# Maker Advent Calendar Day #6: Looking for Light!

By The Pi Hut • Dec 6, 2022 • 20 comments

Welcome to day six of your 12 Projects of Codemas Advent Calendar. Today we'll be using a **light sensor** which - you guessed it - can detect the amount of light around it and provide us with a reading value which we can use in our code.

The light sensor we're using is another analogue component, which means it's able to provide a nice range of values to give us really granular light readings for our projects.

Let's go!

## Box #6 Contents

In this box you will find:

- **1x** Photo Transistor
- **1x** 10k Resistor
- **3x** Male to male jumper wires

## Today’s Project

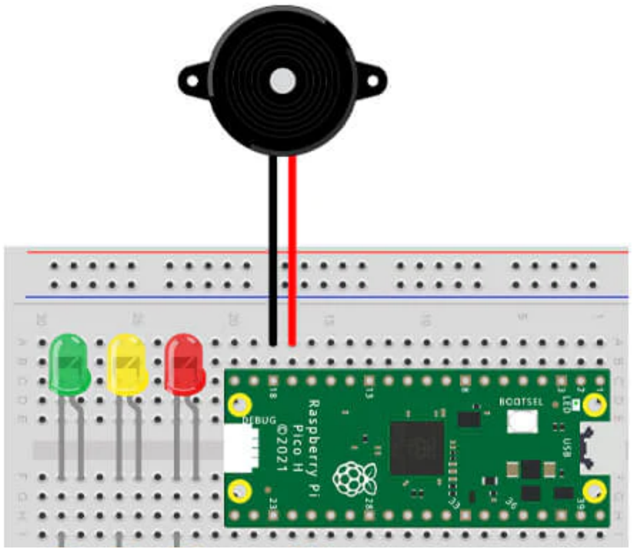
Today we'll be learning how to take analogue readings from our light sensor and converting those into something more useful, as well as some other new tricks to manipulate data and strings. We'll also make a light indication project with our LEDs.

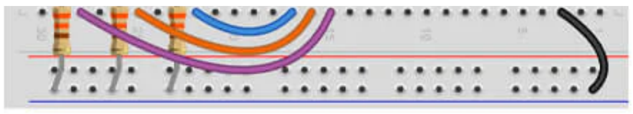
Sensors similar to these are used everywhere in the world around us - in street lamps, vehicles, room lighting and more. They detect the amount of light, and this determines the amount of electrical current that is passed across the the legs (and therefore into our Pico as an input).

A great one to learn and use in many projects!

## Construct the Circuit

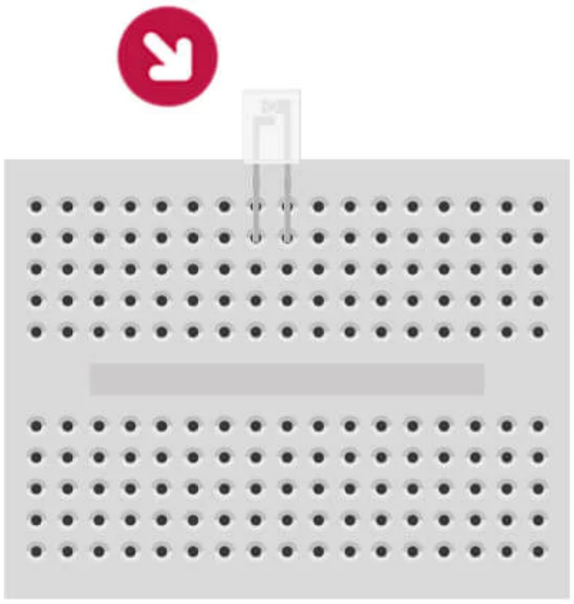
We're going to use the LEDs again and also the buzzer, so leave them in place and remove everything else. Your circuit should look like this to start with:





Before we start with today's circuit we must stress that **you need to pay close attention to the legs of the light sensor** - if you get the wiring wrong here, your light readings won't make a lot of sense!

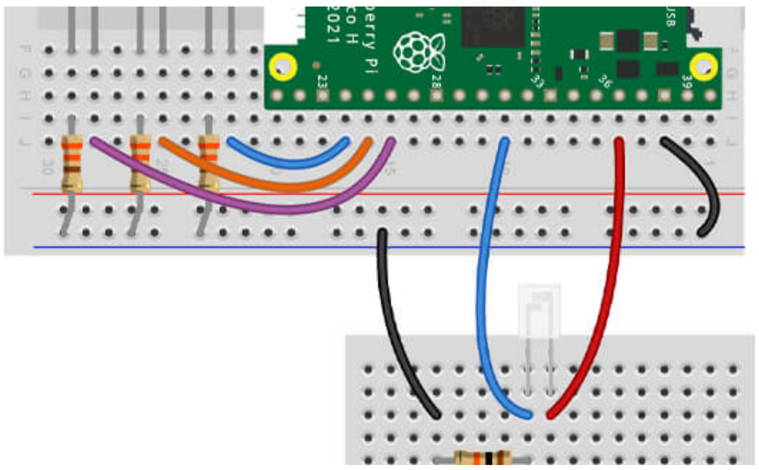
With that in mind, grab your mini breadboard and fit the sensor with the **long leg to the left**. It should look like this:

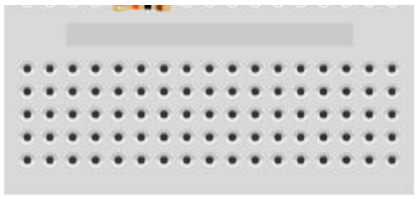


Now we need to add three wires and the resistor:

- Connect the **right** (**short**) leg to the 3.3V pin (***physical pin 36***)
- Connect the **left** (**long**) leg to **GPIO 26** (***physical pin 31***)
- Add the resistor to connect the **left** leg over to another empty unused lane on your breadboard
- Connect the other end of the resistor to a **GND pin** on your Pico (we're using the **blue lane** as it's already connected to GND for our LEDs)

Your circuit should look like this:





## Activity 1: Simple Light Sensing

Let's start with a simple starter program which grabs a light reading for us.

This is just a minimal program to return the analogue value once, which isn't all that useful to humans but we'll convert it to something more practical in the next activity.

### The Code

In this example we:

- ***Import ADC*** to use the ADC function
- Setup the sensor GPIO pin as an ***ADC pin***
- Create a ***variable*** called 'light' which reads the sensor value
- ***Print*** the variable value (the sensor reading)

Copy the code below over to Thonny and give it a try:

```
# Imports
from machine import ADC, Pin

# Define pin for our sensor
lightsensor = ADC(Pin(26))

# Read sensor value and store it in a variable called 'light'
light = lightsensor.read_u16()

# print the value
print(light)
```

## Activity 2: Making Light Readings Usable

We know we can grab a single reading from the sensor with a very short program, but we really need that reading to be in a format that we can use and recognise.

In the example below we convert the reading to a percentage using a little maths. This will come in handy in one of our later boxes when we...*oops nearly dropped a spoiler there!*

We'll show a few examples as there are some other tricks we can use to make our printed values even more useful...

### Example 1:

This example takes the reading and turns it into a percentage.

It divides our reading by 65535 (the maximum of the analogue reading range) then multiplies this by 100 to give us our percentage value:

```
# Imports
from machine import ADC, Pin
from time import sleep

# Define pin for our sensor
lightsensor = ADC(Pin(26))

# Read sensor value and store it in a variable called 'light'
light = lightsensor.read_u16()

# Turn our reading into a percentage of the analogue range
lightpercent = light/65535*100

print(lightpercent)
```

## Example 2:

This example builds on the previous one by limiting the number of decimal places, as we don't really need that much detail for something like a light reading.

To do this we use the built-in '***round***' function in our percentage calculation. You'll see this below where we use ***round(light/65535*100,1)***.

This is taking the light reading, applying the same percentage calculation, and then the final number (***1***) defines how many decimal places to use.

Try the example below, then try changing that last number to 2 or 3 to see what happens (and then try removing the comma and number altogether, which will remove the decimal):

```
# Imports
from machine import ADC, Pin
from time import sleep

# Define pin for our sensor
lightsensor = ADC(Pin(26))

# Read sensor value and store it in a variable called 'light'
light = lightsensor.read_u16()

# Use the round function to limit the decimal places to 1
lightpercent = round(light/65535*100,1)

print(lightpercent)
```

### Example 3:

Our final example adds the % symbol after the reading. We do this by adding some parts to our print line.

Normally when we print the reading, the value is a number (in our example it's a ***float*** to be exact, as it has a decimal). However if we want to print that value along with text at the same time (the % symbol), we need to convert everything to a **string** (text). The print function will not let us mix both together!

We can do this by using the built-in ***str*** function, which converts values to strings.

We use ***str(lightpercent)*** to convert our light value reading to a string, then add another string next to this with ***+"%"***. You can see this in the example below - give it a try:

```
# Imports
from machine import ADC, Pin
from time import sleep

# Define pin for our sensor
lightsensor = ADC(Pin(26))

# Read sensor value and store it in a variable called 'light'
light = lightsensor.read_u16()

# Use the round function to limit the decimal places to 1
lightpercent = round(light/65535*100,1)

# Print our reading and the % symbol
# Convert the reading to a string first using str
print(str(lightpercent) +"%")
```

## Activity 3: Light level LED indicators

We've covered grabbing readings and converting the values, so let's put that to work with a nice little light indication project using our LEDs.

We're going to run an example where we continually check the light readings, and assign our three LEDs to a range of readings - **red** will indicate a low light level, **amber** for normal and **green** for high light levels.

### The Code

In the example below, we add our LEDs back into our code and define the pin numbers. We also define the light sensor pin before we start a ***while loop***.

The while loop grabs a reading from the sensor and converts this to a percentage like we did earlier, then uses ***if statements*** to light a different LED depending on the percentage reading (similar to what we did on day #4).

Copy this over to Thonny, run the code then use your phone's camera light, a torch or simply turn your lights off to see the changes with the program.

Once complete, why not try adding the buzzer code from yesterday's box back in to alert us when the light goes below 30%?

```
# Imports
from machine import ADC, Pin
import time

# Set up the LED pins
red = Pin(18, Pin.OUT)
amber = Pin(19, Pin.OUT)
green = Pin(20, Pin.OUT)

# Define pin for our sensor
lightsensor = ADC(Pin(26))

while True: # Run forever

    # Read sensor value and store it in a variable called 'light'
    light = lightsensor.read_u16()

    # Use the round function to limit the decimal places to 1
    lightpercent = round(light/65535*100,1)

    # Print our reading percentage with % symbol
    print(str(lightpercent) +"%")

    # 1 second delay between readings
    time.sleep(1)

    if lightpercent <= 30: # If percentage is less than or equal to 30

        red.value(1) # Red LED on
        amber.value(0)
        green.value(0)

    elif 30 < lightpercent < 60: # If percentage is between 30 and 60

        red.value(0)
        amber.value(1) # Amber LED on
        green.value(0)

    elif lightpercent >= 60: # If percentage is greater than or equal to 60

        red.value(0)
        amber.value(0)
        green.value(1) # Green LED on
```

## Day #6 Complete!

Good job makers! We've learnt how to use another sensor today, as well as introducing some new ways to manipulate values and strings.

We're going to use this sensor again before the end of the calendar as it's a great partner for a component hiding in one of the other boxes...

So, what did we learn today? Today we:

- Learnt how to wire a light sensor circuit
- Learnt how to manipulate data to make it more useful to us and our programs
- Learnt how to convert values to strings with the **str** function
- Learnt how use the ***round*** function to limit decimal places
- Learnt how to combine multiple strings in prints

As always, **please leave the circuit as it is and wait for tomorrow's box before you remove any parts**. Sleep well!

---

*We used Fritzing to create the breadboard wiring diagram images for this page.*

## Featured Products

**Maker Advent Calendar - The 12 Projects of Codemas (inc. Raspberry Pi Pico H)**

£40 incl. VAT

ADD TO CART

## 20 comments

**The Pi Hut**

December 23, 2022 at 8:32 am

@Victoria – Thanks for the feedback. Pop us a quick message via support.thepihut.com with your order number and we'll get a replacement out to you :)

@Victoria – Thanks for the feedback. Pop us a quick message via support.thepihut.com with your order number and we'll get a replacement out to you :)

**Victoria**

December 23, 2022 at 8:31 am

I think I fried my light sensor. Or was broken to begin with? Maybe include an extra next year. More reading about the anode cathode orientation with VCC and GND would've been helpful.

I think I fried my light sensor. Or was broken to begin with? Maybe include an extra next year. More reading about the anode cathode orientation with VCC and GND would've been helpful.

**Paul**

December 23, 2022 at 8:31 am

@ThePiHut and to anyone else having the same issue. Disconnecting the mini-breadboard with today's circuit seemed to magically work.

@ThePiHut and to anyone else having the same issue. Disconnecting the mini-breadboard with today's circuit seemed to magically work.

**The Pi Hut**

December 20, 2022 at 6:05 pm

@Sheila @ Paul – First try an alternative USB port if you haven't already tried that. Then if you have another Micro-USB cable handy (from an old phone/tablet perhaps), give that a try to rule out a faulty cable. If no joy, try reinstalling MicroPython (follow day #1 again, perhaps the alternative method suggested there now). If still no luck, please get in touch via support.thepihut.com and we'll help.

@Sheila @ Paul – First try an alternative USB port if you haven't already tried that. Then if you have another Micro-USB cable handy (from an old phone/tablet perhaps), give that a try to rule out a faulty cable. If no joy, try reinstalling MicroPython (follow day #1 again, perhaps the alternative method suggested there now). If still no luck, please get in touch via support.thepihut.com and we'll help.

**Paul**

December 20, 2022 at 4:37 pm

Been really enjoying this so far! However found today that the pico micro USB port seems to have broken. Had been working up until this point. Tried multiple USB ports and cables however it is no longer picked up by my machine.

Been really enjoying this so far! However found today that the pico micro USB port seems to have broken. Had been working up until this point. Tried multiple USB ports and cables however it is no longer picked up by my machine.

**Sheila**

December 19, 2022 at 10:56 am

Everything was wonderful till today. Unplugged to configure board; the Pico connection makes a sound when I do this. However, when I plugged it back in there was no sound from my computer (like usual) and the Thonny doesn't see it. Any suggestions where to start to look for my error?
Sheila H in WV, USA
PS I did try different USB ports.

Everything was wonderful till today. Unplugged to configure board; the Pico connection makes a sound when I do this. However, when I plugged it back in there was no sound from my computer (like usual) and the Thonny doesn't see it. Any suggestions where to start to look for my error?
Sheila H in WV, USA
PS I did try different USB ports.

**The Pi Hut**

December 15, 2022 at 11:58 am

@Ben Keeping With no delay at all, things are running VERY fast. It's possible that the ADC simply can't keep up.

@Ben Keeping With no delay at all, things are running VERY fast. It's possible that the ADC simply can't keep up.

**The Pi Hut**

December 15, 2022 at 11:58 am

@Julian Bailey Oh no, looks like a packing error. Sorry about that :( If you can ping us a quick message via support.thepihut.com, we'll get a replacement pack sent to you ASAP.

@Julian Bailey Oh no, looks like a packing error. Sorry about that :( If you can ping us a quick message via support.thepihut.com, we'll get a replacement pack sent to you ASAP.

**The Pi Hut**

December 15, 2022 at 11:58 am

@Karen Dewson When we import sleep from time, we're only using the part we need and not wasting time/resources opening the entire module (but it's a VERY minor thing). There's also a bit of a preference thing for many, but at this basic level it really doesn't matter which way you do things.

@Karen Dewson When we import sleep from time, we're only using the part we need and not wasting time/resources opening the entire module (but it's a VERY minor thing). There's also a bit of a preference thing for many, but at this basic level it really doesn't matter which way you do things.

**Ben Keeping**

December 15, 2022 at 11:52 am

I found that if I dont have a time.sleep() in the loop then I get odd numbers from the ADC, usually either 65535 or 370.
Cue lots of ripping the things apart and starting again, until I put the sleep in and then everything was ok.

Is there any reason why that would be? :)

I found that if I dont have a time.sleep() in the loop then I get odd numbers from the ADC, usually either 65535 or 370.
Cue lots of ripping the things apart and starting again, until I put the sleep in and then everything was ok.

Is there any reason why that would be? :)

**Julian Bailey**

December 15, 2022 at 11:52 am

Hi,

Just opened my day 6 box there is a bag with cables and a resistor but no photodiode :(

Julian

Hi,

Just opened my day 6 box there is a bag with cables and a resistor but no photodiode :(

Julian

**Harold**

December 15, 2022 at 11:52 am

I am still loving it. My only recommendation would be to place links to the prior and next lesson on each day's page. I always have to flip back to the main page before I can link to the next lesson. A minor inconvenience but an inconvenience.
Thank you for a wonderful time

I am still loving it. My only recommendation would be to place links to the prior and next lesson on each day's page. I always have to flip back to the main page before I can link to the next lesson. A minor inconvenience but an inconvenience.
Thank you for a wonderful time

**Karen Dewson**

December 15, 2022 at 11:52 am

Examples 1, 2 and 3 above say 'from time import sleep'? It does work, but it also works if you put 'import time'… Have I missed something?

BTW, I'm loving the Advent Calendar! Just the right level for me. Thank you Pi Hut 😊

Examples 1, 2 and 3 above say 'from time import sleep'? It does work, but it also works if you put 'import time'… Have I missed something?

BTW, I'm loving the Advent Calendar! Just the right level for me. Thank you Pi Hut 😊

**The Pi Hut**

December 12, 2022 at 11:27 am

@Al – If you pop back to day 3, we first covered if, elif and else statements there, with some descriptions around what they stand for and what they do.

To add the buzzer in, add the usual setup lines (the pin and frequency), then within the "if lightpercent <= 30:" indented block (after the LED lines), simply increase the buzzer volume with "buzzer.duty_u16(10000)" add a delay using "time.sleep(2)" then mute the buzzer again afterwards with "buzzer.duty_u16(0)".

@Al – If you pop back to day 3, we first covered if, elif and else statements there, with some descriptions around what they stand for and what they do.

To add the buzzer in, add the usual setup lines (the pin and frequency), then within the "if lightpercent <= 30:" indented block (after the LED lines), simply increase the buzzer volume with "buzzer.duty_u16(10000)" add a delay using "time.sleep(2)" then mute the buzzer again afterwards with "buzzer.duty_u16(0)".

**The Pi Hut**

December 12, 2022 at 11:20 am

@Gordon – We had similar issues when the connection was weak or the sensor was the wrong way around. Double-check your circuit and re-seat all of your cables. Also, make sure your Pico is pressed fully into your breadboard so that none of the legs/pins are showing.

@Gordon – We had similar issues when the connection was weak or the sensor was the wrong way around. Double-check your circuit and re-seat all of your cables. Also, make sure your Pico is pressed fully into your breadboard so that none of the legs/pins are showing.

**Gordon**

December 12, 2022 at 11:19 am

My readings stay within 95% and 97% regardless of whether I'm sitting in the full light or dark?

My readings stay within 95% and 97% regardless of whether I'm sitting in the full light or dark?

**Marek**

December 9, 2022 at 12:42 am

@giles

Nice! This is something that I really miss in those tutorials – having everything work together

@giles

Nice! This is something that I really miss in those tutorials – having everything work together

**giles**

December 7, 2022 at 1:27 am

Still loving the advent calendar, great quality tutorials!

Today I have made code that uses everything we have developed over the previous days. My project does the following:
Monitors the light level
Converts light level to a 4 bit value (0 – 15)
Displays the light level in binary on the 4 LEDs
Plays 'We Wish You a Merry XMas' at the same time
With Lyrics!
With volume controlled by the potentiometer
Uses buttons to start and stop everything

See the code HERE:

https://github.com/gilesknap/pico-xmas/blob/main/README.md#day-6-project

Still loving the advent calendar, great quality tutorials!

Today I have made code that uses everything we have developed over the previous days. My project does the following:
Monitors the light level
Converts light level to a 4 bit value (0 – 15)
Displays the light level in binary on the 4 LEDs
Plays 'We Wish You a Merry XMas' at the same time
With Lyrics!
With volume controlled by the potentiometer
Uses buttons to start and stop everything

See the code HERE:

https://github.com/gilesknap/pico-xmas/blob/main/README.md#day-6-project

**Al**

December 12, 2022 at 11:22 am

Please could you put up the code to add the buzzer in. I can't figure it out.

Could you also explain 'elif'. I get the 'if' but don't know what 'el' refers to. I've looked back at Day 4 but it doesn't explain it there either.
Thanks

Please could you put up the code to add the buzzer in. I can't figure it out.
Could you also explain 'elif'. I get the 'if' but don't know what 'el' refers to. I've looked back at Day 4 but it doesn't explain it there either.
Thanks

**David cartwright**

December 12, 2022 at 11:29 am

Hi. The first LED project stated the power must flow from the anode (long leg) to the cathode (short leg), but here this seems to be the other way around; the text requires the long leg fitted to the left which looks like it would cause power to flow cathode to anode?

Hi. The first LED project stated the power must flow from the anode (long leg) to the cathode (short leg), but here this seems to be the other way around; the text requires the long leg fitted to the left which looks like it would cause power to flow cathode to anode?

## Leave a comment

All comments are moderated before being published.

Name

E-mail

Message

SUBMIT

## Related Posts

**Let it Glow Maker Advent Calendar Day #1: Let's Get Started!**

The Pi Hut • Nov 30, 2023

Welcome to day #1 of Let it Glow – your 12-day maker advent calendar packed with blinky stuff and other components to control your blinky things with! Everyone's welcome here – absolute...

Read more

**Coding the Waveshare RP2040 Matrix**

Tony Goodhew • Sep 15, 2023

The Waveshare RP2040 Matrix development board is an inexpensive, super-compact RP2040 microcontroller with a tiny 5x5 addressable RGB LED matrix on the front. This tiny development board is packed with...

Read more

**MicroPython Skill Builders - #6 Temperature & Humidity Sensors**

Tony Goodhew • Aug 24, 2023

Welcome to another instalment in our MicroPython Skill Builders series by Tony Goodhew, aiming to improve your coding skills with MicroPython whilst introducing new components and coding techniques - using a Raspberry...

Read more

**Handy Links**

All Products

FAQs

Popular Searches

Search

Site Reviews

**Got any questions?**

Contact Us / Support Portal

Can I Cancel My Order?

Has My Order Shipped Yet?

Where Is My Order?

Do You Ship To {insert country name}

How Much Is Shipping?

**Terms & Conditions**

Delivery

Lithium Shipping

Pre-Orders

Privacy Statement

Policies

Terms of Service

Company Info

FAQ

Klarna FAQ

Quick Start Guide

Search

Support Portal

**Our Store Sections**

Raspberry Pi

Maker Store

micro:bit

Arduino

Gifts

**Follow us**

Raspberry Pi
Approved Reseller

We accept