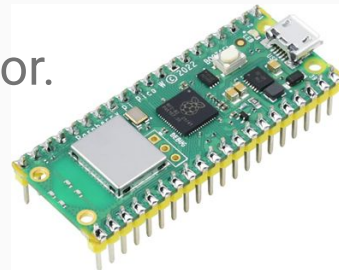


# Formación Raspberry Pi Pico y HuskyLens

- Presentación (15m): Formación, Materiales. Evaluación del nivel de los asistentes
- Introducción a Python (30m). Python, Thonny, Hola Mundo
- Raspberry Pi Pico (30m). micropython. Firmware de micropython
- HuskyLens (30m). Capacidades. Uso de la cámara de manera independiente

## Descanso (15m)

- Primeros montajes electrónicos (75m). Protoboard. Led. Pulsador. Potenciómetro
- HuskyLens & Pico (30m)
- Revisión del material de la formación (15m)

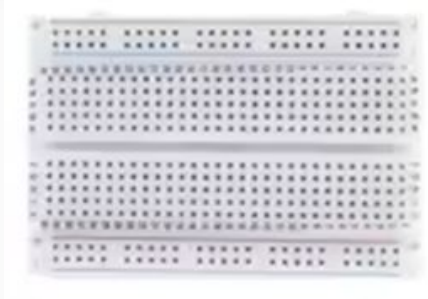


# Formación Pico

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



# ¿Qué hay dentro de la caja?



- **Microcontrolador:** Raspberry Pi Pico W.
- **Conectores y estructura:**
  - 5 cables.
  - Breadboard.
  - 40 pins extra.
- **Resistencias:**
  - 5 resistencias de 220  $\Omega$ .
  - 5 resistencias de 10 k $\Omega$ .



# ¿Qué hay dentro de la caja?



- **Dispositivos de salida:**
  - 5 LEDs de diferentes colores.
  - 1 LED RGB.
- **Dispositivos de entrada:**
  - 2 pulsadores.
  - 1 potenciómetro.
  - 2 fotorresistencias de 2 M $\Omega$ .

# Raspberry Pi Pico W

La Raspberry Pi Pico W es una placa de desarrollo pequeña y potente, diseñada para controlar proyectos electrónicos como robots, sensores, luces y motores.

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



# ¿Qué características tiene?

- 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](#)



# ¿Para qué se utiliza?

Para enseñar a los estudiantes los fundamentos de la electrónica y la automatización:

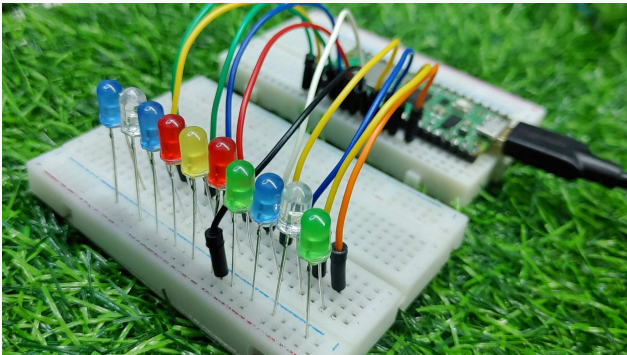
- **Automatización** (encender luces, controlar relés, abrir puertas).
- **Robótica** (control de motores, sensores y actuadores).
- **Internet de las cosas (IoT)** (enviar datos a internet, domótica).
- **Sensores** (temperatura, luz, humedad, movimiento).
- **Control de pantallas y LEDs** (matrices LED, displays OLED/LCD).
- **Sistemas portátiles** (dispositivos de bajo consumo con batería).
- **Proyectos educativos** (aprendizaje de programación y electrónica).
- **Instrumentación** (medición de señales analógicas y digitales).
- **Audio y señales** (generación de tonos, análisis de señales).



# ¿Cómo funciona todos los microcontroladores?

Combinando el **hardware** (parte física) con el **software** (programación):

1. **Hardware:** ¿Cómo se conectan y funcionan los componentes físicos?
2. **Software:** ¿Cómo dar órdenes a la Raspberry Pi Pico W mediante código?





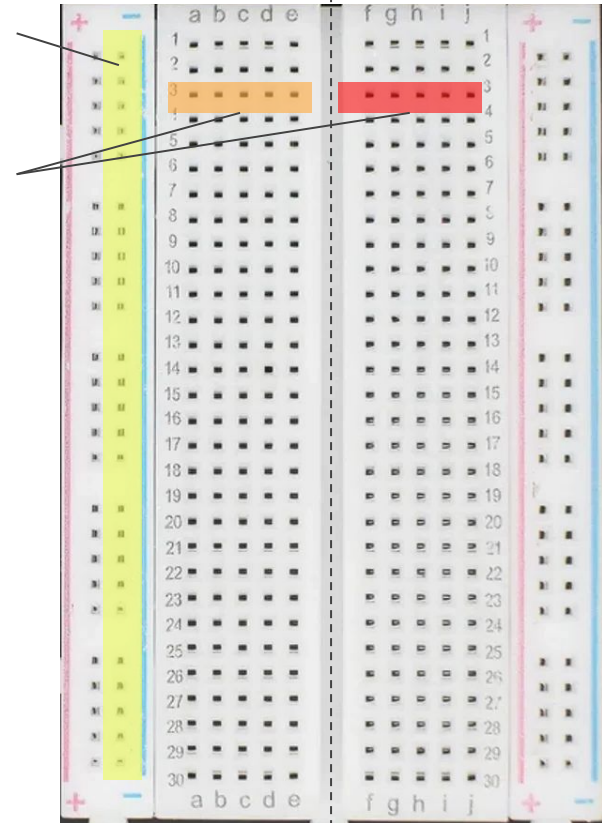
# El hardware: Protoboard

## 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 *fghij*). Estas dos partes **NO están conectadas entre ellas**.

Interconectados  
por columnas

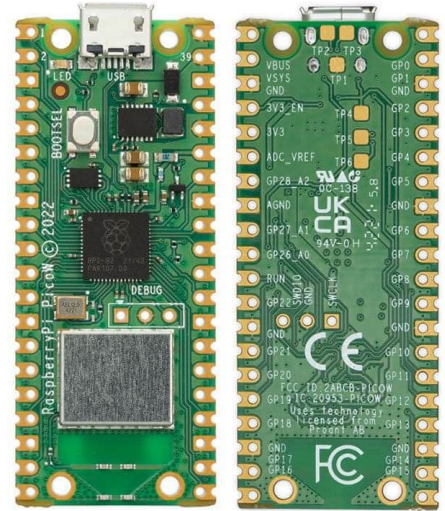
Interconectados  
por filas (derecha e  
izquierda)



# 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 .

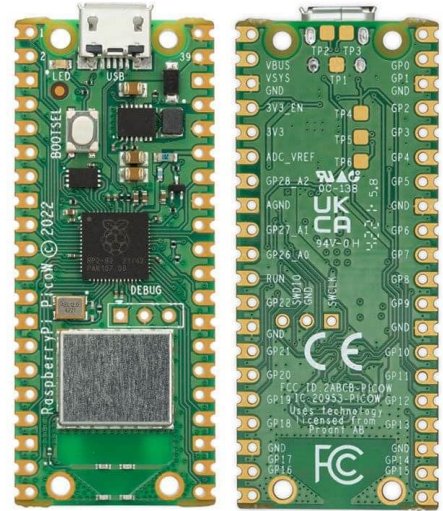


# El hardware: las entradas y salidas (GPIO)

## ¿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).



# El hardware: GPIO

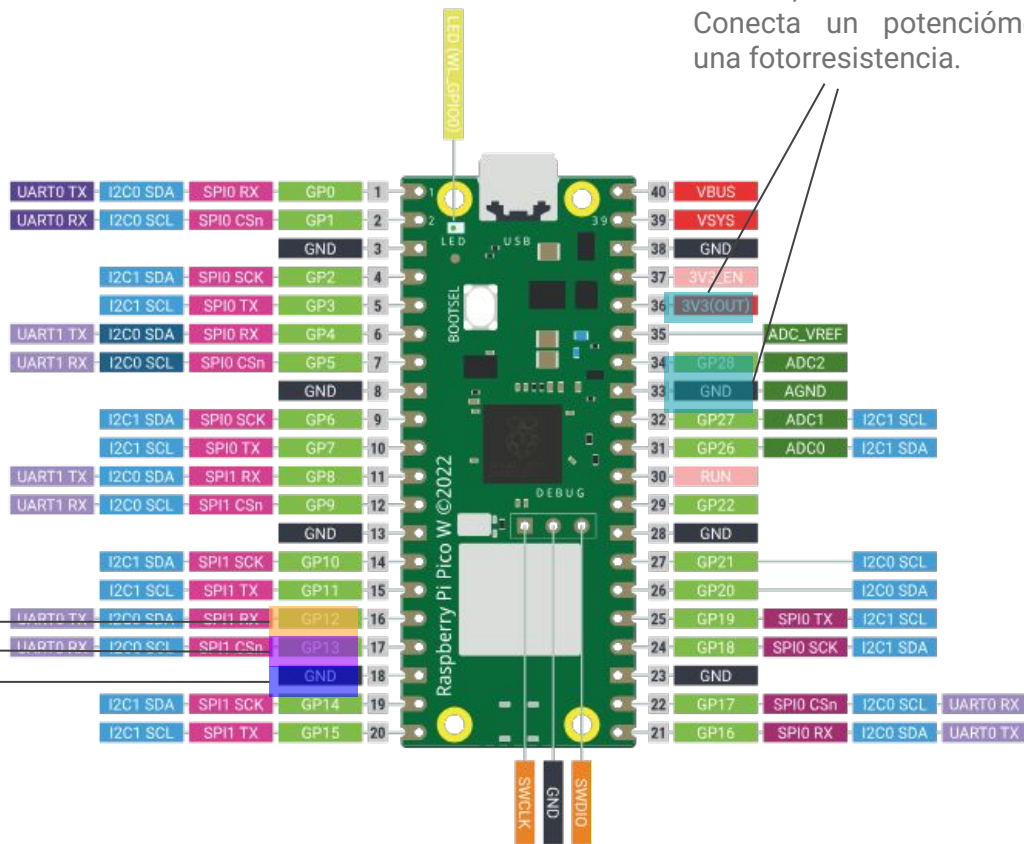
## Los pines:

GPIO14:  
Conecta  
un botón

GND:  
Cierra  
un circuito

GPIO15:  
Conecta  
un LED

GPIO26, GPIO27 o GPIO28:  
Conecta un potenciómetro o  
una fotorresistencia.



RP2040

- Power
- Ground
- UART / UART (default)
- GPIO, PIO, and PWM
- ADC
- SPI / SPI (default)
- I2C / I2C (default)
- System Control
- Debugging

Infinion 43439

- GPIO

[Imagen ampliada](#)

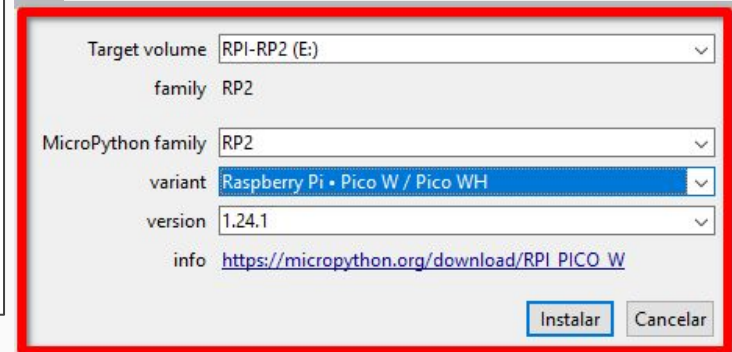
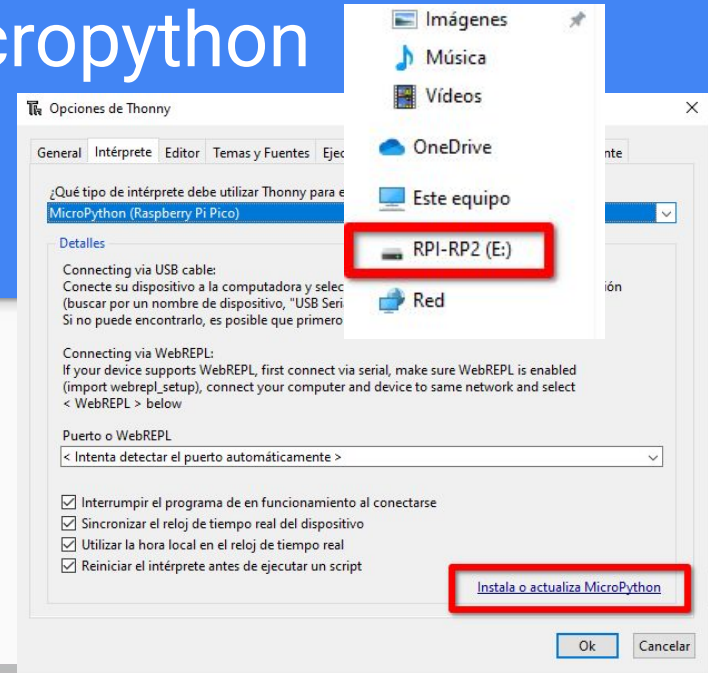
# Instalación del firmware de Micropython

Thonny:



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

1. Descarga e instala el software Thonny (<https://thonny.org/>)
2. Conecta la Pico W al ordenador mediante un cable USB.
3. Abre Thonny, ve a "Ejecutar", pestaña "Intérprete":
  - Selecciona "MicroPython (Raspberry Pi Pico)".
  - Pulsa en "Instala o actualiza Micropython".
  - Selecciona las características de tu Pico
  - Pulsa Instalar
4. Ya puedes seleccionar tu puerto para usar tu Pico

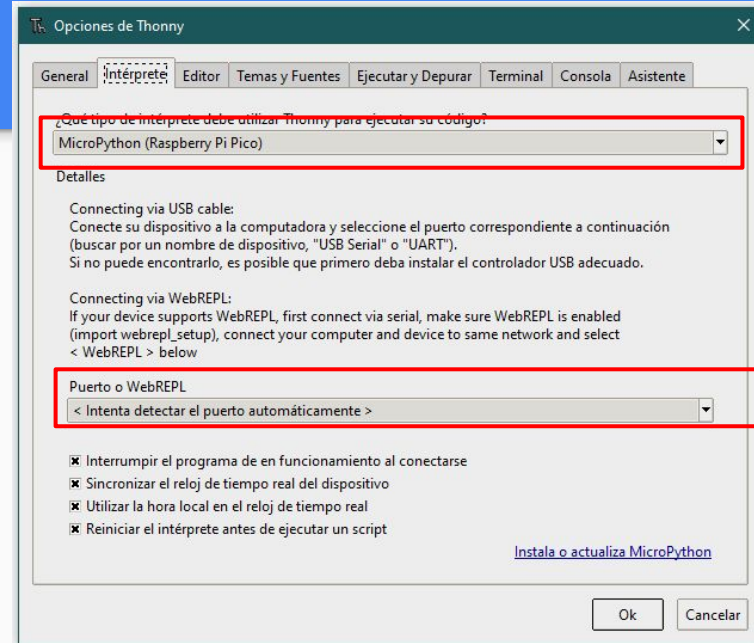


# El software para programar

Thonny: 

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

1. Conecta la Pico W al ordenador mediante un cable USB.
2. Abre Thonny, ve a "Ejecutar", pestaña "Intérprete":
  - Selecciona "MicroPython (Raspberry Pi Pico)".
  - "Puerto o WebREPL" -> selecciona el puerto.





# Práctica 0: LED integrado en la Pico

## Parte de software:

### Código python

- 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*

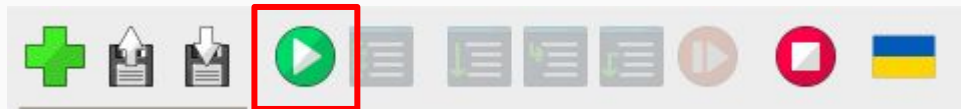
*Enciende el LED*

*Espera 3 segundos*

*Apaga el LED*




- Ejecuta el programa:





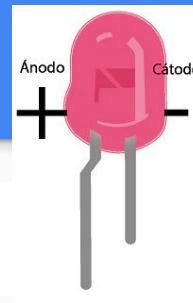
# Práctica 1: Y fue la luz

## Parte de hardware:

- **Objetivo:** encender un LED 3 segundos
- **Material necesario:**
  - Breadboard.
  - Microcontrolador.
  - LED.
  - 2 cables.
  - Resistencia 220Ω 

## Tu circuito:

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





# Práctica 1: Y fue la luz

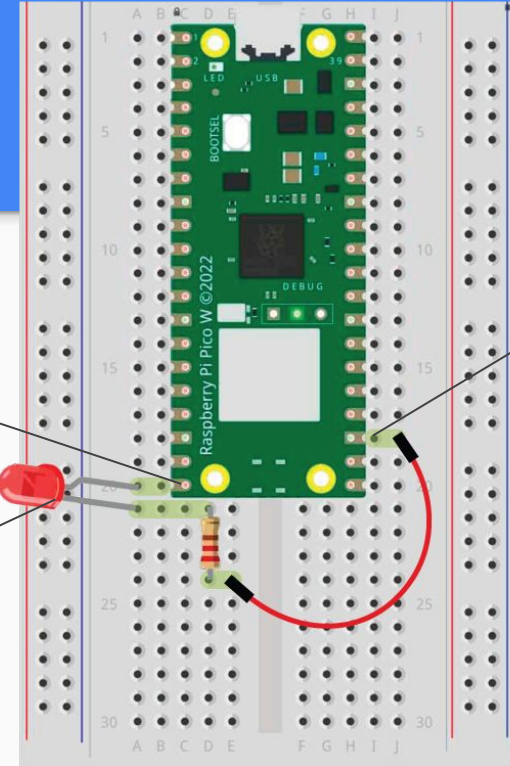
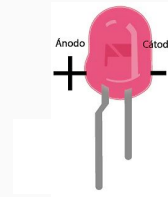
## Parte de hardware

### Tu circuito

Conecta los elementos de esta forma:

GPI015  
Salida digital

GND  
Tierra



**GPI015 -> LED -> resistencia -> GND**



# Práctica 1: Y fue la luz

## Parte de software

### Código python

- 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*


- Ejecuta el programa:



# Práctica 2: Pulsa y brilla



## 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):
  - GPIO14 -> Botón -> GND

# Práctica 2: Pulsa y brilla

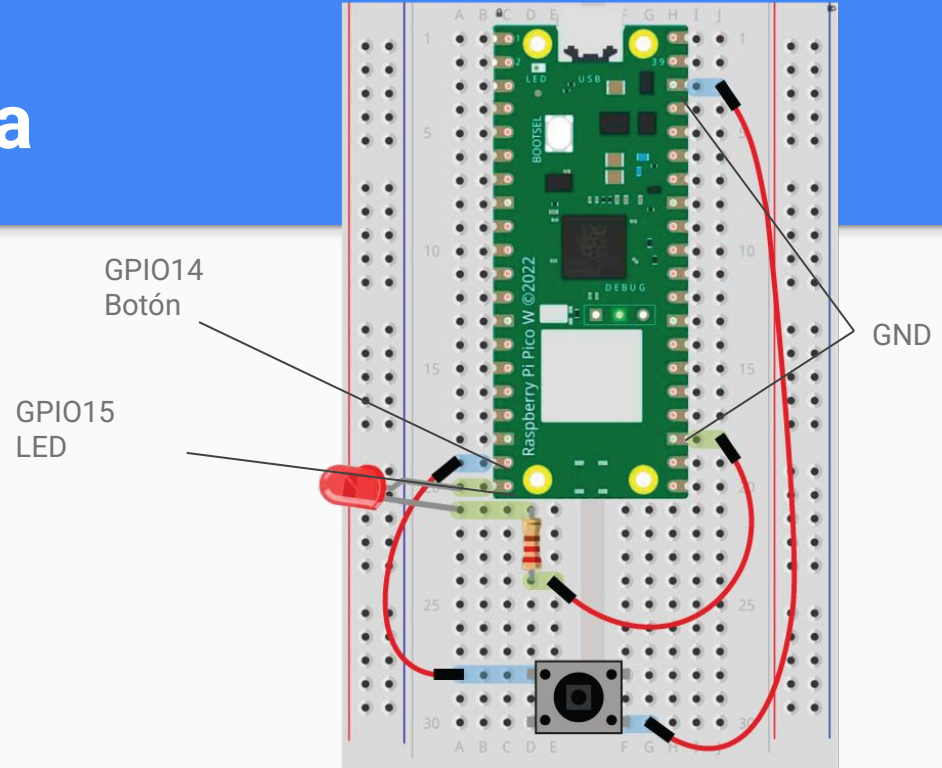
## Parte de hardware

## Tu circuito

Conecta los elementos de esta forma:

**GPIO15 -> LED -> resistencia-> GND**

**GPIO14 -> Botón -> GND**





# Práctica 2: Pulsa y brilla

## 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_DOWN)
while True:
    if boton.value():
        led.on()
    else:
        led.off()
```

- Ejecuta el programa




- Importa la librería para controlar los pines GPIO*
- Configura el GPIO15 como salida para controlar el LED*
- Configura el GPIO14 como entrada con resistencia pull-down*
- Bucle infinito para comprobar continuamente el estado del botón*
  - Si el botón está presionado (valor 1, HIGH)
    - Enciende el LED
  - Si el botón no está presionado (valor 0, LOW)
    - Apaga el LED

# Práctica 3: Ajusta la intensidad



## 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

# Práctica 3: Ajusta la intensidad

Parte de hardware:

Tu circuito

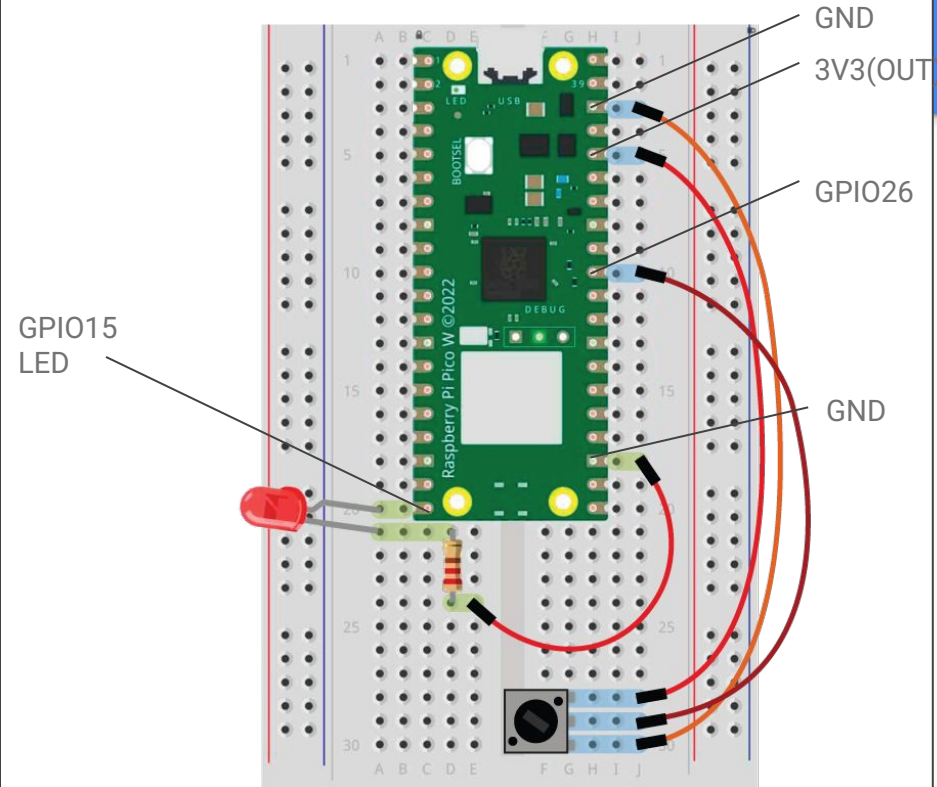
Conecta los elementos de esta forma:

**GPI015 -> LED -> resistencia -> GND**

**Pata superior -> 3V3(OUT)**

**Pata de en medio -> GPIO26**

**Pata inferior -> GND**





# Práctica 3: Ajusta la intensidad

## 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





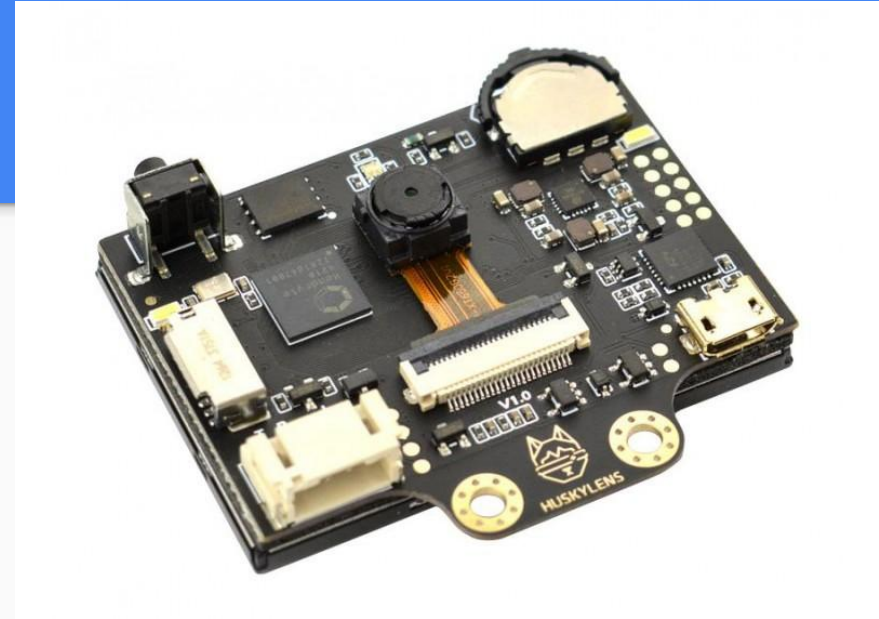
# Comprobaciones básicas a realizar

Por ello, las comprobaciones básicas a realizar son:

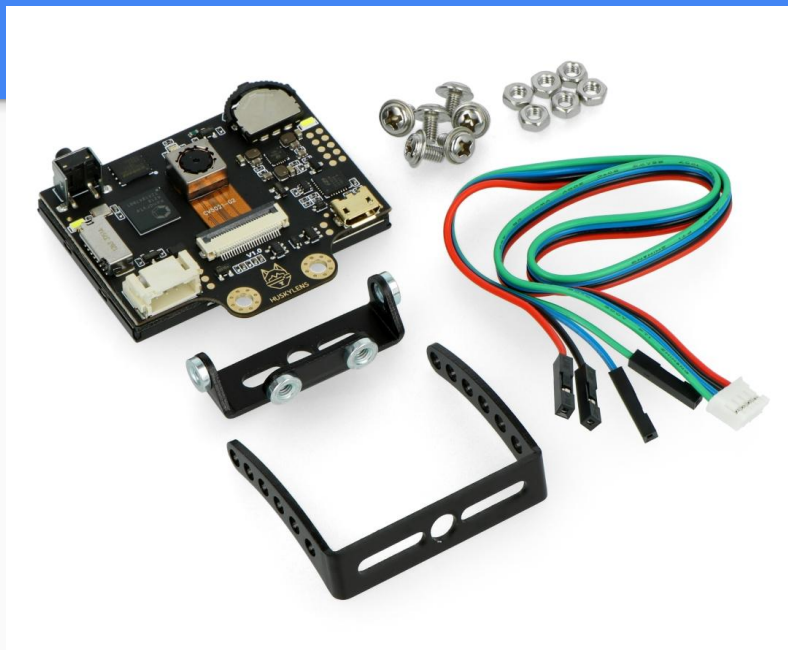
1. Asegúrate de que el cable sea de datos y no solo de carga.
2. Verificar el estado del firmware.
3. Cambiar de placa.
4. Depurar código con otra placa.
5. Depurar con MicroPhyton.

# Agenda de Huskylens

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

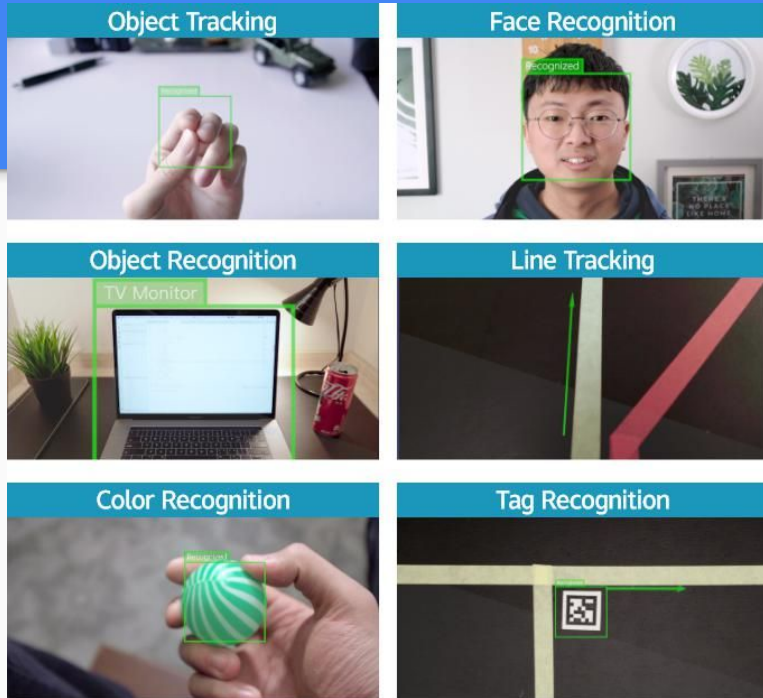


# ¿Qué hay dentro de la caja?



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

# ¿Para qué sirve Huskylens?

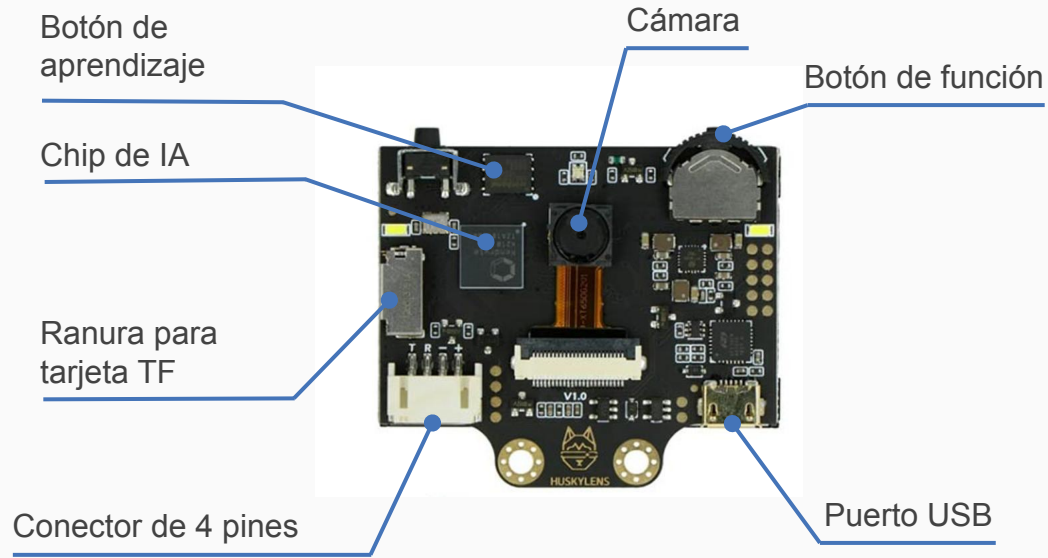


Cámara con inteligencia artificial diseñada para proyectos de robótica, IoT y automatización:

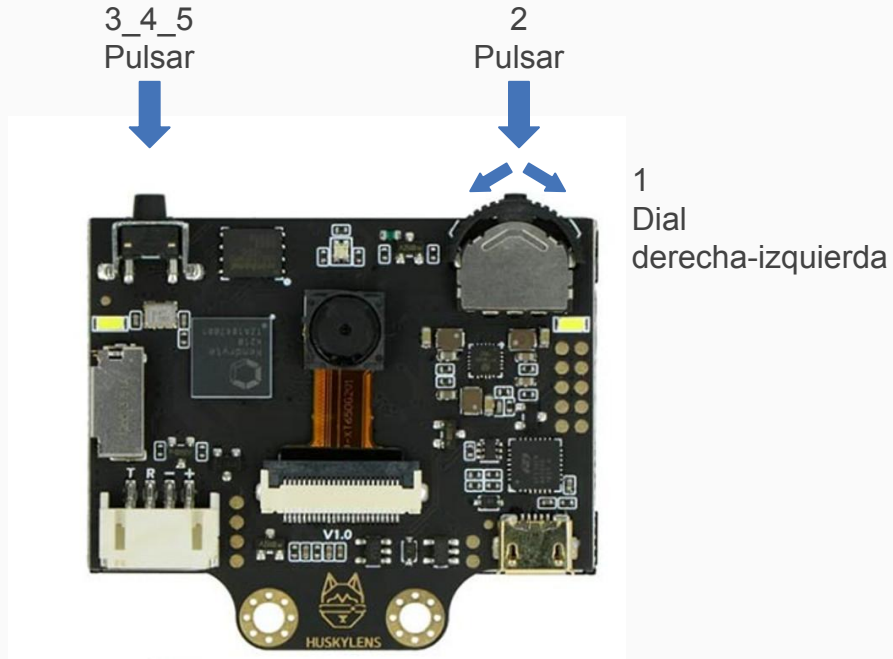
1. Reconocimiento facial.
2. Seguimiento de objetos.
3. Reconocimiento de objetos.
4. Seguimiento de línea.
5. Detección de color.
6. Detección de etiquetas.
7. Clasificación de objetos.

¿Es más potente ¿Huskylens o la Pico?

# HuskyLens

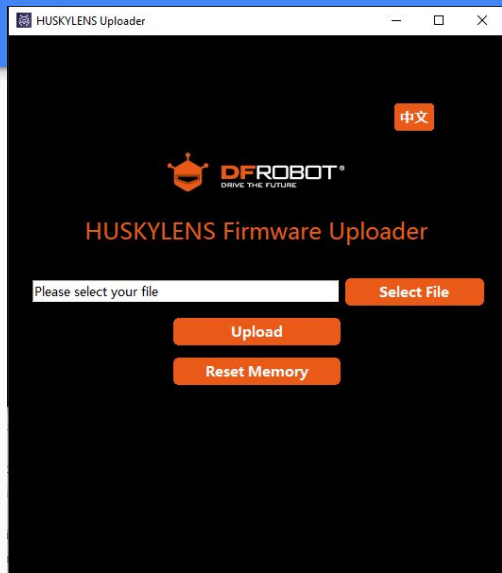


# Botones Huskylens




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.

# Actualizar Firmware



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.

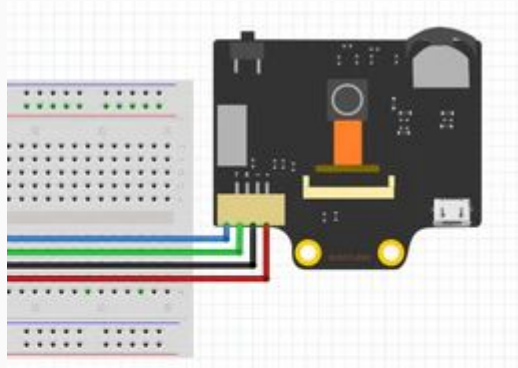
**Nota 1:** Si el cargador solicita el puerto COM, pulsa en el símbolo  de la barra inferior izquierda, haz clic con el botón derecho y selecciona "Administrador de dispositivos".

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

# ¿Cómo funciona el Huskylens?

Combinando el **hardware** (parte física) con el **software** (programación):

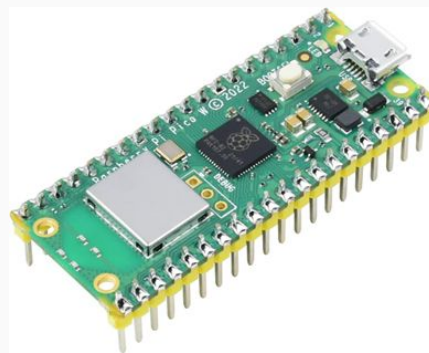
1. **Hardware:** ¿Cómo se conectan y funcionan los componentes físicos?
2. **Software:** ¿Cómo dar órdenes a HuskyLens mediante código?
3. ¿Necesita HuskyLens software externo?





# ¿Cómo funciona el Huskylens?

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



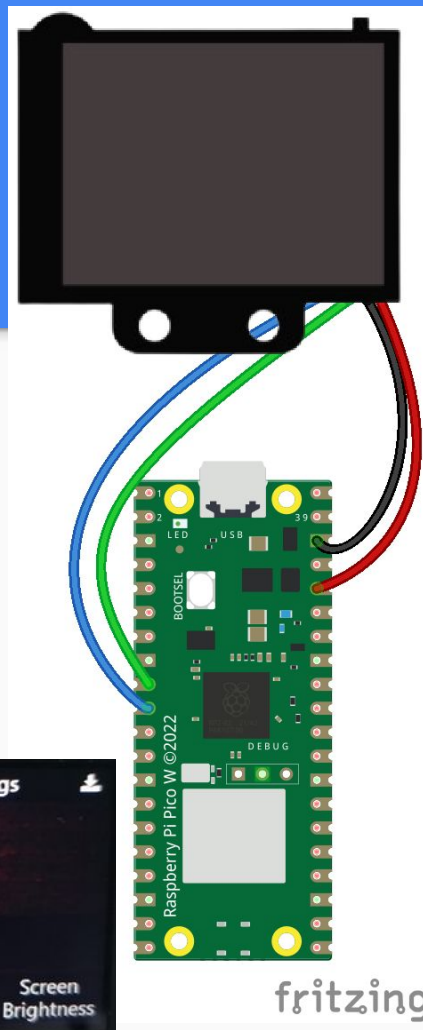
# ¿Cómo se conecta la HuskyLens con Raspberry Pi Pico W?

## 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



# El software

[Tutorial detallado](#)

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. Copia la [librería](#) con el nombre “pyhuskylens.py”
5. Guarda DENTRO de la Pico.
6. EJECUTA.



# Práctica 1: Primer reconocimiento

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

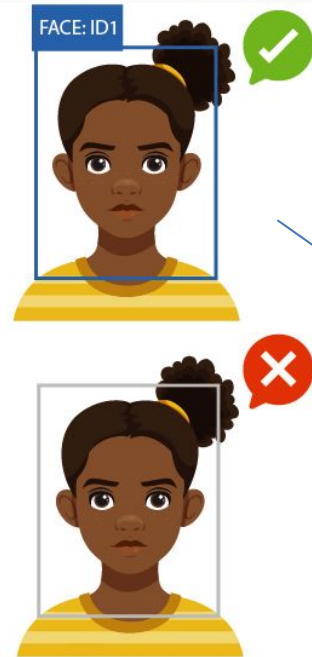
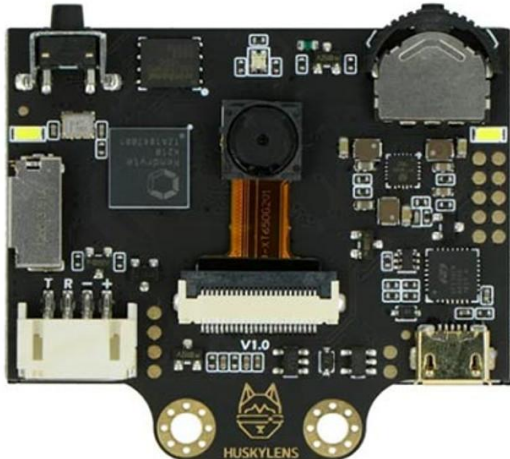
Sigue estos pasos:

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.

# Práctica 1: Primer reconocimiento

Pulsar para aprender

Seleccionar "Face Recognition"



X	No se detecta rostro
X	No se detecta rostro
X	No se detecta rostro
X	No se detecta rostro
X	No se detecta rostro
X	No se detecta rostro
X	No se detecta rostro
X	No se detecta rostro
😊	Rostro reconocido
😊	Rostro reconocido
X	No se detecta rostro
X	No se detecta rostro

# Comprobaciones básicas a realizar

Por ello, las comprobaciones básicas a realizar son:

1. Comprobar la alimentación de Huskylens
2. Versión de Firmware.
3. Revisar la lente de la cámara.
4. Revisar las conexiones a la placa de control.

# Consultas urgentes

Para una duda específica, contacta con tu dinamizador.

# Encuesta

Por favor realiza [esta encuesta](#)



# ¡Muchas gracias!

by [@javacasm](#)