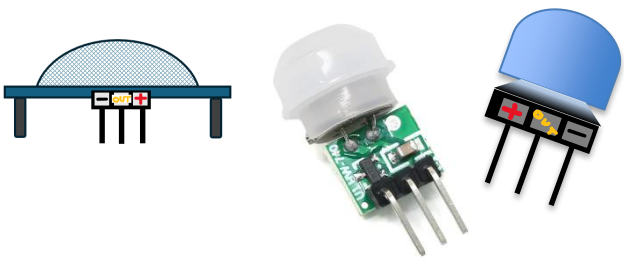
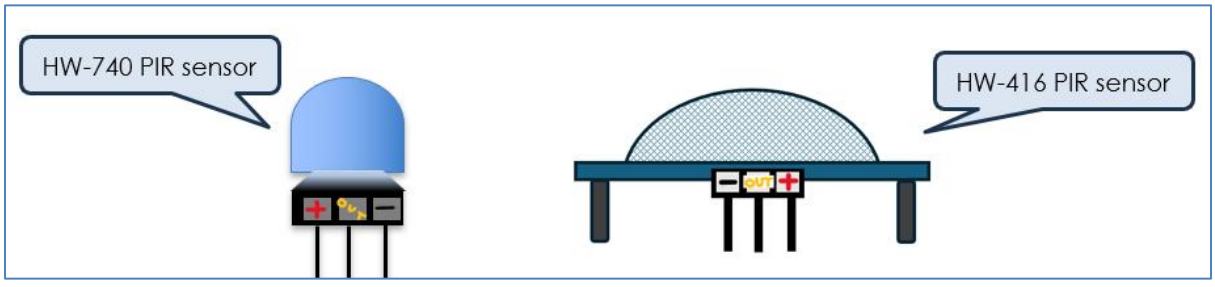


Mini PIR Sensors

Passive Infrared sensors are great for detecting motion.
This guide will show you how to add one to your Pico breadboard project to trigger outputs such as the green on-board LED or a LED strip.

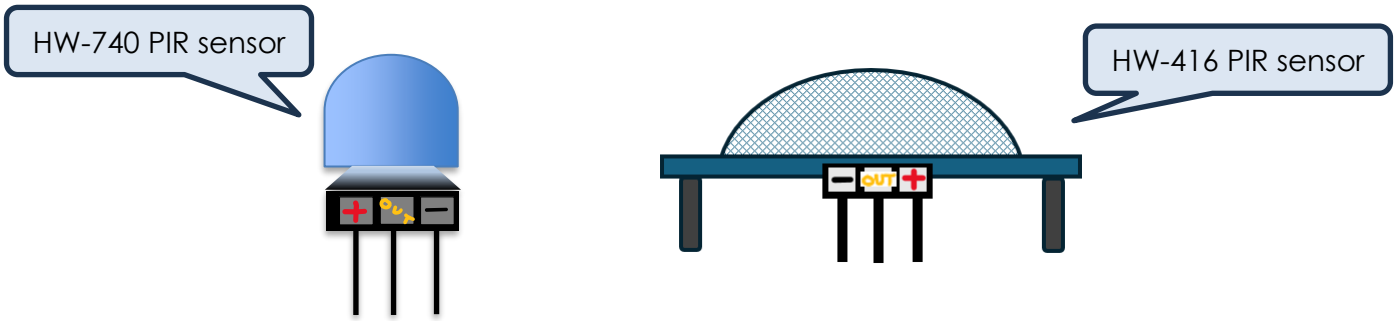
Contents

Mini PIR Sensors 1
Introduction 2
Introducing the HW-740 PIR Sensor: 2
Introducing the HW-416 PIR Sensor: 3
Test your connection from computer to Pico 4
Test Code (to test wired connection to PC): 4
Test the PIR sensor 5
Test Code (to test the PIR sensor): 5
LED Strip and PIR motion detection 5
Test Code (to check the LED strip): 6
LED Strip and PIR 6
Test Code (to test the PIR sensor): 6
Solution code for PIR and LED together: 8
Challenge 1: Different colours 8
Challenge 2: More complex lighting 8
Using 'Interrupt' Code 9
LED Strip and PIR with Interrupt 10
Test Code for interrupt: 10
Challenge 3: Flashing sequence 11
Challenge 4: Buzzer 11
Challenge 4: Toggle 12
Sources: 12



Introduction

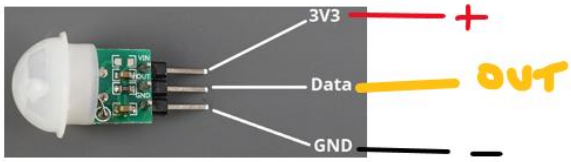
Passive Infra-Red (PIR) sensors detect movement. We can use them to trigger the program to turn on motors, sensors or LED lights. There are **two** types of PIR Sensor we are using here. They look a little different but do the same thing.

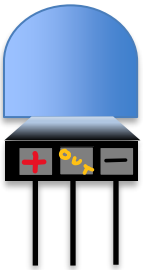


The HW-740 has the advantage of being small.
The HW-416 has small screws built into it that allow the sensitivity to be adjusted.
Both the PIR sensors mentioned here have 3 pins.
The wiring diagram below is the same for each of these models

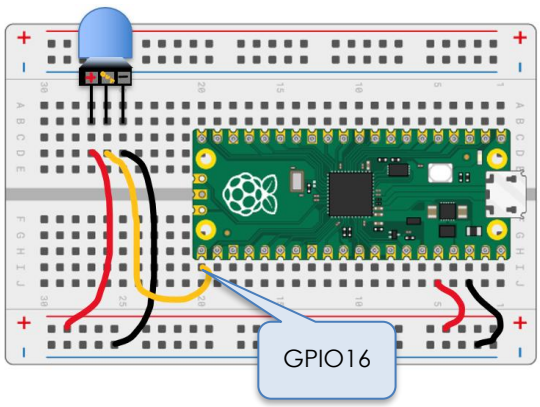
Introducing the HW-740 PIR Sensor:

Mini PIR Sensor: u1 hw-740
https://thepihut.com/products/breadboard-friendly-mini-pir-motion-sensor-with-3-pin-header?variant=39749999788227





This PIR sensor can plug directly into a breadboard.

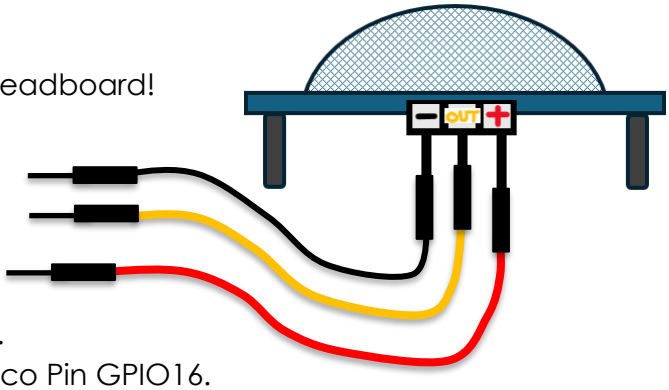


The HW 740 PIR sensor picks up targets around the 5m range.
It can be powered from 3 to 12V DC as it has an onboard regulator.
The middle pin outputs a signal going high for 2 seconds when motion is detected.
I have used a yellow wire to indicate the signal pin.

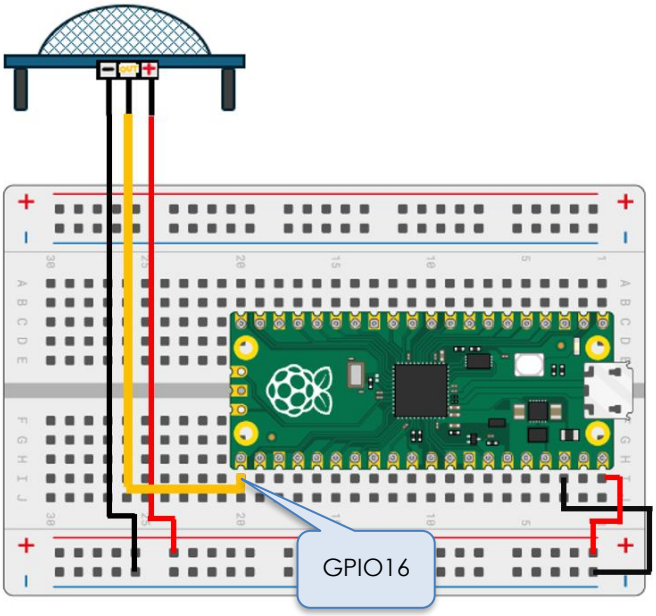
Introducing the HW-416 PIR Sensor:

The Pi Hut also sells this HW-416 PIR Motion Sensor. (I think this is also known as HC-SR501).

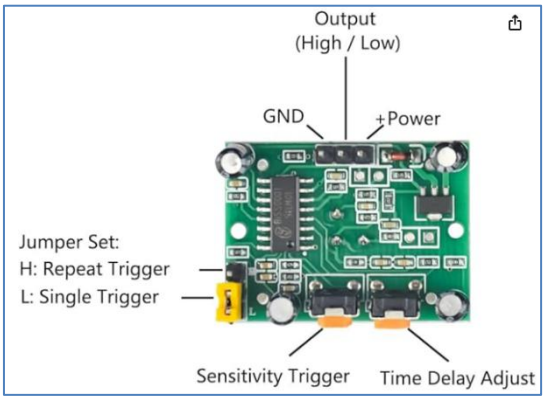
We do not push the pins directly into the breadboard!
Add jumper wires like this.



Then connect the wires to the breadboard.
Note the signal wire (yellow) connects to Pico Pin GPIO16.



This PIR sensor has manually adjustable settings, but it does not plug directly into a breadboard. We can use jumper wires to connect this sensor.



The HW 416 PIR sensor can have its **sensitivity** adjusted using two twistable knobs. Turn it clockwise to increase the **distance**. Maximum range is about 7 meters. Source.
The other knob controls the time delay. The time delay is how long the sensor pauses after sensing motion. Turn clockwise to increase delay.

There is also a yellow **jumper clip** that allows us to choose between "repeat" or "single" trigger mode. In single trigger mode, a PIR sensor's **delay timer** starts when motion is first detected, and any subsequent motion during that time does not restart the timer. In repeat (or re-triggerable) mode, each new motion detection resets and restarts the time delay, keeping the sensor's output active as long as motion continues to be detected within the range.

These PIR sensors run on 3.3 volts or 5v or even 12v.
Power taken from the Pico is 3.3 volts (or 5v from VBUS) so we can power these sensors from the breadboard.

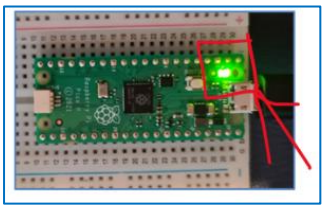
Test your connection from computer to Pico

Before we go any further, lets test our connection between the PC and the Pico.
Connect a Micro USB cable to the Pico and connect it to the PC.
Open up the Thonny App.
Use this test code to check if the connection is working.
The on-board green LED should light up.
Note – this code does nothing with the PIR sensor.

Test Code (to test wired connection to PC):

This is the standard code we always use to check the Pico is connected to the PC properly. It is always worth checking this as some USB slots and cables sometimes cause problems.

```
from machine import Pin
import time
led = Pin(25, Pin.OUT)
led.value(1)
time.sleep(3)
led.value(0)
```



Hopefully the green on-board LED did light up. We can now test the PIR sensor.
Now we know the connection from the PC to the Pico is working, we can test the PIR sensor is working.

Test the PIR sensor

Use the following test code to see whether the PIR sensor detects motion, and triggers an output.

Test Code (to test the PIR sensor):

```
from machine import Pin
from time import sleep
pir_pin = Pin(16, Pin.IN)
led_pin = Pin (25, Pin.OUT)

while True:
    if pir_pin.value():
        print("MOTION DETECTED!")
        led_pin.value(1)
    sleep(1)
    led_pin.value(0)
    print("no motion...")
```

```
Shell
no motion...
no motion...
no motion...
MOTION DETECTED!
no motion...
MOTION DETECTED!
no motion...
MOTION DETECTED!
no motion...
```

Run the program in Thonny. Wave your hand past the sensor. Does the shell show "MOTION DETECTED"?.
The green on-board LED should also come on.

If it is not working, check the signal wire is connected to GPIO 16.
Also check this line of code. It should tell the Pico to 'listen' on pin GPIO 16.

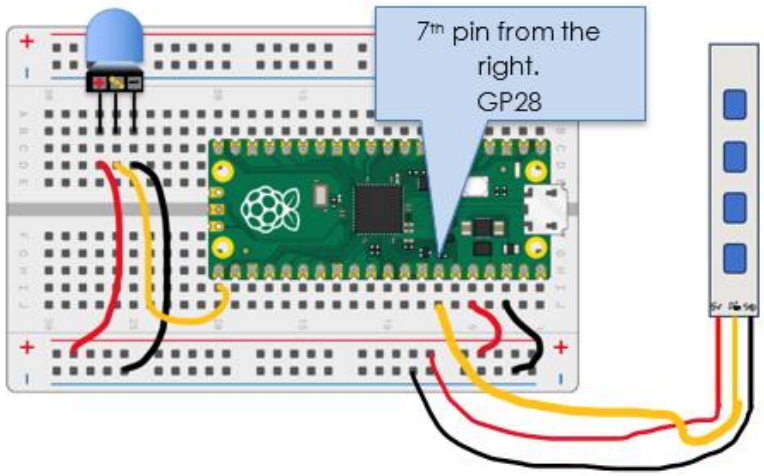
```
3 pir_pin = Pin(16, Pin.IN)
```

Now we have proved our PIR sensor is inputting data to the Pico, we can develop our project.

LED Strip and PIR motion detection

Let's add a LED strip to the breadboard!
We have used LED strip lighting in previous guides.

Make sure you have the neopixel library installed.
If this makes no sense, see detailed instructions in the "LED strip Intro" guide in chapter 2.



Before we try anything with the PIR sensor, we need to check our LED strip is working as it should. This is because the LED strips sometimes have broken wires or connections.

We are always trying to break bigger problems down into smaller steps.
This is called decomposing a problem.

We are now testing the LED strip (not the PIR sensor).
Use this test code to check the LED strip is working.

Test Code (to check the LED strip):

```
from machine import Pin
import time
from neopixel import Neopixel

#variables for the LED strip
numpix = 14
strip = Neopixel(numpix, 0, 28, "GRB")

#colour variables
red = (255, 0, 0)
blank = (0, 0, 0)

#commands
strip.fill(red)
strip.show()
time.sleep(2)
strip.fill(blank)
strip.show()
```

Hopefully, your LED strip did light up.
We can now make the PIR sensor work ***with*** the LED Strip.

LED Strip and PIR

We now need to combine the test code for the PIR sensor, and the test code for the LED strip.

Below is the same test code we used before to check the PIR sensor was wired correctly.

Test Code (to test the PIR sensor):

```
from machine import Pin
from time import sleep
pir_pin = Pin(16, Pin.IN)
led_pin = Pin (25, Pin.OUT)

while True:
    if pir_pin.value():
        print("MOTION DETECTED!")
        led_pin.value(1)
    sleep(1)
    led_pin.value(0)
    print("no motion...")
```

Now we need to copy some of the LED test code into the PIR test code program.

Below, you can see I have highlighted code that controls the LED strip.
Copy this LED strip test code.

```
from machine import Pin
import time
from neopixel import Neopixel

#variables for the LED strip
numpix = 14
strip = Neopixel(numpix, 0, 28, "GRB")

#colour variables
red = (255, 0, 0)
blank = (0, 0, 0)

#commands
strip.fill(red)
```

Paste it into the PIR Test code.

```
from machine import Pin
from time import sleep
from neopixel import Neopixel

#variables for the LED strip
numpix = 14
strip = Neopixel(numpix, 0, 28, "GRB")

#colour variables
red = (255, 0, 0)
blank = (0, 0, 0)

pir_pin = Pin(16, Pin.IN)
led_pin = Pin (25, Pin.OUT)

while True:
    if pir_pin.value():
        print("MOTION DETECTED!")
```

Ok, we are now going to edit the PIR test code.

Find the while loop that outputs "MOTION DECTECTED"
Let's make the LED strip respond when motion is detected.

Add these commands highlighted in yellow:

```
while True:
    if pir_pin.value():
        print("MOTION DETECTED!")
        led_pin.value(1)
        strip.fill(red)
        strip.show()

    sleep(1)
    led_pin.value(0)
    print("no motion...")
    strip.fill(blank)
    strip.show()
```

Now run the program to see if the LED strip lights up with movement.
Did your LED strip light up when you moved your hand over the PIR sensor?

Solution code for PIR and LED together:

```
from machine import Pin
from time import sleep
from neopixel import Neopixel
#variables for the LED strip
numpix = 14
strip = Neopixel(numpix, 0, 28, "GRB")
#colour variables
red = (255, 0, 0)
blank = (0,0,0)

pir_pin = Pin(16, Pin.IN)
led_pin = Pin (25, Pin.OUT)

while True:
    if pir_pin.value():
        print("MOTION DETECTED!")
        #LED commands
        strip.fill(red)
        strip.show()

    sleep(1)
    #LED commands
    strip.fill(blank)
    strip.show()

    print("no motion...")
```

Challenge 1: Different colours

Can you make the LED strip light up one colour when there is no motion detected and another colour when motion is detected?

Challenge 2: More complex lighting

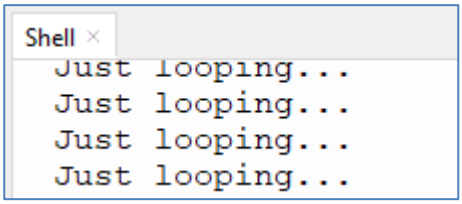
Can you make a more complex lighting sequence when motion is detected e.g. making them fade or zoom. See chapter 2 for guides on how to make LED strips behave this way.

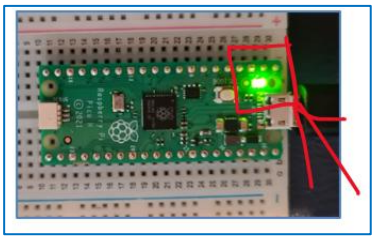
Using 'Interrupt' Code

Imagine you have your LED strip doing something on a Loop.
For example, you might have your LED strip flashing red and green on a loop.
We need a way to stop this action to respond to the input from the PIR sensor.
Using an 'interrupt' means that the Pico will stop whatever it is doing to respond to the PIR sensor. This is useful if the Pico is making LED lights do something in a while loop.

```
from machine import Pin
import time

pir_pin = Pin(16, Pin.IN)

led_pin = Pin(25, Pin.OUT)

def pir_interrupt(pin):
    if pin.value() == 1:
        led_pin.value(1)
    else:
        led_pin.value(0)

# Configure the interrupt on the PIR pin for both rising and falling edges
pir_pin.irq(trigger=(Pin.IRQ_RISING | Pin.IRQ_FALLING), handler=pir_interrupt)

looping = True
while looping == True:
    print("Just looping...")
    time.sleep(1)
```

When you run this program, you should see the shell output a text message every two seconds. The code for this is a while loop.





Move your hand over the sensor.
You should see the on-board green LED light up.

This happened because the while loop was interrupted.

Allowing the Pico to iterate (loop) one thing, but "listen" for the PIR sensor, and then do another, is very useful.

More on this later.

LED Strip and PIR with Interrupt

We are now going to use the PIR test code and the interrupt test code.
We can use lines from one in the other.
This will allow us to make the LED strip do one thing, then sense the PIR input, then do another.

Let's go back to the interrupt code we used before.

Test Code for interrupt:

```
from machine import Pin
import time

pir_pin = Pin(16, Pin.IN)

led_pin = Pin(25, Pin.OUT)

def pir_interrupt(pin):
    if pin.value() == 1:
        led_pin.value(1)
    else:
        led_pin.value(0)

# Configure the interrupt on the PIR pin for both rising and falling edges
pir_pin.irq(trigger=(Pin.IRQ_RISING | Pin.IRQ_FALLING), handler=pir_interrupt)

looping = True
while looping == True:
    print("Just looping...")
    time.sleep(1)
```

Remember, this code iterates "Just Looping…".
At the same time, it is listening for the PIR to interrupt it.
Let's change what happens when the PIR generates an interrupt.

Copy this code from the **LED strip test code**

```
from machine import Pin
import time
from neopixel import Neopixel

#variables for the LED strip
numpix = 14
strip = Neopixel(numpix, 0, 28, "GRB")

#colour variables
red = (255, 0, 0)
blank = (0, 0, 0)

#commands
strip.fill(red)
```

Paste the code into the PIR Interrupt test code.

```
from machine import Pin
import time
from neopixel import Neopixel

#variables for the LED strip
numpix = 14
strip = Neopixel(numpix, 0, 28, "GRB")

#colour variables
red = (255, 0, 0)
blank = (0, 0, 0)

pir_pin = Pin(16, Pin.IN)

led_pin = Pin(25, Pin.OUT)

def pir_interrupt(pin):
    if pin.value() == 1:
```

Add an additional colour variable:

```
#colour variables
red = (255, 0, 0)
blue = (0, 102, 255)
blank = (0, 0, 0)
```

Add this blue colour as a command, to the while loop.

```
pir_pin.irq(trigger=(Pin.IRQ_RIS
looping = True
while looping == True:
    print("Just looping...")
    strip.fill(blue)
    strip.show()
    time.sleep(1)
```

Now add the 'red' colour command to the interrupt function:

```
def pir_interrupt(pin):
    if pin.value() == 1:
        led_pin.value(1)
        strip.fill(red)
        strip.show()
    else:
        led_pin.value(0)
```

Now test your program.
Did it change colour when it detected movement?
Did the colour go back to blue after a couple of seconds?

Challenge 3: Flashing sequence

The interrupt code allows us to loop a lighting sequence. This gets interrupted when the sensor triggers. Edit the while loop to flash a sequence of colours. Or make it zoon or fade.

Challenge 4: Buzzer

Can you add another output device such as a buzzer or servo?
There are a range of buzzers, individual LEDs and other output devices in room GF6.

Challenge 4: Toggle

Can you use 'toggle' to make the LED strip brighten or dim with the wave of a hand. This change is not on a timer. It should stay that brightness until the next wave of the hand.

You can remind yourself how to use 'toggle' in the 'Pico-Buttons Intro' guide.

```
while True:
    if button.value() == 0:
        led.toggle()
    else:
        print("Button is not Pressed")
    time.sleep(0.1)
```

Sources:

https://randomnerdtutorials.com/raspberry-pi-pico-motion-pir-micropython/