Maker Advent Calendar Day #11: OMG OLED!

By The Pi Hut • Dec 11, 2022 • 28 comments

Welcome to day eleven of your 12 Projects of Codemas Advent Calendar. Today we're playing with a really fun and useful component that you'll use again and again in your projects - a mini I2C OLED display!

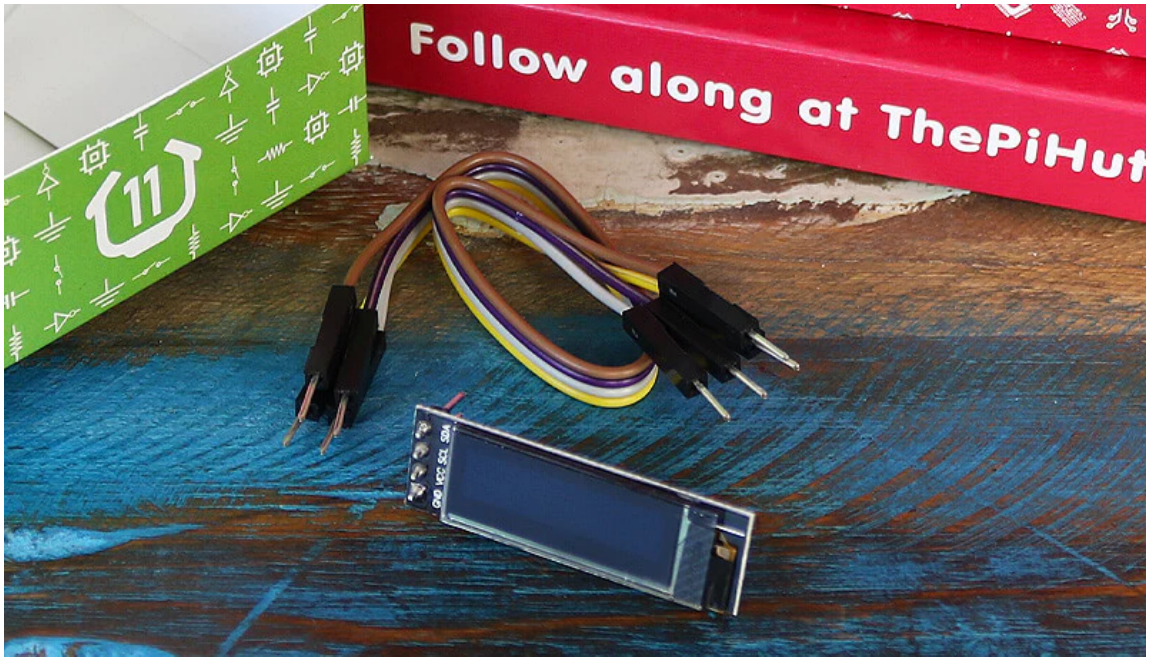
Whilst only teeny-tiny, these little displays are great for showing data from your project such as sensor readings, scores, pin status, alerts and other useful information. We're going to combine it with some components from earlier boxes to show you just how handy these are.

Let's go!

Box #11 Contents

In this box you will find:

- **1x** Pre-soldered 0.96" OLED I2C display (128x32)
- **4x** Male to male jumper wires



Today's Project

Today we're going to display data on our mini I2C OLED display. They're relatively easy to code but they do require the installation of a ***library*** (which we import at the start of our program) but we'll guide you through this.

What is an OLED?

The display in your box is an OLED display. **OLED** stands for **O**rganic **L**ight-**E**mitting **D**iode. You may have seen this mentioned on advertisements for TVs as many models use this technology.

It's a type of digital display technology that uses LEDs and layers of thin organic film between two electrodes. When electrical current is applied, the display emits light. Our code tells the display where to show light and when.

What is I2C?

I2C, sometimes referred to as **IIC**, stands for **I**nter-**I**ntegrated **C**ircuit. It's another type of communication protocol (remember 1-wire from day #8?) which allows multiple I2C devices to communicate to a controller like our Pico.

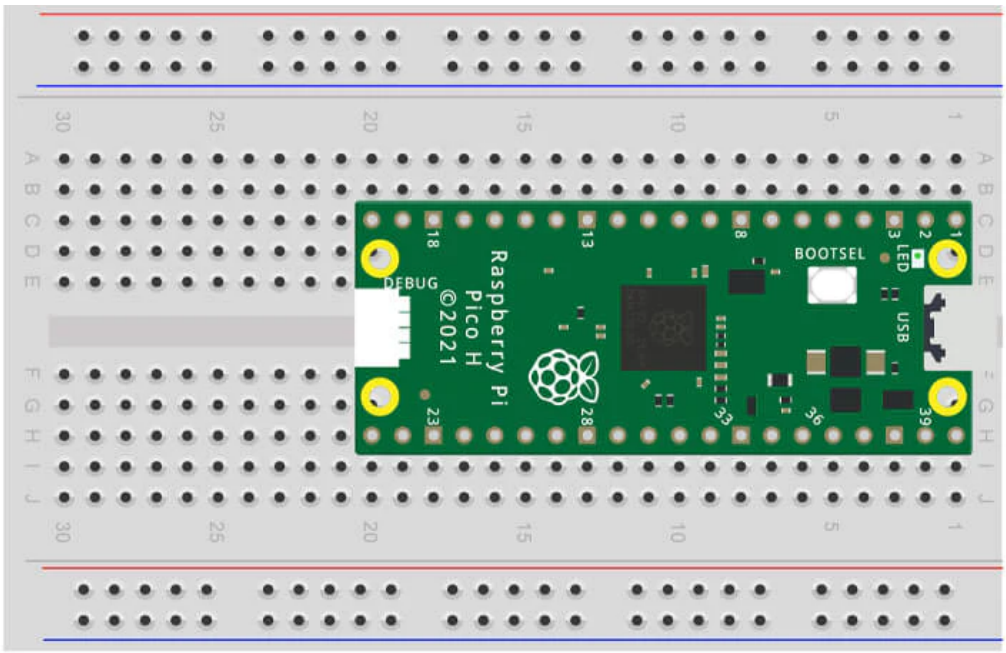
I2C requires just two wires to communicate (along with 3.3V and GND for our display) and has benefits over some other communication options - but we won't bore you with that just now as it's not going to be relevant until you're much further along on your maker journey.

To use I2C we need to import it in our code, which we'll show you in just two ticks!

Construct the Circuit

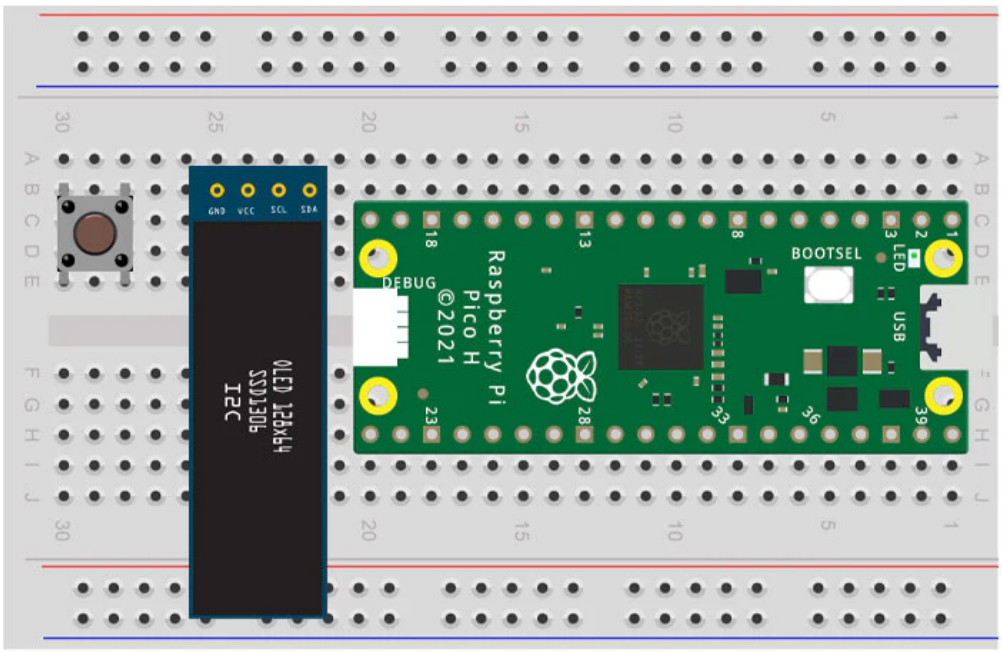
First, **disconnect your Raspberry Pi Pico** from your PC, ensuring it's powered down.

Today we're finally removing the LEDs and buzzer as 'input' components (like buttons and sensors) are more relevant and useful with this display. If you haven't already, remove everything so that you're back to a clean breadboard with just your Pico fitted, like this:



Next, fit one of your buttons and the OLED display into the breadboard like we have below (**make sure you leave a gap above both for us to add our jumper wires**).

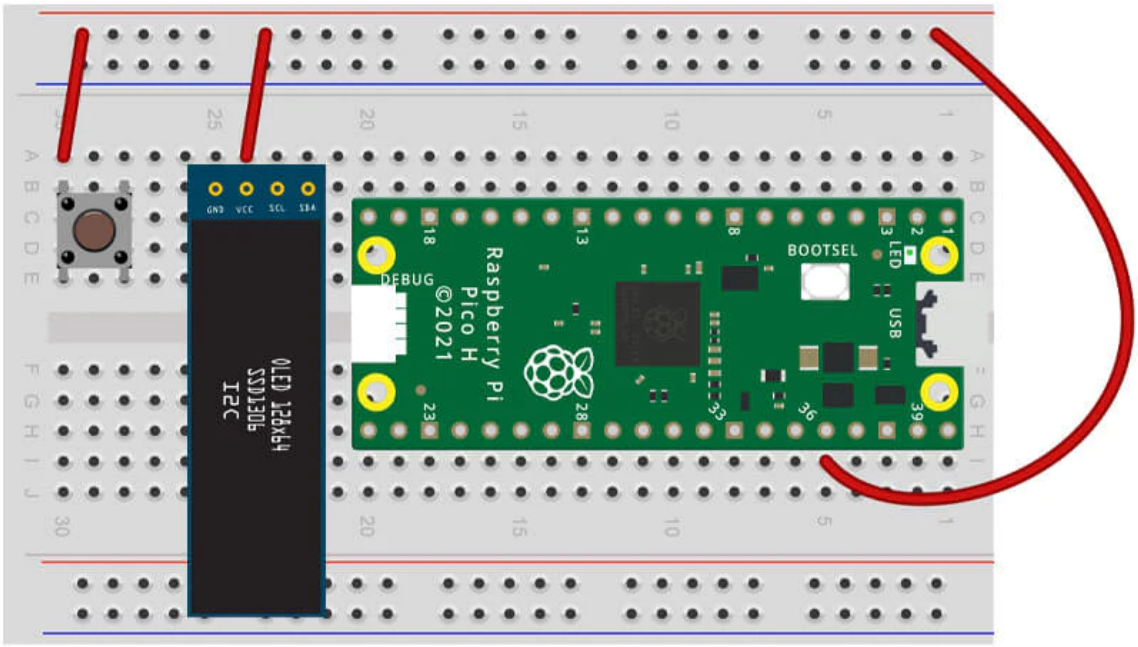
You may also want to **peel off the display's protective film** at this stage:



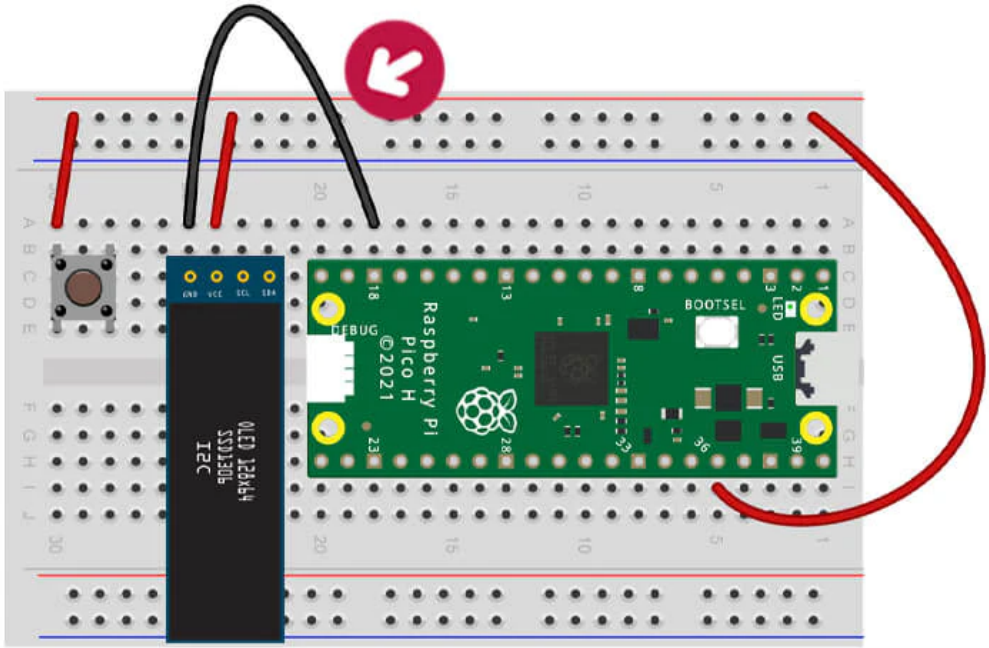
Next we'll add our 3.3V connections.

Run a jumper wire from the ***3.3V pin*** (***physical pin 36***) to the upper ***red*** channel, then connect the ***left button pin***

to the same ***red*** channel, and the ***second OLED pin*** labelled '***VCC***' to the same channel as well, like below:

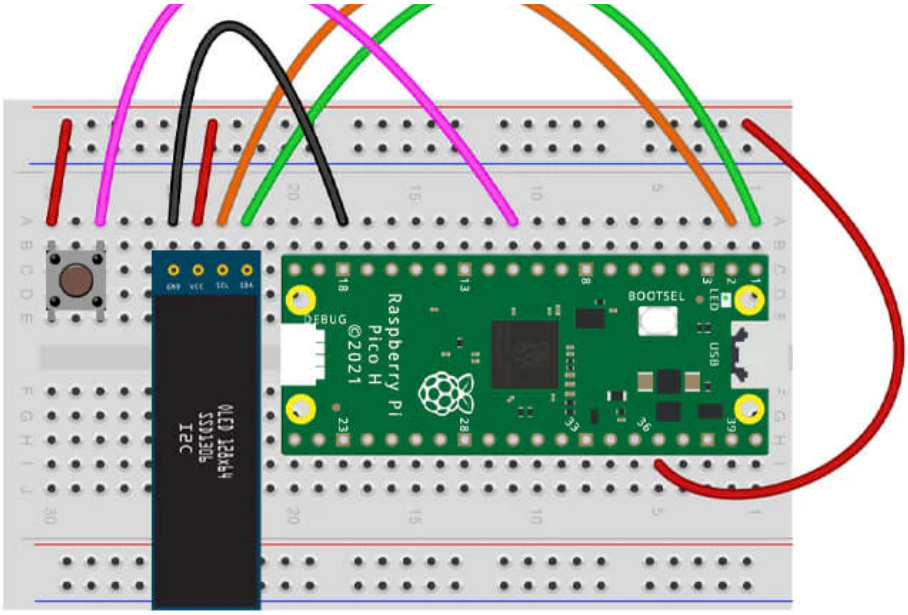


Next is our GND pin. Connect the ***first OLED pin*** labelled '***GND***' to the nearest GND pin on ***physical pin 18***, like we have below:



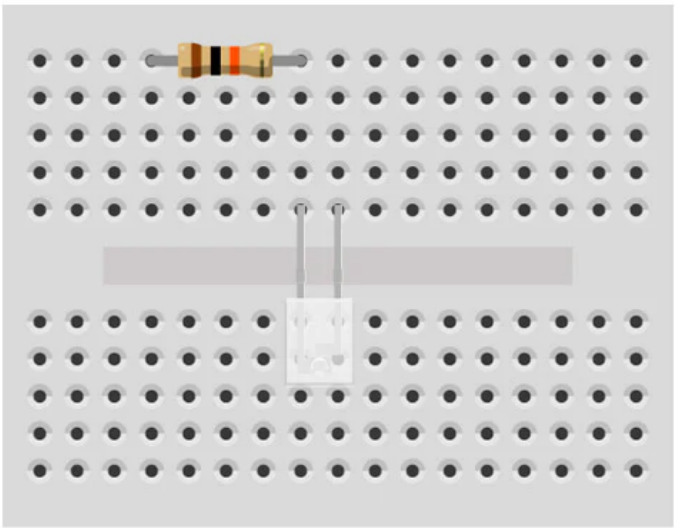
Now for the three GPIO pins which will talk to our Pico:

- Connect the ***right button leg*** to ***GPIO8*** (***Physical pin 11***)
- Connect the ***3rd OLED pin*** labelled '***SCL***' to ***GPIO1*** (***Physical pin 2***)
- Connect the ***4th OLED pin*** labelled '***SDA***' to ***GPIO0*** (***Physical pin 1***)



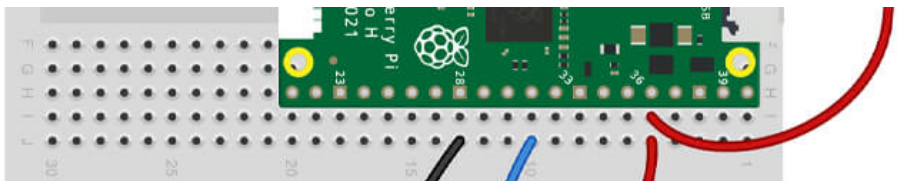
We'll be using the light sensor from box #6 in our examples today, so we need to wire that up as well. We'll use our mini breadboard for this.

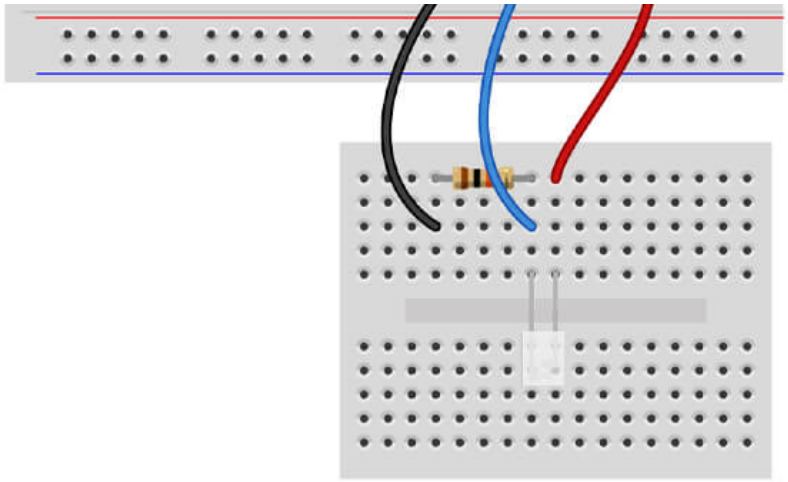
Fit the light sensor and the 10k resistor into the breadboard as we have below. **The long leg of the light sensor needs to go to the left**:



Now to wire up the light sensor to our Pico:

- Connect the ***left resistor leg*** to the ***GND*** connection on ***physical pin 28***
- Connect the ***left light sensor leg*** (between the leg and the resistor) to ***GPIO26*** (***Physical pin 31***)
- Connect the ***right light sensor leg*** to ***3.3V*** on ***physical pin 36***





Now turn your breadboards 90 degrees so that the OLED is facing you. This will make it easier to view and read the text we show on it.

Install Code Libraries

Plug your Pico back into your computer's USB port and make sure it's recognised then let's install what we need to make this display work.

Our display needs a code library package that isn't included in MicroPython by default, so we need to install that before we can start using this display. We will then ***import*** this library when we write our code

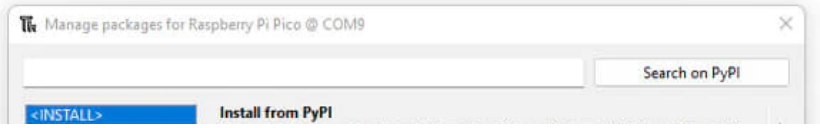
Luckily this is very easy to do and all handled within Thonny, so let's get it installed. The following may vary slightly depending on the type of computer you're using.

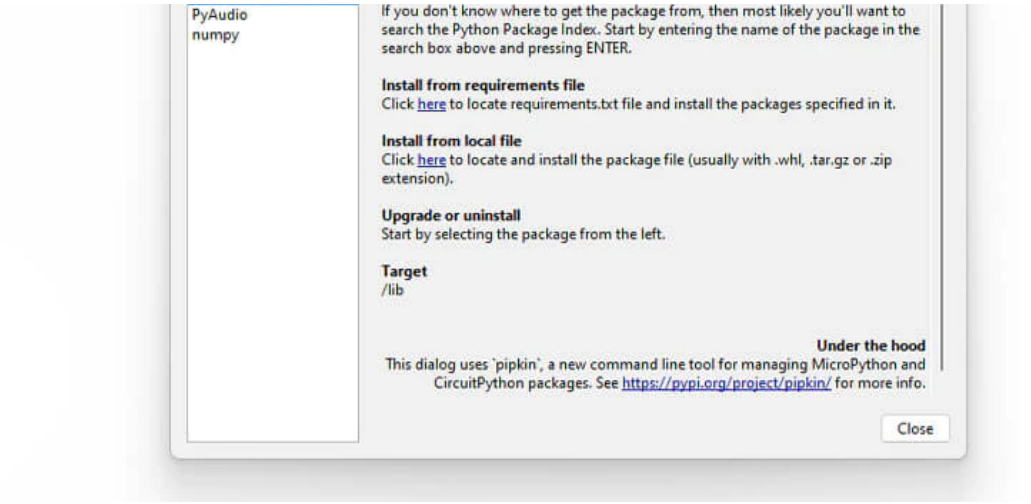
Important: If the steps below don't work for you and you see certificate errors or other similar messages, you may need to install the library manually.

To do this:

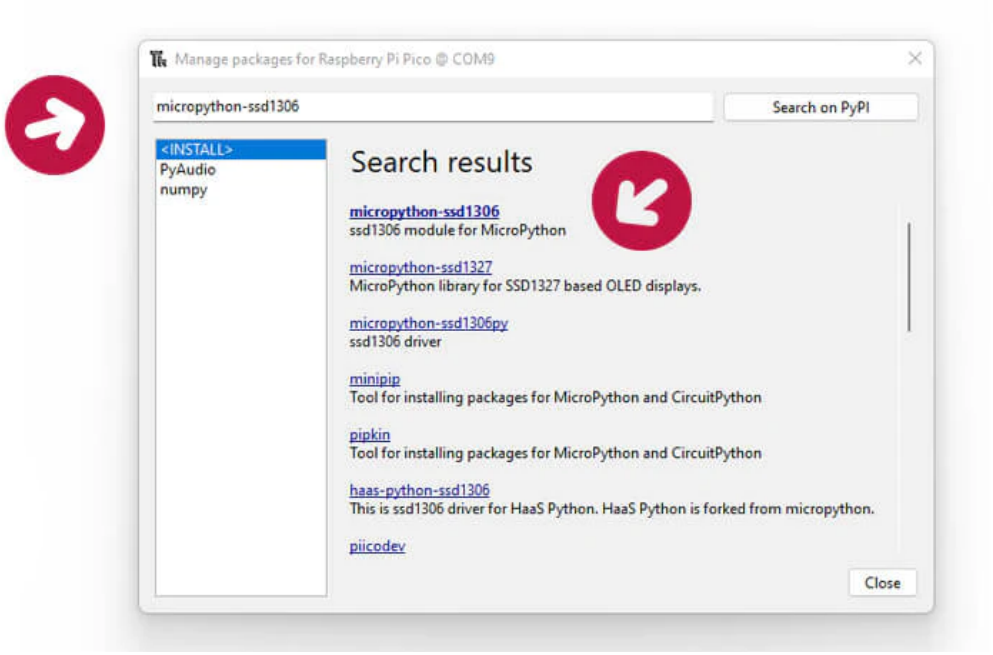
- Go to this link
- Select all of the code on that page (use Ctrl+A) and copy it over to Thonny
- In Thonny, select **File > Save as** and choose **Raspberry Pi Pico** as the destination
- Call the file **ssd1306.py**
- Select **OK**
- **IMPORTANT** - You must now **close this ssd1306.py file** (use the little X on that tab) then open a new one for your code (File > New) otherwise you'll be overwriting that file and making a big mess of it :)

In Thonny, from the top menu bar select ***Tools > Manage packages***. You should see something similar to the box below :

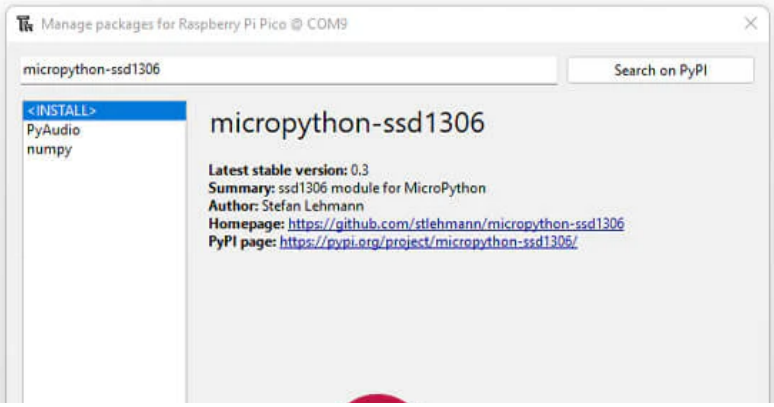


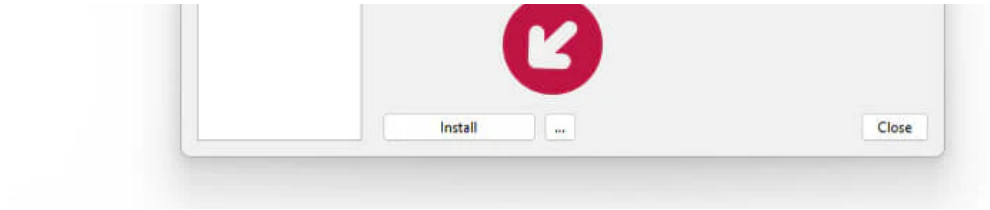


In the search bar, enter "**micropython-ssd1306**" then select the search button. You should see results like we have below, including the micropython-ssd1306 module we need at the top of the list:

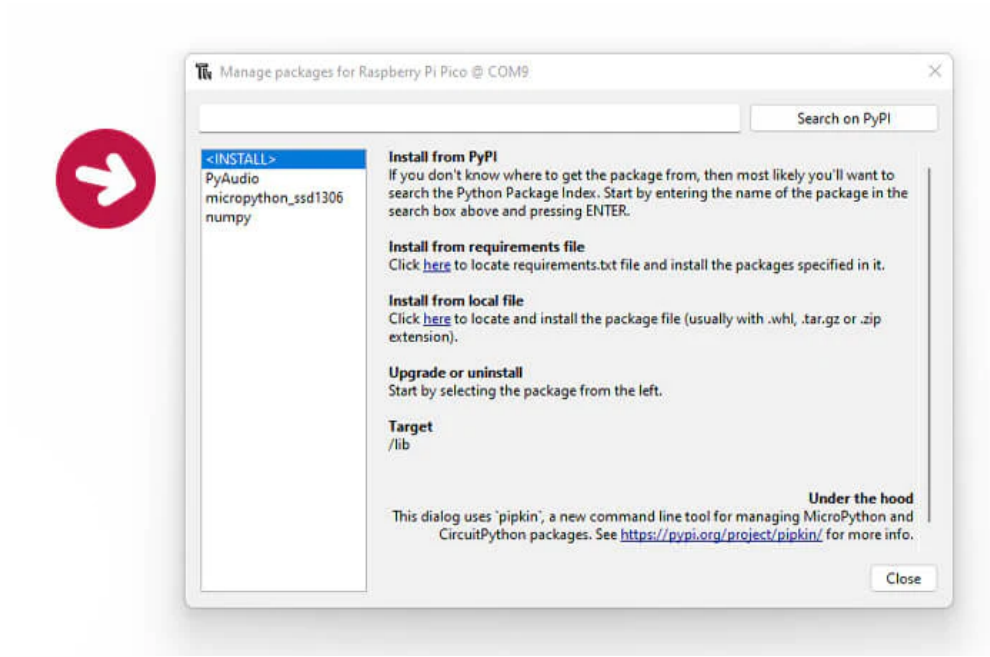


Select that line to go to the information screen for that module, then select the '***Install***' button at the bottom:





Once the package has installed it will show on your installed packages list on the left. It's now ready to use!



Activity 1: Simple Text Display

We'll start by showing a single line of text on the display to kick things off. Of course, it'll be the traditional "*Hello World!*"...

I2C and Display Setup

To use this display, we need to ***import I2C*** as well as the library we have just installed. You'll see this in the first few lines along with the other imports as always.

We then need to set up I2C and the display which you'll see on ***line 7***. This includes the GPIO pins we're using (***GPIO0*** and ***GPIO1***) which are the SDA and SCL pins, used for I2C connections. After this line we always have to wait around 1 second otherwise I2C can get all stroppy and fall over!

Next we define the display size (in pixels) and the type of driver chip it uses. Our display uses an ***SSD1306*** driver and is ***128x32*** pixels, which is reflected on ***line 13***. That's all the setup out of the way.

Displaying text

When we want to display something on the display, we always go through the same process:

- Clear the display

- Define what content we want to show on the display
- Push the content to the display

If we don't clear the display every time, it will write the new content on top of the old content, which makes a big mess of jumbled-up characters. Not good!

Our example shows us clearing the display with ***display.fill(0)***, then sending "*Hello World!*" with ***display.text("Hello World!",0,0)***, and finally pushing this to the display with ***display.show()***.

The Code

We'll cover what the the ***0,0*** arguments do in the next activity, but for now, copy this over to Thonny and see it for yourself!

```
# Imports
from machine import Pin, I2C
from ssd1306 import SSD1306_I2C
import time

# Set up I2C and the pins we're using for it
i2c=I2C(0,sda=Pin(0), scl=Pin(1), freq=400000)

# Short delay to stop I2C falling over
time.sleep(1)

# Define the display and size (128x32)
display = SSD1306_I2C(128, 32, i2c)

# Clear the display first
display.fill(0)

# Write a line of text to the display
display.text("Hello World!",0,0)

# Update the display
display.show()
```

Activity 2: Multiple Lines of Text

Let's go one step further, adding more lines and changing the position of the text, but first let's cover those arguments...

Arguments

You'll notice there are some arguments (***0,0***) after the text we push to the display:

```
display.text("Hello World!",0,0)
```

The ***first argument*** determines how many ***pixels across*** the display the content should start (the ***x-axis***) and the ***second argument*** determines how many ***pixels from the top*** the content should start (the ***y-axis***) going

downwards.

Our display is **128x32 pixels**, so we have 128 pixels running across on the x-axis, and 32 pixels top to bottom on the y-axis.

In our example below, we've managed to squeeze three lines of text on by setting each one at a different height, which just takes a bit of trial and error. The first line is at ***0***, the next line starts at ***12*** and the final line starts at ***24***. If we add more lines they won't be fully visible.

We've also moved ***line 2*** over to the right by entering ***50*** for the first argument, again, just with trial and error.

Give the example below a try, and then have a play with the argument values to see the changes for yourself.

```
# Imports
from machine import Pin, I2C
from ssd1306 import SSD1306_I2C
import time

# Set up I2C and the pins we're using for it
i2c=I2C(0,sda=Pin(0), scl=Pin(1), freq=400000)

# Short delay to stop I2C falling over
time.sleep(1)

# Define the display and size (128x32)
display = SSD1306_I2C(128, 32, i2c)

# Write three lines to the display
display.text("Line 1",0,0)
display.text("Line 2",50,12)
display.text("Line 3",0,24)

# Update the display
display.show()
```

Activity 3: The Endless Counter!

Small OLED displays are great for displaying data, and they can refresh quickly enough to show data changes at a very fast rate.

Let's create an endless (*well, kind of!*) counter to show another way of using these displays. When you come to make your own projects, things like this will be very useful.

We don't use any physical inputs here, just a counter in our code that increases by ***+1*** every loop.

The Code

A new problem - our counter is a number (an ***integer***) but our fussy little display will only show text (a ***string***), so we need to ***convert our counter to a string*** every time we run the loop.

Converting Integers to Strings

We convert the counter integer to a text string using the following line: ***display.text((str(counter)),0,24).*** So instead of entering text in brackets like we did in the previous activity, we use ***(str(counter)*** to turn our counter variable into a string for our display.

It's otherwise quite straightforward, adding ***+1*** to our counter at the end of every loop to continually increase it. You can play with the time delay within the while loop to see just how fast it can run:

```
# Imports
from machine import Pin, I2C
from ssd1306 import SSD1306_I2C
import time

# Set up I2C and the pins we're using for it
i2c=I2C(0,sda=Pin(0), scl=Pin(1), freq=400000)

# Short delay to stop I2C falling over
time.sleep(1)

# Define the display and size (128x32)
display = SSD1306_I2C(128, 32, i2c)

counter = 0 # Start our counter at zero

while True: # Loop forever

    display.fill(0) # Clear the display

    print(counter) # Print the current count

    # Show the counter on the display
    # The display library expects strings only
    # Counter is a number (integer) so we convert it to text (a string) with 'str'
    display.text("The Endless",0,0)
    display.text("Counter!",0,12)
    display.text((str(counter)),0,24)

    # Update the display
    display.show()

    # Short delay
    time.sleep(0.1)

    # Add 1 to our counter
    counter += 1
```

Activity 4: The Naughty or Nice Game!

We're now going to create a game that decides your festive fate! Let's make a *Naughty or Nice game*, which will tell you if you're on the good list this year or not (don't worry kids, it's just a bit of fun - we're *sure* you've all

been good this year!).

This project shows '*Naughty*' and '*Nice*' with a marker that very quickly switches between the two. When you click the button, it stops on whatever option the marker is on at that moment - your aim is to land on '*Nice*'.

The Code

Our example below uses all of our usual ingredients (imports, display setup, button pins...) and creates a ***state variable***, which we've used before. We use this variable to switch between the two ***if statements*** within our ***while loop***. But why do we want to do that?

We want our display to constantly move the **>** marker between '*Naughty*' and '*Nice*'. To do that we have two if statements which are triggered by our state variable being either ***0*** or ***1***. Each if statement starts by showing the text with the marker at a different position.

If the state is ***0***, the text pushed to the display shows the marker on '*Naughty*'. If the state is ***1***, the marker is on '*Nice*'. Each if statement also changes the state, so that on the next loop, the other if statement will trigger - which makes the marker jump between the two.

Each of the if statements then has a ***nested if statement*** inside it which checks if the button is pressed at that *very* moment. If it is, the code inside it updates the display to confirm which option was selected, followed by a 2 second pause:

```
# Imports
from machine import Pin, I2C
from ssd1306 import SSD1306_I2C
import time

# Set up the button
button = Pin(8, Pin.IN, Pin.PULL_DOWN)

# Set up I2C and the pins we're using for it
i2c=I2C(0,sda=Pin(0), scl=Pin(1), freq=400000)

# Short delay to stop I2C falling over
time.sleep(1)

# Define the display and size (128x32)
display = SSD1306_I2C(128, 32, i2c)

# Create variables
state = 0

while True: # Loop forever

    time.sleep(0.1) # Delay before changing marker

    if state == 0: # If state is 0

        display.fill(0) # Clear the display
        display.text("Naughty or Nice?",0,0) # Line 1
```

```
        display.text(">Naughty  Nice",0,24) # Line 3

        display.show() # Update the display

        state = 1 # State change

        if button.value() == 1: # If button clicked

            display.fill(0) # Clear the display
            display.text("Oh no!",0,0) # Line 1
            display.text(">Naughty  Nice",0,24) # Line 3

            display.show()  # Update the display

            time.sleep(2) # Delay

    elif state == 1: # If state is 1

        display.fill(0) # Clear the display
        display.text("Naughty or Nice?",0,0) # Line 1
        display.text(" Naughty >Nice",0,24) # Line 3

        display.show()  # Update the display

        state = 0 # State change

        if button.value() == 1: # If button clicked

            display.fill(0) # Clear the display
            display.text("Yay!",0,0) # Line 1
            display.text(" Naughty >Nice",0,24) # Line 3

            display.show()  # Update the display

            time.sleep(2) # Delay
```

Activity 5: Show me the light!

Showing sensor readings on your OLED is quite possibly one of the most useful and fun ways to put the display to good use.

We're going to run an example where we continually check the light sensor reading and push this to the display - it can even show the % symbol!

The example below includes similar code from the light sensor box (day #6) and the same OLED display code and approach we've taken above, so most of this should make sense if you've been following along.

The Code

The same initial OLED imports and setup code is present, and we've added the light sensor pin setup back in, linking it to ***GPIO26*** along with the required ***ADC import***.

We then start a ***while loop*** which takes a reading from the sensor on ***line 25***. Now, this line does a lot of things at once so let's break it down:

- It creates a variable called 'light'
- It takes a sensor reading
- It turns the reading into a percentage
- It rounds the reading percentage to one decimal place.

We did something similar on day #6 but this time we're doing everything in a single line of code. We've avoided this so far as it can be a little overwhelming, but we're on day #11 so we think you'll be OK!

We then push this data to our display. We use the first line of the display to show some simple text, and the second line to show the reading. Just like the previous examples, we have to convert our 'light' variable to a ***string*** to allow the display to use it. We also add the **%** symbol to the end.

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

```
# Imports
from machine import Pin, I2C, ADC
from ssd1306 import SSD1306_I2C
import time

# Set up I2C and the pins we're using for it
i2c=I2C(0,sda=Pin(0), scl=Pin(1), freq=400000)

# Short delay to stop I2C falling over
time.sleep(1)

# Define the display and size (128x32)
display = SSD1306_I2C(128, 32, i2c)

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

while True:

    # Delay
    time.sleep(0.5)

    # Read sensor value, turn it into a percentage, round to 1 decimal
    # Store this in a variable called 'light'
    light = round((lightsensor.read_u16())/65535*100,1)

    # Print the light reading percentage
    print(light)

    # Clear the display
    display.fill(0)

    # Write two lines to the display
    # The line turns our light variable into a string, and adds '%' to the end
```

```
display.text("Light level:",0,0)
display.text((str(light) + "%"),0,12)

# Update the display
display.show()
```

Day #11 Complete!

Aren't OLED displays just bags of fun? We've only just scratched the surface as well! There are lots of clever ways you can use these kinds of displays with graphics, fonts and other tricks, but we wanted to keep our activities relatively simple for beginners.

You'll find a ton of resources and other examples on the internet that you can use - we even have our own graphics tutorial which can apply to these displays too (with a few tweaks to the code).

Let's recap, what did we learn on day #11? Today we:

- Learnt how to wire an OLED to our Raspberry Pi Pico
- Introduced the I2C communication protocol
- Leant how to install packages in Thonny
- Learnt how to code an I2C OLED display, including:
  - How to write text to the display
  - How to write multiple lines of text
  - How to alter the position of text
  - How to display sensor data on OLED displays
  - A few little tricks like markers!
- Re-used knowledge and components from previous boxes, such as:
  - State variables
  - Converting integers to strings
  - Nested if statements
  - Light sensors
  - ...and more!

Ok folks, we're nearly at the end of our twelve days **:(** You can remove the circuit from today as tomorrow is an entirely different kind of component - we can't wait! See you tomorrow...

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

28 comments

The Pi Hut

January 11, 2023 at 1:19 pm

@Allan We've updated the text to indicate the light sensor resistor value (10k). Thanks for the suggestion :)

@Allan We've updated the text to indicate the light sensor resistor value (10k). Thanks for the suggestion :)

Allan

January 11, 2023 at 12:49 pm

Can you please update this to specify the value of the light sensor resistor to save us looking back at #6?

Can you please update this to specify the value of the light sensor resistor to save us looking back at #6?

Jon Taylor

January 11, 2023 at 12:48 pm

Skip my last message. I solved it when I realised those pins are i2c1 pins (whatever that means!)

Skip my last message. I solved it when I realised those pins are i2c1 pins (whatever that means!)

Jon

January 11, 2023 at 12:48 pm

Really enjoying all of these exercises and examples.

Is there a reason this doesn't work if you shift the connection to a different pair of i2c GPIO pins? For example I have shifted the display to use GP2 and GP3 which are both i2c SDA and SCL pins according to the schematic. When I run the code (with the pin numbers changed in the i2c call) I get the error "ValueError: bad SCL pin"

Really enjoying all of these exercises and examples.

Is there a reason this doesn't work if you shift the connection to a different pair of i2c GPIO pins? For example I have shifted the display to use GP2 and GP3 which are both i2c SDA and SCL pins according to the schematic. When I run the code (with the pin numbers changed in the i2c call) I get the error "ValueError: bad SCL pin"

Rowan Summerfield

January 11, 2023 at 12:47 pm

Ha, this always seems to happen when I write in! Well in case it helps others, my issue was happening because the right leg of the lightsensor was in the wrong hole on the breadboard, just 1 hole to the right instead of next to the other one :')

Ha, this always seems to happen when I write in! Well in case it helps others, my issue was happening because the right leg of the lightsensor was in the wrong hole on the breadboard, just 1 hole to the right instead of next to the other one :')

Rowan Summerfield

January 11, 2023 at 12:47 pm

I was hoping there would be a display in the kit somewhere! I was very happy to open today's box!
Question about the light sensor, it keeps floating between values 1.2 and 1.3% and not moving beyond that when I put my hand over it or put a torch right up close- any ideas? ((I tried copy and pasting your code in to eliminate code errors and still get it))

I was hoping there would be a display in the kit somewhere! I was very happy to open today's box!
Question about the light sensor, it keeps floating between values 1.2 and 1.3% and not moving beyond that when I put my hand over it or put a torch right up close- any ideas? ((I tried copy and pasting your code in to eliminate code errors and still get it))

Sheila

January 11, 2023 at 12:46 pm

Same error as Paul also syntax error. Oh, my and it was all going so well!!! Please help!!!
Sheila H in WV, USA
Hay! I'm 80 and really trying.

Same error as Paul also syntax error. Oh, my and it was all going so well!!! Please help!!!
Sheila H in WV, USA
Hay! I'm 80 and really trying.

Ben

December 28, 2022 at 1:08 pm

Had a little look through the graphics tutorial, suddenly there is a lot of fun to be had :)
I added a little bar meter to the light meter using these two lines…
display.rect(0,24,100,8,1)
display.fill_rect(0,24,int(light),8,1)

Had a little look through the graphics tutorial, suddenly there is a lot of fun to be had :)
I added a little bar meter to the light meter using these two lines…
display.rect(0,24,100,8,1)
display.fill_rect(0,24,int(light),8,1)

Emma

December 28, 2022 at 1:06 pm

Anybody else tried some ASCII-Art? I tried finding some good ones but since the display fits only around 3 lines it needs to be kinda minimalistic. I ended up with a bunny/owl/pikachu-shaped design which looks kinda cute and was curious if maybe someone else experimented and found other cool/cute/christmassy designs to try?

Here's mine:
display.text("(__/)",0,0)
display.text("(O.O)_",0,12)
display.text("(> <)/",0,24)

Anybody else tried some ASCII-Art? I tried finding some good ones but since the display fits only around 3 lines it needs to be kinda minimalistic. I ended up with a bunny/owl/pikachu-shaped design which looks kinda cute and was curious if maybe someone else experimented and found other cool/cute/christmassy designs to try?

Here's mine:
display.text("(__/)",0,0)
display.text("(O.O)_",0,12)
display.text("(> <)/",0,24)

The Pi Hut

December 23, 2022 at 8:38 am

@Neil – It might be the angle of the images you've uploaded, but it looks like you don't have the SCA/SCL pins wired correctly (looks like you're using physical pins 1 and 3 instead of 1 and 2?).

@Neil – It might be the angle of the images you've uploaded, but it looks like you don't have the SCA/SCL pins wired correctly (looks like you're using physical pins 1 and 3 instead of 1 and 2?).

Andriy

December 23, 2022 at 8:33 am

I had the same exact issue
File "", line 13, in

File “ssd1306.py”, line 110, in init
File “ssd1306.py”, line 36, in init
File “ssd1306.py”, line 71, in init_display
File “ssd1306.py”, line 115, in write_cmd
OSError: [Errno 5] EIO

Solution that worked for me:

from machine import Pin, I2C
from ssd1306 import SSD1306_I2C
import time

Change freq from 400000 to 40000
i2c=I2C, scl=Pin(1), freq=40000) Add device scanning
devices = i2c.scan() Define the display and size (128×32) AND add addr parameter
display = SSD1306_I2C(128, 32, i2c, addr=devices0)
while True:
display.fill(0)
display.text(“Test”,0,0)
display.show()

I had the same exact issue
File “”, line 13, in
File “ssd1306.py”, line 110, in init
File “ssd1306.py”, line 36, in init
File “ssd1306.py”, line 71, in init_display
File “ssd1306.py”, line 115, in write_cmd
OSError: [Errno 5] EIO

Solution that worked for me:

from machine import Pin, I2C
from ssd1306 import SSD1306_I2C
import time

Change freq from 400000 to 40000
i2c=I2C, scl=Pin(1), freq=40000) Add device scanning
devices = i2c.scan() Define the display and size (128×32) AND add addr parameter
display = SSD1306_I2C(128, 32, i2c, addr=devices0)
while True:
display.fill(0)
display.text(“Test”,0,0)
display.show()

Neil

December 23, 2022 at 8:35 am

Like others I too am getting the following error:

Traceback (most recent call last):
File “”, line 13, in
File “ssd1306.py”, line 110, in *init*
File “ssd1306.py”, line 36, in *init*
File “ssd1306.py”, line 71, in init_display
File “ssd1306.py”, line 115, in write_cmd
OSError: [Errno 5] EIO

I have reset and replaced all the jumpers to no avail. I have also tried simplifying the board to remove the sensor and button but nothing I try works. I have uploaded a couple of pictures here: https://imgur.com/a/72fb3AY in the hope that someone might be able to give me a pointer which would be a great Xmas present right now!

Like others I too am getting the following error:

Traceback (most recent call last):
File “”, line 13, in
File “ssd1306.py”, line 110, in *init*
File “ssd1306.py”, line 36, in *init*
File “ssd1306.py”, line 71, in init_display
File “ssd1306.py”, line 115, in write_cmd
OSError: [Errno 5] EIO

I have reset and replaced all the jumpers to no avail. I have also tried simplifying the board to remove the sensor and button but nothing I try works. I have uploaded a couple of pictures here: https://imgur.com/a/72fb3AY in the hope that someone might be able to give me a pointer which would be a great Xmas present right now!

Gavin

December 20, 2022 at 4:38 pm

I had the same errors as Julie and Thomas. Try removing and reseating the jumper wires, I did and the error went away as one want in as far as it could be..

I had the same errors as Julie and Thomas. Try removing and reseating the jumper wires, I did and the error went away as one want in as far as it could be..

Julie

December 19, 2022 at 10:59 am

I have installed ssd1306 package and it shows as present in Thonny. But now getting error messages
Traceback (most recent call last):
File “”, line 13, in
File “ssd1306.py”, line 110, in *init*
File “ssd1306.py”, line 36, in *init*
File “ssd1306.py”, line 71, in init_display

File “ssd1306.py”, line 115, in write_cmd

OSError: [Errno 5] EIO

Anyone else fixed this?

I have installed ssd1306 package and it shows as present in Thonny. But now getting error messages

Traceback (most recent call last):

File “”, line 13, in

File “ssd1306.py”, line 110, in *init*

File “ssd1306.py”, line 36, in *init*

File “ssd1306.py”, line 71, in init_display

File “ssd1306.py”, line 115, in write_cmd

OSError: [Errno 5] EIO

Anyone else fixed this?

Joe Edwards

December 19, 2022 at 11:01 am

Help! I’m working on a Mac and have run into a complete road block. First I don’t have a folder called Raspberry Pi Pico so I have created one and saved the code in it as ssd1306.py. However Thonny has greyed out the Manage Packages option so I cannot get any further. What do I do? Should I have had a ‘destination’ called Raspberry Pi Pico and if so where should not be – in with Thonny?

Help! I’m working on a Mac and have run into a complete road block. First I don’t have a folder called Raspberry Pi Pico so I have created one and saved the code in it as ssd1306.py. However Thonny has greyed out the Manage Packages option so I cannot get any further. What do I do? Should I have had a ‘destination’ called Raspberry Pi Pico and if so where should not be – in with Thonny?

The Pi Hut

December 15, 2022 at 11:42 am

If anyone is having trouble after copying the library code over and saving it to the Pico, you MUST make sure that you then open a NEW window/tab using File > New for your own code, otherwise you’re essentially going to save back over the SSD1306.py file which is why it’s not working. We’ll add this to the page shortly.

If anyone is having trouble after copying the library code over and saving it to the Pico, you MUST make sure that you then open a NEW window/tab using File > New for your own code, otherwise you’re essentially going to save back over the SSD1306.py file which is why it’s not working. We’ll add this to the page shortly.

Paul

December 15, 2022 at 11:39 am

@Michael – Please could you detail how you fixed this issue. I have the same issue, which I sent as a comment four days ago, but for some reason wasn’t published. Thanks

@Michael – Please could you detail how you fixed this issue. I have the same issue, which I sent as a

comment four days ago, but for some reason wasn't published. Thanks

Rob

December 15, 2022 at 11:39 am

Failed for me. I had to use the LINK for the code ssd1306.py. Got that installed in the Pico and that all seemed fine. Now if i try and run the code from activity one I get the following

Traceback (most recent call last):
File "", line 3, in
File "ssd1306.py", line 3, in
ImportError: can't import name SSD1306_I2C

Failed for me. I had to use the LINK for the code ssd1306.py. Got that installed in the Pico and that all seemed fine. Now if i try and run the code from activity one I get the following

Traceback (most recent call last):
File "", line 3, in
File "ssd1306.py", line 3, in
ImportError: can't import name SSD1306_I2C

Michael

December 15, 2022 at 11:39 am

Ignore my message, I wasn't using File>new, so the code was overwriting the ssd1306.py

Ignore my message, I wasn't using File>new, so the code was overwriting the ssd1306.py

Michael

December 14, 2022 at 5:21 pm

Hi,
I've set up the circuit as shown, saved the .py to the pico and done the manage package part, but when trying to run the first code I get
"Traceback (most recent call last):
File "", line 3, in
File "ssd1306.py", line 3, in
ImportError: can't import name SSD1306_I2C"

Not sure what I'm doing wrong but if you want me to email pictures to support I can do

Hi,
I've set up the circuit as shown, saved the .py to the pico and done the manage package part, but when trying to run the first code I get
"Traceback (most recent call last):
File "", line 3, in
File "ssd1306.py", line 3, in

ImportError: can't import name SSD1306_I2C"

Not sure what I'm doing wrong but if you want me to email pictures to support I can do

Thomas

December 15, 2022 at 11:39 am

We've tried to get activity 1 working using both methods but neither seems to be working, Its displaying the following message in Thonny:
Traceback (most recent call last):
File "", line 13, in
File "ssd1306.py", line 110, in *init*
File "ssd1306.py", line 36, in *init*
File "ssd1306.py", line 71, in init_display
File "ssd1306.py", line 115, in write_cmd
OSError: [Errno 5] EIO

We've made sure that all components are properly pushed down and in the right location.

We've tried to get activity 1 working using both methods but neither seems to be working, Its displaying the following message in Thonny:
Traceback (most recent call last):
File "", line 13, in
File "ssd1306.py", line 110, in *init*
File "ssd1306.py", line 36, in *init*
File "ssd1306.py", line 71, in init_display
File "ssd1306.py", line 115, in write_cmd
OSError: [Errno 5] EIO

We've made sure that all components are properly pushed down and in the right location.

Paul

December 15, 2022 at 11:39 am

I've successfully installed the package, and I can see it in the left hand bar, but keep getting this error when trying to run the first block of code.

Traceback (most recent call last):
File "", line 3, in
File "ssd1306.py", line 3, in
ImportError: can't import name SSD1306_I2C

Any clues anyone? Thanks

I've successfully installed the package, and I can see it in the left hand bar, but keep getting this error when trying to run the first block of code.

Traceback (most recent call last):
File "", line 3, in
File "ssd1306.py", line 3, in
ImportError: can't import name SSD1306_I2C

Any clues anyone? Thanks

The Pi Hut

December 12, 2022 at 10:40 am

@David Addis A few other users had this issue, so we've just added a manual method to the installation instructions (you may need to hit F5 to refresh the web page)

@David Addis A few other users had this issue, so we've just added a manual method to the installation instructions (you may need to hit F5 to refresh the web page)

David Addis

December 12, 2022 at 10:39 am

Is there an alternative way to install the SSD1306 library, as when I follow the instructions, and click on the top search result, it says 'Could not find the package info from PyPI. Error code <urlopen error [SSL:CERTIFICATE_VERIFY_FAILED] certificate verify failed: certificate has expired (_ssl.c:997)>'

Is there an alternative way to install the SSD1306 library, as when I follow the instructions, and click on the top search result, it says 'Could not find the package info from PyPI. Error code <urlopen error [SSL:CERTIFICATE_VERIFY_FAILED] certificate verify failed: certificate has expired (_ssl.c:997)>'

The Pi Hut

December 12, 2022 at 10:33 am

@REP We haven't had this issue, however we've found a discussion on it here (https://github.com/thonny/thonny/issues/1986). It seems to be an issue that affects certain user's operating systems but it's not clear why. There are a couple of workarounds on that page, one of which involves installing/updating a certificate which you have to download...we don't advise doing that unless you really know what you;re doing and the potential risks involved.

The other suggested solution is relatively easy. You go here (https://raw.githubusercontent.com/stlehmann/micropython-ssd1306/master/ssd1306.py), highlight all the code and copy that over to Thonny. Then select File > Save As and choose Raspberry Pi Pico when the option pops up. Give it the same file name of ssd1306.py and save. This now gives your Pico the library file it needs when you run the code on it.

@REP We haven't had this issue, however we've found a discussion on it here (https://github.com/thonny/thonny/issues/1986). It seems to be an issue that affects certain user's operating systems but it's not clear why. There are a couple of workarounds on that page, one of which involves

installing/updating a certificate which you have to download…we don't advise doing that unless you really know what you;re doing and the potential risks involved.

The other suggested solution is relatively easy. You go here (https://raw.githubusercontent.com /stlehmann/micropython-ssd1306/master/ssd1306.py), highlight all the code and copy that over to Thonny. Then select File > Save As and choose Raspberry Pi Pico when the option pops up. Give it the same file name of ssd1306.py and save. This now gives your Pico the library file it needs when you run the code on it.

The Pi Hut

December 12, 2022 at 10:10 am

@Steve – Please get in touch with our support team via support.thepihut.com and we'll help with that. If you can include a clear picture of your circuit that would help too. We've had a few customers not quite pushing the Pico into the breadboard fully (+ other little errors), so a picture really helps. Thanks.

@Steve – Please get in touch with our support team via support.thepihut.com and we'll help with that. If you can include a clear picture of your circuit that would help too. We've had a few customers not quite pushing the Pico into the breadboard fully (+ other little errors), so a picture really helps. Thanks.

Steve Harte

December 12, 2022 at 10:08 am

I cannot get the display to be recognised at all. Doing an i2c.scan() returns 0. I have copy/pasted your code and checked the connections 20 times. Do I have a faulty display?

I cannot get the display to be recognised at all. Doing an i2c.scan() returns 0. I have copy/pasted your code and checked the connections 20 times. Do I have a faulty display?

REP

December 12, 2022 at 10:22 am

Hi guys,

I'm getting this error when trying to install the ssd1306 library:
Could not find the package info from PyPI. Error code: <urlopen error [SSL: CERTIFICATE_VERIFY_FAILED] certificate verify failed: certificate has expired (_ssl.c:997)>

Hi guys,

I'm getting this error when trying to install the ssd1306 library:
Could not find the package info from PyPI. Error code: <urlopen error [SSL: CERTIFICATE_VERIFY_FAILED] certificate verify failed: certificate has expired (_ssl.c:997)>

Leave a comment

All comments are moderated before being published.

This site is protected by reCAPTCHA and the Google Privacy Policy and Terms of Service apply.

Name

E-mail

Message

SUBMIT

Related Posts

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...

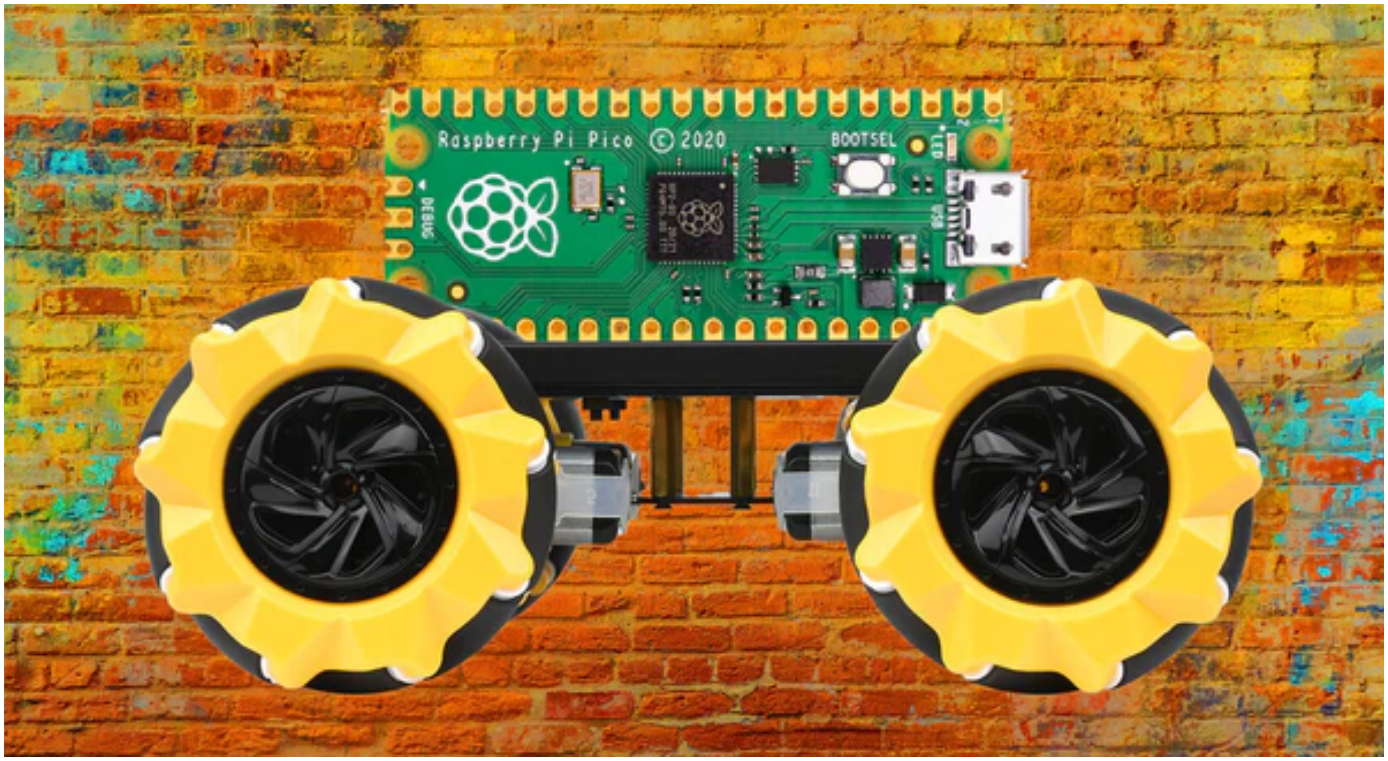
Read more

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



Build a Raspberry Pi Pico Robot with Mecanum Wheels

Tony Goodhew • Jul 24, 2023

In this tutorial, we'll show you how to build and code a Raspberry Pi Pico robot using mecanum wheels! Normal wheels allow the robot to move forwards, backwards, spin and...

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