

# Raspberry Pi Pico + HuskyLens v1.7

by [@javacasm](#)

1. Programando la Pico
  - Instalación de Thonny
  - Primeros programas con Python en Thonny
  - Instalación del firmware micropython en la Pico

## 2. Electrónica con la Pico

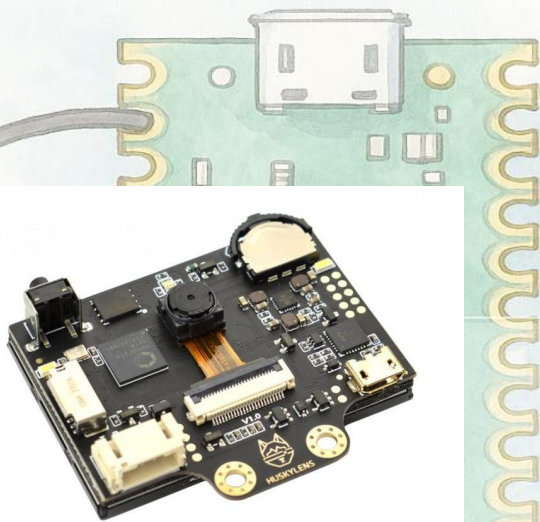
- Encendiendo LEDs
- Pulsadores
- Controlando el brillo
- ADC - Resistencia variable

## 3. HuskyLens

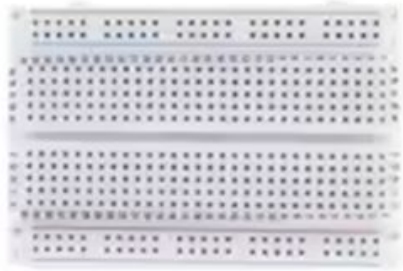
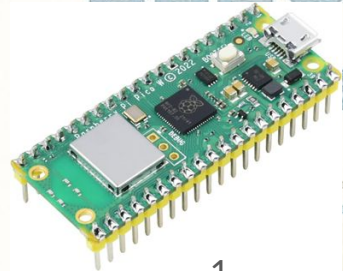
- Usando la Huskylens
- Conectando Huskylens+Pico
- Controlando la HuskyLens desde la Pico

## Apéndice: micro:bit

- Conectando la HuskyLens a la micro:bit



# ¿Qué debe llevar un kit de electrónica?



1.



1. **Microcontrolador:** Raspberry Pi Pico W
2. 10 cables Dupont de colores M-M y H-H
3. Mini Breadboard, mejor si es la [PicoBB](#)
4. **Resistencias de distintos valores**
  - a. 220  $\Omega$  para LEDs
  - b. 10 k $\Omega$
5. **Actuadores - Dispositivos de salida:**
  - a. LEDs de diferentes colores.
  - b. LED RGB.
  - c. Neopixels
  - d. Relé
6. **Dispositivos de entrada:**
  - a. Pulsadores.
  - b. Potenciómetro o resistencia variable.
  - c. Fotorresistencias LDR de 2 M $\Omega$
  - d. Sensor de temperatura



# Raspberry Pi Pico W

La Raspberry Pi Pico es un microcontrolador diseñado para proyectos de computación física y tareas específicas y aplicaciones de bajo nivel, como controlar sensores, motores, LEDs u otros dispositivos electrónicos

Es ideal para principiantes y expertos que quieran aprender programación y electrónica.

## Aplicaciones:

La Raspberry Pi Pico es ideal para:

- Proyectos de IoT (especialmente con Pico W).
- Control de dispositivos como sensores, motores, pantallas o LEDs.
- Aprendizaje de programación y electrónica.
- Prototipado de productos electrónicos.
- Proyectos de robótica, automatización del hogar y sistemas embebidos.





# Características de la Raspberry Pi Pico W (la de la foto)

## ELECTRONICA

### Raspberry Pi Pico

- WiFi integrado para conectarse a internet (2.4 GHz).
- Bluetooth 5.2, con compatibilidad Bluetooth LE y Bluetooth Classic.
- Microcontrolador potente (RP2040) con doble núcleo, rápido y eficiente.
- Fácil de programar con MicroPython o C/C++, perfecto para principiantes.
- WPA3 y modo Punto de Acceso para conectar hasta 4 dispositivos.
- 26 pines GPIO, que permiten conectar sensores, luces, motores y más.
- Bajo consumo de energía, ideal para proyectos portátiles.

[Diferencias entre Raspberry Pi y Raspberry Pico](#)



# ELECTRONICA

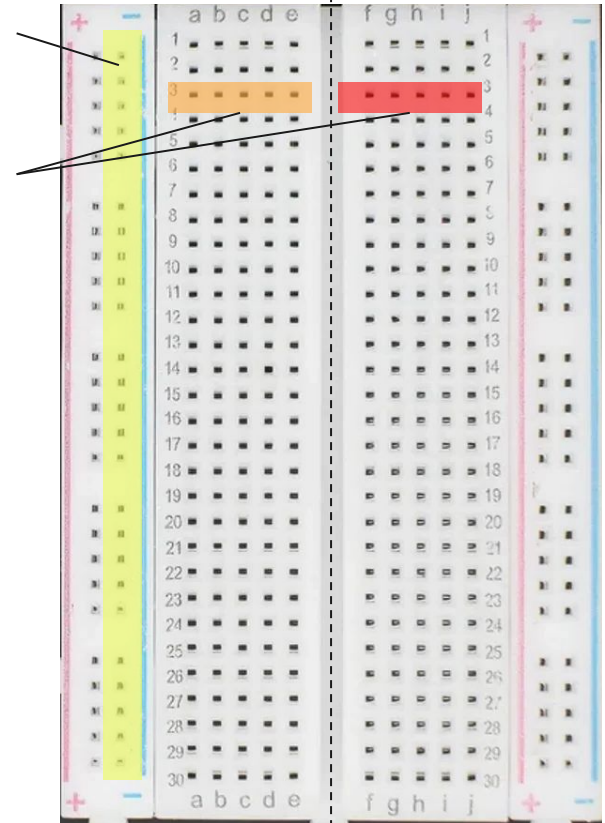
## El hardware: Protoboard

### La breadboard:

- Las columnas laterales (+ -) están **conectadas por columnas** entre ellas:
  - si se conecta algo en dos agujeros de la misma columna, estarán conectados.
- Los agujeros de en medio, están **conectados por filas** entre ellos:
  - Si se conecta algo en una misma fila, estarán conectados.
  - Las filas están divididas en dos partes (*abcd* y *fghi*). Estas dos partes **NO están conectadas entre ellas**.

Interconectados  
por columnas

Interconectados  
por filas (derecha e  
izquierda)

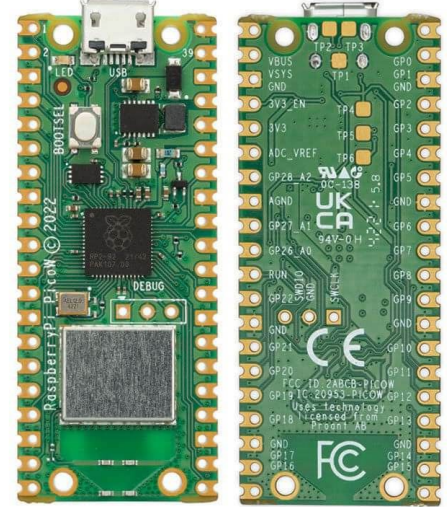


# ELECTRONICA

## El hardware: Raspberry Pi Pico

### La placa Raspberry Pi Pico W:

- En la parte posterior de la placa encontrarás las etiquetas con los nombres y números de cada pin.
- Cada número corresponde a **un tipo de conexión** según tus necesidades. Para empezar, debes conocer:
  - **GND:** Es la toma de tierra. Cualquier circuito debe tener una conexión a GND para cerrarlo. Escoge cualquiera, hay varios.
  - **GPIO:** Son entradas o salidas digitales y se pueden utilizar para:
    - LEDs y botones.
    - Sensores analógicos .





# ELECTRONICA

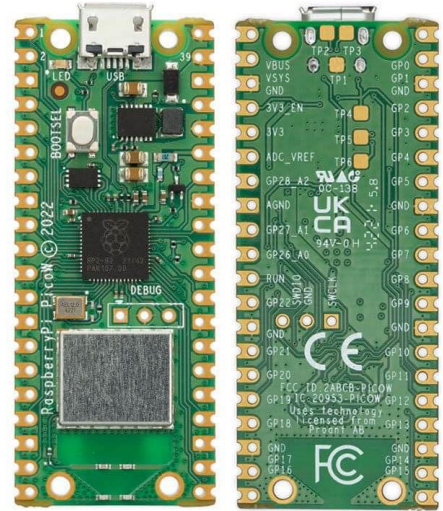
## Raspberry Pi Pico

### El hardware: las entradas y salidas (GPIO)

#### ¿Qué GPIO debo usar?

Depende de lo que debas conectar:

- **Salidas digitales** (ej. LEDs): Cualquier GPIO, pero usaremos el GPIO15.
- **Entradas digitales** (ej. Botones): Cualquier GPIO, pero usaremos el GPIO14.
- **Entradas analógicas** (potenciómetro y fotorresistencia: **Solo los GPIO26, GPIO27 y GPIO28** tienen capacidad ADC para leer valores analógicos).



# ELECTRO

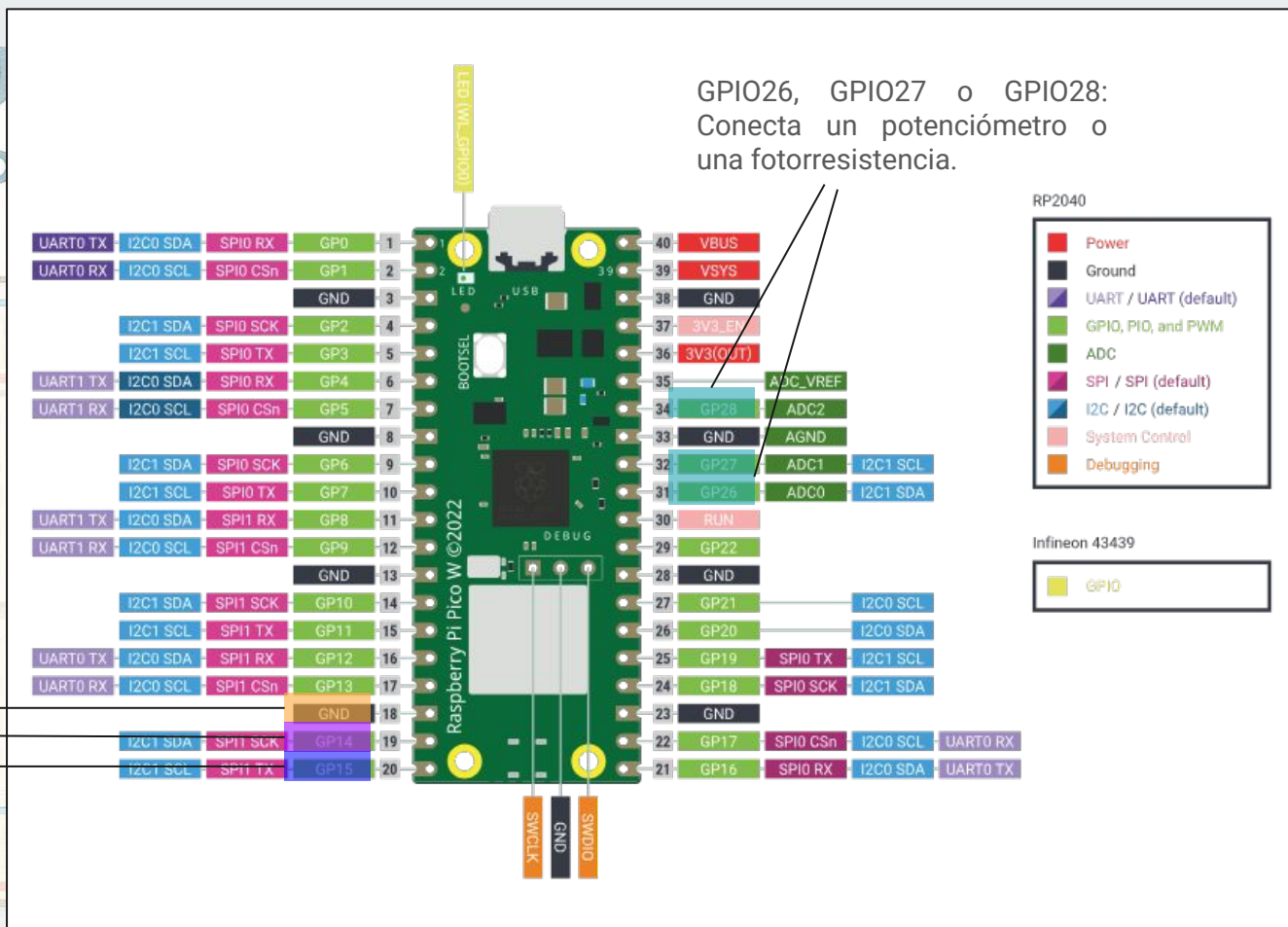
## Raspberry P

## GPI014: Conecta un botón

GND:  
Cierra  
un circuito

## GPI015: Conecta un LED

## Imagen ampliada





# Instalación de Thonny

Thonny:



Para dar órdenes se necesita un entorno para programar. Thonny es un software sencillo para programar con **lenguaje Python**.

1. Descarga la versión de Thonny (<https://thonny.org/>) para tu sistema operativo.
2. Instálalo (no necesita permisos de administrador)
3. Selecciona el idioma que quieras y la configuración "Standard"

## Thonny



Download version **4.1.7** for  
Windows • Mac • Linux

### Official downloads for Windows

1.

**Installer with 64-bit Python 3.10**, requires 64-bit Windows 8.1 / 10 / 11  
[thonny-4.1.7.exe \(21 MB\)](#)

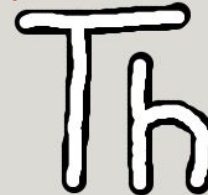
**Installer with 32-bit Python 3.8**, suitable for all Windows versions since 7  
[thonny-py38-4.1.7.exe \(20 MB\)](#)

**Portable variant with 64-bit Python 3.10**  
[thonny-4.1.7-windows-portable.zip \(31 MB\)](#)

**Portable variant with 32-bit Python 3.8**  
[thonny-py38-4.1.7-windows-portable.zip \(29 MB\)](#)

**Re-using an existing Python installation** (for advanced users)  
`pip install thonny`

3.



Language:

Initial settings:

Let's go!

# Instalación del firmware de Micropython

Thonny:



Vamos a instalar el firmware de Micropython en la Pico

1. Conecta la Pico W al ordenador mediante un cable USB.
2. Aparecerá una unidad (como un pendrive) RPI-RP2
3. Abre Thonny, ve a menú "Ejecutar", pestaña "Intérprete":

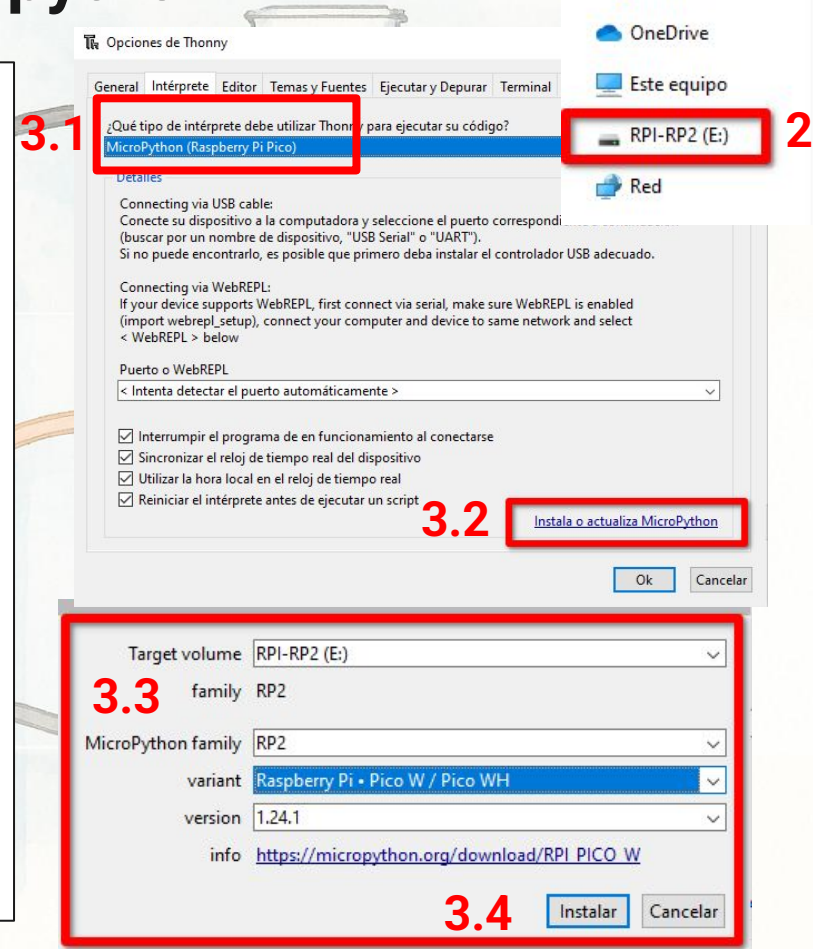
3.1 Selecciona "MicroPython (Raspberry Pi Pico)".

3.2 Pulsa en "Instala o actualiza Micropython".

3.3 Selecciona las características de tu Pico

3.4 Pulsa Instalar

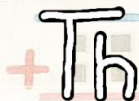
4. Ya puedes seleccionar tu puerto para usar tu Pico



# Thonny: entorno de programación

## ELECTRONICA Raspberry Pi Pico

Thonny:



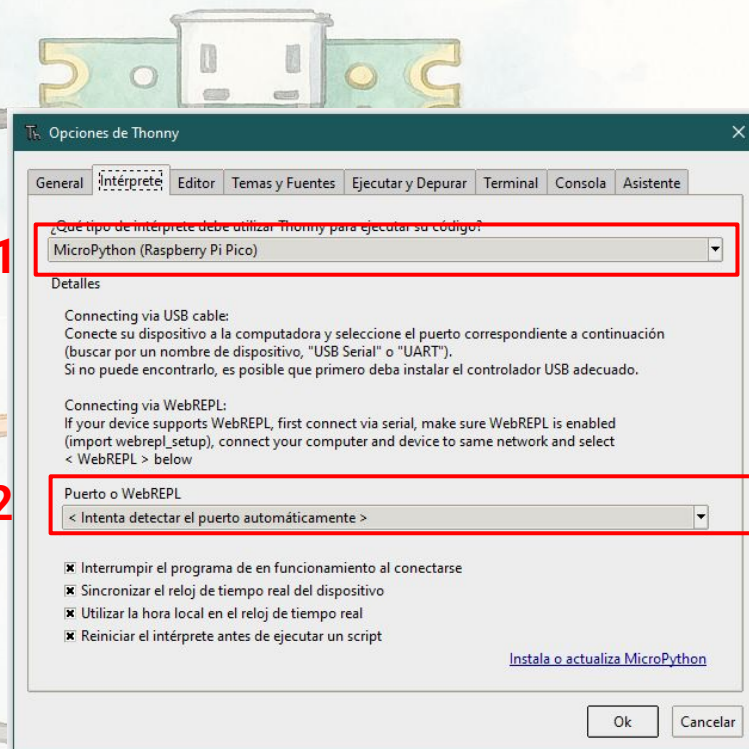
Usaremos la aplicación Thonny para programar con **lenguaje Python**.

1. Conecta la Pico W al ordenador mediante un cable USB.
2. Abre Thonny, ve a "Ejecutar", pestaña "Intérprete":

2.1 Selecciona "MicroPython (Raspberry Pi Pico)".

2.2 En "Puerto o WebREPL" -> selecciona el puerto.

3. Puedes pulsar abajo a la derecha, cuando Thonny detecte tu placa.



MicroPython (Raspberry Pi Pico) • Board in FS mode @ /dev/ttyACM0



# Encendemos el LED integrado en la Pico

## ELECTRONICA Raspberry Pi Pico

1. Copia esta programación a Thonny:

```
from machine import Pin
import time
led = Pin('LED', Pin.OUT)
led.on()
time.sleep(3)
led.off()
```

*Importa la librería para controlar los pines GPIO*

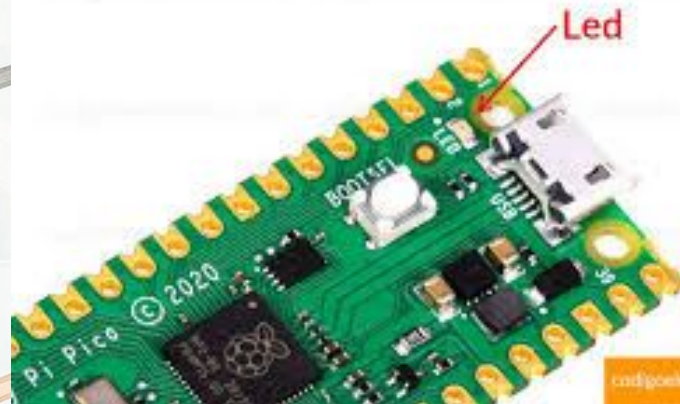
*Importa la librería para controlar el tiempo*

*Configura el Pin interno como salida para controlar el LED de la placa*

*Enciende el LED*

*Espera 3 segundos*

*Apaga el LED*



**¡¡Si copias y pegas el código  
corrige las comillas!!**

2. Ejecuta el programa.



# Parpadeamos el LED integrado en la Pico

## ELECTRONICA

### Raspberry Pi Pico

1. Copia esta programación a Thonny:

```
from machine import Pin
import time
led = Pin('LED', Pin.OUT)
led.on()
time.sleep(3)
led.off()
```

*Importa la librería para controlar los pines GPIO*

*Importa la librería para controlar el tiempo*

*Configura el Pin interno como salida para controlar el LED de la placa*

*Enciende el LED*

*Espera 3 segundos*

*Apaga el LED*

**¡¡Si copias y pegas el código  
corrige las comillas!!**

2. Ejecuta el programa.



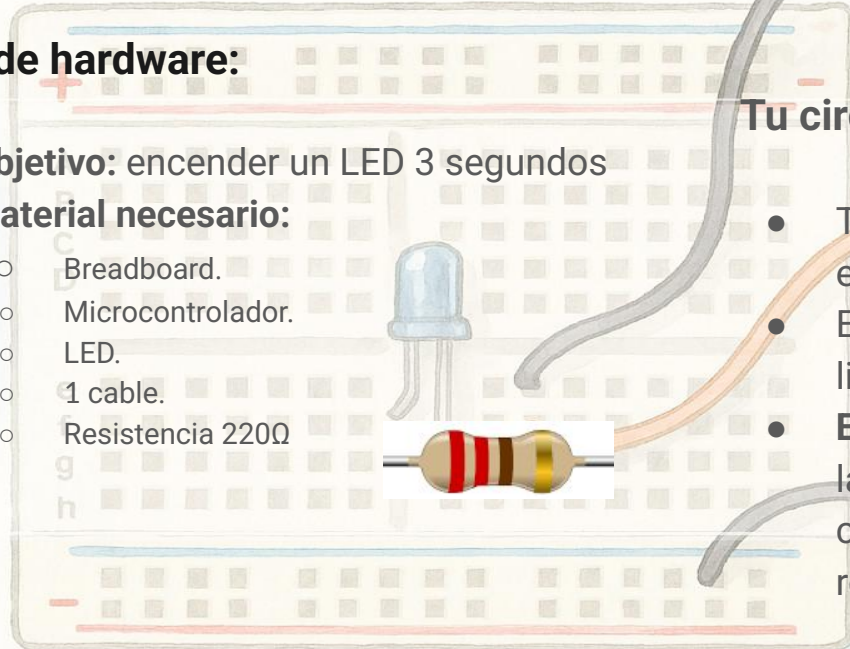
# Encendiendo y apagando un LED

## ELECTRONICA

### Raspberry Pi Pico

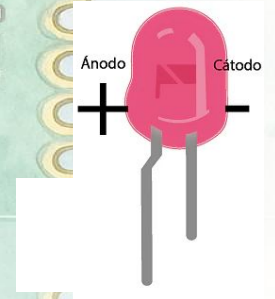
#### Parte de hardware:

- **Objetivo:** encender un LED 3 segundos
- **Material necesario:**
  - Breadboard.
  - Microcontrolador.
  - LED.
  - 1 cable.
  - Resistencia 220 $\Omega$



#### Tu circuito:

- Tu circuito debe empezar en el GPIO\_15 y terminar en un GND.
- El LED necesita una resistencia de 220 $\Omega$  para limitar la corriente.
- **El circuito va de + a -** : Esto significa que la pata larga del LED (+) se conecta al GPIO\_15 y la pata corta (-) se conecta a GND, pasando antes por la resistencia.
  - GPIO15 → LED → resistencia → GND

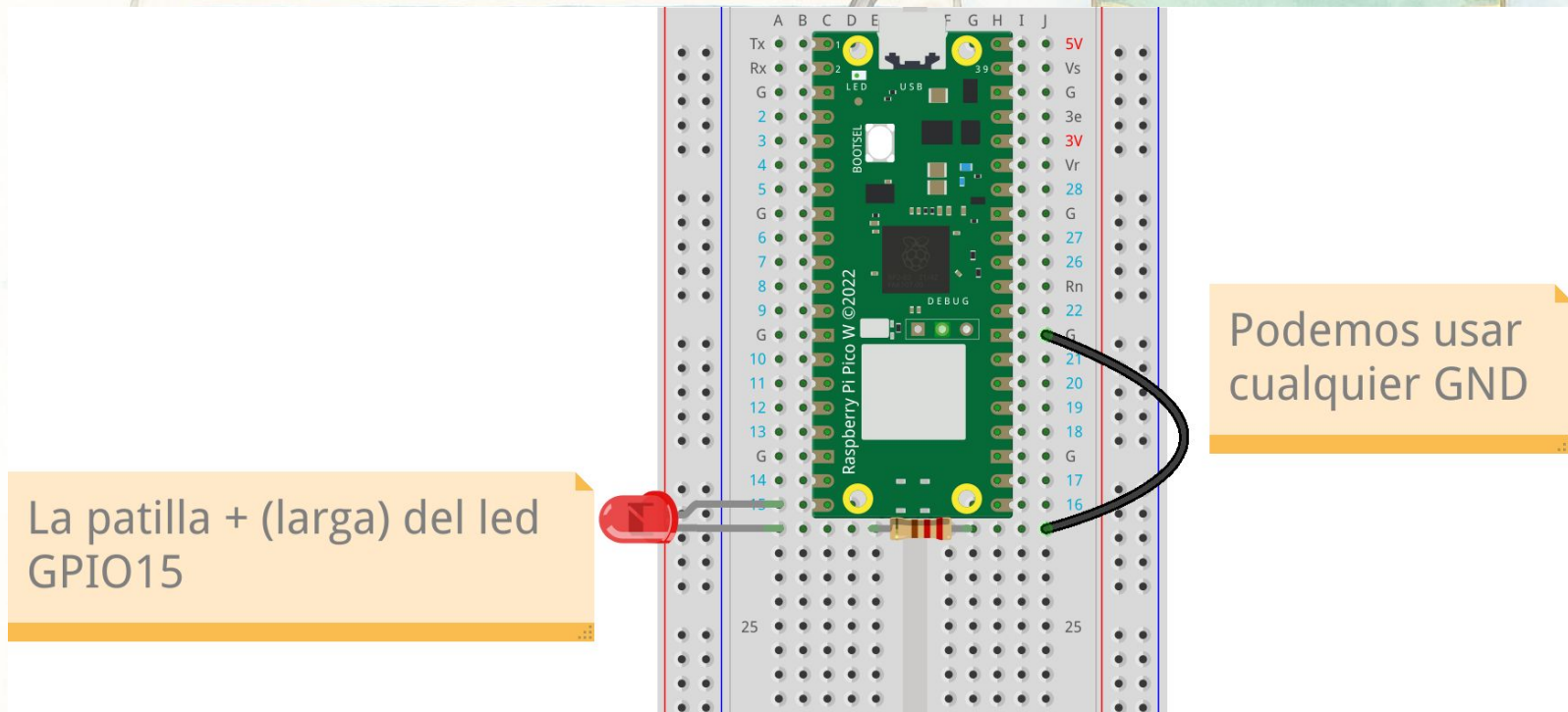




# Conectamos LED externo

## ELECTRONICA

### Raspberry Pi Pico



# Encendiendo un led

## ELECTRONICA

### Raspberry Pi Pico



#### Código python

1. Copia esta programación a Thonny:

```
from machine import Pin
import time
led = Pin(15, Pin.OUT)
led.on()
time.sleep(3)
led.off()
```

*Importa la librería para controlar los pines GPIO*

*Importa la librería para controlar el tiempo*

*Configura el GPIO15 como salida para controlar el LED*

*Enciende el LED*

*Espera 3 segundos*

*Apaga el LED*

2. Ejecuta el código



# Encendiendo y apagando manualmente

## ELECTRONICA

### Raspberry Pi Pico



#### Parte de hardware

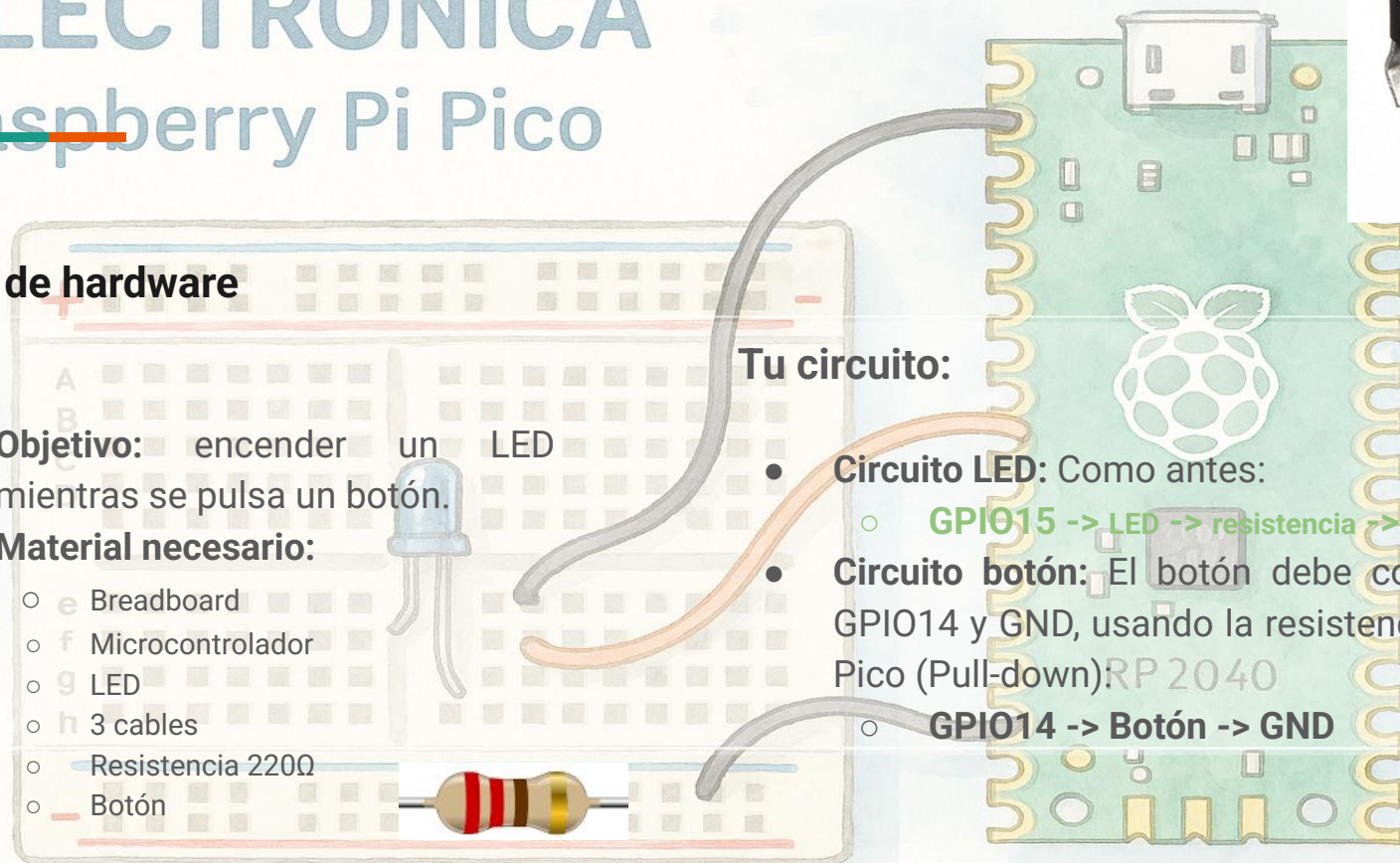
- **Objetivo:** encender un LED mientras se pulsa un botón.

- **Material necesario:**

- Breadboard
- Microcontrolador
- LED
- 3 cables
- Resistencia 220Ω
- Botón

#### Tu circuito:

- **Circuito LED:** Como antes:
  - GPIO15 -> LED -> resistencia -> GND
- **Circuito botón:** El botón debe conectarse entre GPIO14 y GND, usando la resistencia interna de la Pico (Pull-down) RP2040
  - GPIO14 -> Botón -> GND





# Encendiendo y apagando manualmente

## ELECTRONICA Raspberry Pi Pico



### Parte de hardware

### Tu circuito

Conecta los elementos de esta forma:

GPIO15 -> LED -> resistencia -> GND

GPIO14 -> Botón -> GND

Conectamos el pulsador a GPIO14

Conectamos el pulsador 3V

fritzing

# Encendiendo y apagando manualmente

## ELECTRONICA

### Raspberry Pi Pico



#### Parte de software:

#### Código python

- Copia esta programación a Thonny:

```
from machine import Pin
led = Pin(15, Pin.OUT)
boton = Pin(14, Pin.IN, Pin.PULL_UP)
while True:
    if boton.value():
        led.off()
    else:
        led.on()
```

*Importa la librería para controlar los pines GPIO*  
*Configura GPIO15 como salida para controlar el LED*  
*Configura GPIO14 como entrada con resistencia pull-up (Activo por defecto)*  
*Bucle infinito para comprobar continuamente el estado del botón*

- Si el botón está presionado (valor 0, LOW)
  - Apaga el LED
- Si el botón no está presionado (valor 1, HIGH)
  - Enciende el LED

- Ejecuta el programa



# Controlando el brillo del LED (PWM)

## ELECTRONICA

### Raspberry Pi Pico

#### Parte de hardware:

- **Objetivo:** Regular la intensidad de brillo del LED con un potenciómetro.
- **Material necesario:**
  - Breadboard.
  - Microcontrolador.
  - LED.
  - 4 cables.
  - Resistencia 220Ω
  - Potenciómetro.

#### Tu circuito:

- **Circuito LED:** Como antes:
  - GPIO15 -> LED -> resistencia -> GND
- **Circuito potenciómetro:** El potenciómetro tiene tres patas y dos soportes (se encajan en el espacio del medio de la breadboard). Con las patas a la derecha:
  1. Pata superior -> 3V3(OUT)
  2. Pata de en medio -> GPIO26
  3. Pata inferior -> GND





# Ajustando manualmente el brillo

## ELECTRONICA

### Raspberry Pi Pico

#### Hardware:

- **Objetivo:** Regular la intensidad de brillo del LED con un potenciómetro.
- **Material necesario:**
  - Breadboard.
  - Microcontrolador.
  - LED.
  - 4 cables.
  - Resistencia 220Ω
  - Potenciómetro.

#### Tu circuito:

- **Circuito LED:** Como antes:
  - GPIO15 -> LED -> resistencia -> GND
- **Circuito potenciómetro:** El potenciómetro tiene tres patas y dos soportes (se encajan en el espacio del medio de la breadboard). Con las patas a la derecha:
  1. Pata superior -> 3V3(OUT)
  2. Pata de en medio -> GPIO26
  3. Pata inferior -> GND



# Ajustando manualmente el brillo

## ELECTRONICA Raspberry Pi Pico

Parte de hardware:

Tu circuito

Conecta los elementos de esta forma:

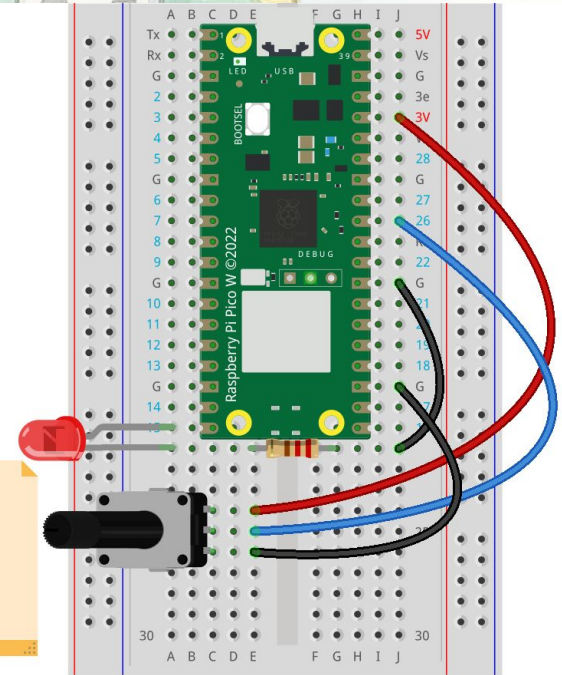
GPIO15 -> LED -> resistencia -> GND

Pata superior -> 3V3(OUT)

Pata de en medio -> GPIO26

Pata inferior -> GND

Un extremo - 3V  
El pin central - GPIO26  
El otro extremo - GND



fritzing

# Ajustando manualmente el brillo

## ELECTRONICA

### Raspberry Pi Pico



#### Parte de software: Código python

- Copia esta programación a Thonny:

```
from machine import Pin, ADC, PWM
import time
pot = ADC(26)
led = PWM(Pin(15))
led.freq(1000)
while True:
    valor = pot.read_u16()
    led.duty_u16(valor)
    time.sleep(0.01)
```

*Importa la librería para controlar los pines GPIO, leer valores analógicos, controlar la intensidad LED y añadir pausas.  
Configura el potenciómetro como entrada analógica.  
Configura el LED en el GPIO15 como salida PWM.  
Fija la frecuencia del PWM en 1000 Hz.*

*Inicia un bucle infinito para leer el potenciómetro continuamente:*

- *Lee el valor del potenciómetro en un rango de 0 a 65535.*
- *Ajusta la intensidad LED según el valor del potenciómetro.*
- *Pequeña pausa de 10ms para estabilidad.*

- Ejecuta el programa





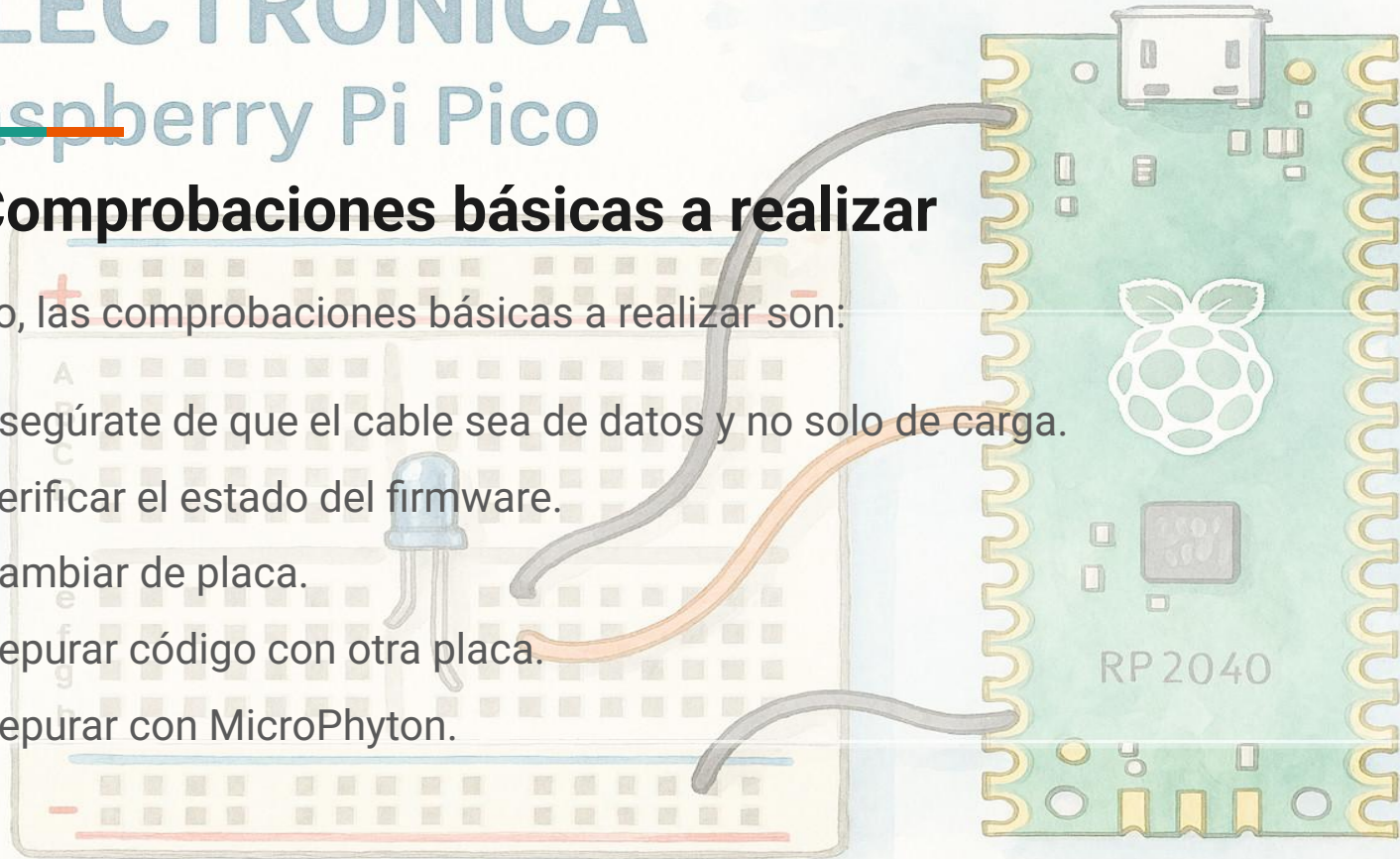
# ELECTRONICA

## Raspberry Pi Pico

### Comprobaciones básicas a realizar

Por ello, las comprobaciones básicas a realizar son:

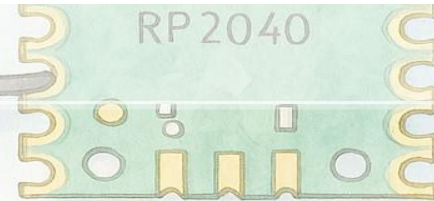
1. Asegúrate de que el cable sea de datos y no solo de carga.
2. Verificar el estado del firmware.
3. Cambiar de placa.
4. Depurar código con otra placa.
5. Depurar con MicroPython.



# ELECTRONICA

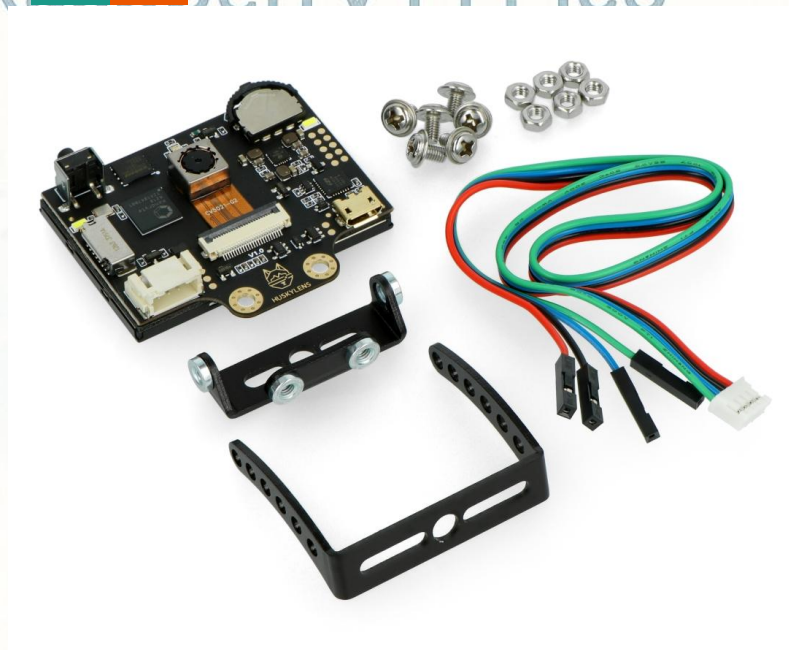
## Agenda de Huskylens

1. Presentación del producto y elementos principales.
2. El hardware.
3. El software.
4. Prácticas.
5. Comprobaciones básicas a realizar.



# ¿Qué hay dentro de la caja?

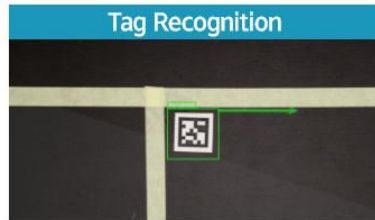
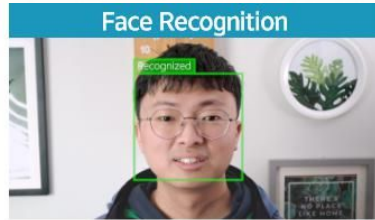
## Raspberry Pi Pico



- HuskyLens.
- 6 tornillos M3.
- 6 tuercas M3.
- Soporte de montaje pequeño.
- Soporte de altura.
- Cable sensor Gravity de 4 pines.



# ¿Para qué sirve Huskylens?



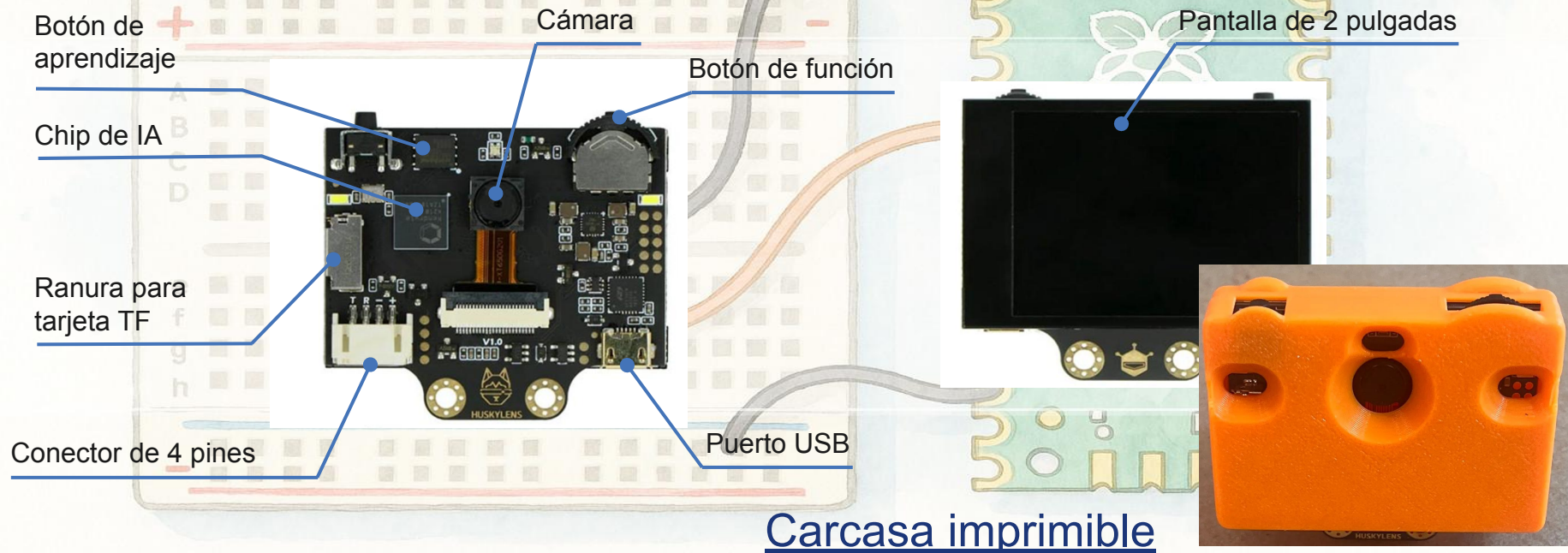
Cámara con inteligencia artificial diseñada para proyectos de robótica, IoT y automatización:

1. Reconocimiento facial.
2. Seguimiento de objetos.
3. Reconocimiento de objetos.
4. Seguimiento de línea.
5. Detección de color.
6. Detección de etiquetas.
7. Clasificación de objetos.

¿Es más potente ¿Huskylens o la Pico?

# ELECTRONICA

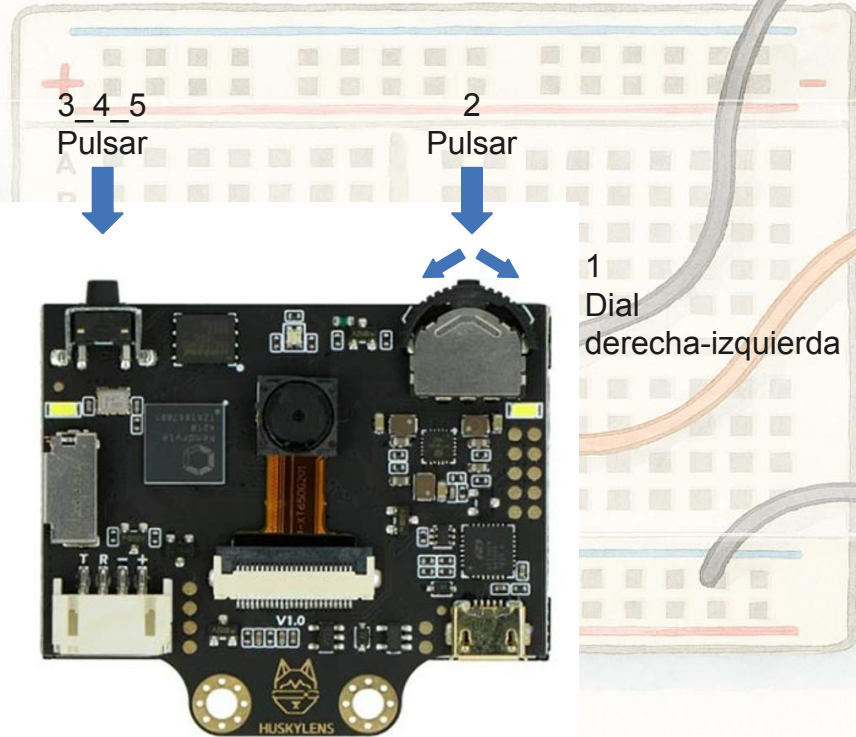
## Huskylens Raspberry Pi Pico



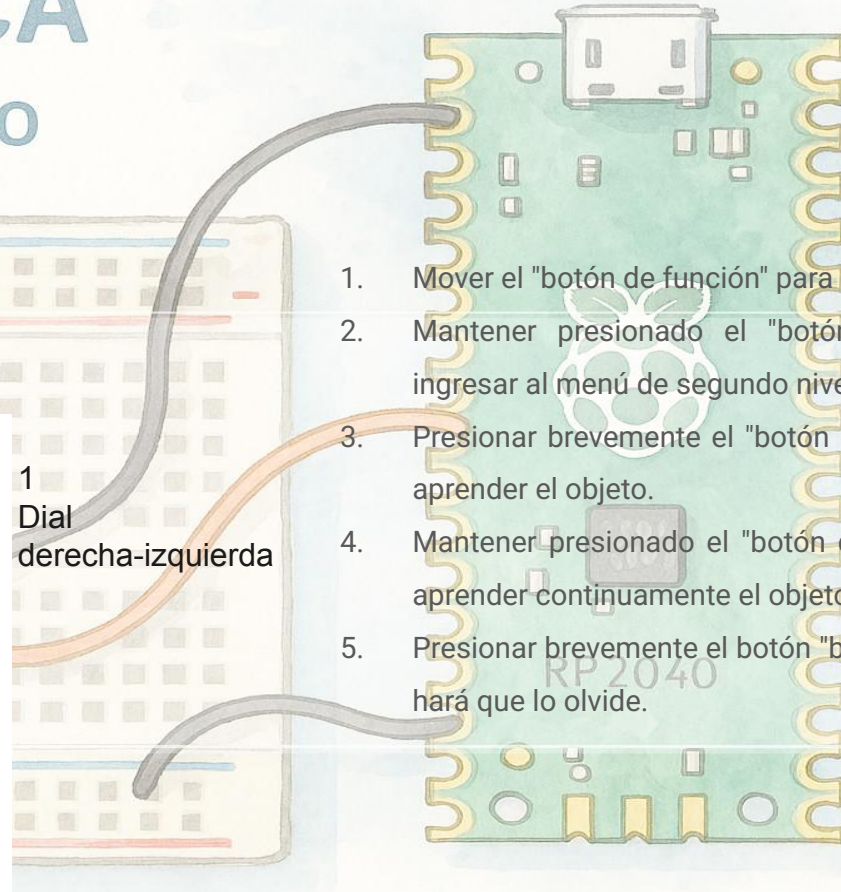
# ELECTRONICA

## Raspberry Pi Pico

### Botones Huskylens



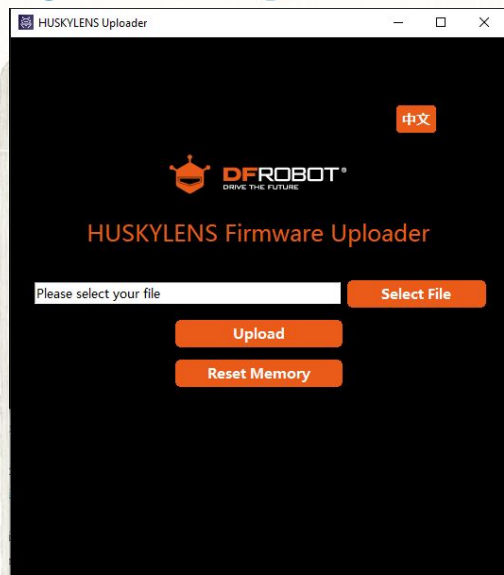
1. Mover el "botón de función" para cambiar funciones.
2. Mantener presionado el "botón de función" para ingresar al menú de segundo nivel y para seleccionar.
3. Presionar brevemente el "botón de aprendizaje" para aprender el objeto.
4. Mantener presionado el "botón de aprendizaje" para aprender continuamente el objeto
5. Presionar brevemente el botón "botón de aprendizaje" hará que lo olvide.






# ELECTRONICA

## Actualizar Firmware Raspberry Pi Pico



1. Descarga el cargador de HuskyLens (Uploader)
  - Descomprime el archivo.
2. Descarga el Driver USB.
3. Descarga el firmware más moderno
4. Conecta HuskyLens a un puerto USB
5. Ejecuta el cargador:
  - Selecciona el archivo del Driver USB (select file).
  - Pulsa Upload.

**Nota 1:** Si el cargador solicita el puerto COM, pulsa en el símbolo  de la barra inferior izquierda, haz clic con el botón derecho y selecciona "Administrador de dispositivos".

**Nota 2:** Si falla la carga, pulse "Reset Memory". Deja pasar un tiempo hasta encenderse dos luces en la HuskyLens.

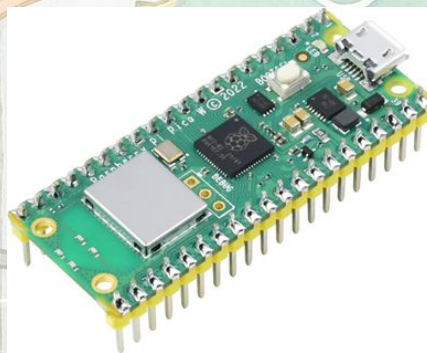


# ¿Cómo funciona el Huskylens?

## ELECTRONICA

### Raspberry Pi Pico

HuskyLens **NO** funciona de forma independiente; **NO** necesita una placa controladora, como Raspberry Pi Pico, para interpretar los datos y ejecutar acciones.



# Conecta la HuskyLens con Pico

## ELECTRONICA

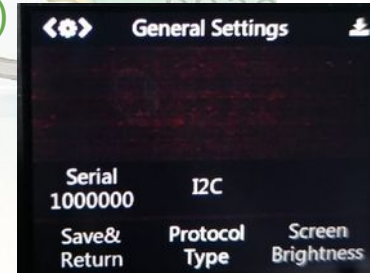
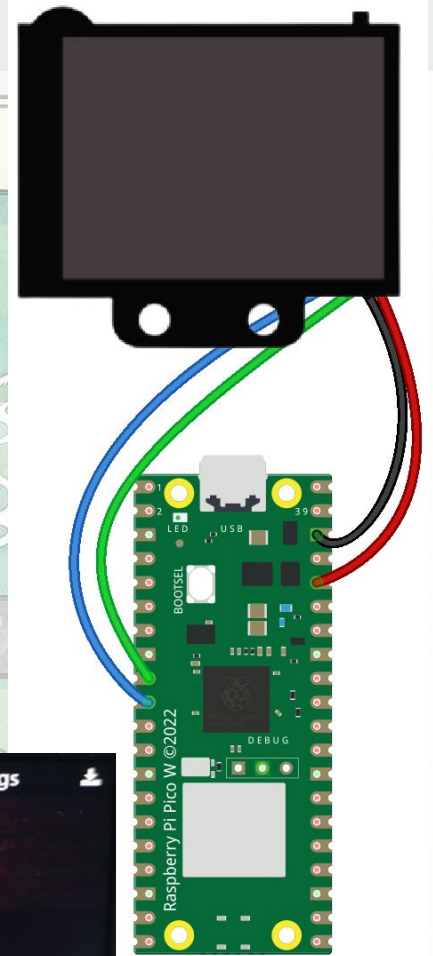
### Raspberry Pi Pico

#### Hardware

Conecta los cables I2C de la HuskyLens a los pines de la Raspberry Pi Pico de esta forma:

- VCC → 3,3V (Pin 36. Se encarga de la alimentación)
- GND → GND (Pin 38, o cualquier GND. Es la toma de tierra)
- SDA → GPIO 6 (SDA) (Pin 6. Permite paso de datos)
- SCL → GPIO 7 (SCL) (Pin 7. Sincroniza)

Configura tu HuskyLens para comunicación I2C



fritzing

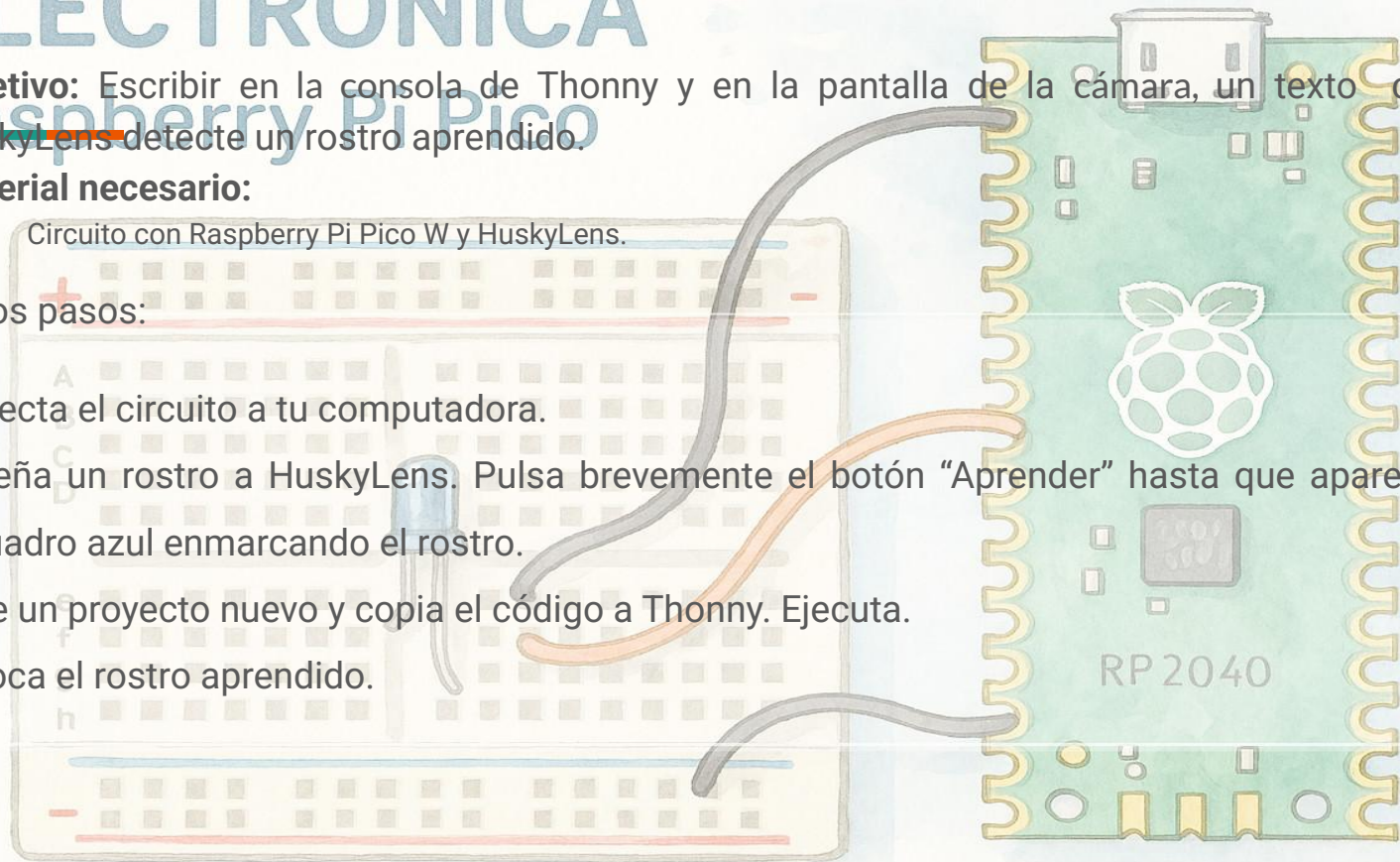


# Detectando caras desde la Pico

- **Objetivo:** Escribir en la consola de Thonny y en la pantalla de la cámara, un texto cuando HuskyLens detecte un rostro aprendido.
- **Material necesario:**
  - Circuito con Raspberry Pi Pico W y HuskyLens.

Sigue estos pasos:

1. Conecta el circuito a tu computadora.
2. Enseña un rostro a HuskyLens. Pulsa brevemente el botón "Aprender" hasta que aparezca un recuadro azul enmarcando el rostro.
3. Abre un proyecto nuevo y copia el código a Thonny. Ejecuta.
4. Enfoca el rostro aprendido.



# Memorizando Caras



# ELECTRONICA

## Raspberry Pi Pico

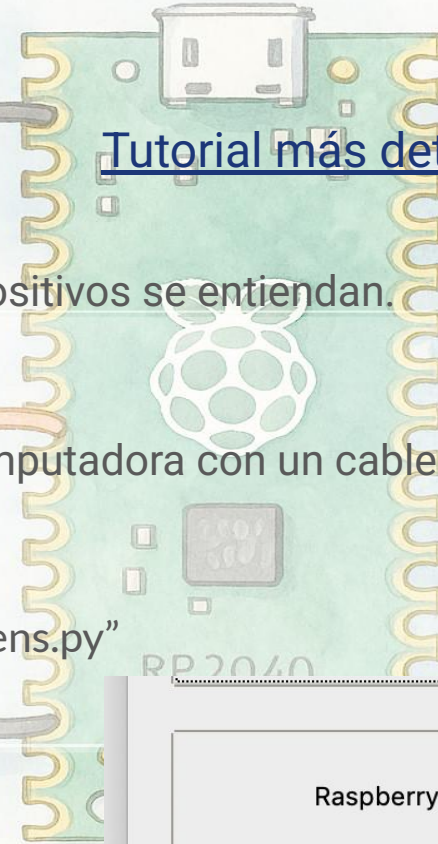
### El software

Es imprescindible usar [una librería](#) para que ambos dispositivos se entiendan.

Sigue estos pasos:

1. Conecta la Raspberry Pi Pico con el circuito a tu computadora con un cable USB.
2. Abre Thonny y conecta el USB de la Pico
3. Abre un programa nuevo y Copia el [programa](#)
4. Descarga y Copia la [librería](#) con el nombre "pyhuskylens.py"
5. Guarda DENTRO de la Pico.
6. EJECUTA.

[Tutorial más detallado](#)



Raspberry Pi Pico



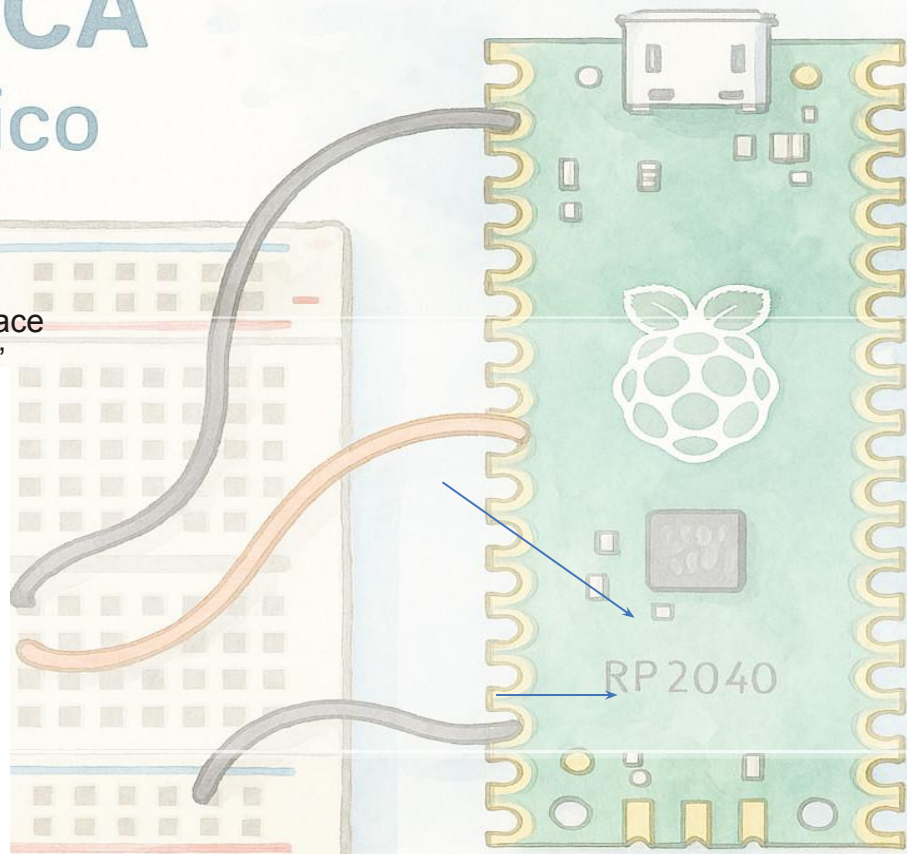
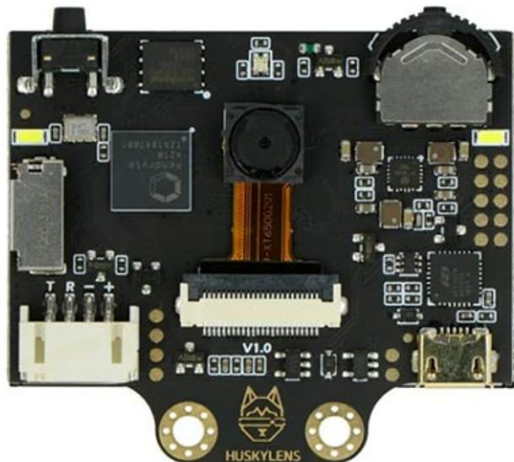
# Detectando caras desde la Pico: Pruebas

## ELECTRONICA

### Raspberry Pi Pico

Pulsar para  
aprender

Seleccionar "Face  
Recognition"



# ELECTRONICA

## Raspberry Pi Pico

### Comprobaciones básicas a realizar

Por ello, las comprobaciones básicas a realizar son:

1. Comprobar la alimentación de Huskylens ¡la cámara consume mucho!
2. Revisa si la cámara está configurada con el tipo de detección que quieres usar.
3. Revisar las conexiones a la placa de control.



# ELECTRONICA

## Raspberry Pi Pico

### Apéndice: Usando Pico + HuskyLens con microblocks

<http://microblocks.fun/> Entorno visual de bloques para programar la Pico incluso con la HuskyLens

The screenshot shows the MicroBlocks web interface in a browser. The interface includes a sidebar with categories like Output, Input, Pins, Control, Operators, Variables, Data, and My Blocks. A 'Connect' button is in the top right. A 'File Open' dialog is open, showing a file named 'HuskyLens' under the 'AI' category. A 'graph' block is visible in the workspace. A 'when started' block is followed by a 'forever' loop containing 'set user LED' and 'wait 500 milliseconds' blocks.

1 Conectamos la Pico

2 Actualizamos el firmware

3 Podemos usar bloques...

4 Importar librerías

5 AI → HuskyLens

¡¡Gracias Javier!!



# Apéndice II: Usando micro:bit + HuskyLens con Makecode



## Proyecto Portero automático con Makecode

Microsoft | micro:bit Bloques JavaScript

Buscar...

- Básico
- Entrada
- Música
- LED
- HuskyLens**
- Maqueen v4
- Maqueen v5
- Radio
- Bucles

**Código "porterillo"**

```
al presionarse el botón A
  El botón A es el del Porterillo
  HuskyLens request data once and save into the result
  si HuskyLens check if frame is on screen from the result entonces
    LED izquierdo encender
    LED derecho apagar
    Los leds de maqueen actúan como el cierre de la puerta
    mostrar icono
  si no
    LED izquierdo apagar
    LED derecho encender
    mostrar icono
```

**2 Inicialización**

```
al iniciar
  Usamos comunicaciones I2C
  huskyLens initialize I2C until success
  HuskyLens switch algorithm to Face Recognition
```

**3**

Usaremos Makecode y las extensiones de maqueen y HuskyLens

[Vídeo](#)

# ELECTRONICA

## Raspberry Pi Pico

**¡Muchas gracias!**



by [@javacasm](#)