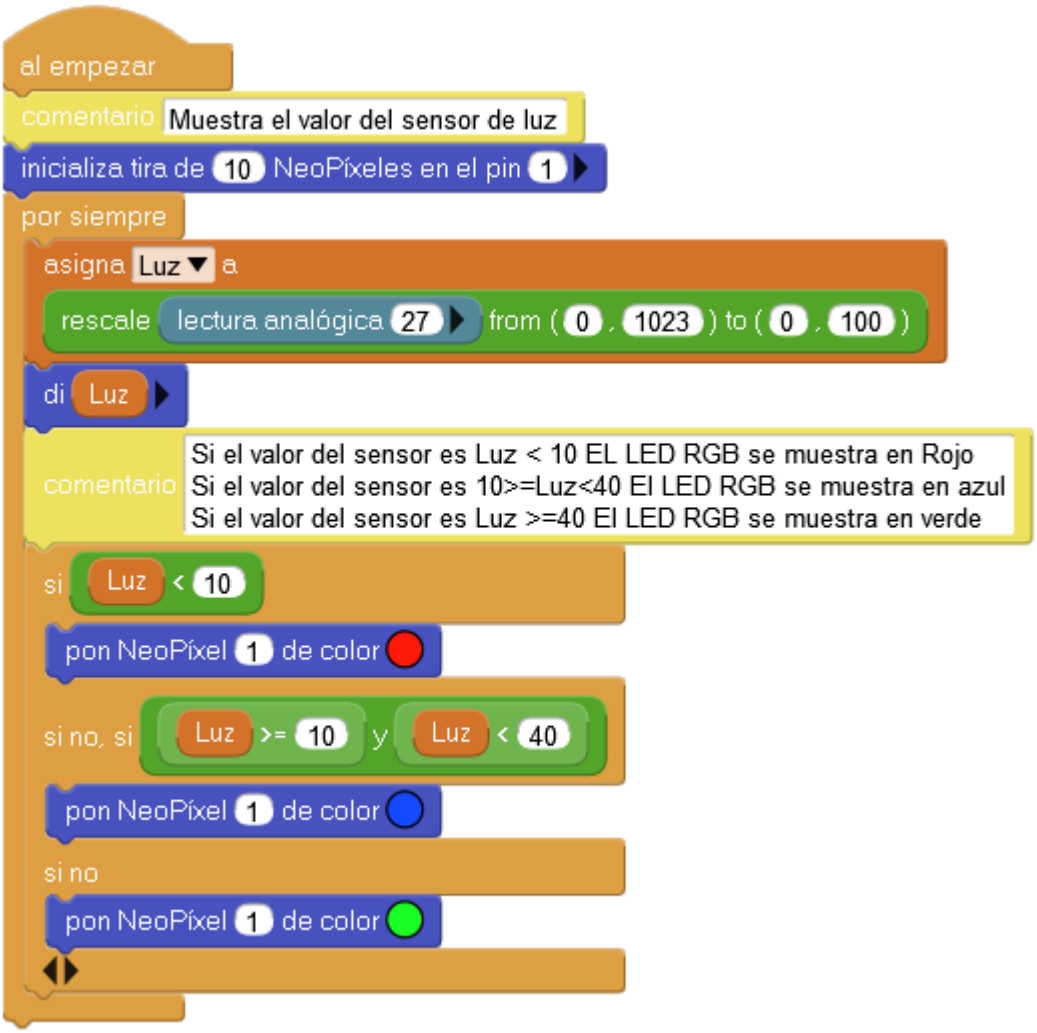
Programa:

El programa comienza con la designación del pin en el que colocaremos el dispositivo Neopixel, pin **GP1**.

Asignamos la variable Luz al valor mapeado del sensor de 0 a 100 con la función "**rescale**.."

Mostramos el valor de la variable Luz con la función "**di**.."

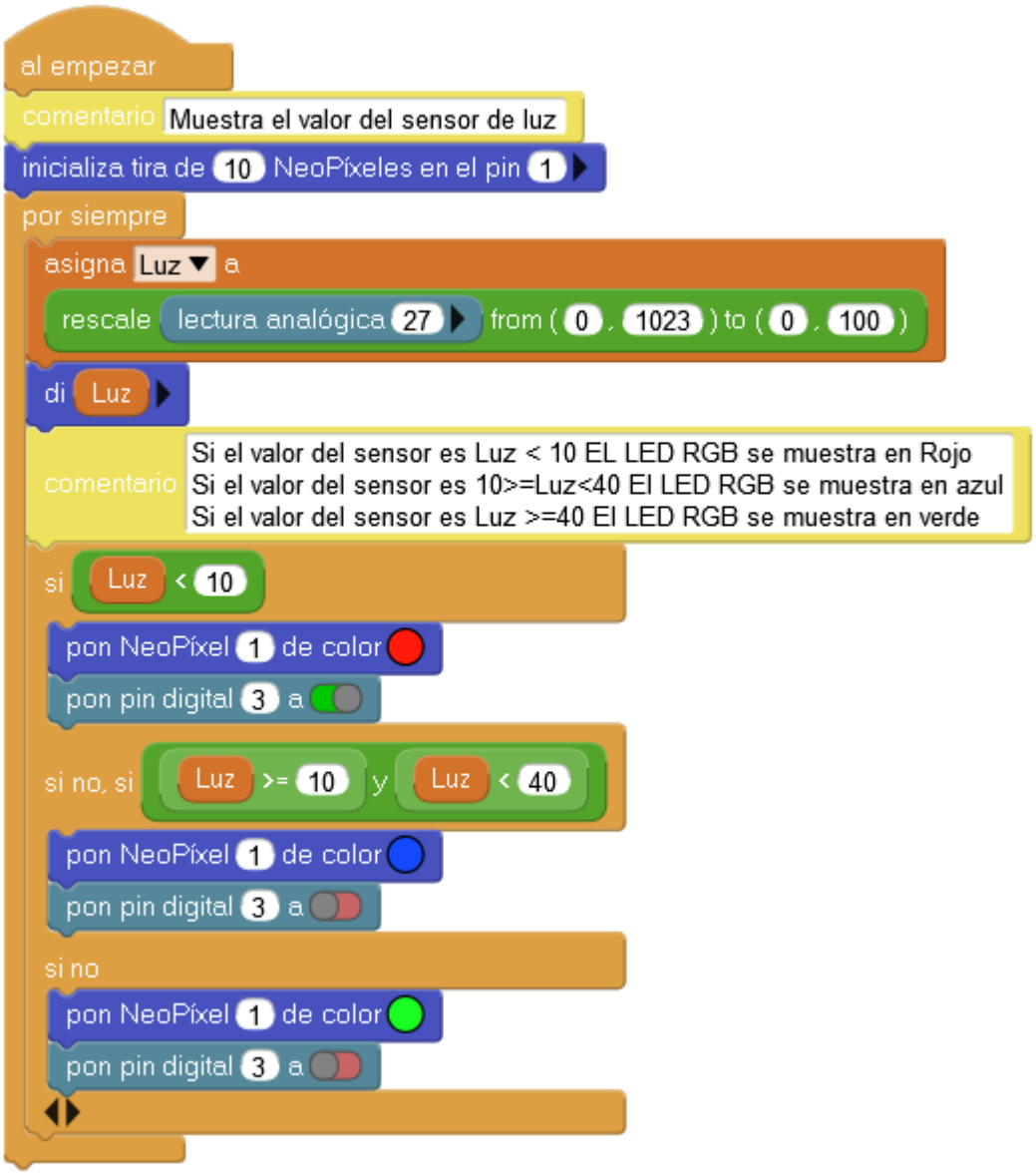
Seguidamente colocaremos los bloques de función encadenados en los que se testean los valores de la luz para cada tipo de rango, tal como hemos explicado anteriormente



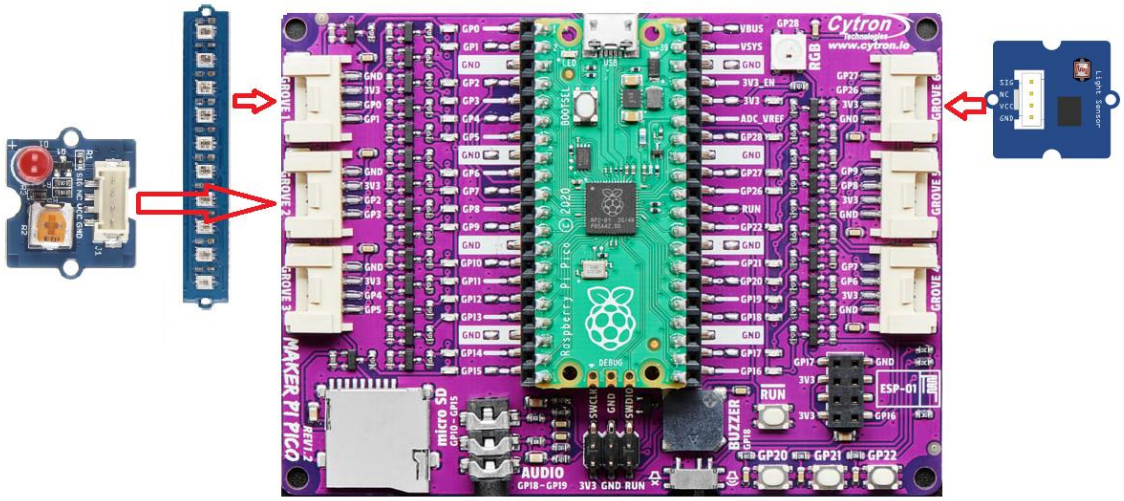
Actividades de Ampliación

1. Queremos realizar las modificaciones necesarias para que nuestro sistema de monitorización del valor de la cantidad de Luz pueda dar orden de activación de una lampara cuando la cantidad de Luz < 10. Para nuestra simulación usaremos un LED de color Rojo que se conectara en el pin GP3.

   La imagen siguiente muestra la solución al ejercicio.



Este es el montaje del ejercicio



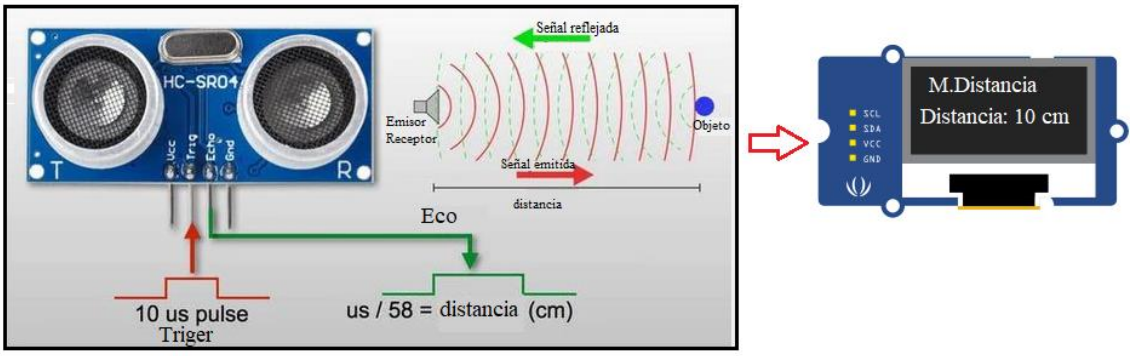
6.18. Medida de Distancia Básico

Objetivo

Se pretende conocer el funcionamiento de un dispositivo para medir distancias basado en ultrasonidos.

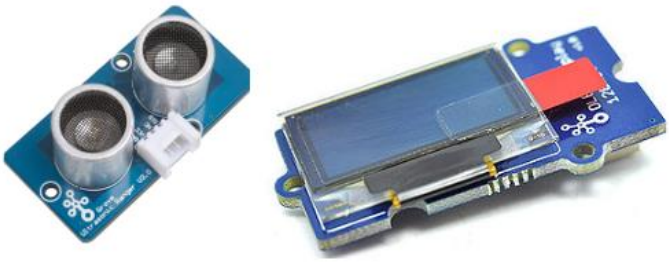
Funcionamiento:

Vamos a manejar el sensor HC-SR04 que conectaremos a nuestra tarjeta. La medida que realicemos la mostraremos en un display OLED



Entradas salidas:

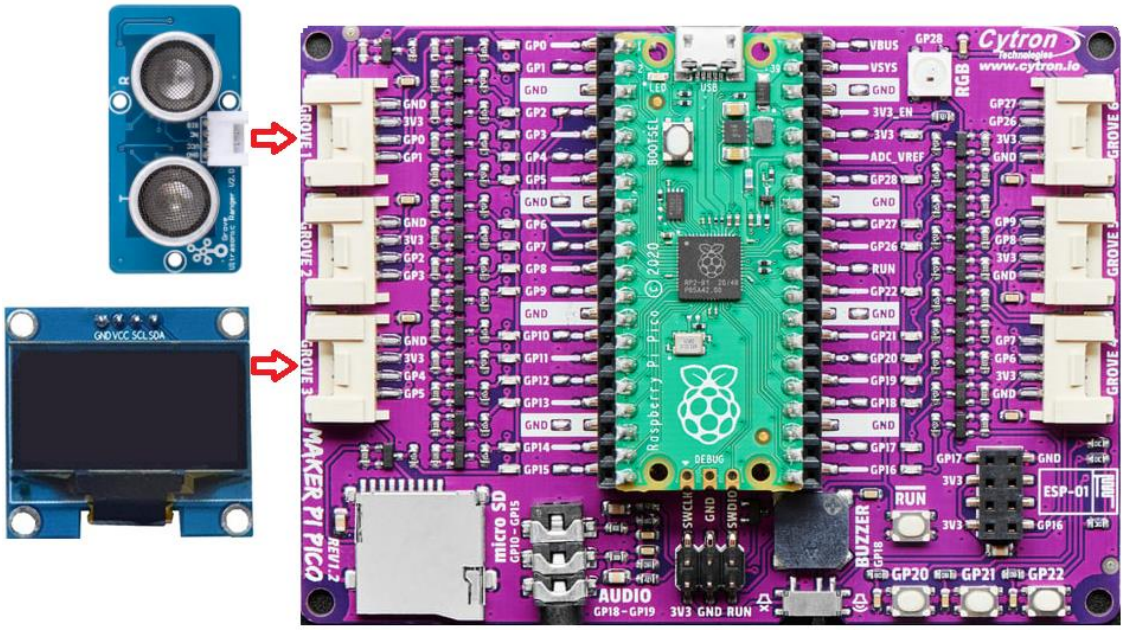
Los dispositivos que usaremos



Sus conexiones a la tarjeta

- **GP4** **SDA** del dispositivo **OLED**
- **GP5** **SCL** del dispositivo **OLED**
- **GP0** **Trig** del Dispositivo **HC SR04**
- **GP1** **Eco** del dispositivo **HC SR04**

Esquema de montaje:

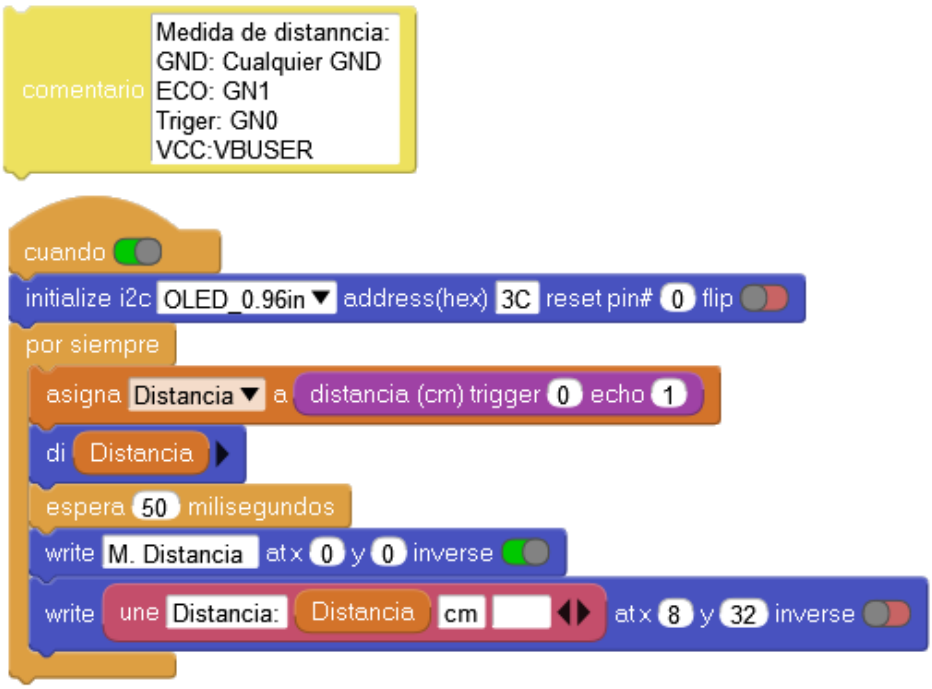


Programa:

El programa sería el de la siguiente figura.

Básicamente hemos seguido los pasos:

- Cargar las librerías "**Distance**" y "**OLE Graphics**"
- Crear la variable "**Distancia**" y reasignar su valor al valor entregado por el bloque "**distancia (cm) …**" de la librería "**Distance**" cuyos parámetros de pines de conexión serán **GP0** y **GP1**
- Inicializar el display OLED
- Mostrar el valor de la distancia en **MicroBlocks** mediante la instrucción "**di**.."
- Esperar 50 ms para permitir el refresco del valor a leer



- Escribir el valor "**Distancia**" en el display OLED poniendo delante la etiqueta **M.Distancia**

Actividades de ampliación

1. Te proponemos que realices un análisis de la respuesta del sensor y que dibujes una tabla en la que figure la distancia real y la distancia medida con el fin de comprobar el comportamiento del sensor.

6.19. Medida de Distancia Básico con Sonido

Objetivo

Aplicar el uso de un **detector de distancia** para la emisión de un sonido en función de la distancia. Esta es la emulación de los sistemas de aparcamiento asistido en los automóviles.

Funcionamiento:

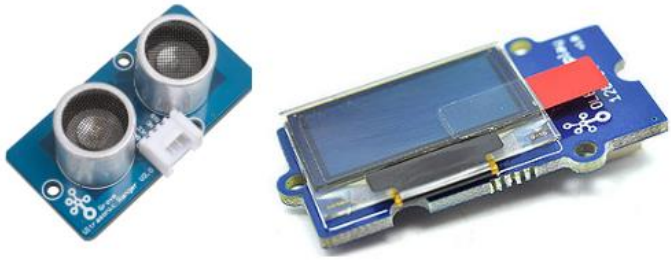
Vamos a montar el mismo ejemplo anterior, pero en este nuevo, incorporaremos una función sonora. Se trata de usar la librería "**Tonos**" con el fin de usar el bloque de generación de un sonido "**toca la frecuencia…**" cuyo valor de frecuencia será variable estando en función de la distancia que nos mide el sensor de distancia.

Multiplicamos el valor de la "Distancia" para conseguir una variación de frecuencias audibles, también podríamos haber usado la función "**rescale**"

.

Entradas salidas:

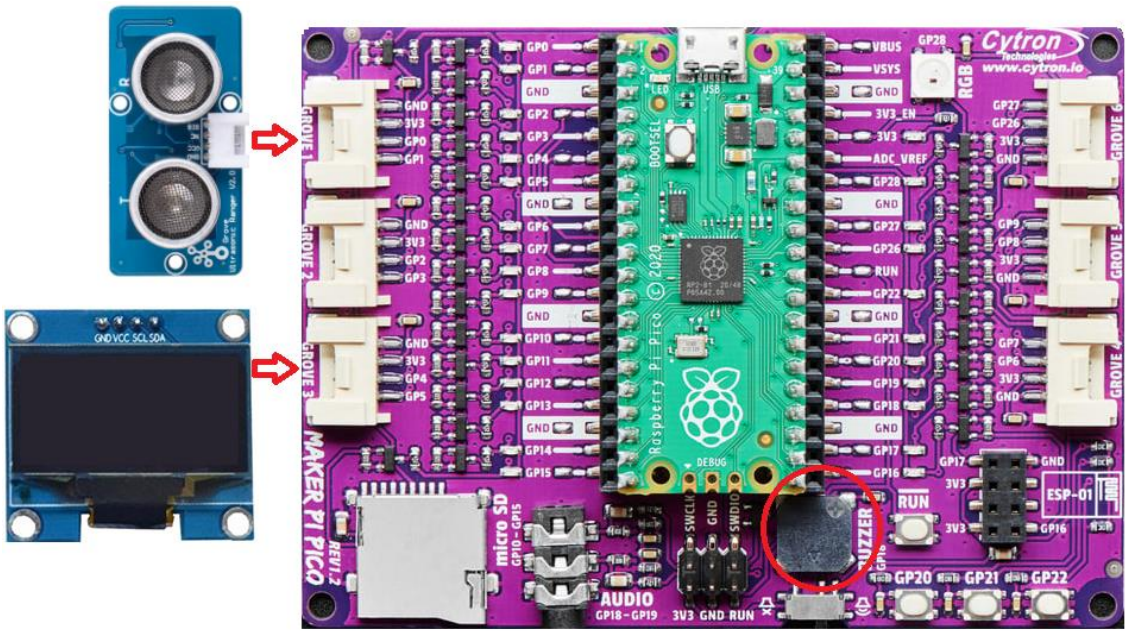
Los dispositivos que usaremos



Sus conexiones a la tarjeta

- **GP4 SDA** del dispositivo **OLED**
- **GP5 SCL** del dispositivo **OLED**
- **GP0 Trig** del Dispositivo **HC SR04**
- **GP1 Eco** del dispositivo **HC SR04**
- **GP18 Buzer** de la tarjeta

Esquema de montaje:



Programa:

El programa sería el que se muestra en la siguiente figura

6.20. Manejo de la librería “Little Numbers” para mostrar valores numéricos en un OLED

Objetivo

Vamos a usar una librería que contiene las imágenes de los dígitos de 0 a 9 de tamaño mayor que el texto convencional del display OLED con el fin de poder mostrar valores numéricos de una manera más visible.

Funcionamiento

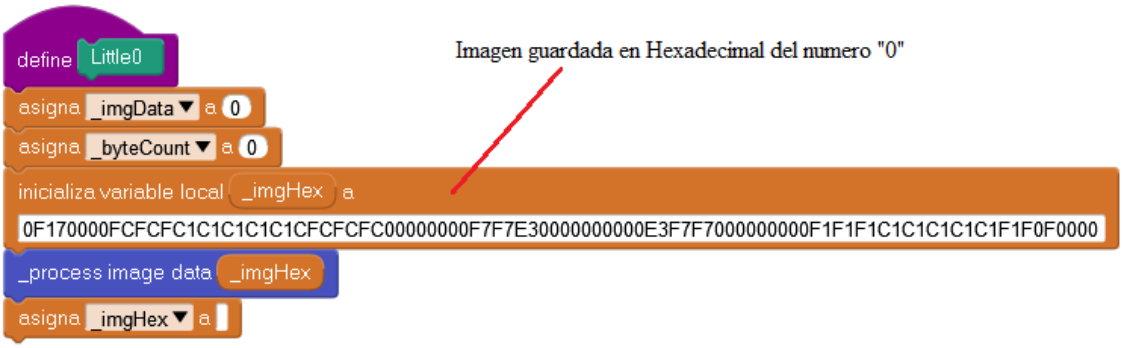
Usaremos la opción de creación de nuevas librerías. La librería que usaremos nos la dan y contiene los bloques siguientes.

Usamos la librería “**Little Numbers**” que se facilita junto con este manual.

Descarga de la las librerías de **Peter Mathijssen**: PeterMathijssen/picobricks-libraries

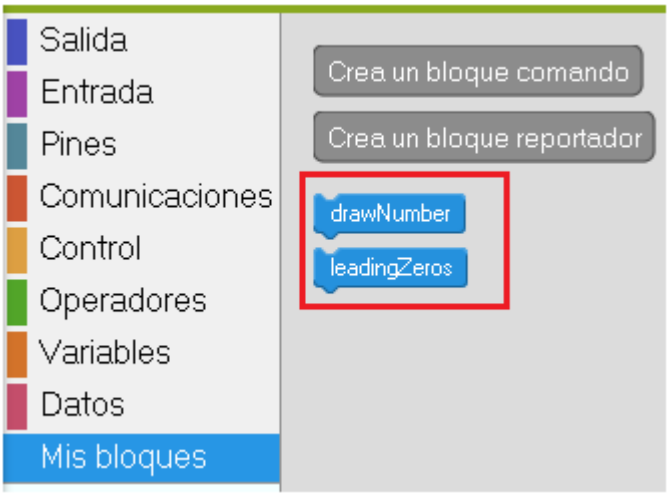
Pagina de Peter Mathijssen

Para incluir una imagen en un display OLED podemos hacerlo siempre que la imagen este en formato Hexadecimal. La siguiente imagen se corresponde con el digito “0”



En el siguiente Documento encontrarás información para crear tus propias librerías de imágenes.

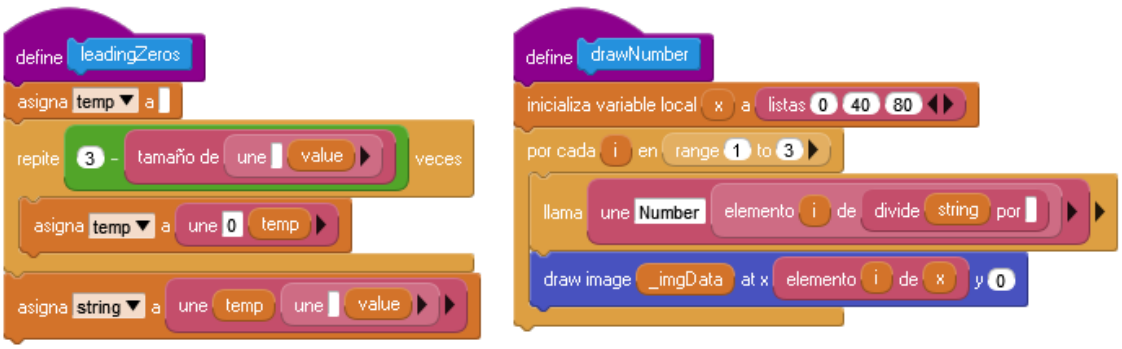
Necesitaremos usar dos librerías de usuario para trabajar en las siguientes aplicaciones además de la librería de imágenes que hemos mencionado. Las funciones de usuario a las que nos referimos tienen la misión de establecer y disponer la escritura de un valor numérico en el display OLED. Se denominan "**drawNumber**" y "**leadingZeros**"



leadingZeros se encarga de completar con ceros a la izquierda el valor a representar. Por ejemplo, se queremos mostrar tres cifras (para escribir de 0 a 999) si el valor en un determinado momento es 6 esta función añadirá al valor dos ceros 006

drawNumber se encarga de representar el numero "valor" que le entrega la función anterior una vez que ha añadido los ceros a la izquierda.

Vemos en la figura los scripts de estas funciones de usuario:



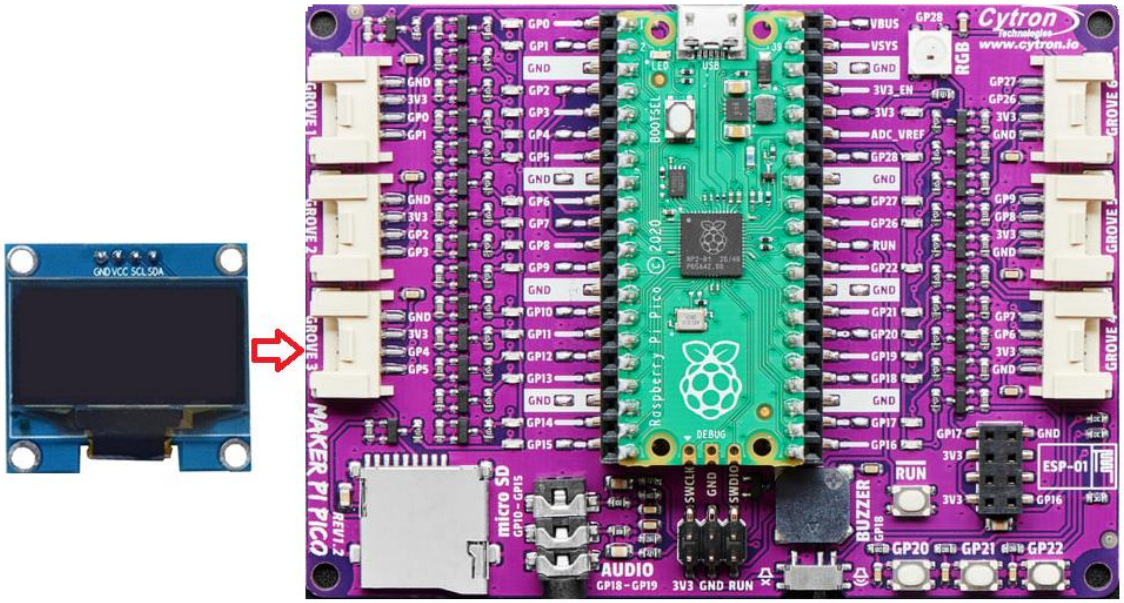
Entradas salidas

Dispositivo OLED

Conexionado

- **GP4 SDA** del dispositivo **OLED**
- **GP5 SCL** del dispositivo **OLED**

Esquema de montaje



Programas

Veamos a continuación un sencillo programa para mostrar un numero de la lista de números "**numbers**" que contiene los nombres de cada uno de los números que forman la librería "**Little Numbers**". La lista se construye con cada uno de los nombres de los bloques de imágenes de la librería, por ejemplo, **Small3.** Para cargar la imagen usaremos el bloque "**llama…**" del grupo de librerías de "**control**"

A esta función le pondremos el parámetro "**elemento** *cualquiera* de **numbers**"

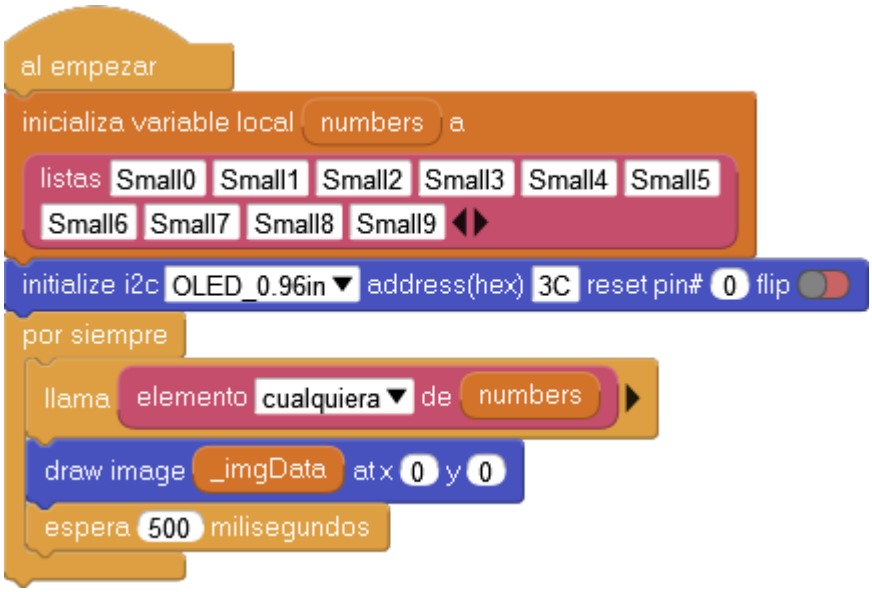
Propuesta de ejemplo 1: Mostrar un elemento aleatorio de la lista numbers

Vamos a realizar el siguiente ejemplo para empezar nuestro estudio del uso de la **librería.**

Para cargar la imagen usaremos el bloque "**llama…**" del grupo de librerías de "**control**"

A esta función le pondremos el parámetro "**elemento** *cualquiera de* **numbers***"*

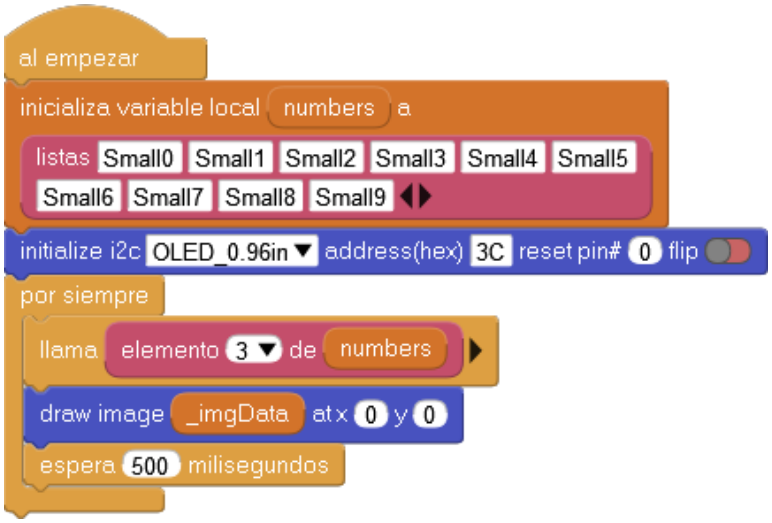
Después de inicializar el dispositivo OLED lo que haremos será mostrar un elemento de la lista "**numbers**" que puede ser en nuestro caso "cualquiera" de manera aleatoria y después usamos el bloque "**draw image** con el parámetro **_imdData**" ***_imdData*** *contiene la imagen leída.* Finalmente ponemos un tiempo de espera de 500 ms para el refresco de pantalla



Propuesta de ejemplo 2: Mostrar una cifra concreta

En el siguiente ejemplo vamos a mostrar en la pantalla uno de los elementos de la lista de números de la librería.

En la siguiente imagen se muestra el elemento 3 de la lista de cifras que se corresponde con la cifra “2”



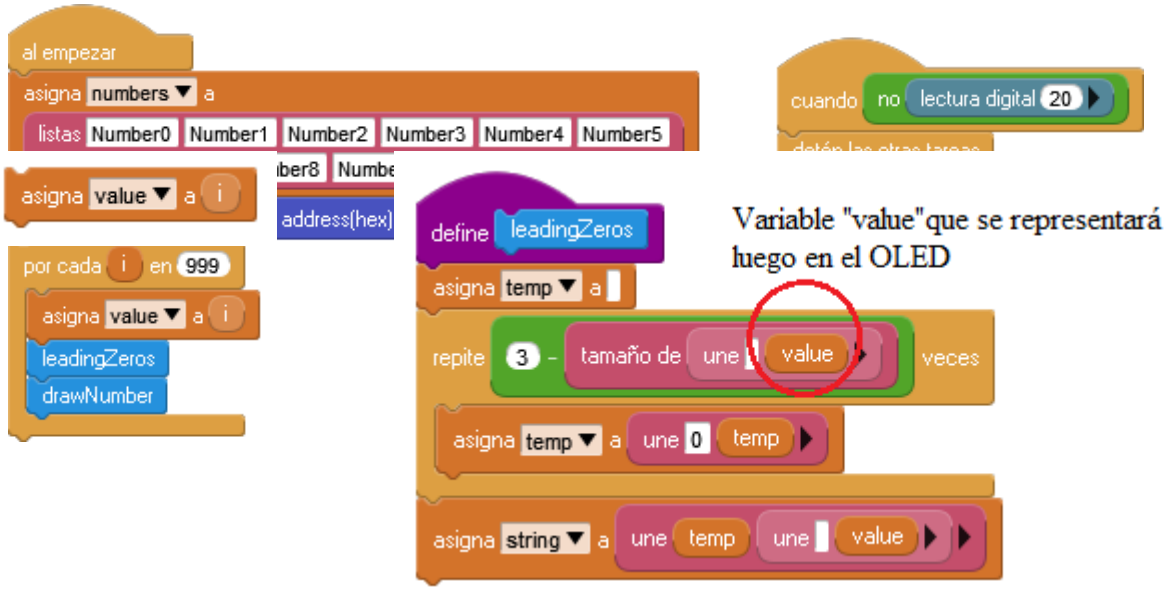
Propuesta de ejemplo 3: Mostrar cuatro cifras aleatorias

En el siguiente programa mostramos 4 cifras cada una de ellas de manera aleatoria.

Básicamente haremos el mismo programa solo que en el bucle incluiremos cuatro llamadas a la función que cargarán un elementos de la lista **numbers** y lo dibujaran en la pantalla con una separación entre números de valor 30 en el eje horizontal.

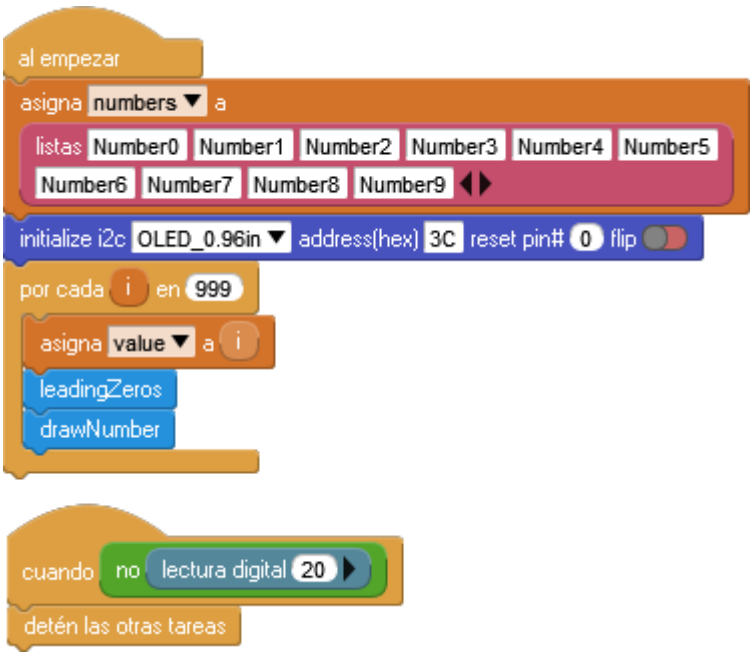
Propuesta de ejemplo 4: Mostrar tres cifras de un contador

Vamos a mostrar el valor de una variable que se incrementa (cuenta).



La variable como vemos es "i" pero la variable que recoge la función "**leadingZeros**" se denomina "**value**" por lo que debemos realizar la asignación de *value al valor i*

El valor de cuenta lo establecemos en "**999**" y se detendrá el contador cuando pulsemos el botón **GP20**



6.21. Medidor Gráfico de un sensor con 4 Dígitos

Objetivo

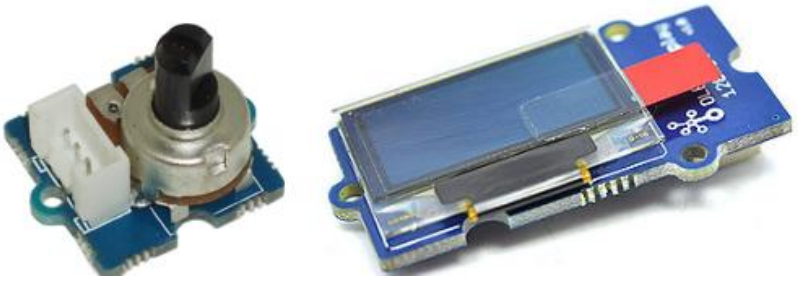
Utilizar la librería "**Little Numbers**" para construir una aplicación que permita mostrar un valor en pantalla con los números mayores que los que permite la librería convencional del dispositivo OLED.

Funcionamiento:

Partimos de la base de que ya disponemos de una librería que tiene hasta 10 bloques, cada uno asociado a un digito, desde el 0 al 9. En la figura siguiente vemos el contenido del bloque dedicado al número 0 "Little 0"

Entradas salidas:

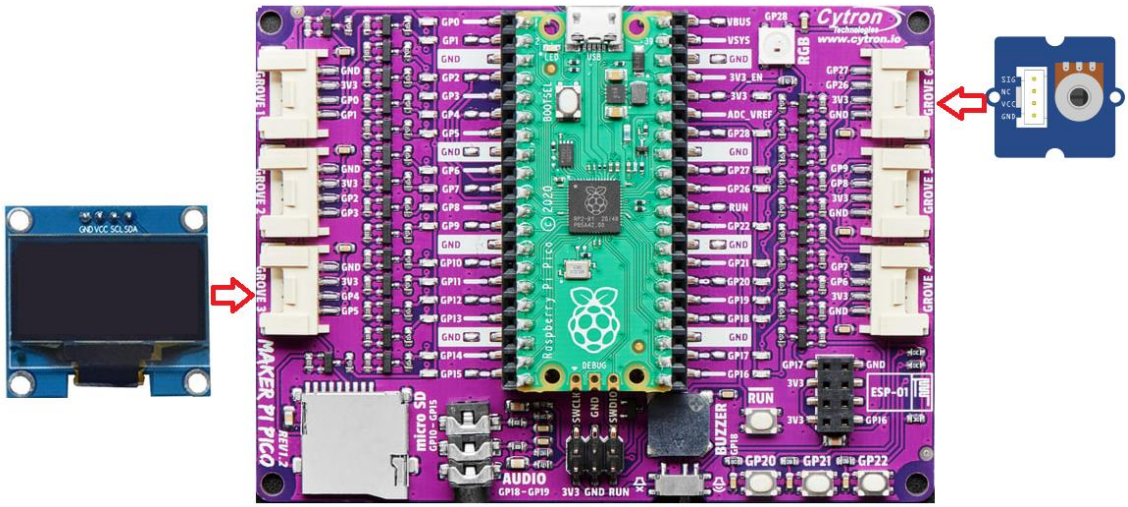
Los dispositivos que usaremos son los de la imagen



Los pines de conexión son los siguientes:

- **GP27 Potenciómetro**
- **GP4 SDA** del dispositivo **OLED**
- **GP5 SCL** del dispositivo **OLED**

Esquema de montaje:



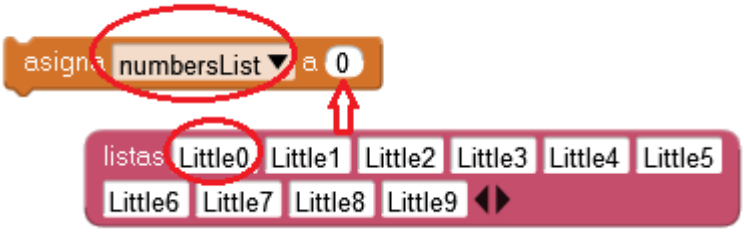
Programa:

Nuestro programa hace uso de dos funciones ya creadas que se facilitan y lo único que haremos será usarlas en la aplicación que vamos a construir.

Los pasos a seguir para la construcción son los siguientes:

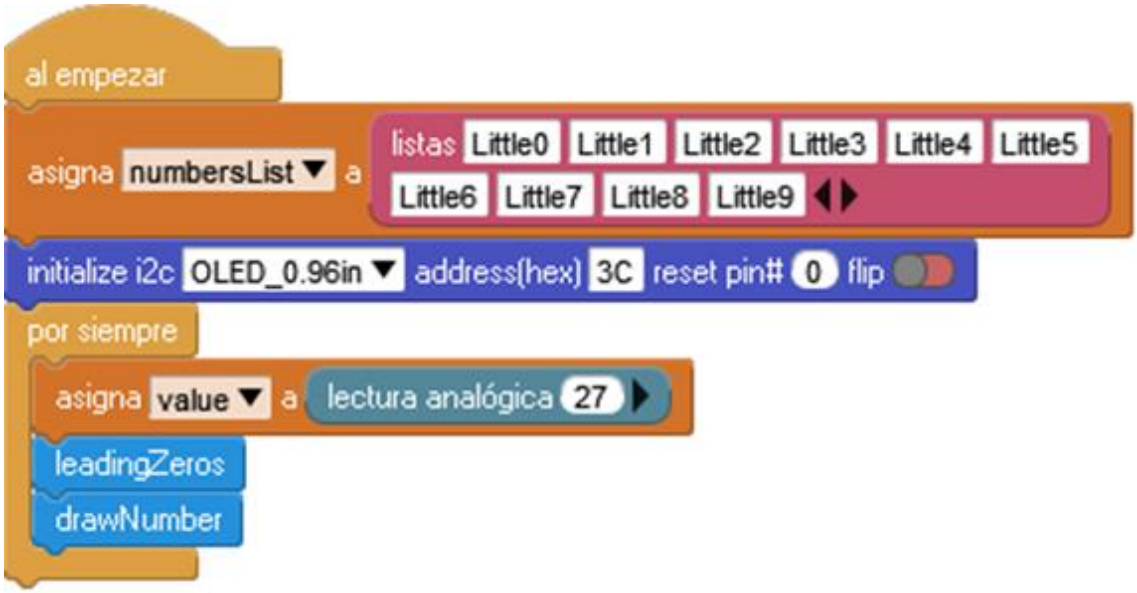
Creamos la variable "**value**" y le asignamos al valor leído en el pin GP27 en donde colocaremos nuestro dispositivo potenciómetro para generar la señal que vamos a mostrar. El nombre de la variable debe ser ese y no otrlo

Creamos una lista variable llamada "**numberList**" y le asociamos una lista con los números de que se corresponde con los elementos de la librería "**Little Numbers**" *Little0 a Little9*



Seguidamente configuramos nuestro display OLED

La señal a mostrar se llama "**value**" porque esta es la variable que reconoce y usan los bloques de librería para mostrar la información en el display **OLED** llamadas **"leadingZeros"** y **"drawNumber"**



6.22. Medidor Gráfico de Luz de 4 Dígitos

Objetivo

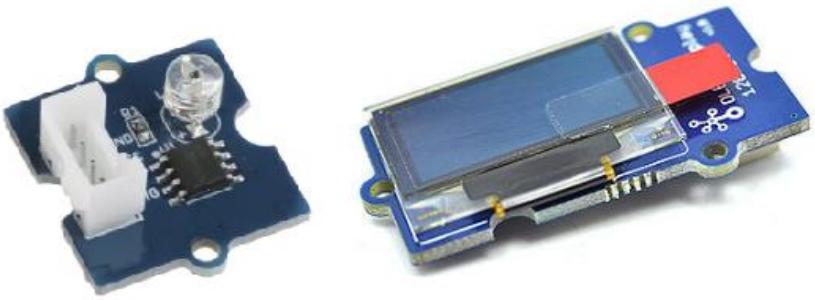
Nuestro ejemplo nos será útil para poder medir la cantidad de luz del medio ambiente y mostrar su valor en el display OLED. Nos basaremos en los anteriores ejemplos en los que hemos usado las librerías para representación numérica

Funcionamiento:

La descripción del funcionamiento ya se ha explicado en la anterior practica

Entradas salidas:

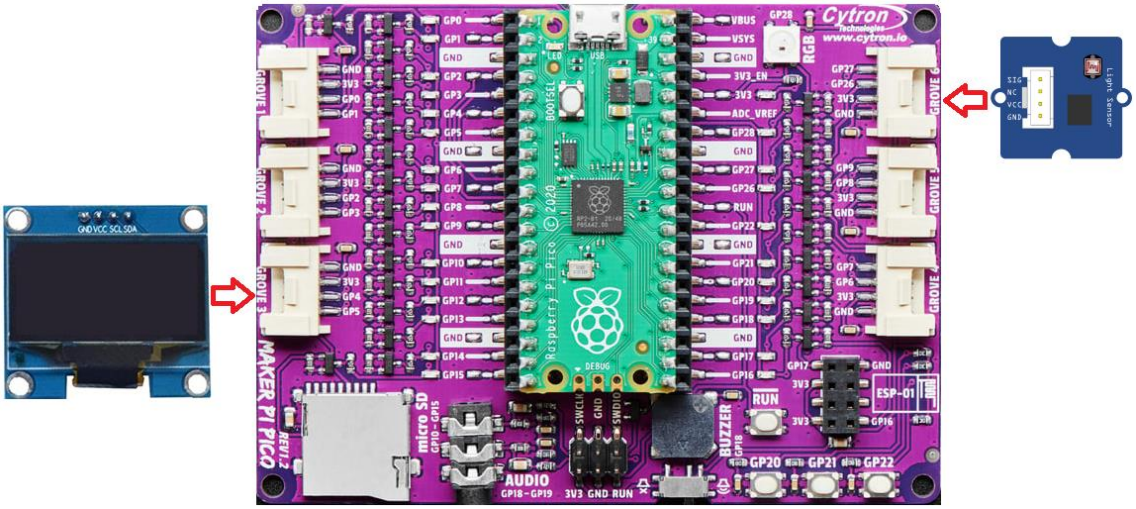
Los dispositivos que usaremos son los de la imagen



Los pines de conexión son los siguientes:

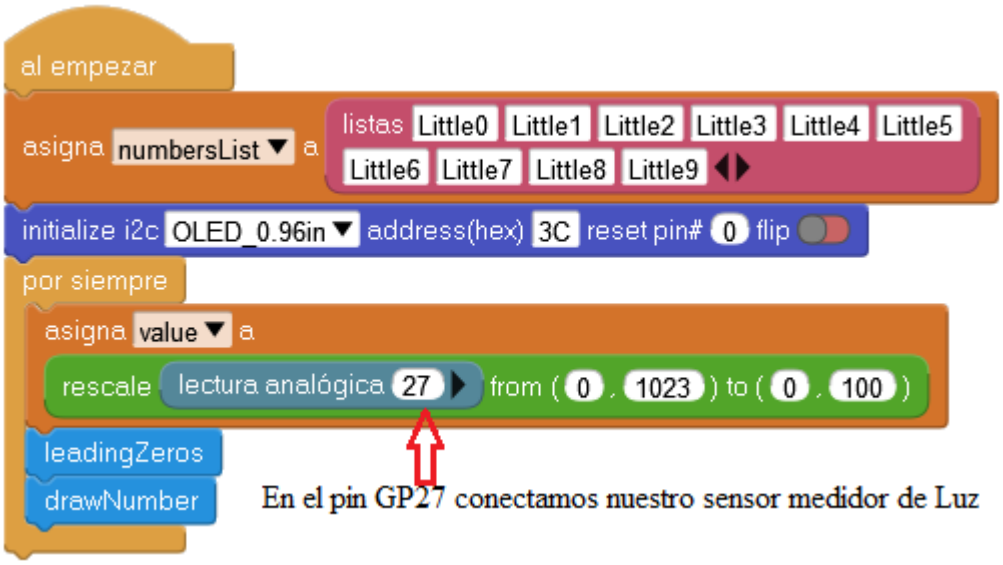
- **GP27** Sensor medidor de Luz
- **GP4 SDA** del dispositivo **OLED**
- **GP5 SCL** del dispositivo **OLED**

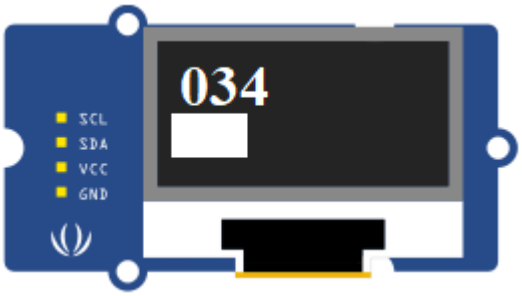
Esquema de montaje:



Programa:

El programa es similar al del ejemplo anterior. La única diferencia es que sustituimos el potenciómetro por un sensor.





6.23. Lotería

Objetivo

Realizar una aplicación que simule una "**lotería**" en la que se establezcan una serie de premios y en cada jugada saldrá de manera aleatoria un premio al pulsar un Botón

Funcionamiento:

Se trata de crear una lista con 6 cifras, cada una de las cuales viene a representar un premio: **Lista [0, 200, 400, 1000, 0, 5000]**

Debemos sacar de esa lista un elemento cuando se pulse un Botón que conectaremos en el pin **GP1** de nuestra tarjeta.

Una vez pulsado el Botón se genera el número y se mostrara en la pantalla **OLED**, tal como vemos en la imagen.

Entradas salidas:

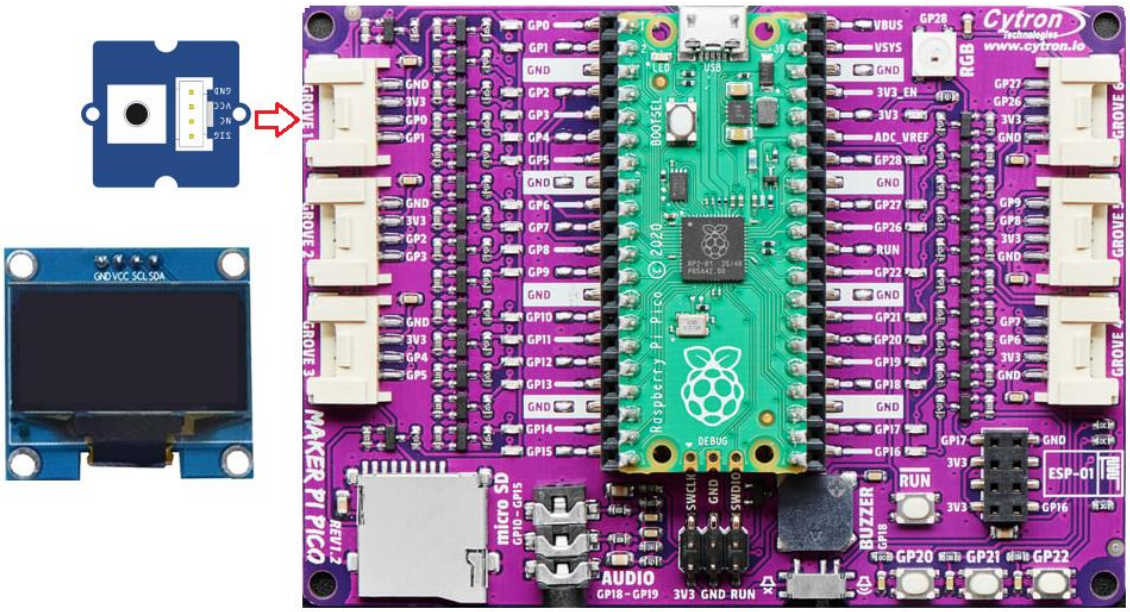
Los elementos que usaremos son los que se muestran.

Estos elementos se conectarán en los pines siguientes

- **GP1** Botón
- **GP4** Pin **SDA** del **OLED**
- **GP5** PIN **SCL** del **OLED**

Esquema de montaje:

La siguiente imagen representa el montaje que usaremos



Programa:

El proceso a seguir para confeccionar el programa es el siguiente.

1. Creación de la variable "**Premios**" que se asociara a la lista de números.
2. Creación de la variable "**Premio**" que representará uno de los elementos de la lista y que se asociará al valor leído en la lista de forma aleatoria que mostraremos en la pantalla **OLED**.
3. Al comenzar declaramos el display **OLED** , escribimos en la pantalla el texto "**LOTERIA**" y definimos la lista de premios llamada "**Premios**"
4. Seguidamente usamos un bloque de tipo "**cuando**" que se activara con la señal procedente del pin **GP1** en el que tenemos el Botón. Este bloque desencadena la generación de un numero aleatorio que se asigna a al parámetro que indica el elemento a extraer de la lista de **Premios** y que se asigna a la variable "**Premio**".

Actividades de Ampliación

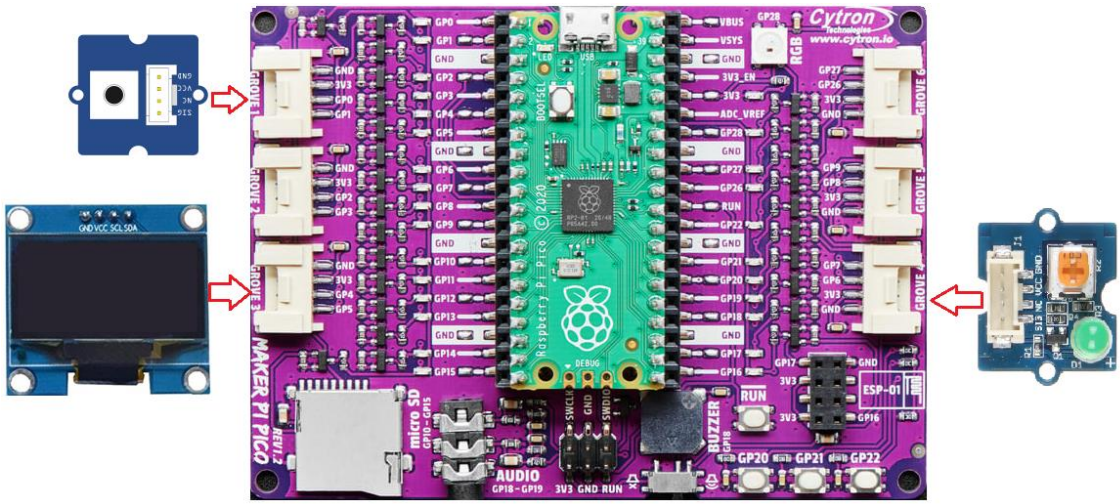
1. Crear un programa en el que dada una lista de 4 números saquemos uno al azar y comprobemos si se trata de uno de los elementos guardados (en nuestro caso el numero 8)

Nuestro programa se creará de acuerdo al siguiente orden de tareas:

1. Creación de la variable "**Lista**" que se asociara a la lista de números.
2. Creación de la variable "**Numero**" que representará uno de los elementos de la lista y que se asociará al valor del premio que mostraremos en la pantalla **OLED**.
3. Al comenzar declaramos el display **OLED** y definimos la lista de premios
4. Seguidamente usamos un bloque de tipo "**cuando**" que se activara con la señal procedente del pin **GP1** en el que tenemos el Botón. Este bloque desencadena la generación de un numero aleatorio que se asigna a al parámetro que indica el elemento a extraer de la "**Lista**" y que se asigna a la variable "**Numero**".

5. Seguidamente se plantea un condicional "**si .. si no.**." que testeará si el número premiado se corresponde con el numero "**8**" que hemos seleccionado como el premio:
    - Si **Numero=8.** Escribe en **OLED** "***HAS GANADO***", muestra sobre el bloque en el entorno **MicroBlocks** el texto "***HAS GANADO***" y activa la salida pin **GP7** que tiene el LED indicador de premio.
    - Si **Numero no es 8.** Escribe en **OLED** "***HAS PERDIDO***", muestra sobre el bloque en el entorno **MicroBlocks** el texto "***HAS PERDIDO***" y desactiva la salida pin **GP7**

Este sería el montaje



Este sería el programa

6.24. PWM Potenciómetro-LED

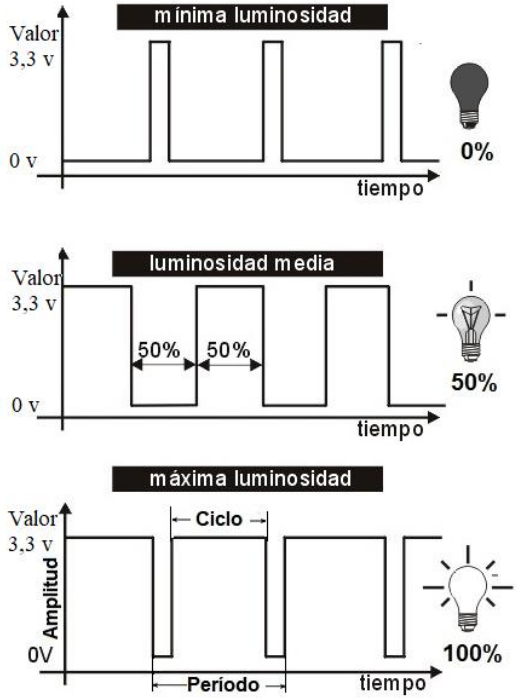
Objetivo

Con esta práctica, aprenderemos sobre **PWM**, es decir, modulación de ancho de pulso en Raspberry Pi Pico. Cambiaremos gradualmente la luminancia de un LED a través de la programación. Conseguiremos este efecto con modulación de ancho de pulso (PWM).

Funcionamiento:

La modulación de ancho de pulso o PWM es una técnica común utilizada para variar el ancho de los pulsos en un tren de pulsos. PWM tiene muchas aplicaciones, como controlar servos y controladores de velocidad, limitando la potencia efectiva de motores y LED.

Nuestro ejercicio consiste en habilitar un pin de Raspberry Pi Pico como salida PWM y depositar en el un valor que permitirá cambiar el valor medio de salida.



La función para generar una señal **PWM** desde el software **MicroBlocks** se encuentra en la librería "**Pines**".

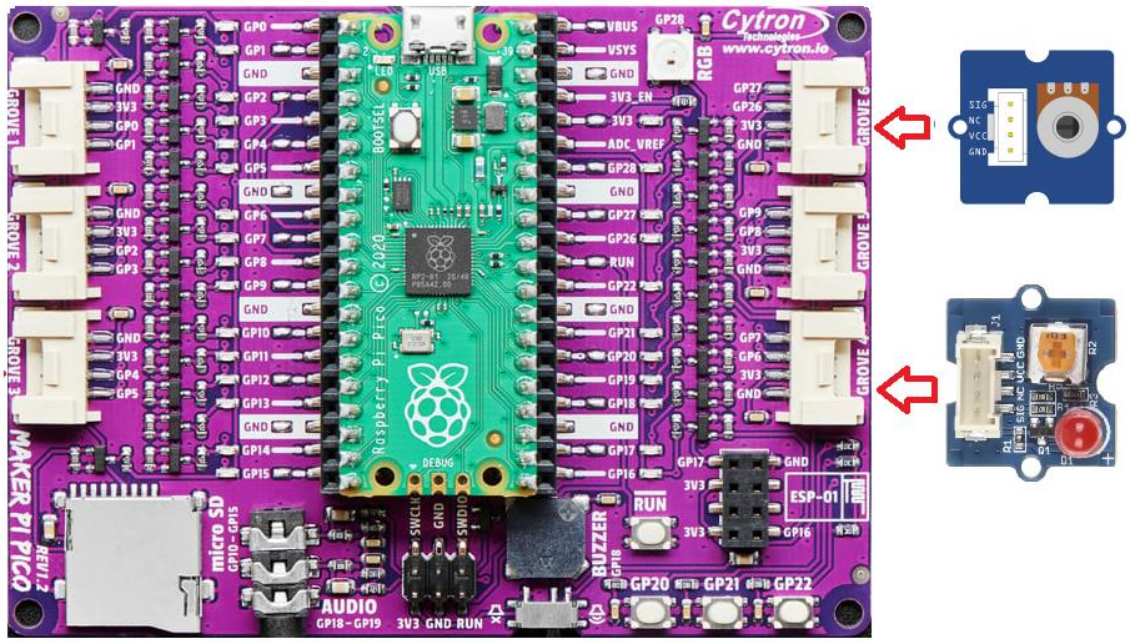
Genera una señal de modulación de ancho de pulso (PWM) en el pin dado que se aproxima a un nivel de potencia de 0-1023. PWM funciona encendiendo y apagando el pin rápidamente. La potencia se controla variando el ciclo de trabajo: el porcentaje de tiempo durante cada ciclo que el pin está encendido. Un valor de 0 significa que el pin está apagado, mientras que un valor de 1023 significa potencia máxima (es decir, el pin está encendido el 100 % del tiempo). Un valor de 512 da como resultado un ciclo de trabajo del 50 %; el pin está encendido la mitad del tiempo y apagado la mitad del tiempo. PWM se puede utilizar para controlar el brillo de un LED o la velocidad de un motor.

Algunas placas tienen uno o dos pines equipados con convertidores de digital a analógico (DAC). Cuando este bloque se usa con dichos pines, el pin emite un voltaje constante en lugar de una señal PWM. Un valor de cero significa 0 voltios, un valor de 1023 significa el voltaje total de la fuente de alimentación (3,3 V o 5 voltios) y 512 es la mitad del voltaje total.

Entradas salidas:

- **GP7** LED de salida PWM
- **GP27** Potenciómetro selector de valor de salida PWM

Esquema de montaje:



Programa:

El programa siempre está ejecutándose por ello hemos seleccionado el bloque de inicio "**cuando**" y seleccionando el valor de disparo de este bloque con un valor "1" permanente.

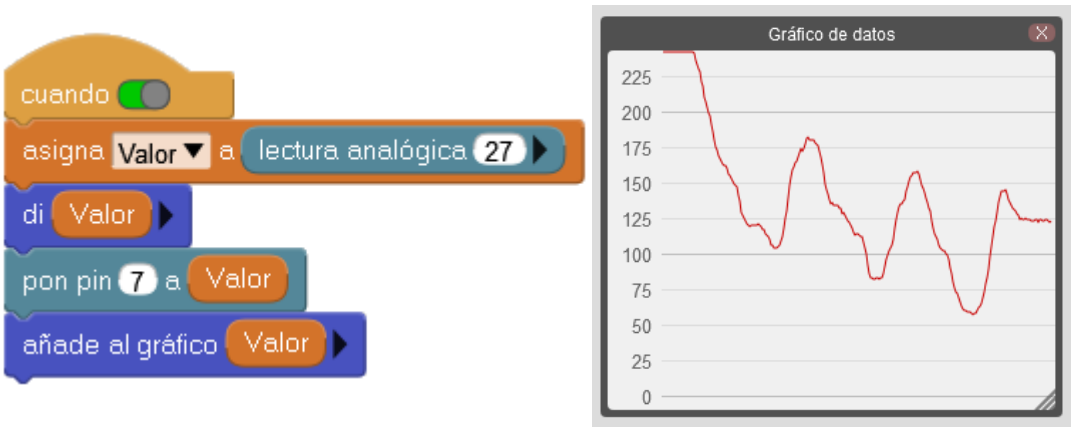
Cuando el bloque "cuando" se está ejecutando, verifica repetidamente una condición booleana. Cuando la condición se vuelve verdadera, se ejecutan los bloques debajo del sombrero. Si la condición sigue siendo verdadera al final de la ejecución, los bloques se ejecutarán nuevamente y ese proceso se repetirá hasta que la condición se vuelva falsa.

El programa consiste en la creación de una variable a la que llamaremos "**Valor**" y le asignaremos el valor leído en el pin **GP27** que como sabemos es una entrada analógica

Seguidamente escribiremos el "**Valor**" en la propia pantalla de **MicroBlocks** cuando este se encuentra en modo "**Run**"

A continuación, llevamos el "**Valor**" al pin **GP7** con el bloque "**pon pin 7** *valor*" que veremos cambiar su iluminación.

Seguidamente con la función "**añade al gráfico** *valor*" mostramos en la ventana "**Gráfico de datos**" el grafico de variación de la variable "**Valor**"



6.25. Sensor de Presencia

Objetivo

Con esta aplicación vamos a probar el funcionamiento del sensor de movimiento o presencia conectado a la tarjeta de experimentación.

Funcionamiento:

El sensor de movimiento PIR ajustable es un sensor de movimiento infrarrojo pasivo fácil de usar, que puede detectar movimiento de objetos infrarrojos de hasta 3 metros. Cualquier objeto de infrarrojos se mueve en su rango de detección, el sensor emite UNA señal ALTA en su pin SIG. Además, puede ajustar el tiempo ALTO SIG hasta 130s a través del potenciómetro, además, puede ajustar el rango de detección a través del otro potenciómetro.

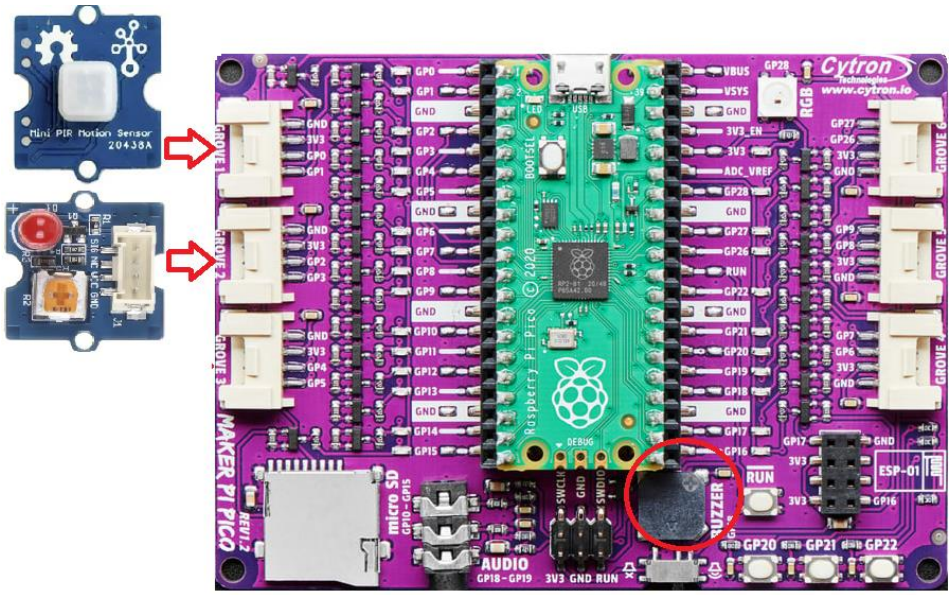
Entradas salidas:

Los dispositivos que usaremos son

Los pines de conexión son:

- **GP1** **Sensor PIR** de movimiento
- **GP3** Diodo **LED** de alarma
- **GP18** **Buzer** integrado en la tarjeta

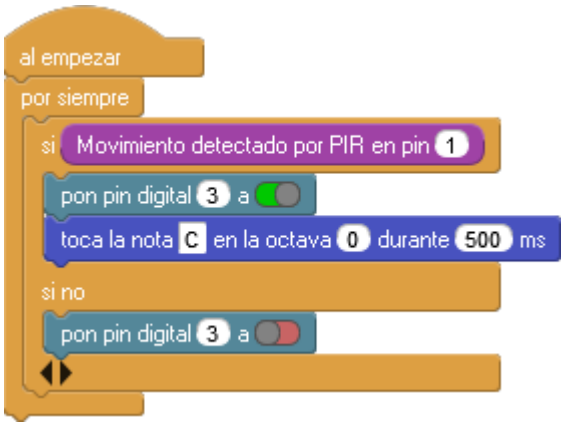
Esquema de montaje:



Programa:

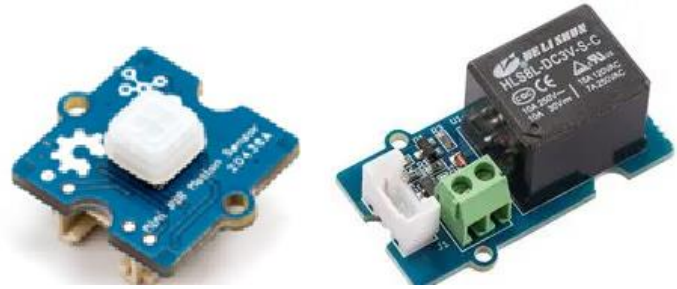
Nuestro programa será el siguiente

- Utilizaremos las librerías "**IR Motion (PIR)**" y "**Tonos**"
- Designaremos la ubicación el Buzer en el pin **GP18** que es el pin que tiene designado la tarjeta.
- Dentro de un bloque "**por siempre**" colocaremos un condicional cuya parte de condición será la señal que genera el detector PIR de movimiento (de tipo digital), colocado en el pin **GP1** que se pondrá en valor "1" cuando se detecta un movimiento.
- Seguidamente dentro de este condicional doble se coloca en la primera parte un bloque de activación del LED indicador de alarma de movimiento que se conecta en el pin **GP3** seguido de un bloque de emisión de tono "**toca la nota** *C* **en la octava** *0* **durante** *500* **ms**"
- En la parte "si no" del condicional debemos apagar el LED conectado en pin GP3

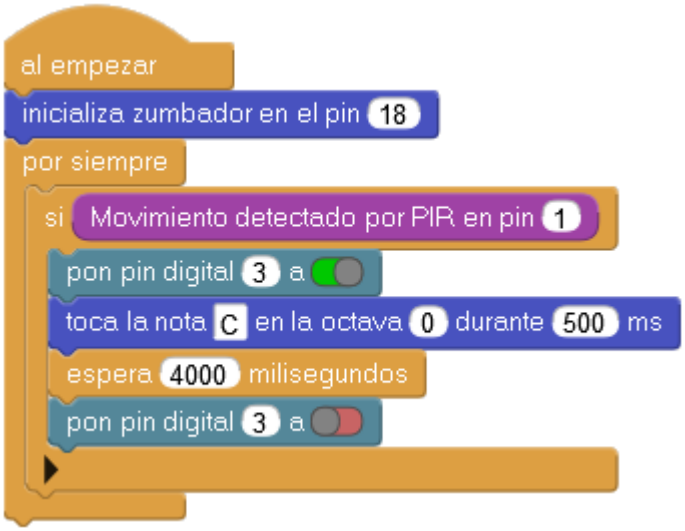


Actividades de Ampliación

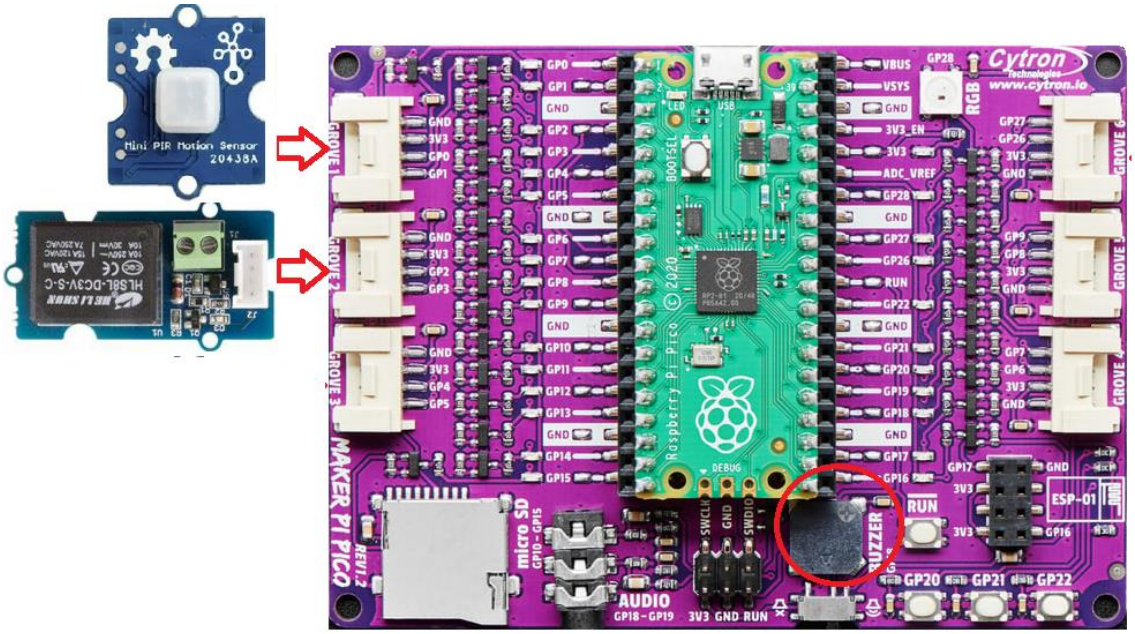
1. Queremos construir una aplicación que en una habitación detecte el sensor movimiento y mantenga la iluminación activada durante un tiempo de **4 segundos**. Usaremos una unidad de relé para activar la lampara de la habitación que conectaremos en el pin **GP3**. El sensor lo conectaremos en el pin **GP1**



Programa



Montaje



6.26. Termómetro

Objetivo

Mediremos el valor de la temperatura y lo mostraremos en un display OLED

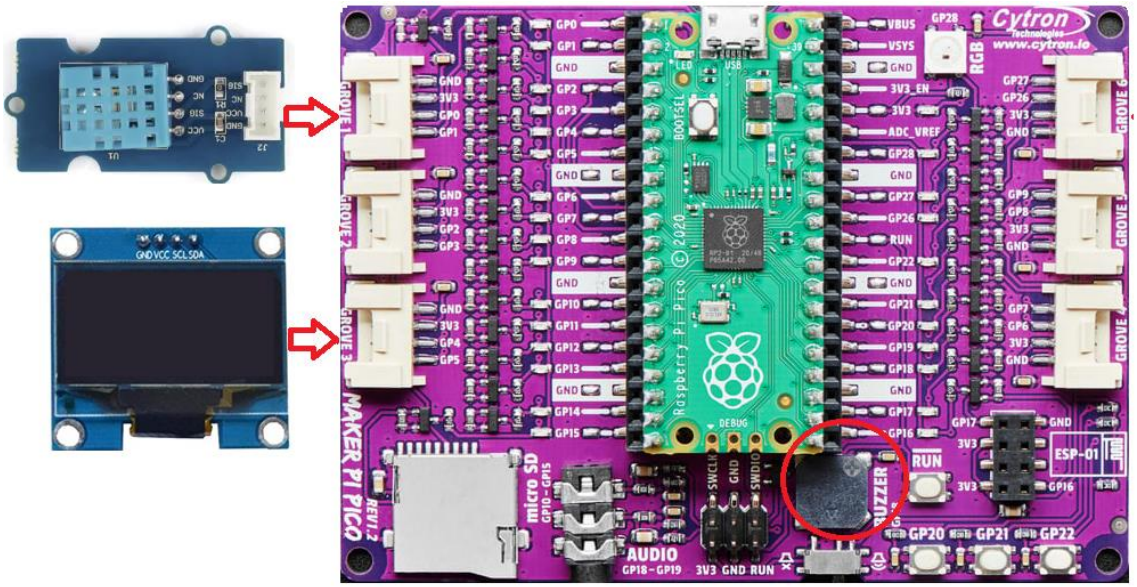
Funcionamiento:

Para medir con el sensor DHT11 usaremos un pin digital en modo entrada a través del cual se obtiene el valor de la temperatura y llevaremos este valor al display OLED

Entradas salidas:

- **GP1** **Sensor DHT11**
- **GP4** Pin **SDA** del **OLED**
- **GP5** PIN **SCL** del **OLED**

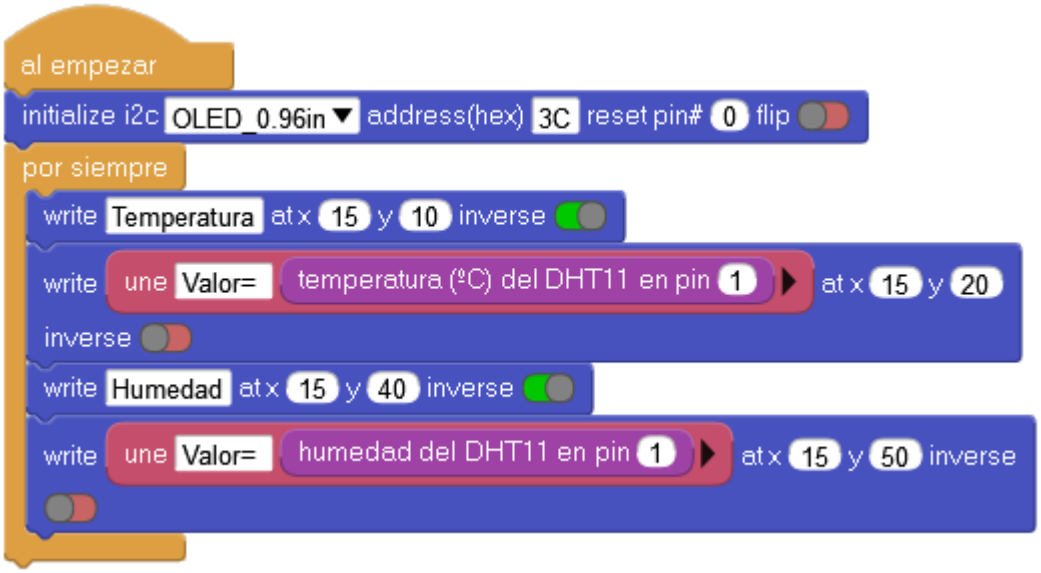
Esquema de montaje:



Programa:

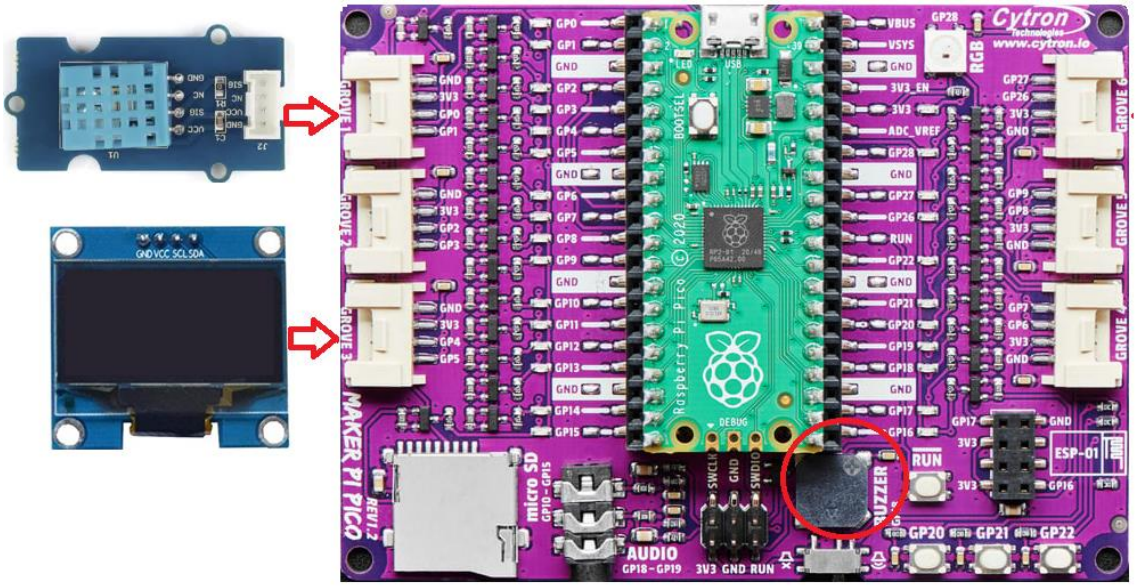
Actividades de Ampliación

1. Aprovechar la capacidad del sensor **DHT11** de medir la humedad para mostrar en el OLED ambas magnitudes, **temperatura y humedad**.
   Se trata de incluir la información de la humedad que recogemos mediante el bloque e incorporarlo en el display **OLED**




GND VCC SCL SDA
Temperatura
Valor= 18
Humedad
Valor= 50

Montaje


GND VCC SCL SDA
MAKER PI PICO
Raspberry Pi Pico
Cytron
RGB
BUZZER
AUDIO
micro SD
ESP-01
RUN
DEBUG
GROVE 1
GROVE 2
GROVE 3
GROVE 4
GROVE 5
GROVE 6

6.27. Trazador de Datos

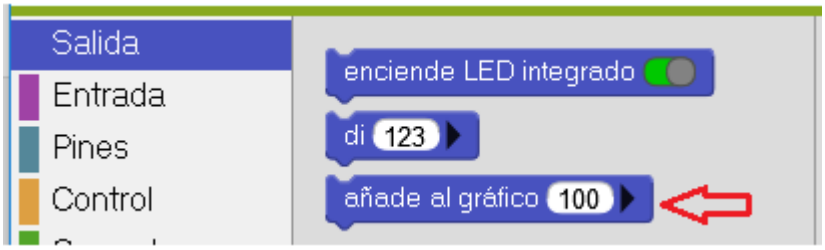
Objetivo

Con este ejemplo vamos a recoger un valor de señal analógica en el pin GP27 mediante la conexión de un potenciómetro. El valor recogido lo vamos a mostrar por un lado en la pantalla Oled y por otro en el propio software **MicroBlocks** mediante la opción "Grafico"

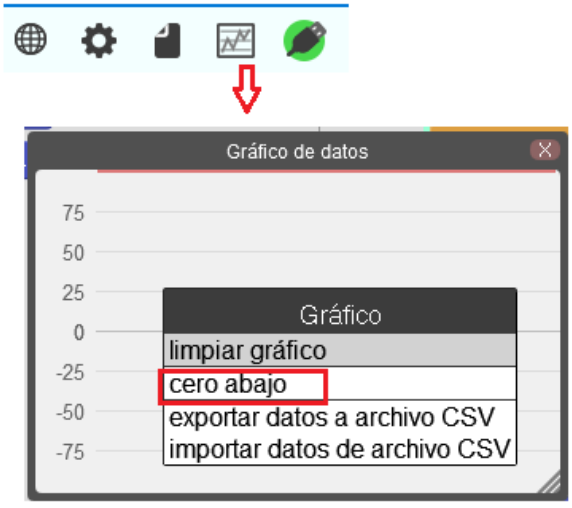
Funcionamiento:

Para realizar esta aplicación usaremos la señal del pin GP27 que oscilará entre 0 y 1023 y este valor lo asociaremos a una variable que debemos crear y a la que llamaremos "**valor**" y llevaremos al display **OLED**.

La opción para trazar el valor en la pantalla del PC se realizará con la ayuda de la instrucción "añade al gráfico". La recogida del valor la haremos cada 10 ms por lo que en el bucle de lectura colocaremos un retardo. Esto es bueno hacerlo para evitar que colapse el algoritmo por velocidad.



Para el trazado grafico seleccionamos pulsando en el botón correspondiente y marcamos la opción " cero **abajo**" para que el trazado sea de los valores positivos ya que la lectura del puerto **GP27** no da valores negativos.



Si lo deseamos también podemos exportar los valores en formato CSV para abrirlos con una hoja de cálculo.

Para ver los datos podemos abrir la ventana del trazador, pero al ser el valor más elevado 1023 de nos sale de la ventana. Debemos ajustar el tamaño y las variaciones para poder verlas. Podríamos también realizar un "**escalado**" de la señal llevándola desde el rango 0 a 1023 al rango 0 a 100 haciendo uso de la instrucción "**rescale**" que está en la librería de

"Operadores" teniendo habilitada la opción de "muestra los bloques avanzados" que se selecciona en

Asi quedaría el valor escalado a colocar en la instrucción de trazado

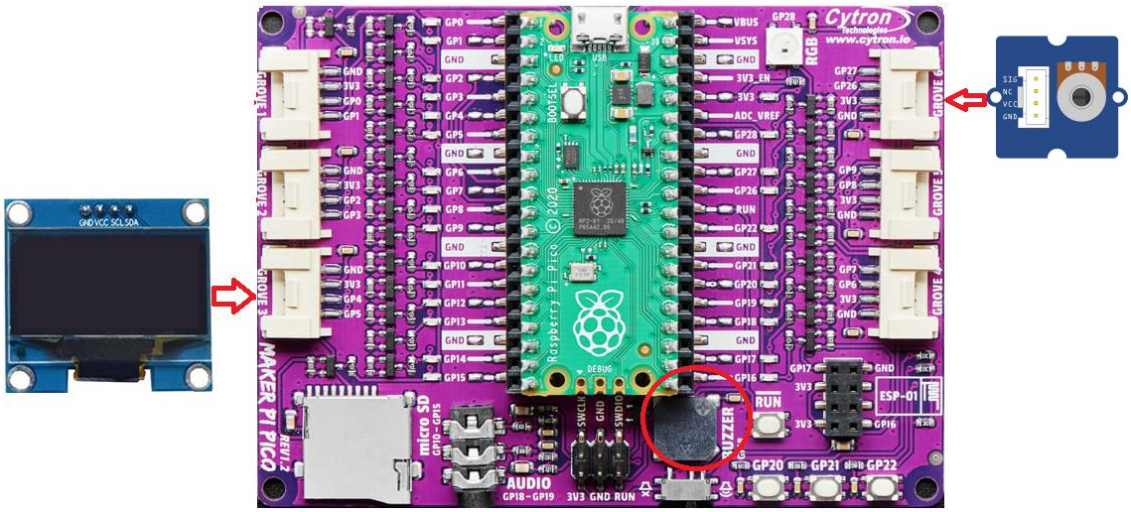
Entradas salidas:

Utilizaremos los siguientes dispositivos conectados a nuestra tarjeta.

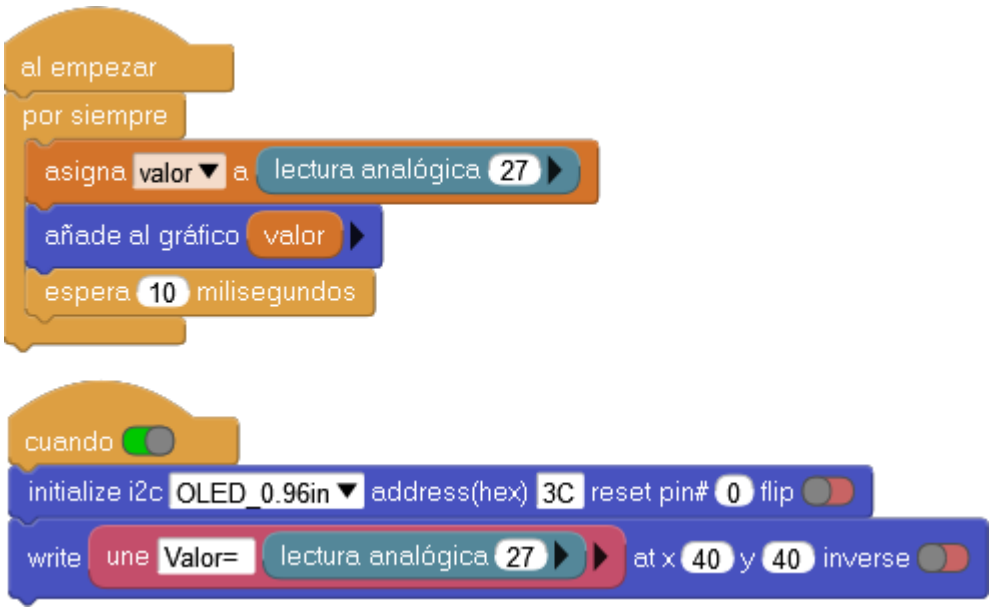
Las conexiones serán:

- GP27 Potenciómetro
- GP4 Pin SDA del dispositivo OLED
- GP5 Pin SCL del dispositivo OLED

Esquema de montaje:

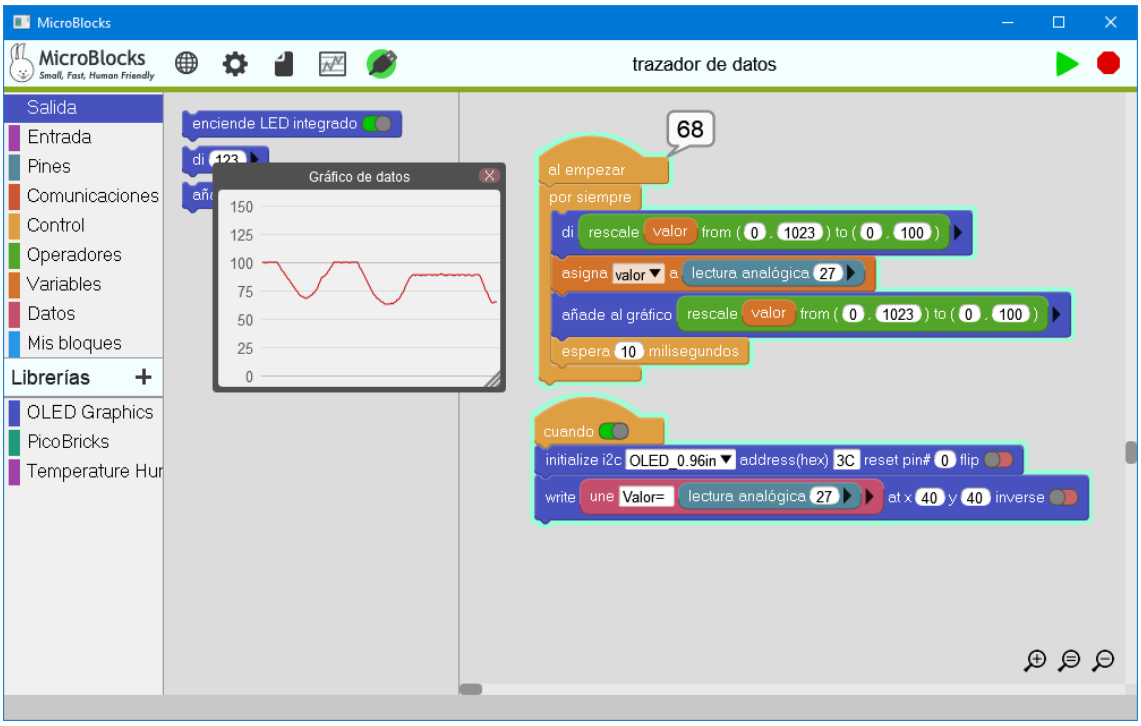


Programa:

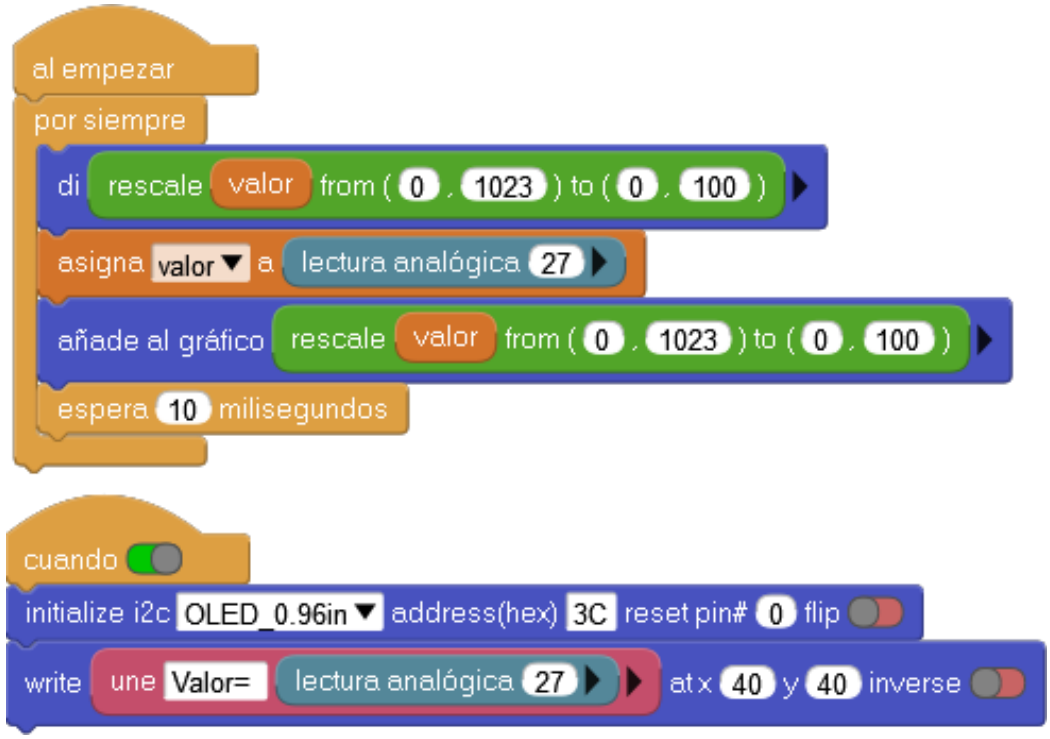


Actividades de ampliación

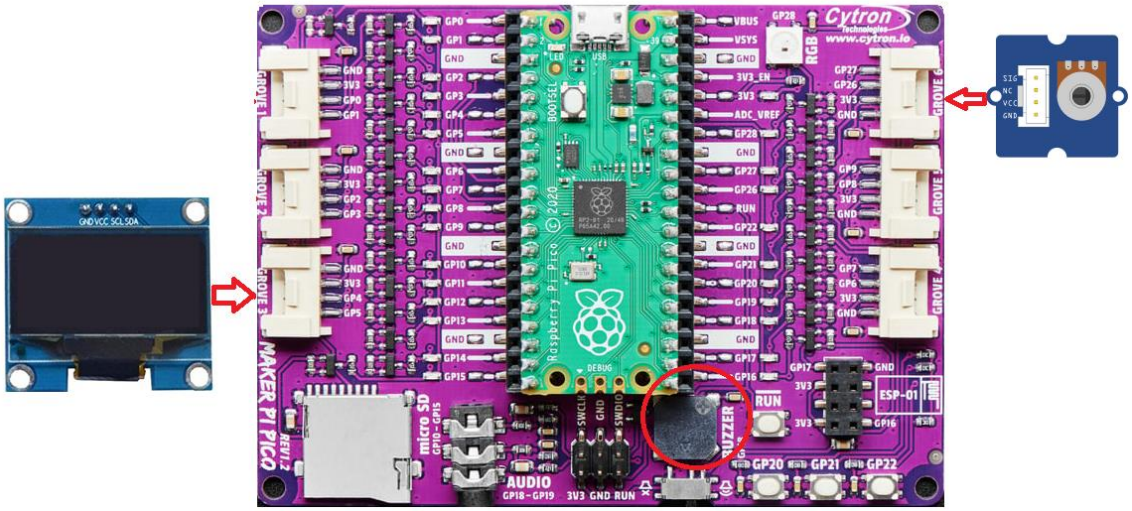
1. Realizar el escalado de la señal para que se pueda ver bien. Pasar a la escala0 a 100 y a la vez queremos que se muestre el valor en cada instante, para lo que recurriremos a la instrucción “di” de la librería “Salida



En la siguiente imagen se muestra el programa con las modificaciones propuestas.



Montaje


GND VCC SCL SDA
MAKER PI PICO
Cytron
www.cytron.io
GROVE 1
GROVE 2
GROVE 3
GROVE 4
GROVE 5
GROVE 6
GROVE 7
RGB
BUZZER
RUN
ESP-01
micro SD
AUDIO
GP18 - GP19
3V3 GND RUN
GP20 GP21 GP22
SIG
NC
VCC
GND

6.28. Realización de un ejemplo usando politonos

Objetivo

Con esta práctica queremos abordar el tratamiento del sonido (politonos) por parte de nuestra tarjeta.

Funcionamiento:

Explicación complementaria sobre el manejo de los ficheros aceptados por la librería de "Politonos" de MicroBlocks.

Usando Politonos

Unas explicaciones previas sobre el formato de politono RTTTL que usa la librería "Politonos" de Microblock

Lenguaje de transferencia de texto de tonos de llamada (RTTTL)

Información (ENG):

Ring Tone Text Transfer Language (**RTTTL**) fue desarrollado por Nokia para ser utilizado para transferir tonos de llamada al teléfono celular por Nokia.

El formato RTTTL es una cadena dividida en tres secciones: nombre, valor predeterminado y datos.

La sección de nombre consiste en una cadena que describe el *nombre* del tono de llamada. No puede tener más de 10 caracteres y no puede contener dos puntos ":". (Sin embargo, dado que la especificación Smart Messaging permite nombres de hasta 15 caracteres de longitud, algunas aplicaciones que procesan RTTTL también lo hacen).
La sección de valor predeterminado es un conjunto de valores separados por comas, donde cada valor contiene una clave y un *valor* separados por un *carácter* =, que describe ciertos valores predeterminados que deben cumplirse durante la ejecución del tono de llamada. Los nombres posibles son

- **D**. Duración
- **o** - octava
- **B** – ritmo, tempo

La sección de *datos* consiste en un conjunto de cadenas de caracteres separadas por comas, donde cada cadena contiene una duración, *tono*, *octava* y *puntos* opcionales (lo que aumenta la *duración* de la nota a la mitad).

El formato de la notación RTTTL es similar al lenguaje de macros de música que se encuentra en las implementaciones BASIC presentes en muchas de las primeras microcomputadoras.

Especificación técnica

Para ser reconocido por los programas de tonos de llamada, un tono de llamada de formato RTTTL / Nokring debe contener tres elementos específicos: nombre, configuración y notas.

Por ejemplo, aquí está el tono de llamada RTTTL para Haunted House:

```
1HauntHouse: d=4,o=5,b=108: 2a4, 2e, 2d#, 2b4, 2a4, 2c, 2d, 2a#4, 2e., e, 1f4, 1a4, 1d#, 2e., d, 2c., b4, 1a4, 1p, 2a4,
 2e, 2d#, 2b4, 2a4, 2c, 2d, 2a#4, 2e., e, 1f4, 1a4, 1d#, 2e., d, 2c., b4, 1a4
```

Las tres partes están separadas por dos puntos.

- **Parte 1**: nombre del tono de llamada (aquí: "HauntHouse"), una cadena de caracteres representa el nombre del tono de llamada
- **Parte 2**: configuración (aquí: d = 4, o = 5, b = 108), donde "d =" es la duración predeterminada de una nota. En este caso, el "4" significa que cada nota sin especificador de duración (ver más abajo) se considera por defecto una nota de cuarto. "8" significaría una octava nota, y así sucesivamente. En consecuencia, "o =" es la octava predeterminada. Hay cuatro octavas en el formato Nokring/RTTTL. Y "b=" es el tempo, en "pulsaciones por minuto".
- **Parte 3:** las notas. Cada nota está separada por una coma e incluye, en secuencia: un especificador de duración, una nota musical estándar, ya sea a, b, c, d, e, f o g, y un especificador de octava. Si no hay ningún especificador de duración u octava, se aplica el valor predeterminado.

Duraciones

Las duraciones musicales estándar se denotan con las siguientes notaciones:

- **1** – nota entera
- **2** – media nota
- **4** – Nota de cuarto
- **8** – Octava nota
- **16** – Decimosexta nota
- **32** – Trigésimo segundo Nota

Los patrones de ritmo punteados se pueden formar añadiendo un carácter de punto (".") al final de un elemento de duración/compás/octava.

Tonos

- **P** – descanso o pausa
- A – **A**
- **A#** – A♯ / B♭
- B – **B** / C♭
- C – **C**
- **C#** – C♯/D♭
- D – **D**
- **D#** – D♯ / E♭
- E – **E** / F♭
- F – **F** / E♯
- **F#** – F♯ / G♭
- G – **G**
- **G#** – G♯ / A♭

Octava

El formato RTTTL permite octavas comenzando desde la A por debajo del do medio y subiendo cuatro octavas. Esto se corresponde con la incapacidad de los teléfonos celulares para reproducir ciertos tonos audiblemente. Estas octavas están numeradas desde el tono más bajo hasta el tono más alto de 4 a 7.

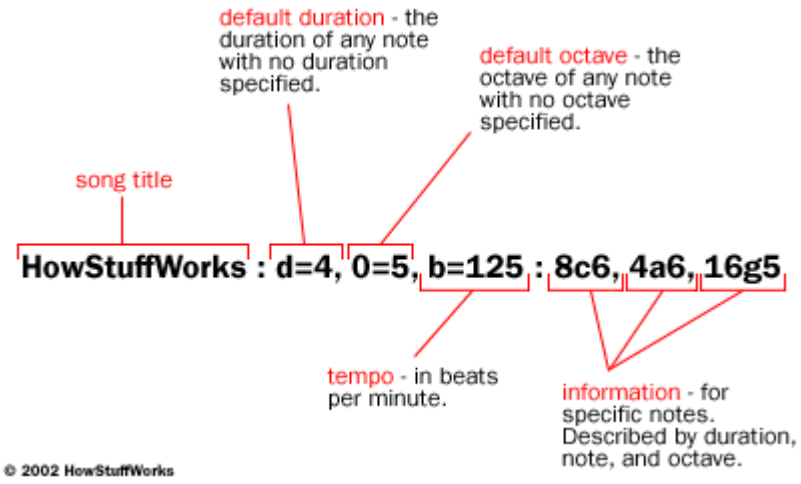
La octava debe dejarse fuera de la notación en el caso de un descanso o pausa en el patrón.

Ejemplo

Un ejemplo del formato RTTTL sería:

fifth:d=4,o=5,b=63:8P,8G5,8G5,8G5,2D#5

Fuente: Wikipedia



Ejemplos de melodías

The_Simpsons:d=4,o=5,b=160:c.6,e6,f#6,8a6,g.6,e6,c6,8a,8f#,8f#,8f#,2g,8p,8p,8f#,8f#,8f#,8g,a#.,8c6,8c6,8c6,c6

Indiana:d=4,o=5,b=250:e,8p,8f,8g,8p,1c6,8p.,d,8p,8e,1f,p.,g,8p,8a,8b,8p,1f6,p,a,8p,8b,2c6,2d6,2e6,e,8p,8f,8g,8p,1c6,p,d6,8p,8e6,1f.6,g,8p,8g,e.6,8p,d6,8p,8g,e.6,8p,d6,8p,8g,f.6,8p,e6,8p,8d6,2c6

TakeOnMe:d=4,o=4,b=160:8f#5,8f#5,8f#5,8d5,8p,8b,8p,8e5,8p,8e5,8p,8e5,8g#5,8g#5,8a5,8b5,8a5,8a5,8a5,8e5,8p,8d5,8p,8f#5,8p,8f#5,8p,8f#5,8e5,8e5,8f#5,8e5,8f#5,8f#5,8f#5,8d5,8p,8b,8p,8e5,8p,8e5,8p,8e5,8g#5,8g#5,8a5,8b5,8a5,8a5,8a5,8e5,8p,8d5,8p,8f#5,8p,8f#5,8p,8f#5,8e5,8e5

Mas ejemplos:

- Zip file of Mixed Tunes 1 (450 tunes)
- Zip file of Mixed Tunes 2 (375 tunes)
- Zip file of Mixed Tunes 3 (10,000 tunes)
- Zip file of TV Theme Tunes (50 tunes)
- Zip file of Christmas Tunes (70 tunes)

Explicación del montaje a realizar

Con los elementos de nuestra tarjeta, sin necesidad de otro dispositivo de E/S vamos a realizar nuestro montaje.

Básicamente usaremos las librerías:

Nuestra tarjeta tiene asignado el pin GP18 como salida de sonido incluyendo ya un dispositivo Buzer.

Vamos a utilizar los pines GP20, GP21 y GP22 que son entradas a las que ya en la tarjeta se han conectado 3 dispositivos Pulsadores.

LO que haremos será utilizar el bloque de función

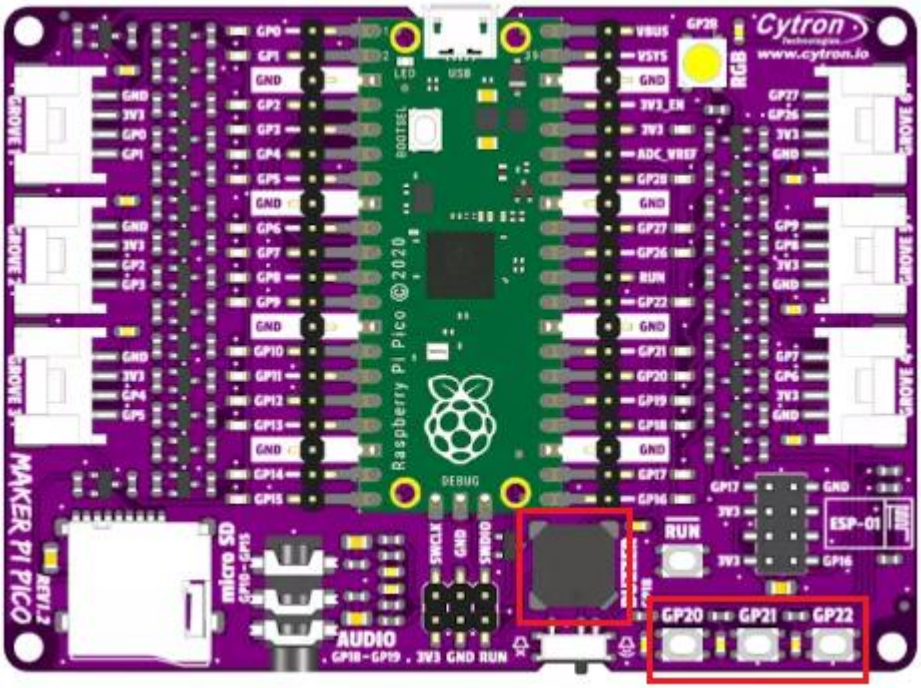
colocando como parámetro el pin 18.

Crearemos 4 nuevos bloques mediante el procedimiento explicado de “Mis bloques” que serán los que contendrán hasta cuatro melodías que hemos seleccionado para nuestra aplicación: Bolero, Contador, Rock, Para Elisa y Beatles.

Entradas salidas:

- **PG18** Buzer
- **PG20**, **PG21** y **PG22** Botones integrados en la tarjeta.

Esquema de montaje:



Programa:

Lo primero será cargar las librerías “Politonos “ y “Tonos”.

Seguidamente se crearán los bloques de usuario que contengan las melodías,

Seleccionamos de la librería "**Mis Bloques**" la opción "**Crear un bloque comando**"

Colocaremos dentro del bloque "toca el politono" la secuencia de la música escrita en el lenguaje de politonos que hemos explicado:

```
Bolero:d=4,o=5,b=80:c6,8c6,16b,16c6,16d6,16c6,16b,16a,8c6,16c6,16a,c6,8c6,
16b,16c6,16a,16g,16e,16f,2g,16g,16f,16e,16d,16e,16f,16g,16a,g,g,16g,16a,16b
,16a,16g,16f,16e,16d,16e,16d,8c,8c,16c,16d,8e,8f,d,2g
```

El resto de melodías es el que se muestra.

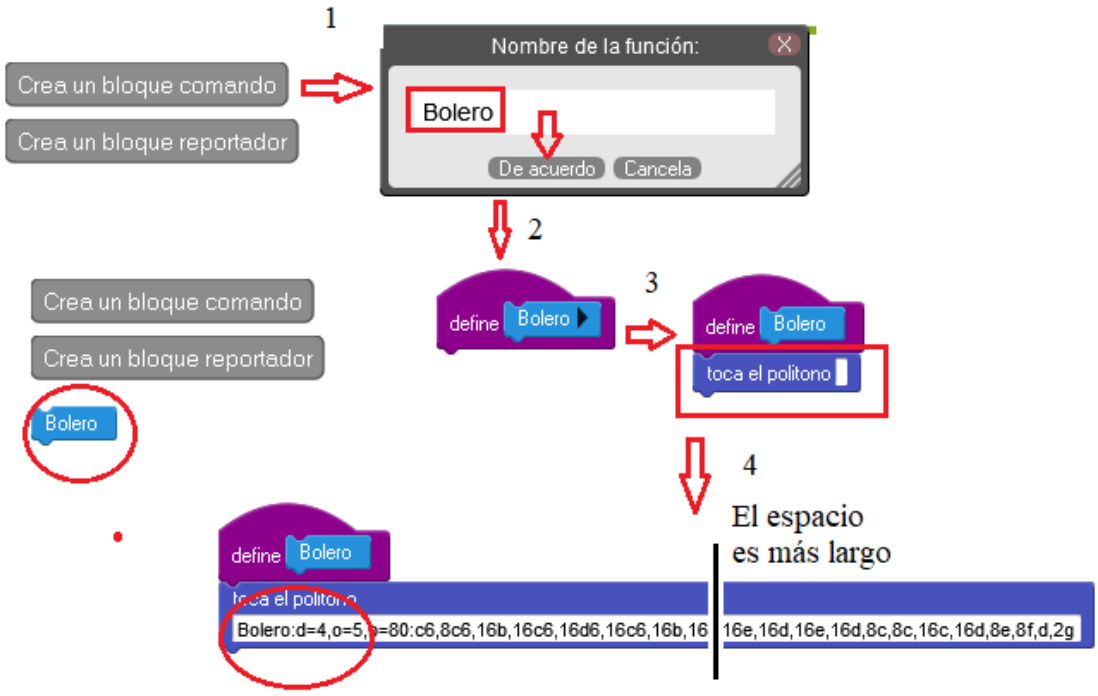
```
Beatles - Hard days night :
d=4,o=5,b=100:16c#6,16d6,16c#6,8e6,8p,8e6,16p,8e6,p,16e6,16e6,16d6,16e6,
8g6,8p,16e6,16d6,16e6,8c#6,
```

```
Countdown:d=4,o=5,b=125:p,8p,16b,16a,b,e,p,8p,16c6,16b,8c6,8b,a,p,8p,16c6
,16b,c6,e,p,8p,16a,16g,8a,8g,8f#,8a,g.,16f#,16g,a.,16g,16a,8b,8a,8g,8f#,e,c6,2b
.,16b,16c6,16b,16a,1b
```

```
Beethoven - Fur Elise :
d=4,o=5,b=140:8e6,8d#6,8e6,8d#6,8e6,8b,8d6,8c6,a,8p,8c,8e,8a,b,8p,8e,8g#,8b
,c6,p,8e,8e6,8d#6,8e6,8d#6,8e6,8b,8d6,8c6,a,8p,8c,8e,8a,b,8p,8e,8c6,8b,2a,
```

```
rock:d=4,o=6,b=112:32a#.6,32f.6,16c.6,32a#.6,32f.6,16c.6,32a#.6,32f.6,16c.6,32
a#.6,32f.6,16c.6,1g6,32a#.6,32f.6,16c.6,32a#.6,32f.6,16c.6,32a#.6,32f.6,16c.6,32
a#.6,32f.6,16c.6,1g6,32a#.6,32f.6,16d.6,32a#.6,32f.6,16d.6,32a#.6,32f.6,16d.6,32
a#.6,32f.6,16d.6,1g6,32a#.6,32f.6,16d.6,32a#.6,32f.6,16d.6,32a#.6,32f.6,16d.6,3
2a#.6,32f.6,16d.6,1g6,32f.6,32d.6,16a#.5,32f.6,32d.6,16a#.5,32f.6,32d.6,16a#.5,
32f.6,32d.6,16a#.5,1g6,32f.6,32d.6,16a#.5,32f.6,32d.6,16a#.5,32f.6,32d.6,16a#.5
,32f.6,32d.6,16a#.5,1g6,32f.6,32d#.6,16c.6,32f.6,32d#.6,16c.6,32f.6,32d#.6,16c.6
,32f.6,32d#.6,16c.6,1g6,32f.6,32d#.6,16c.6,32f.6,32d#.6,16c.6,32f.6,32d#.6,16c.6
,32f.6,32d#.6,16c.6,32g.6,
```

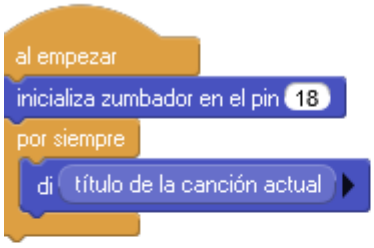
En la siguiente imagen se muestra el proceso a seguir para crear cada una de las melodías

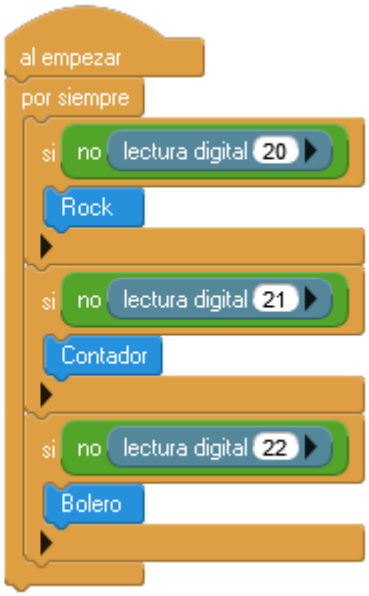


El montaje completo es el que se muestra a continuación.

Por un lado, establecemos un bloque de ejecución en el que se designa el pin de conexión del Buzer y además ponemos un bloque "**di**" que mostrara el título de la canción actual que se obtiene mediante el correspondiente bloque de la librería "**Politono**"

El bucle de ejecución de las distintas melodías politonales se monta usando tres bloques condicionales tipo “si” cuyas condiciones son la pulsación sobre los botones pin PG20, PG21 y PG22





6.29. Sistema de Riego

Objetivo

Aprovechando el dispositivo detector de humedad de Cytron vamos a realizar una sencilla practica en la que vamos a probar su funcionamiento y con ello poder descubrir las grandes posibilidades de este elemento.

Funcionamiento:

Nuestro dispositivo dispone de 4 terminales para su conexión en nuestros sistemas con Raspberry Pi Pico.

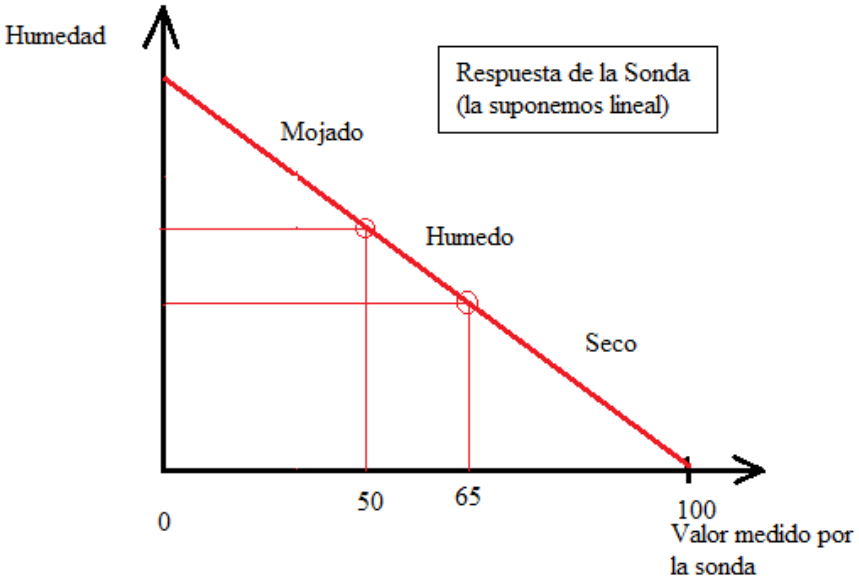
- GND conectable a GND de nuestra tarjeta
- VCC conectable a 3,3 v de nuestra tarjeta
- DIS seña de entrada para habilitar/deshabitar el dispositivo
- UOT señal de salida analógica que mide la humedad

Nuestra aplicación consistirá en colocar este dispositivo en el el conector correspondiente para que la señal OUT se corresponda con el GP27 de nuestra tarjeta.

Vamos a realizar el escalado de la señal que entrega este dispositivo para que la variación sea de 0 a 100.

Mostraremos el valor escalado en el display OLED y además queremos que cuando el grado de “sequedad” medido por la sonda sea >=65 se active una salida LED que conectaremos al pin GP1

Debemos tener en cuenta que el valor dado por la sonda es inversamente proporcional a la humedad, es decir que a mayor humedad menor es el valor dado por la sonda (la tierra está más mojada), por el contrario, cuando la tierra esta seca, el valor medido por la sonda es mayor.



Con los datos del grafico realizamos la programación de nuestro sistema. Queremos que se active el LED colocado en **GP1** cuando el terreno este "seco", es decir cuando el valor de la medida sea <=65. Para conseguirlo colocamos un bloque condicional "**si**.."

- Si **valor**(escalado) **<=65** activamos GP1 Activaríamos la bomba de riego (LED=1)
- En caso contrario desactivamos GP1 Pararíamos la bomba de riego (LED=0)

Los mensajes a colocar en el display OLED serían:

- Si **valor < 50 MOJADO**
- Si **valor >=50 Y <= 65 HUMEDO**
- Si **valor >= 65 SECO**

Entradas salidas:

Los dispositivos que usaremos son los de la figura siguiente.

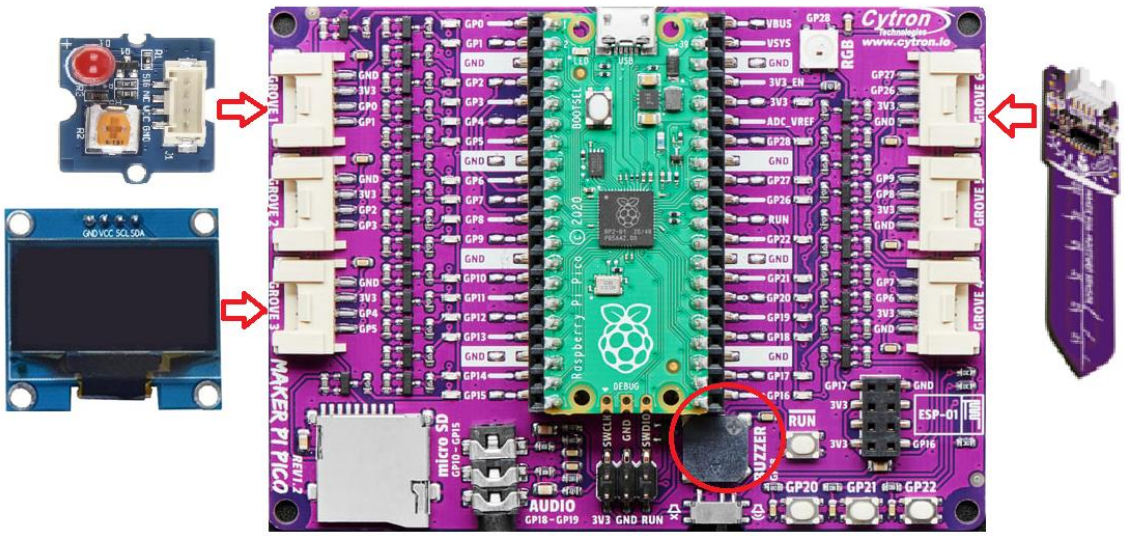
La designación de los pines será

- LED **GP1**
- SONDA **GP27**
- OLED **GP4 (SDA) y GP5 (SCL)**

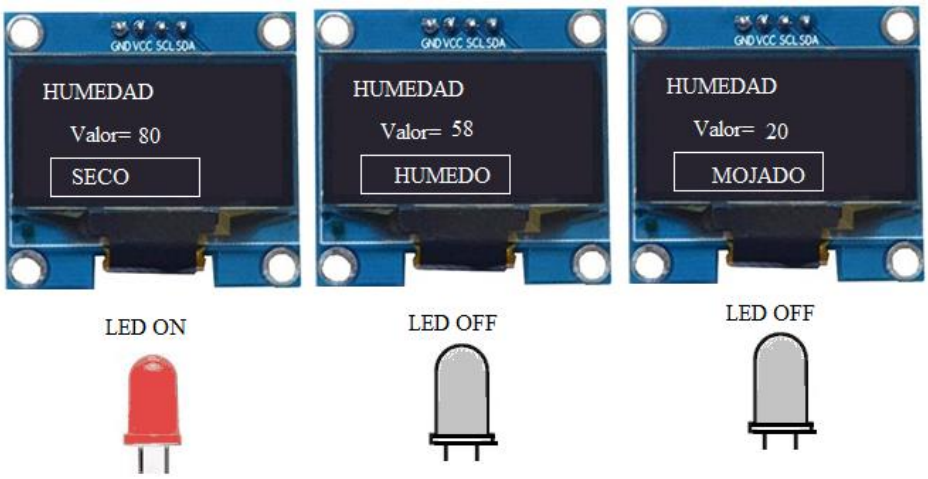
El pin de inhabilitación de la sonda DIS no lo usaremos.

Información sobre la SONDA DE HUMEDAD

Esquema de montaje:



Programa:



Pantallas durante la ejecución.

6.30. Mando por Infrarrojos

Objetivo

Este ejemplo nos servirá para manejar de manera básica el uso de un receptor de infrarrojos para con el poder activar y desactivar una salida digital.

Funcionamiento:

Un mando a distancia es un dispositivo de control que emplea un LED infrarrojo para enviar una señal al receptor. La señal puede ser detectada para controlar cualquier elemento de salida, relé, LED Motor, etc.

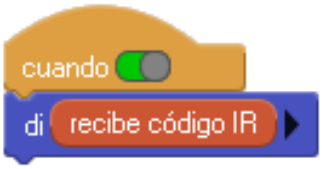
Un mando a distancia emplea un emisor de luz en el infrarrojo cercano, invisible para el ojo humano, pero que puede ser captado con facilidad por un receptor infrarrojo. El mando presenta una serie de teclas, cada una de las cuales emite un código numérico distinto.

El receptor IR (infrarrojo) se utiliza para recibir señales infrarrojas y también se utiliza para la detección de control remoto. El detector IR tiene un demodulador en el interior que busca IR modulado a 38 kHz. El receptor infrarrojo puede recibir señales dentro de 10 metros. El receptor infrarrojo puede recibir señales dentro de 10 metros. Si supera los 10 metros, es posible que el receptor no reciba las señales. A menudo usamos las dos ramitas: el receptor infrarrojo y el emisor infrarrojo para trabajar juntos.

En nuestro ejemplo queremos encender y apagar un LED que conectaremos en el pin GP3 usando los códigos del mando correspondiente a:

- Activar el LED Código de la Tecla 1 código (22)
- Apagar el LED Código de la Tecla 2 código (25)

NOTA: tenemos que averiguar en cada mando los códigos de cada tecla, para lo cual basta montar el siguiente programa

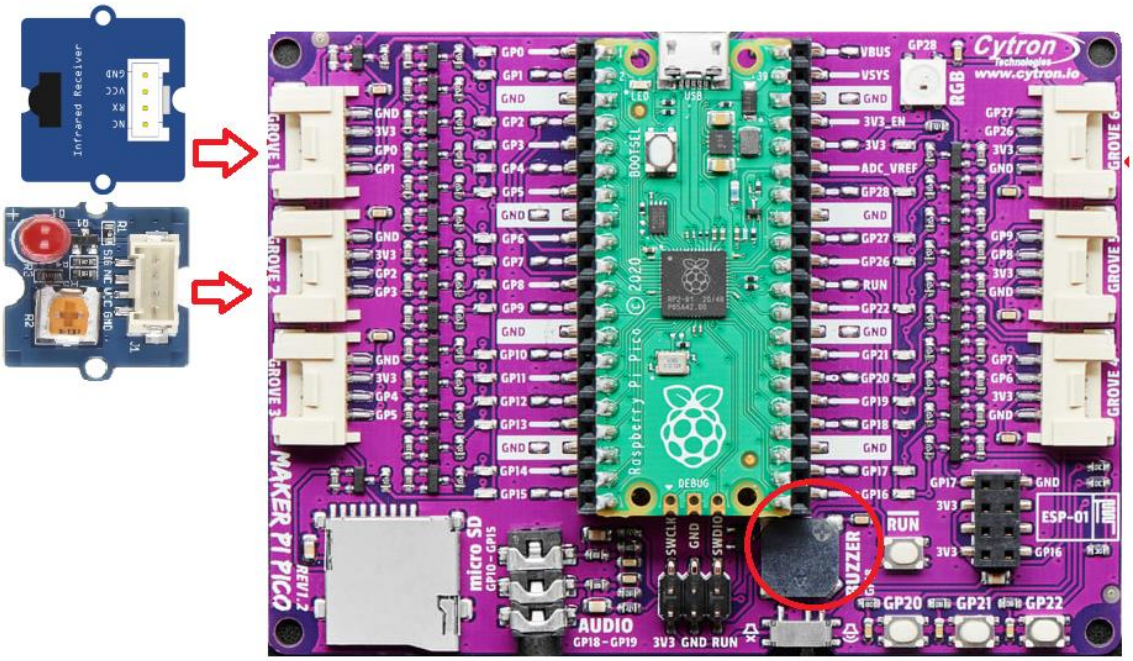


Entradas salidas:

Dispositivos utilizados

- **GP3** LED
- **GP1** Detector Infrarrojos

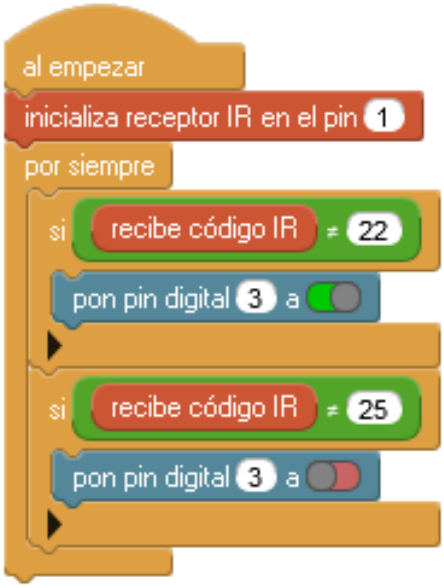
Esquema de montaje:



Programa:

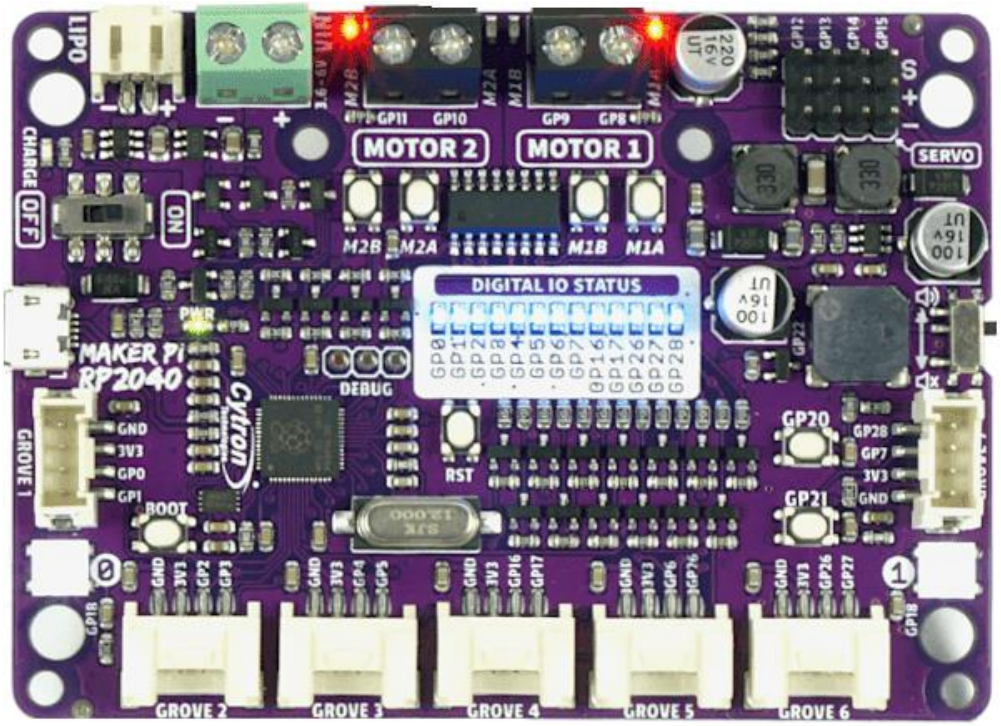
Nuestro programa se monta de la siguiente manera:

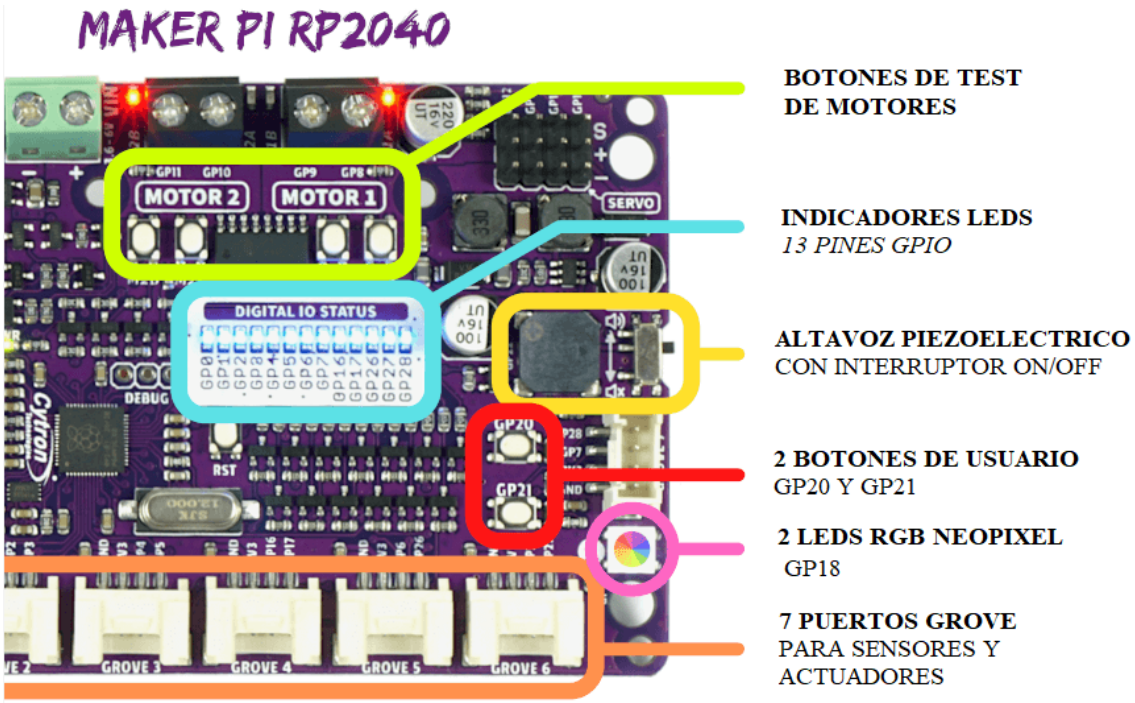
- En primer lugar, designamos el pin GP1 para conectar nuestro detector de infrarrojos
- Seguidamente establecemos dos condicionales, uno para la activación y otro para la desactivación de la salida pin GP3.
- Recibido el código 22 se activa la salida GP3
- Recibido el código 25 se desactiva la salida GP3



7. Prácticas con la tarjeta MAKER PI RP2040

Esquema de la tarjeta





7.1. Control Básico de un Motor de c.c.

Objetivo

Controlar un motor de c.c. que conectaremos en la salida MOTOR1 de la tarjeta MAKER PI RP2040.

Funcionamiento:

Se trata de mover el motor en ambos sentidos de giro y pararlo.

Consultaremos la tabla de control de los motores que nos facilita el fabricante.

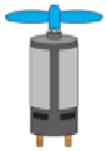
Entrada A (GP8 / GP10)	**Entrada B (GP9 / GP11)**	**Salida A (M1A / M2A)**	**Salida B (M1B / M2B)**	**Motor**
Bajo	**Bajo**	Bajo	Bajo	Freno
Alto	**Bajo**	Alto	Bajo	Adelante*
Bajo	**Alto**	Bajo	Alto	Atras*
Alto	**Alto**	Hi-Z (Open)	Hi-Z (Open)	Costa

Tabla 3: Tabla de verdad del controlador de motor

La dirección real del motor depende de la conexión del motor. Cambiar la conexión (MA y MB) invertirá la dirección.

El motor M1 se controla con los pines GP8 y GP9

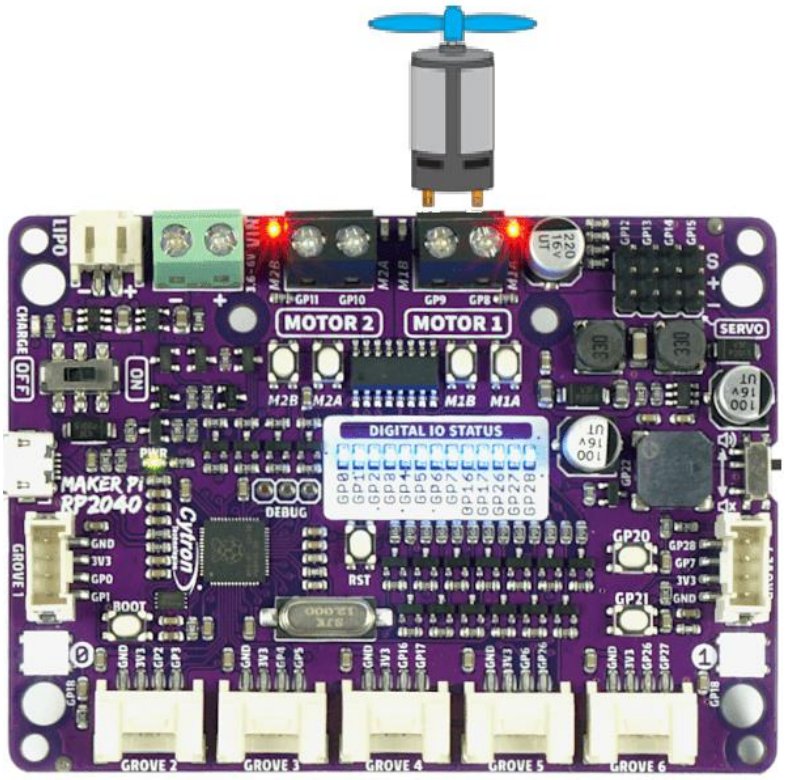
Entradas salidas:



Conexión en los terminales del MOTOR1

GP8 y GP9 son los pines de control del motor M1

Esquema de montaje:



Programa:

El programa se construye colocando en fama secuencial las combinaciones de activación/desactivación de las salidas GP8 y GP9 que son ñlas que controlan el MOTOR1.

De acuerdo con la tabla de funcionamiento para el control de los motores de la tarjeta anteriormente mostrada podemos escribir las secuencias

- **MOTOR ADELANTE**: GP8=1 y GP9=0
- **MOTOR ATRAS**: GP8=0 y GP9=1
- **MOTOR PARADO**: GP8=0 y GP9=0

Los tiempos entre cada acción son de 2000 ms

Se coloca delante de cada secuencia la instrucción "**di…**" para indicar el estado del motor

En la siguiente imagen se muestra el programa completo.

Actividades de Ampliación:

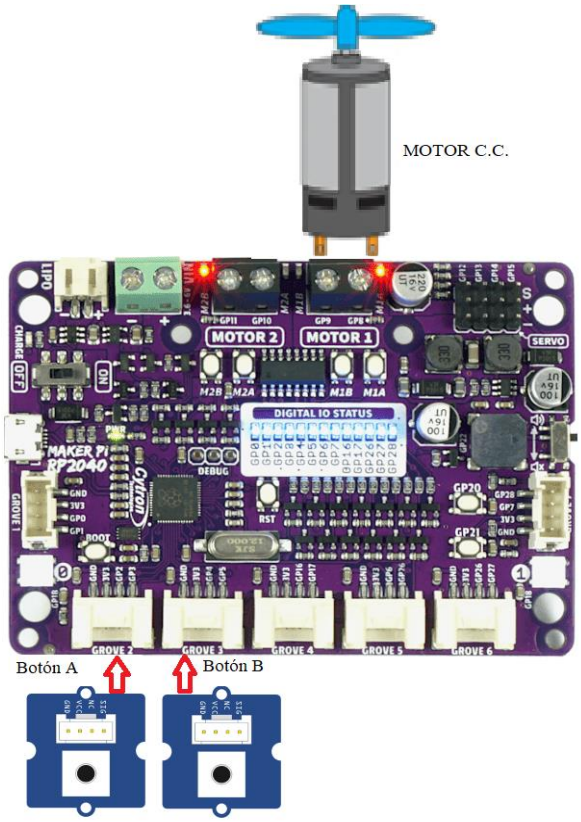
1. Queremos controlar el motor con dos botones uno de paro **Botón A** y uno de marcha **Botón B** que conectaremos a **GP3** y **GP5** y el motor lo seguimos conectando a la salida **MOTOR1**.

Solución:

En este caso tenemos que usar la entrada del pin **GP3** en la que colocamos el **botón A** en la parte de condición de un bloque condicional "**si**". La entrada **GP5** se coloca en el otro condicional.

A la vez que activamos y desactivamos las salida de control del MOTOR1 **GP8** y **GP9** mostramos un texto con el bloque "**di**.." poniendo "*MOTOR FUNCIONANDO*" y "*MOTOR PARADO*"

En la siguiente imagen vemos el programa y también el esquema de montaje.



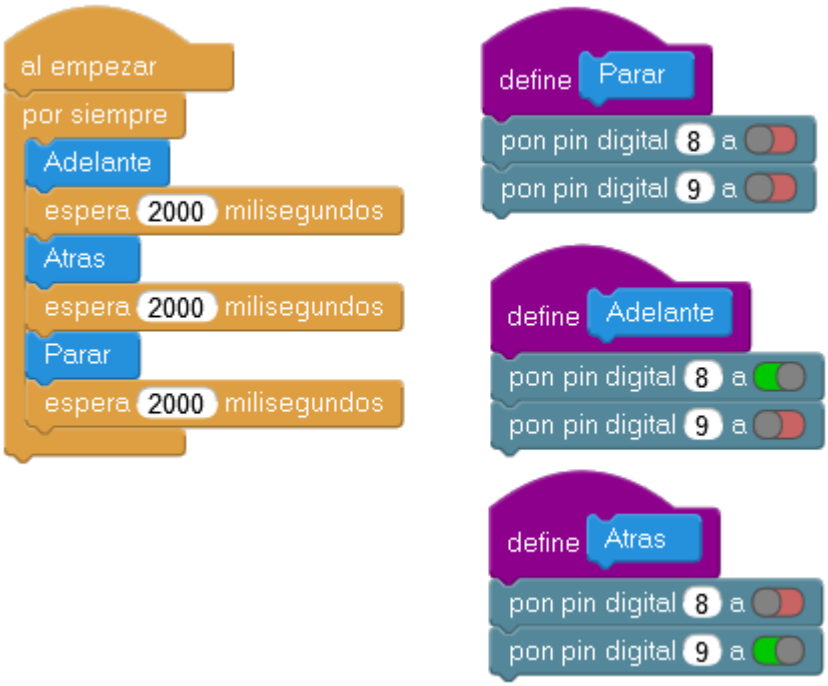
2. Realizar una aplicación en la que se diseñen tres funciones de control del motor: Adelante, Atrás y Parar. Estas funciones las queremos usar en el establecimiento de una secuencia permanente de movimiento tal como se indica en la figura.

Movimiento	**Tiempo**
Adelante	2000 ms.
Atrás	2000 ms
Parar	2000 ms

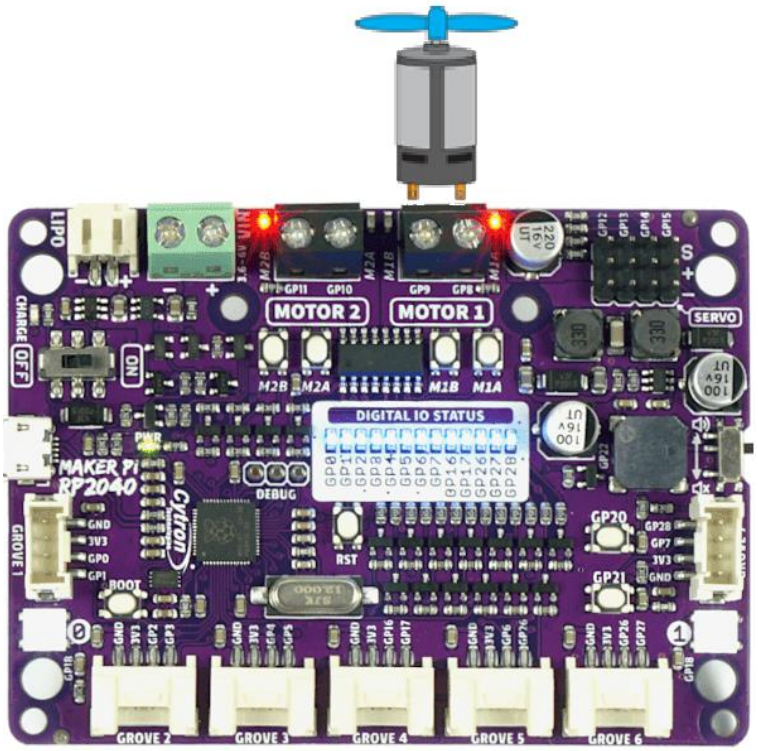
Lo que hacemos es crear cada uno de los movimientos en forma de función, tal como se muestra en la imagen siguiente.

El programa principal será un bucle “por siempre” dentro del cual ponemos cada función de acuerdo a la tabla mostrada anteriormente con los correspondientes retardos.

La siguiente imagen es la del programa completo.



Este sería el esquema del montaje.



7.2. Control de la velocidad de un Motor

Objetivo

Realizar el control de un motor variando la velocidad de giro

Funcionamiento:

La velocidad de giro se puede modificar enviando a los pines de control del motor (al que deba estar con valor "1" un valor analógico, es decir una tensión variable.

Para obtener la señal analógica variable usamos un "**Potenciómetro**" que conectaremos a un pin analógico de entrada de nuestra tarjeta. En nuestro caso colocaremos el potenciómetro en el pin **GP27**

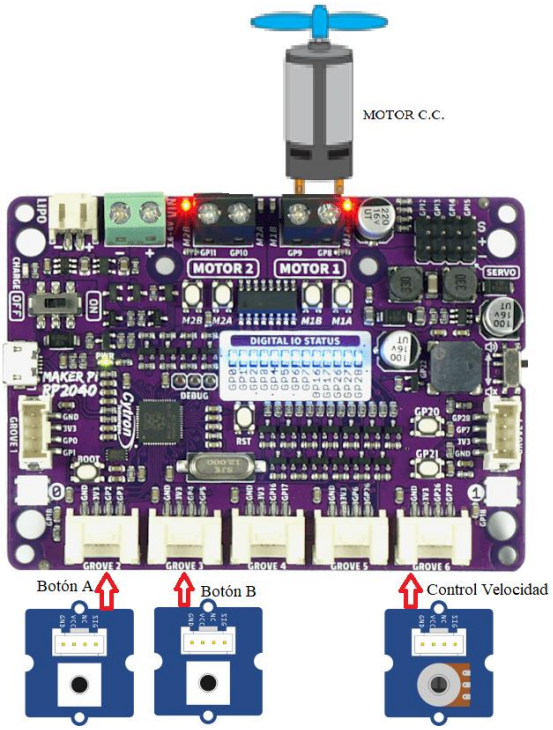
Entradas salidas:

Elementos de conexión a utilizar

Conexionado

- **GP3** Botón A
- **GP9** Botón B
- **GP27** Potenciómetro
- **Terminales MOTOR1**

Esquema de montaje:



Programa:

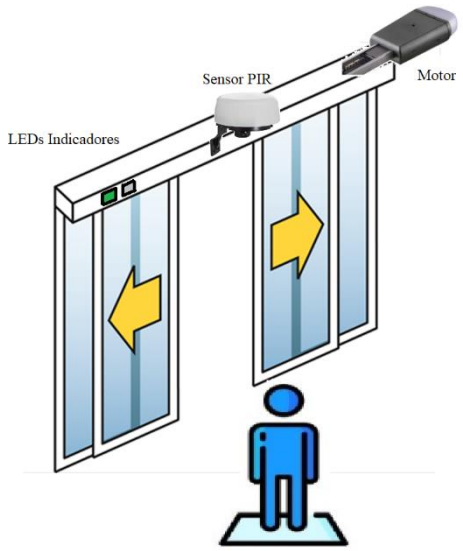
Antes de nada, debemos crear una variable a la que le llamaremos "**Velocidad**" y que asociaremos al valor entregado por el bloque de "**lectura analógica**" desde el pin **GP27.**

Dentro del bucle "**por siempre**" integraremos a continuación los condicionales que permiten el funcionamiento y la parada del **Motor1**. En el pin **GP8** en lugar de poner un estado digital "**1**" ponemos el valor de la **velocidad**.

7.3. Control de la apertura de una puerta automática

Objetivo

Realizar una aplicación de control que permita la apertura de una puerta cuando alguien esta frente a ella y seguidamente se cierre.



Funcionamiento:

Queremos controlar la apertura de una puerta con un sensor de movimiento de tal manera que cuando el detector detecte movimiento se ponga en marcha el motor que abre la puerta y trascurrido un tiempo de **4 seg**. si el detector no detecta movimiento se cierre de nuevo la puerta. Usaremos para la activación un **detector PIR** conectado en el pin **GP3** el motor lo colocaremos en la salida MOTOR1.Queremos señalizar la apertura con un **LED verde** conectado en el pin **GP5** y el cierre con un **LED rojo** conectado en el pin **GP17**

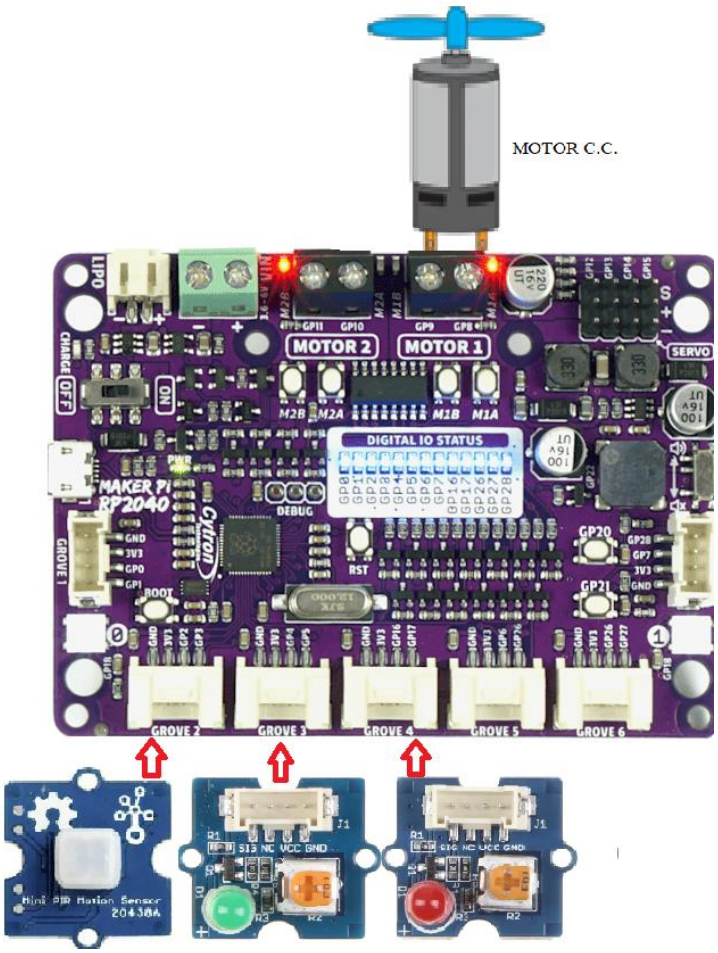
Entradas salidas:

Elementos que se uran

Conexiones

- **GP3** Sensor de movimiento
- **GP5** LED Verde
- **GP17** LED Rojo
- MOTOR1 **GP8** y **GP9**

Esquema de montaje:



Programa:

El programa se basará fundamentalmente en el testeo de un condicional cuya condición será el estado de la entrada a la que hemos conectado el sensor de movimiento que es una señal de tipo digital **GP3**

Si se activa el sensor: En este caso se pone en marcha el **MOTOR1** y a la vez se activa el LED Verde **GP5** y se desactiva el LED Rojo **GP17**. En este caso podemos pasar.

Vamos a suponer que deseamos tener la puerta abierta **4000** ms.

Se cierra la puerta: Una vez trascurrido el tiempo de apertura la puerta iniciará el retorno para su cierre. En esta fase queremos que el LED Verde se ponga en modo intermitente durante **8 secuencias** de encendido/apagado con retardo de **200 ms**.

Puerta Cerrada: Una vez terminados los **8 encendidos/apagados** del LED Verde se supondrá que la puerta se ha cerrado y se parará el motor. En este caso se apagará el **LED Verde** y se activará el **LED Rojo**

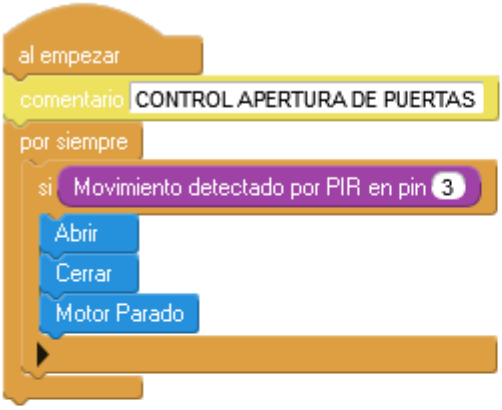
Propuestas de ampliación

1. Realizar el mismo montaje, pero usando tres bloques de función creados por el usuario. Estos bloques serán: Abrir, Cerrar, Motor Parado.

Solución.

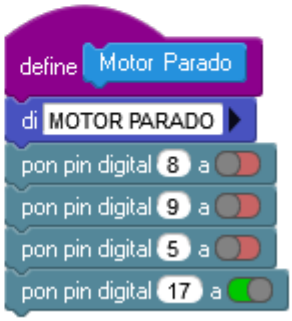
En la siguiente imagen se muestran los tres bloques que hemos mencionado.

El script principal de la aplicación consistirá el colocar un condicional que vigilará la detección de movimiento por parte del sensor. Si se detecta movimiento se ejecutarán las tres funciones, una tras otra en secuencia.


al empezar
comentario CONTROL APERTURA DE PUERTAS
por siempre
si Movimiento detectado por PIR en pin 3
Abrir
Cerrar
Motor Parado


define Cerrar
pon pin digital 8 a
pon pin digital 9 a
repite 8 veces
pon pin digital 5 a
espera 200 milisegundos
pon pin digital 5 a
espera 200 milisegundos


define Abrir
di MOTOR FUNCIONANDO
pon pin digital 8 a
pon pin digital 9 a
pon pin digital 5 a
pon pin digital 17 a
espera 4000 milisegundos


define Motor Parado
di MOTOR PARADO
pon pin digital 8 a
pon pin digital 9 a
pon pin digital 5 a
pon pin digital 17 a

7.4. Control básico de un servo

Objetivo

Con este ejercicio vamos a probar el funcionamiento de un servo conectado a nuestra tarjeta MAKER PI RP2040

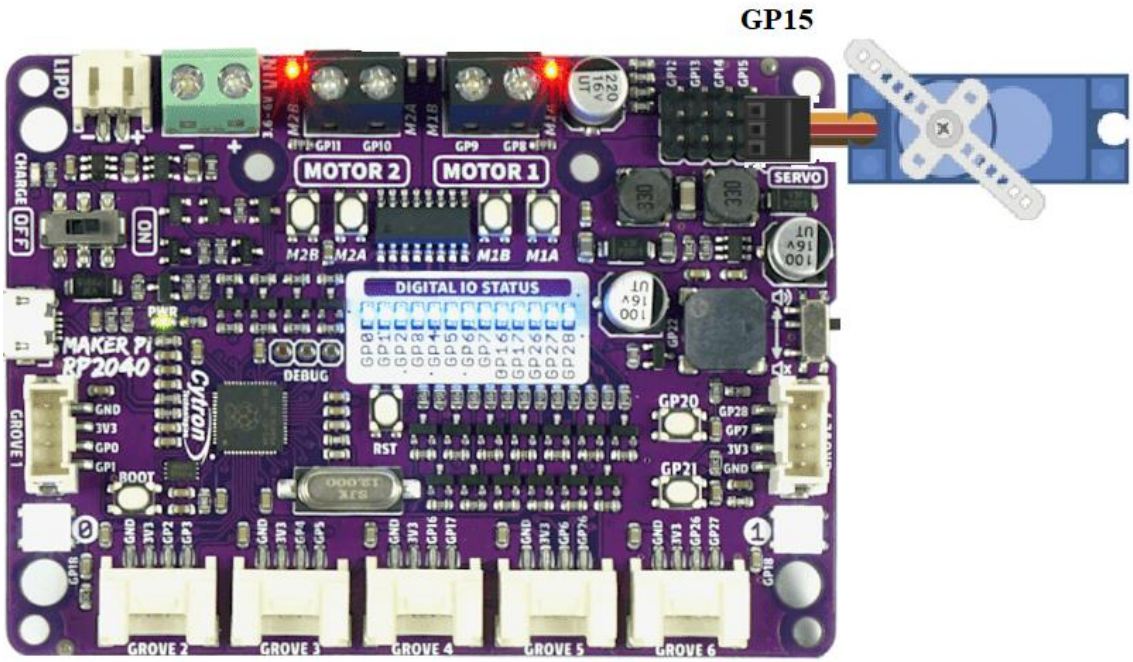
Funcionamiento:

Para realizar la practica contamos con un servo que conectaremos al pin GP15

Entradas salidas:

- **GP15** Conexión del servo

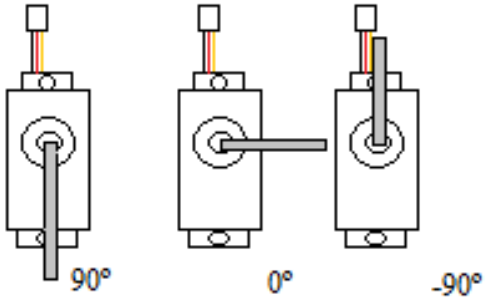
Esquema de montaje:



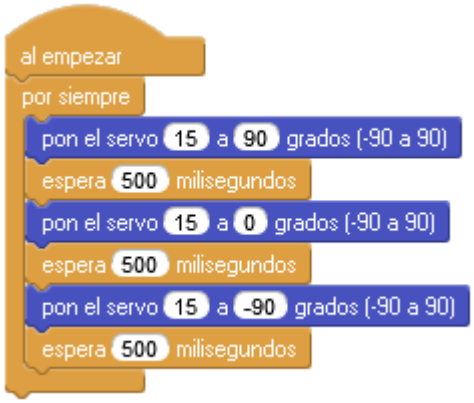
Programa:

Cargaremos la librería "**Sevomotores**"

Usaremos el bloque "**pon servo .. a ..**" y le colocamos los parámetros a cada una de las tres instrucciones que posicionarán el servo en los ángulos de 90, 0, y -90



Designamos una espera de 500 ms. para cada posición

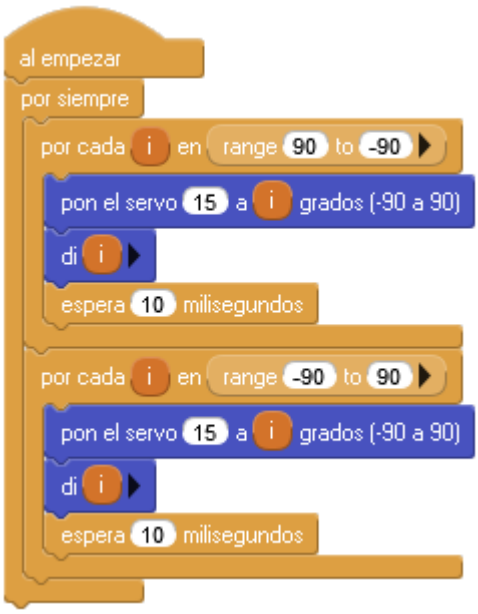


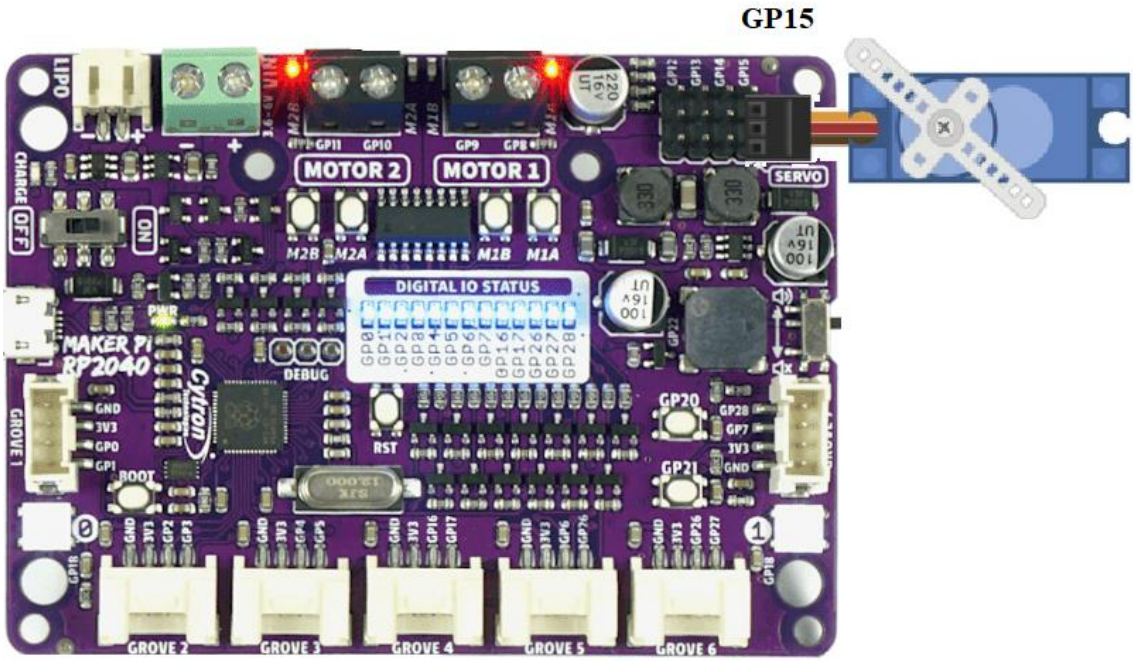
Actividades de Ampliación:

1. Realizar un programa para el control de un servo que le haremos funcionar en dos ciclos de trabajo.

Ciclo 1: Girar en paso de 15º en un rango de 90º a -90º para lo que usaremos una función tipo "**For Next„".** El tiempo entre cada uno de los incrementos de ángulo lo fijaremos en 10 ms

Ciclo 2: Girar en paso de 15º en un rango de -90º a 90º para lo que usaremos una función tipo "**For Next„".** El tiempo entre cada uno de los incrementos de ángulo lo fijaremos en 10 ms





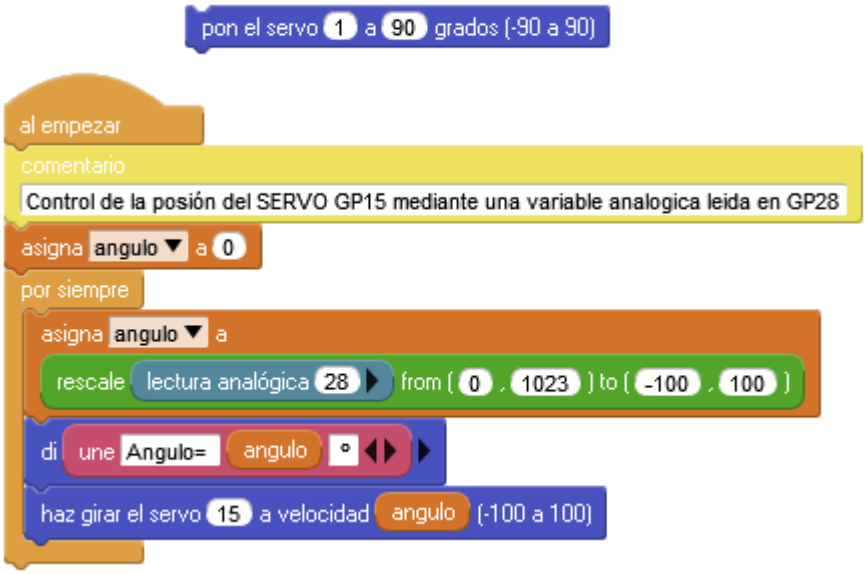
2. Realizar el control del servo colocado en GP15 designando el ángulo de giro en función del valor que entrega un potenciómetro colocado en el GP28. Mostraremos el valor con el bloque de fusión "di.."

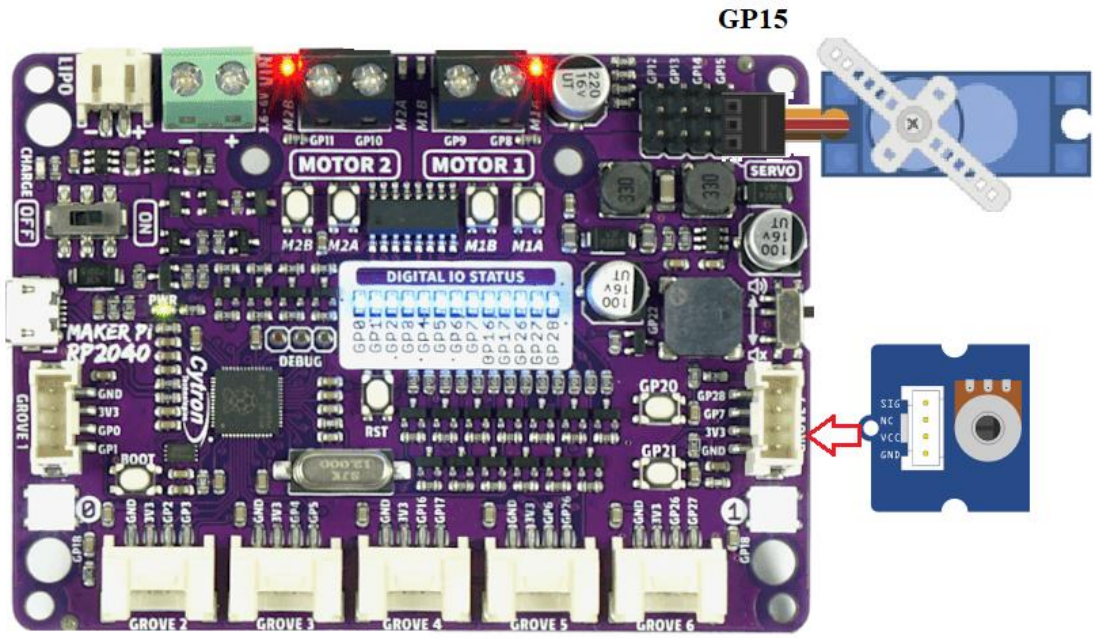
Solución.

Lo primero será definir la variable "**ángulo**" que pondremos a cero antes de entrar en el bucle "**por siempre**".

Dentro del bucle asignamos el valor del ángulo al valor que entrega el potenciómetro (0 a 1023) pero reescalando la variable al entorno -100 y 100 con el fin de ajustar el valor al que admite el bloque de activación del servo.

Seguidamente mostramos el valor del ángulo y después se ejecuta el bloque de posionamiento del servo.





7.5 Control de un servo. Modificación de ángulos de giro

Objetivo

En ocasiones una maquina debe ejecutar una secuencia de movimientos que se realizan usando un servo. En este ejemplo vamos a crear una secuencia de valores angulares que queremos ejecutar cuando se active un Botón conectado en la entrada digital GP5

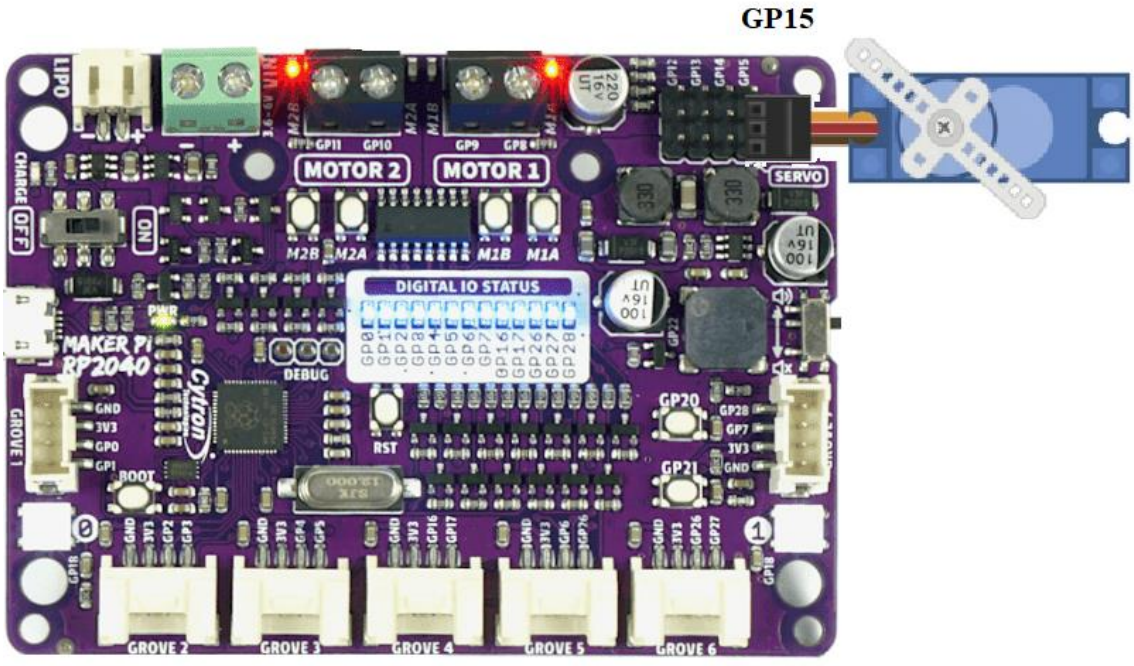
Funcionamiento:

La secuencia de movimientos se va a anotar en una lista que debemos crear. Las posiciones que deseamos serán las siguientes **[0. 90, -90, 0, 90]** la variable de esta lista la bautizaremos con el nombre "**Array_Posiciones**". Las posiciones almacenadas en la lista se irán colocando como valor angular en la instrucción de giro del servo.

Entradas salidas:

- **GP15** Conexión del servo

Esquema de montaje:



Programa:

Crearemos dos variables: "**Valor**" que recogerá el valor del ángulo a girar y "**Array_Posiciones**" que será la lista que contiene los **5** valores de los ángulos.

Nuestro programa tendrá dos scripts uno se ejecutará al arranque del programa mediante el bloque "**al empezar**" y en él se definirá la lista de posiciones angulares.

Nuestro segundo bloque de evento será el que recoja la activación del pin GP5

Lo que haremos será repetir 5 veces (ya que son con los valores que contiene la lista "**Array_Posiciones**") los valores que se asocairarán a la variable "**Angulo**". Los valores se van sacando con el bloque de instrucción "**elemento .. de ..**"

El retardo entre cada ángulo la estableceremos en un valor de 500 ms.

Actividades de Ampliación:

1. Te invitamos a crear una aplicación en la que se establezca para cada posición angular no solo el valor del ángulo sino también el tiempo que debe permanecer en esa posición el servo. Para la solución debes crear dos listas**: Lista de ángulo** y **lista de retardos** que denominaremos**, Array_Posiciones y Array_Tiempos.**

	Posición1	**Posición2**	**Posición3**	**Posición4**	**Posición5**
Angulo	0	90	-90	0	90
Tiempo	1000	500	2000	1000	500

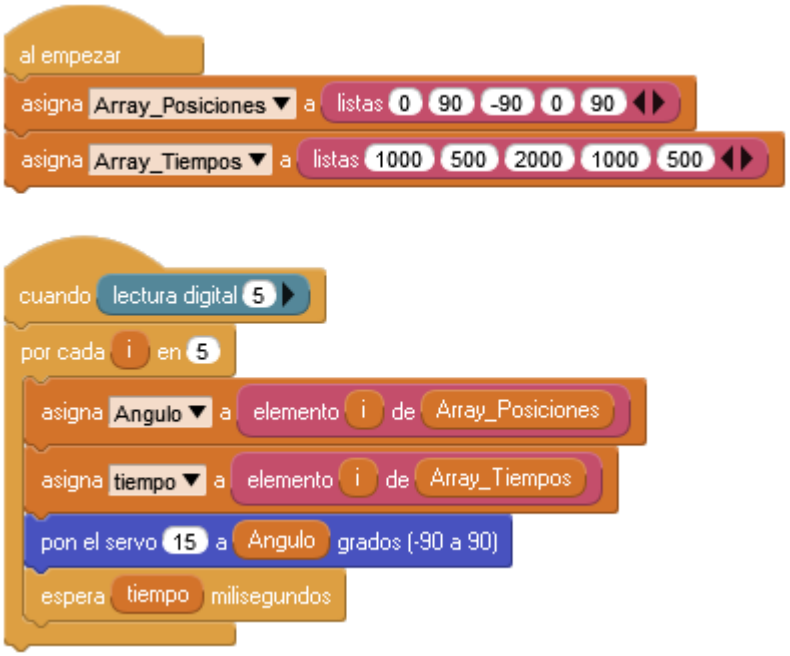
Solución

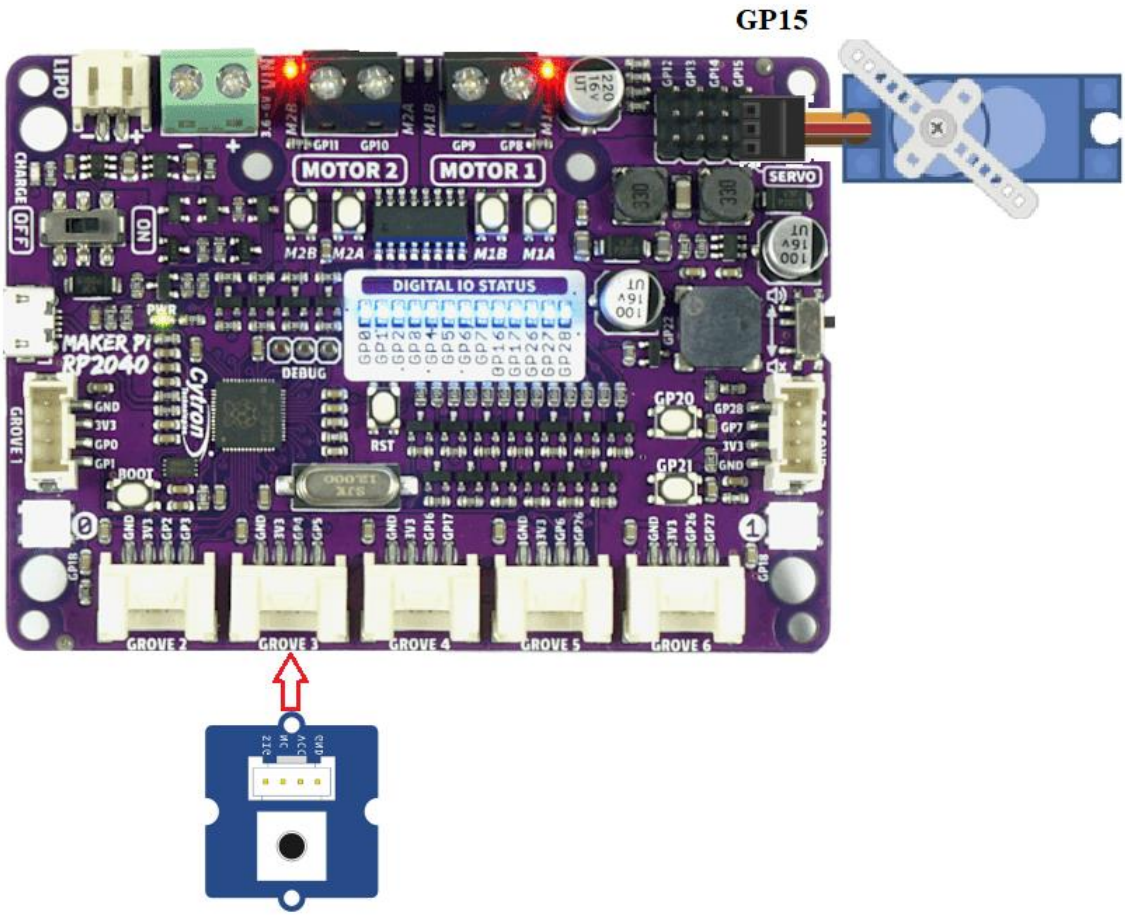
Lo primero será crear las variables.

- **Angulo** Recogerá el ángulo (valor que sacaremos de la lista de Posiciones)
- **Tiempo** Recogerá el retardo (valor que sacaremos de la lista Tiempos)
- **Array_Posiciones** Lista de posiciones angulares [0,90,-90,0.90]
- **Array_Tiempos** Lista de Tiempos [1000,500,2000,1000,500]

Seguidamente se definirán las listas o arrays de datos que se cargarán en el momento de arranque del programa.

El bloque de evento “cuando” estará gobernado por el estado del valor leído en el pin GP5 de la tarjeta. Sise pulsa el botón conectado a este puerto se realizara un bucle tipo “**para cada .. en ..**” **5** veces designando dentro de el los valores Angulo y tiempo que se van leyendo de las listas correspondientes





7.6. Control de servo mediante dos pulsadores.

Objetivo

Con esta aplicación vamos a controlar un servo mediante dos pulsadores. Pulsando uno girará hacia la derecha un ángulo determinado, pulsando el otro conseguiremos que el giro sea en sentido contrario.

Funcionamiento:

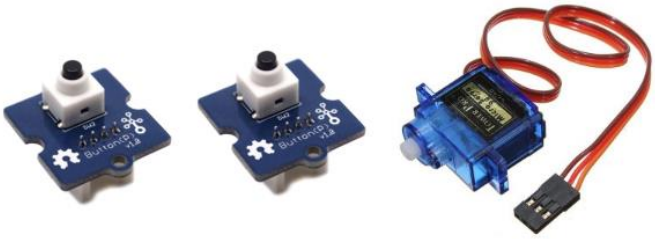
Conectados ambos pulsadores conseguimos con nuestro programa:

- Al pulsar el Pulsador conectado en **PG3** el servo girara un ángulo de 5º
- Al pulsar el pulsador conectado en **PG5** el servo girará un ángulo de -5º

Debemos controlar la variable ángulo en lo que se refiere a los valores que pueda tomar. El servo puede girar entre -90º y 90º, por lo que tenemos que controlar estos valores con el fin de que no se superen.

Entradas salidas:

Los dispositivos a conectar son



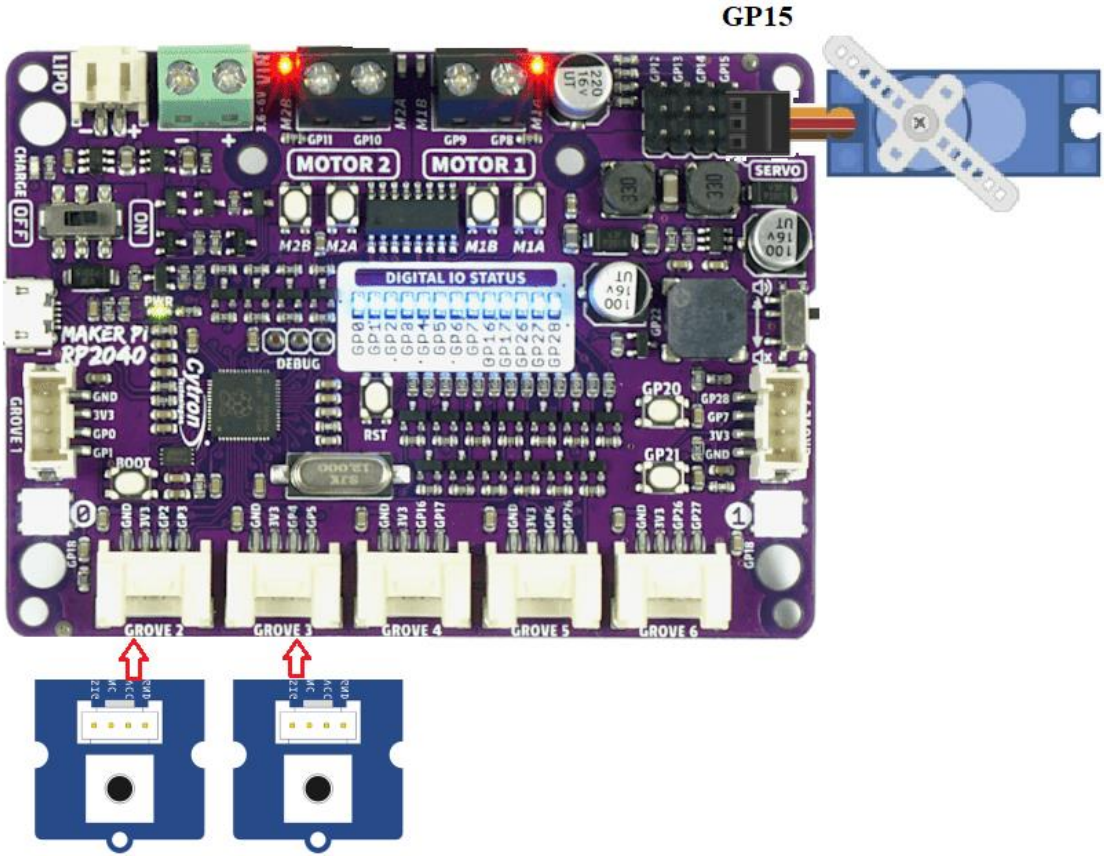
Las conexiones serán

- **GP3** Botón A
- **GP5** Botón B
- **GP15** Servo

Montaje

A continuación se muestra el esquema de montaje.

Esquema de montaje:



Programa:

Para empezar el programa debemos crear la variable "**Angulo**" que pondremos a "**0**" antes de entrar en el bucle de ejecución.

Seguidamente, dentro del bucle creado con el bloque "**por siempre**".

Colocaremos dos condicionales que se encargaran de testear el valor de las señales procedentes de los pines en los que están los botones, **GP3** y **GP5** de tal manera que si se pulsa el **Botón A en GP3** el servo **gira 5º** positivos y si por el contrario pulsamos el **Botón B conectado en GP5** el ángulo de giro será de **-5º**

Seguidamente colocaremos dos condicionales que se encargarán de que el valor del Angulo no supere los 90º ni sea menor que -90º

Finalmente mostraremos el valor del ángulo y ordenaremos el giro del servo colocando la variable "Angulo" en la ranura del parámetro correspondiente.


al empezar
asigna Angulo a 0
por siempre
si lectura digital 3
aumenta Angulo en 5
espera 50 milisegundos
si lectura digital 5
aumenta Angulo en -5
espera 50 milisegundos
si Angulo < -90
asigna Angulo a -90
si Angulo > 90
asigna Angulo a 90
di Angulo
pon el servo 15 a Angulo grados (-90 a 90)

7.7. Control de servo de 360º (vuelta completa).

Objetivo

Probar el funcionamiento de un servo d 360º (vuelta completa)

Funcionamiento:

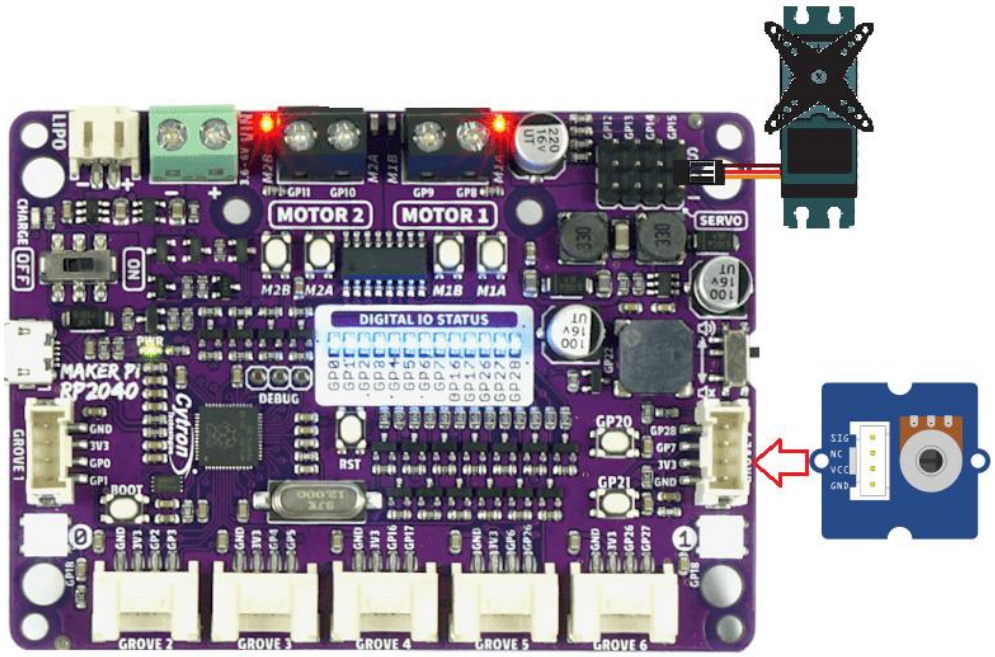
Este tipo de servo es capaz de girar vueltas completas. Lo puede hacer en sentido derecho o izquierdo dependiendo del valor que suministremos en su parámetro de ángulo.

Lo que haremos será obtener una señal analógica a través de un potenciómetro colocado en el pin **GP28** que escalaremos al rango de valor de este tipo de servo que es -100 a 100

Entradas salidas:

- **GP15** Servo de 360º
- **GP28** Potenciómetro

Esquema de montaje:

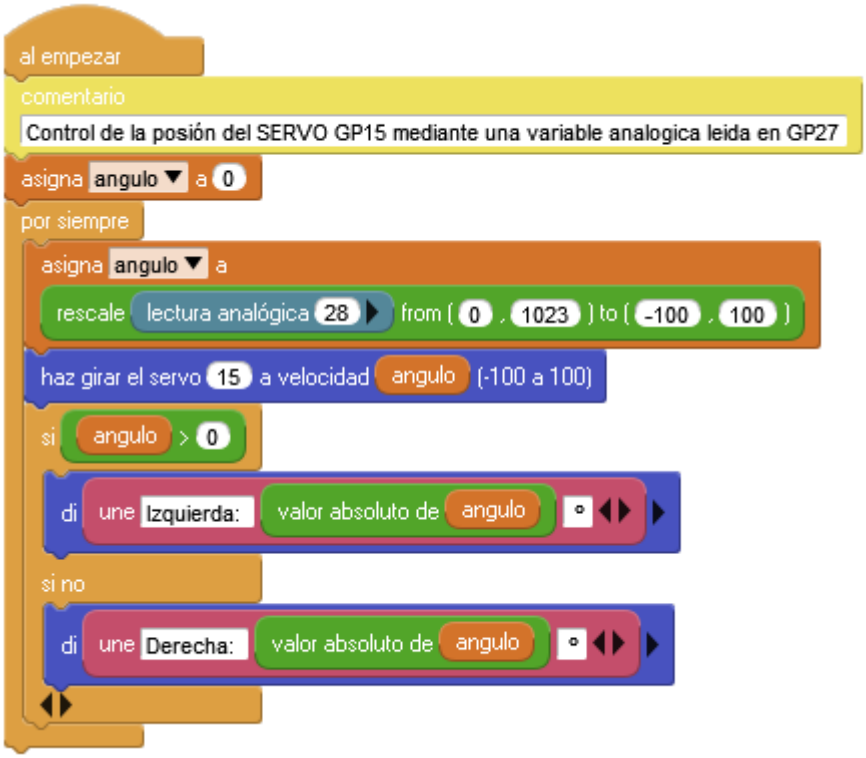


Programa

Para empezar, definimos una nueva variable a la que llamaremos "**angulo**"

Seguidamente dentro del bucle "**por siempre**" reescalamos la señal del pin **GP28** al margen -100 a 100 y a continuación colocamos en el bloque de activación del servo de 360º la variable "**angulo**" y el pin **GP15**.

Queremos mostrar el valor del ángulo y el sentido de giro en la pantalla de Microblocks, para ello testeamos el valor del **angulo >0** o valor positivo lo que significa que mostraremos la etiqueta "Izquierda" seguida el valor del ángulo, si el **ángulo <0** en ese caso el giro será a la izquierda y el ángulo negativo



8. Documentación y materiales útiles.

- Pagina principal de **MicroBlocks**
- Pagina Wiki de **MicroBlocks**
- Crear nuevas librerías con **MicroBlocks**.
- Descarga de la las librerías de **Peter Mathijssen**: PeterMathijssen/picobricks-libraries:
- Página de Peter Mathijssen

Profesor: **José Manuel Ruiz Gutiérrez**
Enero 2023