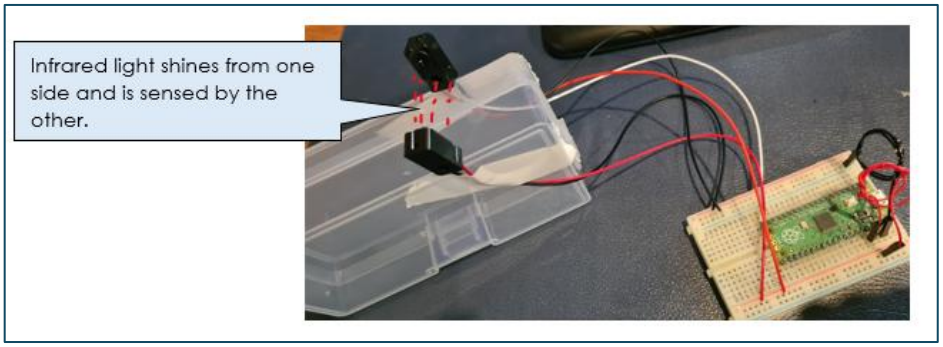
Introduction to Break Beam Sensor

Break beam sensors, also called Light gates, can be used as trip wires or movement sensors. These are great for burglar alarms or lap timers.



Contents

Components

Raspberry Pico H on a breadboard

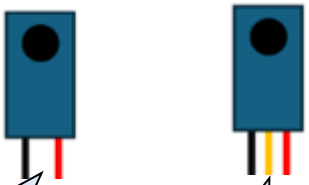
IR Break Beam sensor (3mm or 5mm)

https://thepihut.com/products/ir-break-beam-sensor-5mm-leds

The pack contains two sensors.

One has two wires.

The other has 3.



Red and black are for power

+ & -

The third wire is yellow, or sometimes white. This sends a signal back to the Pico telling it the beam has been broken.

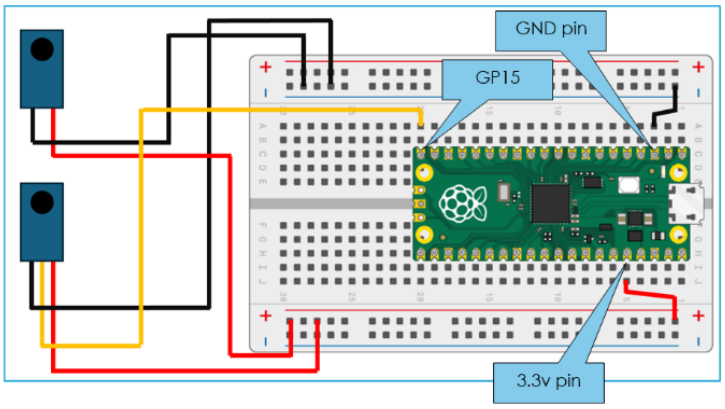
Specification

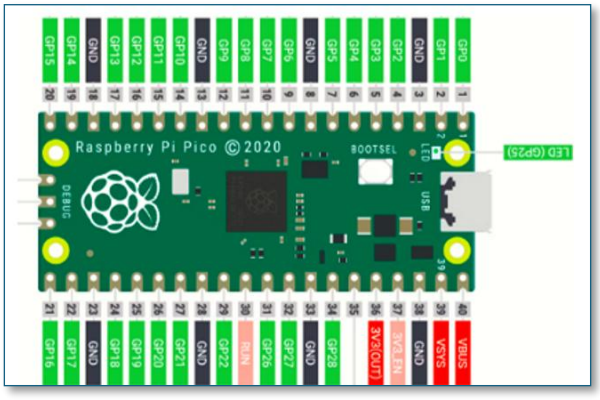
Voltage – It works at 3.3v but try 5v if you have problems.

Distance between the two sensors can be up to approximately 25cm.

Set up

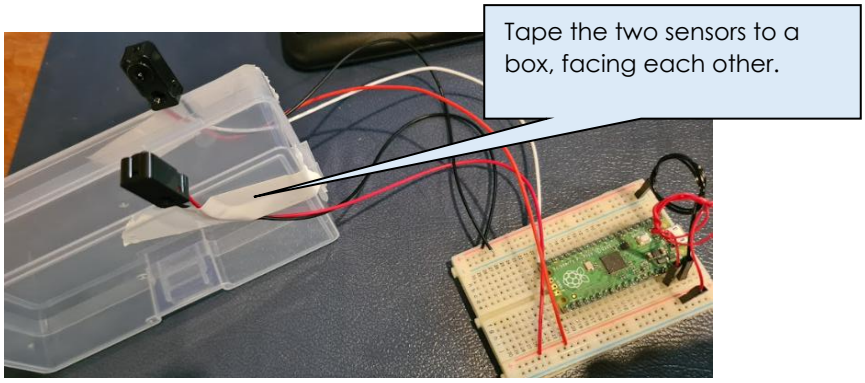
Note that the yellow signal wire is going to pin GP15.





If the wires are a bit floppy use the sharp end of a jumper wire to help it go down into the breadboard hole.

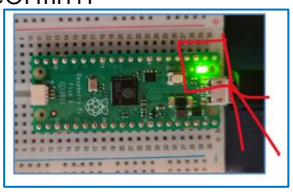
Create a testing setup like this.



Tape the two sensors to a box, facing each other.

Test Code to check Pico connection:

As always, we decompose the problem, testing at each step. We want to confirm the USB connection from the Pico to the PC is working as it should.



Use this test code to make the green on-board LED light up:

```
from machine import Pin
import time
led = Pin(25, Pin.OUT) # links the "led" variable to Pin 25 (the green light)

led.value(1) #turns on light
time.sleep(3) #pause for 3 seconds
led.value(0) #turns off the light
```

Ok, lets move on!

Reading the Break Beam Sensor

The test code (see below) will make a text message output when the beam pin is broken. The test code below will output "ALERT" if the break beam sensor is broken.

Variable **beam_pin** reads value from Pin GP15

Loops endlessly to check beam_pin reading every 0.1 second

```
import machine
import time
from machine import Pin

beam_pin = Pin(15, Pin.IN, Pin.PULL_UP) # Identifies the Pin

old_beam_value = True # output if nothing breaking the beam

while True:
    x = beam_pin.value() # current value from beam_pin
    if x == False:
        print("ALERT")
    else:
        print("all clear")
    time.sleep(0.1)
```

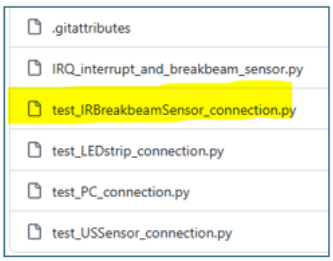
Output if the beam is broken

Using GitHub to copy/paste test code

Copy and paste the code from the GitHub page (this maintains the formatting) to see if you are getting a reading from your break beam sensor.

https://github.com/pythonninja-code/Sensor-Example-Code

You want this file:



Once you are viewing the code you need to **copy it using this button** (or you will lose some indentation and formatting).

I have also included the test code below, in case the school firewall blocks GitHub.

Test Code: Checking the break beam sensor

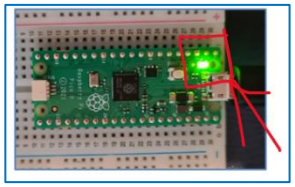
```
import machine
import time
from machine import Pin

beam_pin = Pin(15, Pin.IN, Pin.PULL_UP) # Identifies the Pin

old_beam_value = True # output if nothing breaking the beam

while True:
    x = beam_pin.value() # current value from beam_pin
    if x == False:
        print("ALERT")
    else:
        print("all clear")
    time.sleep(0.1)
```

Challenge 1 – use the green LED

Earlier we tested the Pico's connection to the PC, using the green on-board LED.

Can you make this LED light up when the beam is broken?

Can you make it flash 5 times?



Challenge 2 – use a LED strip

Can you connect up a LED strip.

Write code that makes the LED strip light up red if the beam is broken.

Make the LED strip green if unbroken.

See previous guides to remind yourself how to wire up a LED strip.

Counting the number of breaks

```
1  import machine
2  import time
3  from machine import Pin
4
5  beam_pin = Pin(15, Pin.IN, Pin.PULL_UP)
6  old_beam_value = beam_pin.value()
7  count = 0
8
9  while True:
10     if old_beam_value != beam_pin.value():
11         old_beam_value = not old_beam_value
12         #print(old_beam_value)
13         if old_beam_value == True:
14             count = count + 1
15             print(count)
16     time.sleep(0.01)
17
18
```

The beam is broken. Then as the break is resolved the value returns to True.

This is what is counted.

Line 16 has a time.sleep. This value has been changed to (0.01) so it can identify fast changes.

```
Shell
>>> %Run -c $EDITOR_CONTENT

  MPY: soft reboot
  1
  2
  3
  4
  5
```

IRQ – Interrupt Requests

IRQ interrupts are very useful because they can interrupt a while loop.

The code below does the same thing that the while loop code did.

However instead of a while loop we are using an IRQ.

IRQ = interrupt request.

How an IRQ works

An IRQ has a **trigger** and a **handler**.

The **trigger** is the code that launches the IRQ.

The **handler** is what happens when the IRQ is triggered.

```
15 beam_pin.irq(handler = beam_change, trigger = Pin.IRQ_FALLING | Pin.IRQ_RISING)
```

The trigger is when the beam is broken.

The handler is to call the beam_change() function.

The previous while loop program checked constantly for a change. This is called polling. But this is a dead end that makes it difficult for other things to happen.

The IRQ method listens out for a change on a Pin and then runs a function.

Test Code: IRQ and breakbeam True or False

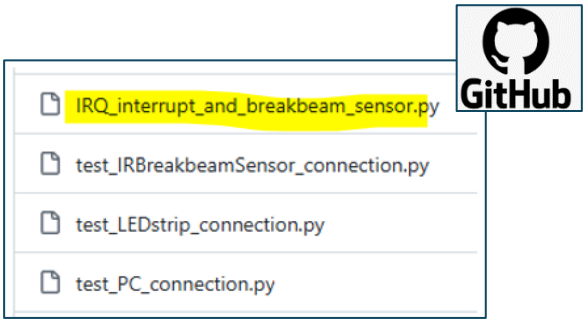
This test code outputs "TRUE" if the beam is NOT broken.

It will output "FALSE" when the IR beam is broken.

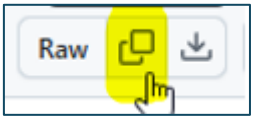
Copy and paste the code from the GitHub page (this maintains the formatting) to see if you are getting a reading from your break beam sensor.

https://github.com/pythonninja-code/Sensor-Example-Code

You want this file:



Copy/paste the code into a new Thonny page.



Understanding the Code

```
1  import machine
2  import time
3  from machine import Pin
4
5  beam_pin = Pin(15, Pin.IN, Pin.PULL_UP)
6  old_beam_value = beam_pin.value()
7
8  def beam_change(pin):
9      broken = pin.value() == 0
10     if not broken:
11         print('TRUE')
12     else:
13         print("FALSE")
14
15 beam_pin.irq(handler = beam_change, trigger = Pin.IRQ_FALLING | Pin.IRQ_RISING)
16
17 while True:
18     time.sleep(0.1)
19
```

This function is called with the IRQ is triggered.

This calls the function

Endless loop.

Test Code: IRQ and breakbeam True or False

I have included the test code below, in case the school firewall blocks GitHub.

```
import machine
import time
from machine import Pin

beam_pin = Pin(15, Pin.IN, Pin.PULL_UP)
old_beam_value = beam_pin.value()

def beam_change(pin):
    broken = pin.value() == 0
    if not broken:
        print('TRUE')
    else:
        print("FALSE")

beam_pin.irq(handler = beam_change, trigger = Pin.IRQ_FALLING | Pin.IRQ_RISING)

while True:
    time.sleep(0.1)
```

Challenge: use the On-Board LED with IRQ

This is a demonstration of how a while loop can do one thing, and then be interrupted.

First, add code that allows us to use the on board LED.

```
led = Pin("LED", Pin.OUT)
```

Add code to the **while True** section so the on-board LED flashes slowly.

```
while True:
    led.value(1)
    time.sleep(1.5)
    led.value(0)
    time.sleep(1.5)
```

Add code to the function so the on-board LED flashes fast 4 times (use a for loop)

```
    print("FALSE")
    for x in range(4):
        led.value(1)
        time.sleep(0.5)
        led.value(0)
        time.sleep(0.5)
```

The Scalextric Project

The following uses for IRQ and the break beam sensor were used in the Scalextric project

Use IRQ and break beam – Measure Velocity

The code below includes an If statement that measures the velocity of whatever broke the beam.

I used this in a slot car racing game.

```
import machine
import time
from machine import Pin

beam_pin = Pin(15, Pin.IN, Pin.PULL_UP)
old_beam_value = beam_pin.value()
last_time = time.ticks_us()

def beam_change(pin):
    global last_time
    broken = pin.value() == 0
    if not broken:
        elapsed_time = time.ticks_diff(time.ticks_us(), last_time)
        elapsed_time = elapsed_time / 1_000_000
        print('beam restored', elapsed_time)
    else:
        print("beam broken")
        last_time = time.ticks_us()

beam_pin.irq(handler = beam_change, trigger = Pin.IRQ_FALLING | Pin.IRQ_RISING)

while True:
    time.sleep(0.1)
```

Test Code

```
import machine
import time
from machine import Pin

beam_pin = Pin(15, Pin.IN, Pin.PULL_UP)
old_beam_value = beam_pin.value()
last_time = time.ticks_us()

def beam_change(pin):
    global last_time
    broken = pin.value() == 0
    if not broken:
        elapsed_time = time.ticks_diff(time.ticks_us(), last_time)
        elapsed_time = elapsed_time / 1_000_000
        print('beam restored', elapsed_time)
    else:
        print("beam broken")
        last_time = time.ticks_us()

beam_pin.irq(handler = beam_change, trigger = Pin.IRQ_FALLING | Pin.IRQ_RISING)

while True:
    time.sleep(0.1)
```