

# Agente Inteligente para “Controlar la Luz del Aula”



Susana Oubiña Falcón

## Índice

|      |                                                      |    |
|------|------------------------------------------------------|----|
| 1.   | Reto .....                                           | 2  |
| 2.   | Planificación del agente .....                       | 2  |
| 3.   | Materiales .....                                     | 2  |
| 4.   | Circuito eléctrico .....                             | 2  |
| 4.1. | Raspberry Pi Pico .....                              | 2  |
| 4.2. | Kit Inventor Nezha .....                             | 5  |
| 4.3. | Cyber Pi .....                                       | 6  |
| 5.   | Programa de control .....                            | 10 |
| 5.1. | Thonny para la Raspberry Pi Pico .....               | 10 |
| 5.2. | MakeCode para la el shield Nezha con micro:bit ..... | 12 |
| 5.3. | mBlock para la Cyber Pi .....                        | 13 |
| 6.   | Funcionamiento .....                                 | 14 |
| 7.   | Posibles mejoras .....                               | 14 |

### 1. Reto

Desarrollar un agente inteligente que controle automáticamente la iluminación del aula en función de la cantidad de luz natural disponible y la presencia de personas.

### 2. Planificación del agente

¿Qué tiene que hacer el agente inteligente?

- ✓ Detectar si hay suficiente luz natural (Sensor de luz).
- ✓ Detectar si hay personas en el aula (Sensor de movimiento).
- ✓ Encender la luz solo **si está oscuro y hay personas presentes**.

### 3. Materiales

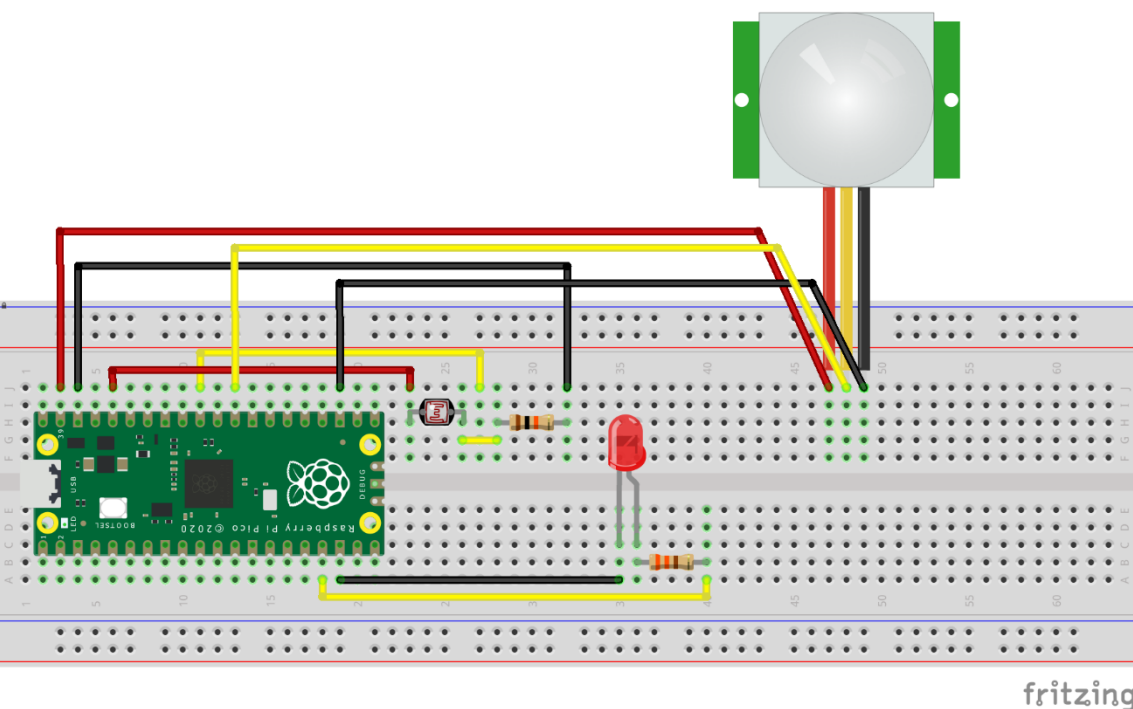
- a) Placa controladora o cerebro: Cyber Pi del mBot 2, Raspberry Pi Pico, Arduino, Nezha
- b) Sensores:
  - **Sensor de luz** (como una LDR o un módulo sensor de luminosidad para la Cyber Pi o Nezha). Detecta la cantidad de luz natural en el aula.
  - **Sensor de movimiento** (como un sensor PIR o un módulo sensor PIR para la Cyber Pi o Nezha): Detecta si hay personas en el aula.
- c) Actuadores:
  - **LED o LED RGB**: Simula la iluminación del aula.
- d) Conexionado: Cables RJ25-I2C (Cyber Pi), cables jumper con Protoboard o cables RJ11 con Nezha
- e) Ordenador: Para programar la placa y monitorear el comportamiento del agente.
- f) Software: mBlock, Thonny, Makecode

### 4. Circuito eléctrico

Dependiendo del cerebro que hemos escogido, el circuito será diferente:

#### 4.1. Raspberry Pi Pico

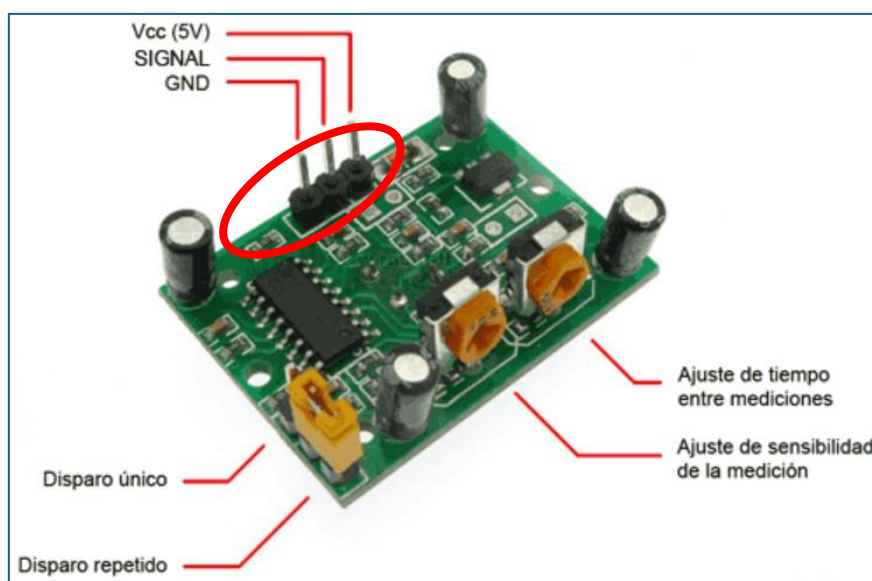
Es importante tener el [Pin Out](#) de la Raspberry Pi Pico



### Conexionado del sensor PIR:

- Conectar el pin VCC (alimentación) del sensor con un pin de 5 V (cable rojo)
- Conectar el pin de SALIDA del sensor a GP22 (cable amarillo)
- Conectar el GND del sensor a un pin GND (tierra) (cable negro)

Este sensor dispone de tres pines que son: **Vcc** y **GND** para la alimentación y el pin OUT, también llamado SEÑAL o de datos para la activación. (Ver siguiente imagen)

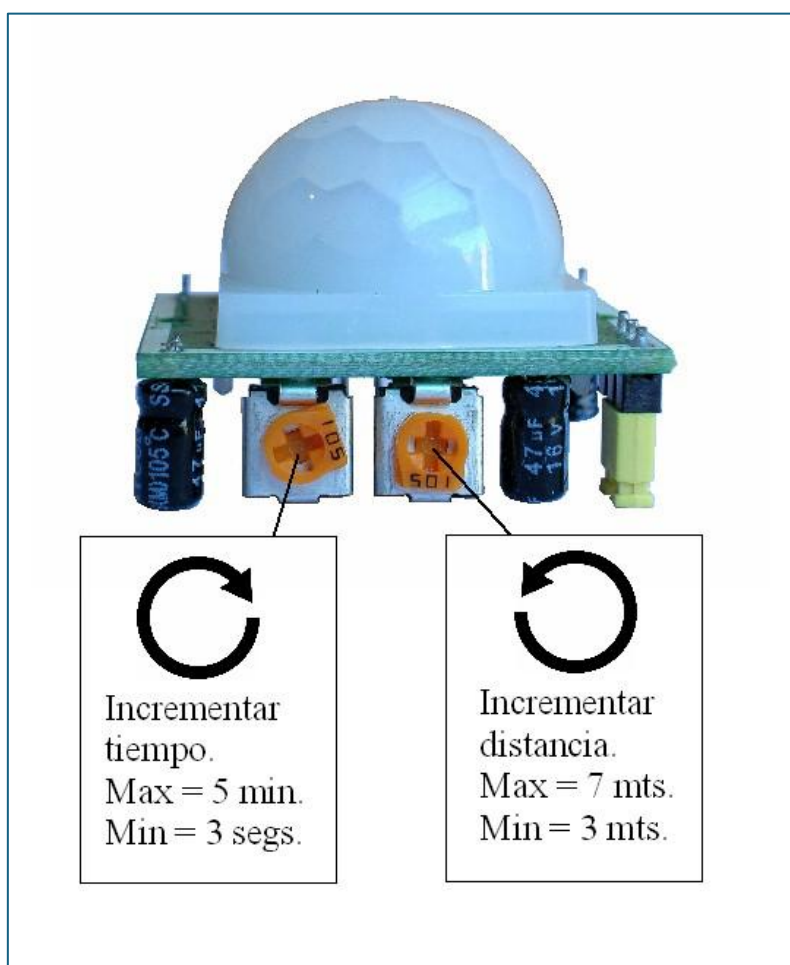


El encapsulado semiesférico de la parte superior, llamada *lente de Fresnel*, se ha diseñado para dejar pasar la **radiación infrarroja** de cualquier objeto. Pensar que:

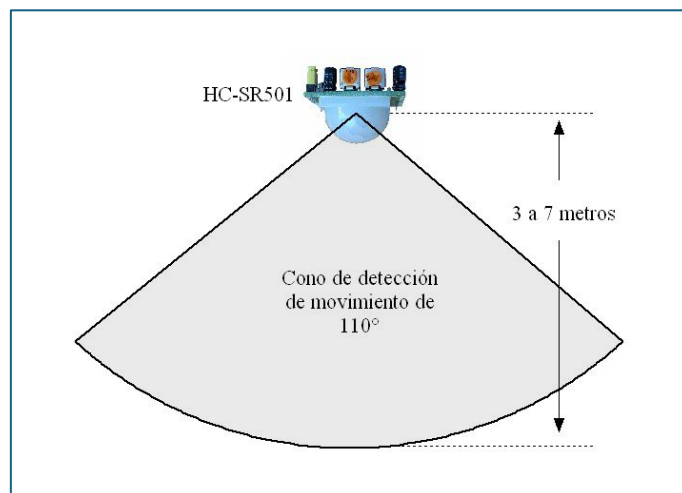
- Todos los cuerpos emiten radiación infrarroja.
- Cuanta más temperatura tenga el objeto que se pone delante del sensor PIR, más radiación infrarroja emitirá y el sensor PIR lo detectará mejor.



**IMPORTANTE:** Ajustar los dos potenciómetros de vuestro PIR antes de comenzar con la práctica. Como es una práctica en el taller, los potenciómetros deberían estar ajustados para obtener una distancia y tiempo de activación mínimos.



Relativo a su cono de detección, podemos considerarlo de  $110^\circ$ :



Al alimentar por primera vez el módulo HC-SR501, durante el **primer minuto** el módulo entra en modo de inicialización. Durante este tiempo el sensor **no puede detectar bien** el movimiento de un objeto.

### Conexión del LED:

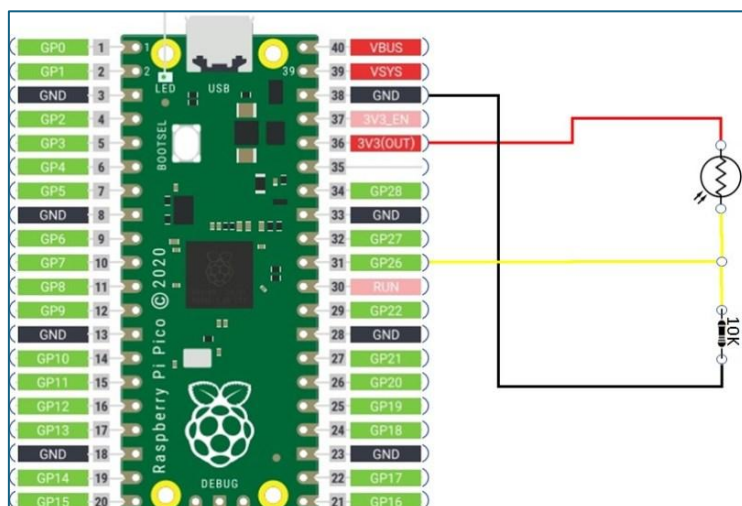
- Conectaremos el LED con una resistencia en serie de  $330\Omega$  al pin GP13, que equivale al pin 17 de la placa. Usaremos como GND el pin 18 de la placa. Si usamos LEDs de 10mm podemos omitir la resistencia.

### Conexión de la LDR:

Podemos implementarla en serie con una R de  $10K\Omega$

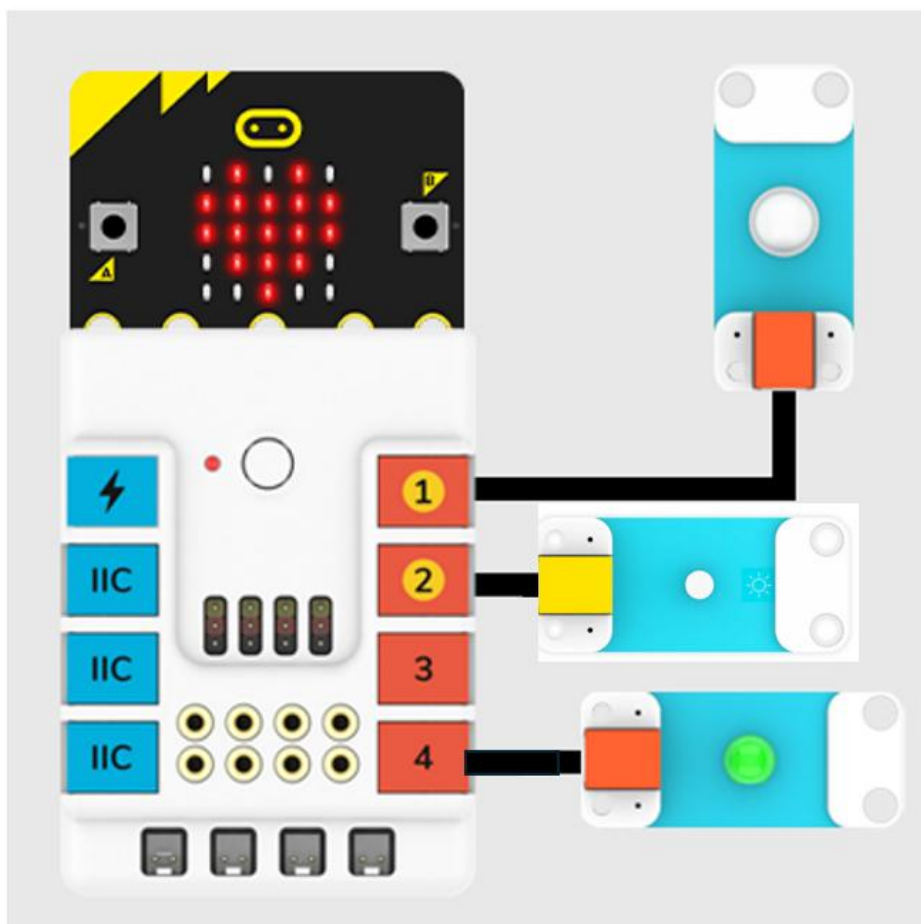
- Conectaremos un extremo de la LDR a **3.3V**.

- Conectaremos el otro extremo al pin analógico **GP26 (ADC0)** y a tierra a través de una resistencia de **10k $\Omega$**  (en un divisor de voltaje).



### 4.2. Kit Inventor Nezha

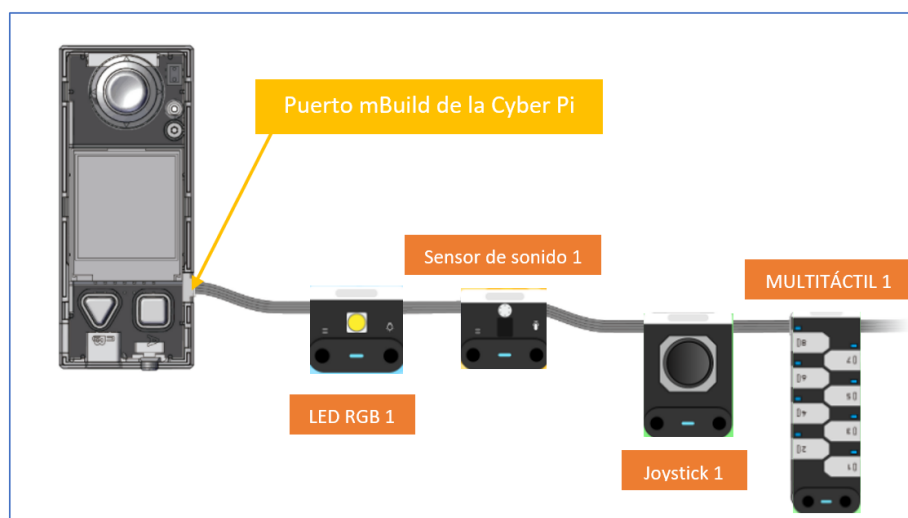
El circuito podría ser el siguiente:



Conectamos el sensor PIR al puerto J1, el sensor de luz al puerto J2 y el LED al puerto J4.

#### 4.3 Cyber Pi

Se pueden conectar varios módulos entre sí, sin especificar el puerto de conexión. La casa comercial aconseja no conectar más de 8 al puerto mBuild que los controlará:



En la imagen anterior todos los módulos son diferentes y, por lo tanto, reciben la numeración 1 a la hora de acceder a ellos mediante el mBlock5:



La conexión entre puerto mBuild y módulos o entre dos módulos se realiza con el cable mBuild de 4 conectores:



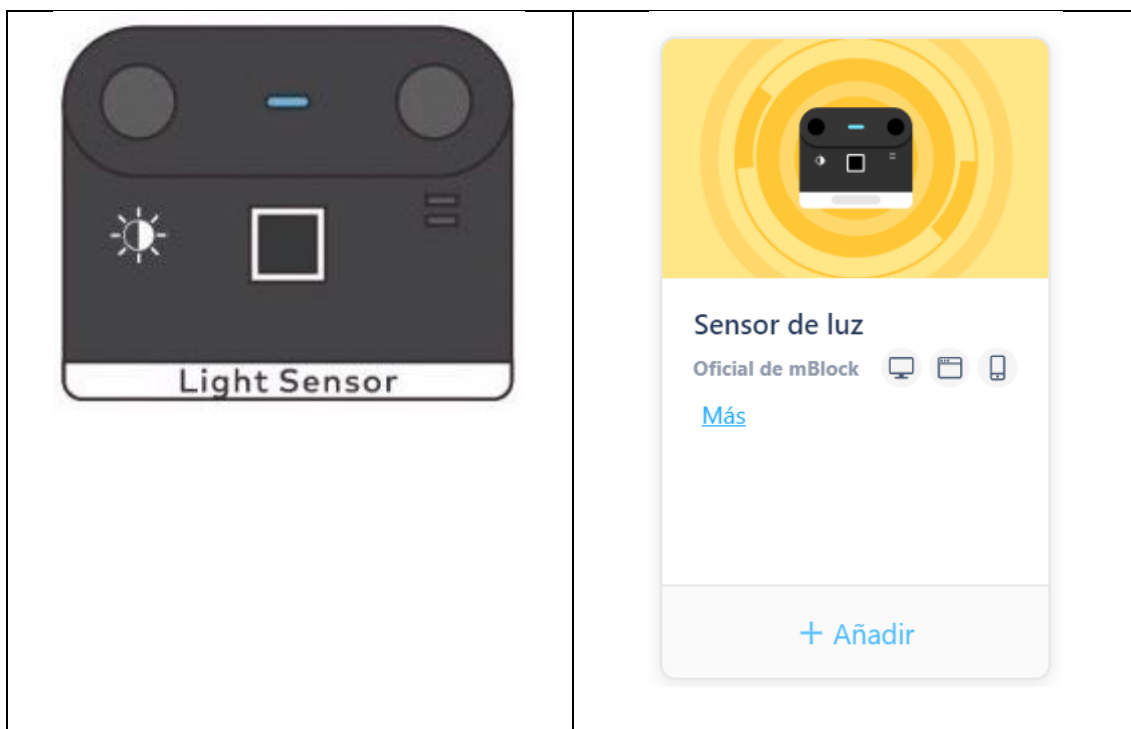
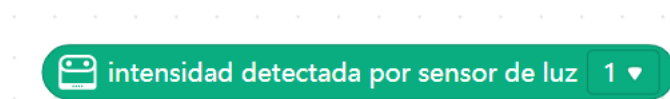
Cada módulo dispone de dos conectores mBuild y por este motivo se pueden conectar en serie, uno seguido del otro:



**Sensor de luz:** Este módulo detecta la intensidad de la luz ambiente. El sensor de luz mBuild se compone de un fotorresistor y, como tal, atendiendo a la cantidad de luz que recibe, proporciona un determinado valor de resistencia. Sus características son:

- Tamaño: 24×20mm
- Valor del rango: 0 ~ 100%
- Corriente de funcionamiento: 15mA

Tras conectar el sensor mBuild bien sea al puerto mBuild de la Cyber Pi o del robot mBot2 y conectar la Cyber Pi al software mBlock5, añadiremos la extensión del sensor en mBlock5 que, en este caso, sólo dispone de un comando:



**Sensor PIR:** Este módulo detecta el movimiento. Sus características son:

- Rango de detección: 3~5 m
- Ángulo de detección del eje X: 80°
- Ángulo de detección del eje Y: 55°

## Agente Inteligente para “Controlar la Luz del Aula”

- Duración tras la activación: 3 s

Tras conectar el sensor mBuild bien sea al puerto mBuild de la Cyber Pi o del robot mBot2 y conectar la Cyber Pi al software mBlock5, añadiremos la extensión del sensor en mBlock5 que, en este caso, sólo dispone de los siguientes comandos:



## 5. Programa de control

El software dependerá el cerebro que usemos.

### 5.1. Thonny para la Raspberry Pi Pico

- Importo las librerías y configuro los pines GP

```
i_primer_AI.py *
1 from machine import Pin, ADC
2 import time
3
4 # Configuración de pines según tu imagen
5 sensor_luz = ADC(26) # LDR conectado a GP26 (ADC0)
6 sensor_pir = Pin(22, Pin.IN, Pin.PULL_DOWN) # PIR conectado a GP22 con resistencia pull-down
7 led = Pin(13, Pin.OUT) # LED conectado a GP13
```

- Defino los umbrales para la luz y el tiempo de encendido

```
9 # Umbral de luz en porcentaje y tiempo de encendido
10 UMBRAL_LUZ = 40 # Ajustar según condiciones de luz ambiental. Con poca luz en % debe ser pequeño
11 UMBRAL_TIEMPO = 5 # En segundos para mantener la luz encendida
```

- Defino la función luz: solo debe medirse en %

```
12
13 def leer_luz():
14     """
15     Lee el valor del sensor de luz y lo convierte en porcentaje.
16     0% = máxima oscuridad, 100% = máxima luz (según Vref de 3.3V).
17     """
18     valor_bruto = sensor_luz.read_u16() # Lectura en rango 0-65535
19     porcentaje_luz = (valor_bruto / 65535) * 100 # Convertir a porcentaje
20     return porcentaje_luz
21
```

- Defino la función movimiento.

```
22 def leer_movimiento():
23     """
24     Lee el estado del sensor PIR y lo devuelve.
25     1 si detecta movimiento, 0 si no.
26     """
27     return sensor_pir.value()
```

- Defino la función controlar luz

```

29 def controlar_luz():
30     """
31     Lógica del agente inteligente para controlar la luz.
32     Si el ambiente está oscuro (porcentaje de luz bajo) y hay movimiento,
33     enciende la luz.
34     """
35     luz = leer_luz()
36     movimiento = leer_movimiento()
37
38     print(f"Luz: {luz:.2f}%, Movimiento: {movimiento}") #Imprimirá 2 decimales en el valor de luz
39
40     if luz < UMBRAL_LUZ and movimiento == 1:
41         led.value(1) # Encender LED
42         print("Luz encendida")
43         time.sleep(UMBRALE_TIEMPO) # Mantener encendida por tiempo definido
44     else:
45         led.value(0) # Apagar LED
46         print("Luz apagada")

```

**NOTA:** La **f** al inicio de `f"Luz: {luz:.2f}%, Movimiento: {movimiento}"` indica que es una **f-string**. Las **f-strings** permiten insertar valores de variables directamente dentro de una cadena, **dentro de las llaves {}**, de forma más limpia y legible.

La parte **.2f** se usa para **formatear números flotantes (números con decimales)**.

- **f:** Significa "**float**", indicando que estás trabajando con un número flotante.
- **.2:** Indica cuántos **decimales** quieres mostrar.

Lógica del agente para controlar la luz:

- Si la luz ambiental es baja (**luz < UMBRAL\_LUZ**) y hay movimiento (**movimiento == 1**), el LED se enciende.
- En caso contrario, el LED se apaga.
- Defino la función agente: ¿Qué debe hacer?

```

47
48 def iniciar_agente(): # Nuevo nombre para la función principal
49     print("Iniciando PIR...")
50     time.sleep(60) # Esperar 1 minuto para inicialización del PIR
51     print("PIR inicializado. Comenzando detección...")
52     try:
53         while True:
54             controlar_luz()
55             time.sleep(0.5)
56     except KeyboardInterrupt:
57         print("Programa detenido")
58
59 # Ejecutar el programa llamando a la nueva función
60 iniciar_agente()

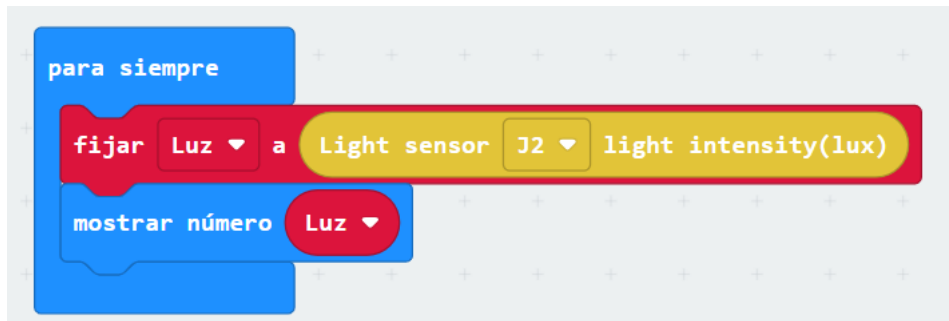
```

El sistema verifica constantemente las condiciones de luz y movimiento cada 0.5 segundos.

## 5.2. MakeCode para la el shield Nezha con micro:bit

Use MakeCode micro:bit: <https://makecode.microbit.org>

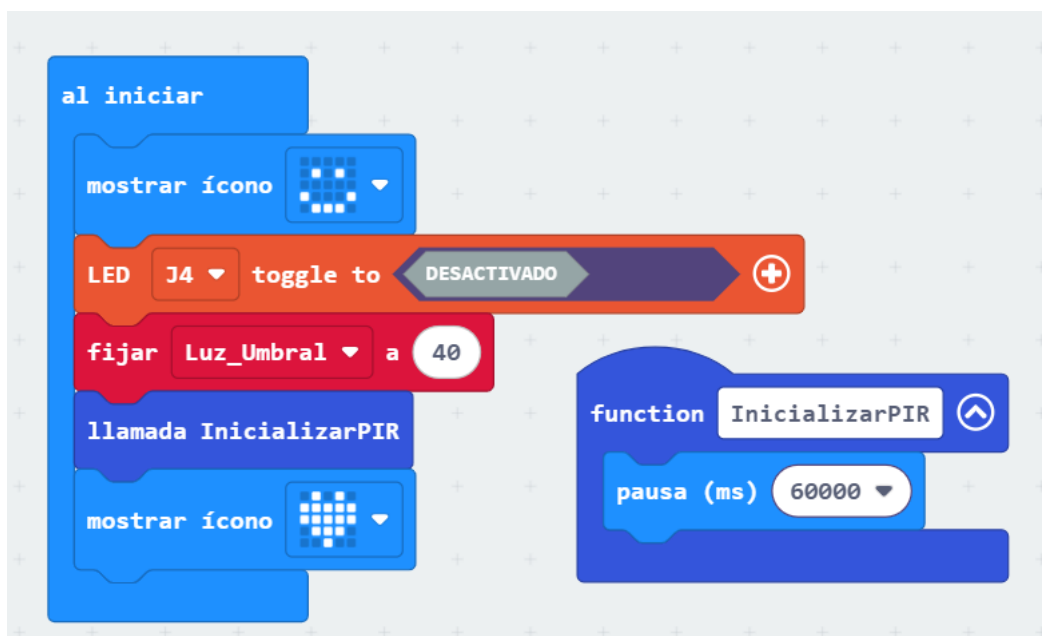
Debemos testear el **valor umbral** de la cantidad de luz en Lux



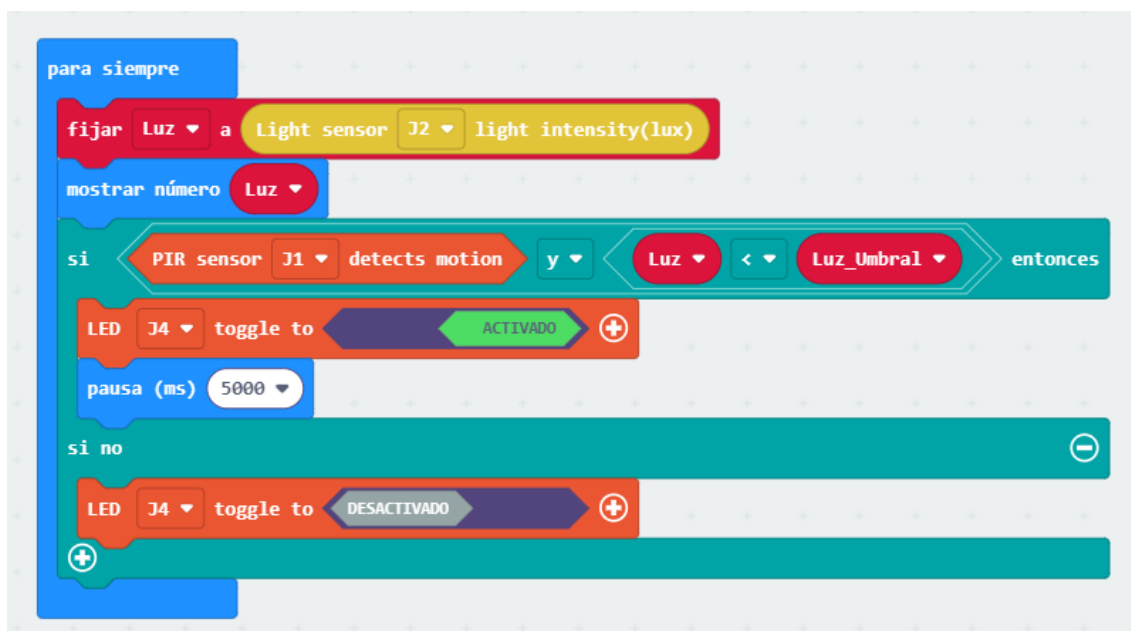
**NOTA:** Lux (lx) El Lux se usa para determinar la cantidad de luz proyectada sobre una superficie (un Lux equivale a un Lumen por metro cuadrado). Nos permite cuantificar la cantidad total de luz visible y la intensidad de la iluminación sobre una superficie.

**Supongamos** que nuestro umbral es de 40 lux. Nuestro programa para el agente inteligente podría ser el siguiente:

Primero apagamos el led, fijamos el umbral y hacemos una espera de 60 segundos para que se inicialice el sensor:



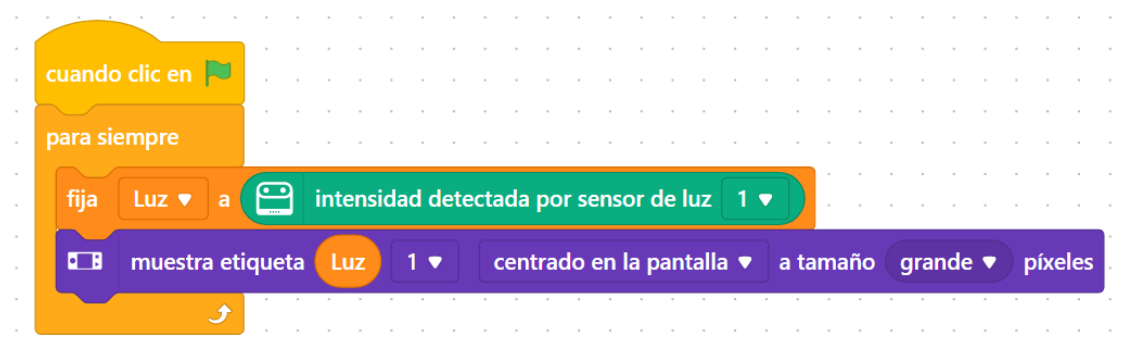
Sólo nos resta aplicar las condiciones de detección de movimiento y luminosidad para activar o no el LED. Se iluminará durante 5 segundos.



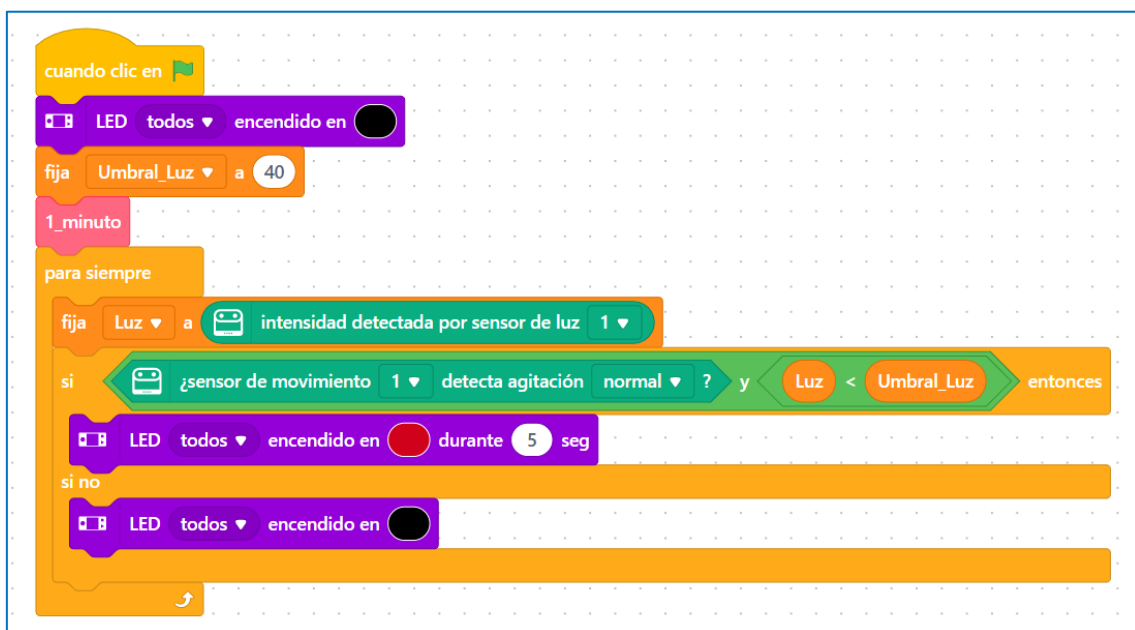
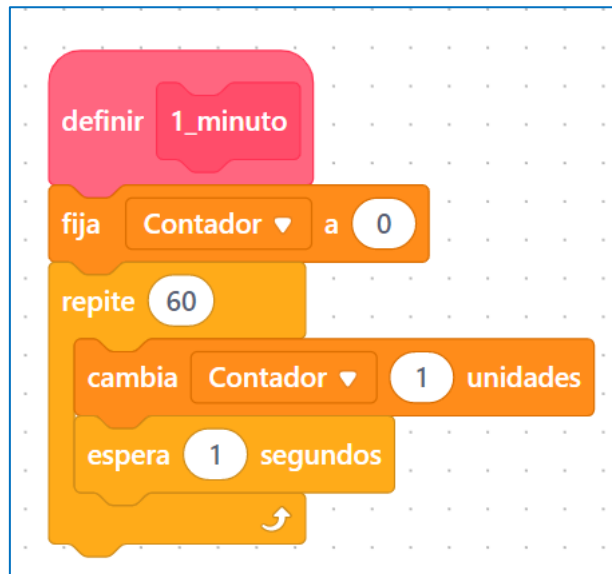
### 5.3. mBlock para la Cyber Pi

Podemos trabajar en modo carga o en modo vivo. Lo vamos a hacer en modo vivo.

Debemos medir la intensidad de la luz para decidir el valor umbral:



**Supongamos** que el valor umbral es 40. Primero apagamos el led, fijamos el umbral y hacemos una espera de 60 segundos para que se inicialice el sensor. El bloque de espera de un minuto podría ser el siguiente:



### 6. Funcionamiento

Vídeo explicativo de proyecto

### 7. Posibles mejoras

- Implementa un registro de datos: vamos a mejorar el agente inteligente añadiendo una funcionalidad para registrar datos de luz y movimiento en un archivo almacenado en drive o en el propio dispositivo. Esto introduce el concepto de **registro y análisis de datos**, lo cual es útil en aplicaciones reales de automatización.
- Graficar los datos