Using an I2C OLED Display Module with the Raspberry Pi

27

BY MATT ON APRIL 8, 2018 I2C, TUTORIALS & HELP

Miniature OLED display modules are a great way to add a small screen to your Raspberry Pi projects. They are available in various sizes but common sizes include 128×32 and 128×64 pixels. The cheaper ones have single colour pixels that are either white, yellow or blue. My device has white pixels and uses an I2C interface which only requires four wires to be connected to the Pi.

In this tutorial I'll explain how I setup my 0.96" OLED display module using the Pi's I2C interface. Once setup it is easy to use Python to place text, draw shapes or even display simple images and animations.

The OLED Module

My OLED display module is a 0.96" I2C IIC SPI Serial 128X64 OLED LCD LED Display Module.

Search ... Search

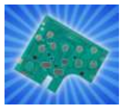
RECENT POSTS

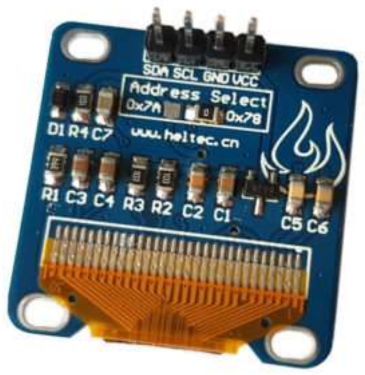
NOVEMBER 13, 2019 1
Ntablet an Open-source Tablet

OCTOBER 21, 2019 2
Pi-Hole OLED Status Screen

SEPTEMBER 22, 2019 0
KKSB Raspberry Pi 4 Steel Case

JUNE 24, 2019 0
Introducing the Raspberry Pi 4


JUNE 19, 2019 2
Gameboy Zero 6 Button Board from Aliexpress



It has four pins. Two are power (Vcc and Gnd) and two are for the I2C interface (SDA and SCL). The header may need to be soldered on before you can use it.

Update Operating System

As with all my projects I started off by creating an SD card with the latest Raspbian image. Then I made sure this was up-to-date by running the following commands :

```
sudo apt-get update
sudo apt-get upgrade
```

This step may take a few minutes if there are lots of packages to update but it usually saves some frustration in the future.

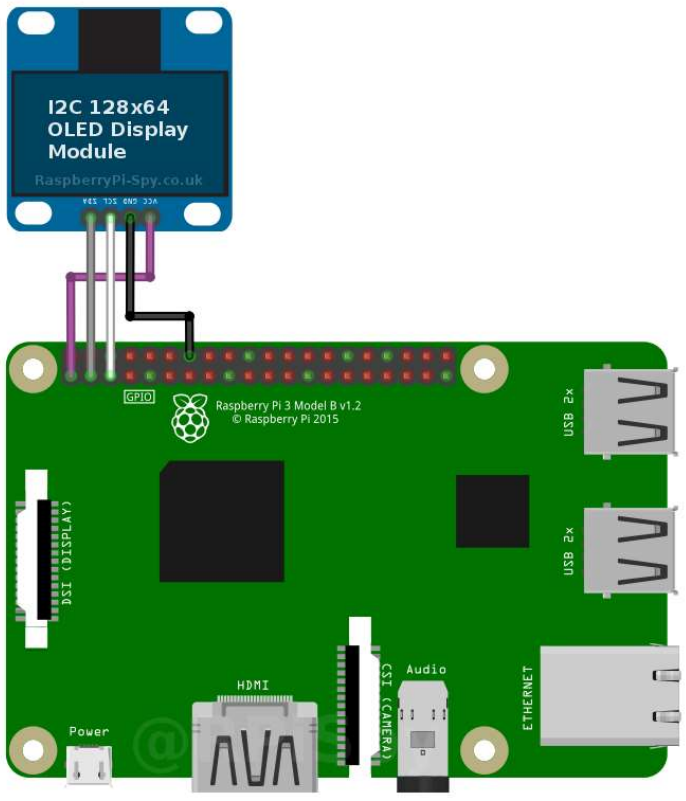
Display Module Setup

My screen had four pins, two for power and two for the I2C interface.

CATEGORIES

1-wire
3D Printing
Add-ons
BBC Micro:bit
BerryClip
Books
Camera Module
Cases
Events
General
Hardware
I2C
Infographics
Interfaces
Minecraft
Model A+
Model B+
News
Pi Models
Pi Zero
Power
Programming
Python
Raspbian
RetroGaming
Robotics
Sensors
Software
SPI
Tutorials & Help

Follow on Twitter 18K



I connected them directly to the Raspberry Pi's GPIO header using the following scheme :

OLED Pin	Pi GPIO Pin	Notes
Vcc	1 *	3.3V
Gnd	14 **	Ground
SCL	5	I2C SCL
SDA	3	I2C SCA

* You can connect the Vcc pin to either Pin 1 or 17 as they both provide 3.3V.

** You can connect the Gnd pin to either Pin 6, 9, 14 , 20, 25, 30, 34 or 39 as they all provide Ground.

Enable I2C Interface

The I2C interface is disabled by default so you need to enable it. You can do this within the raspi-config tool on the command line by running :

```
sudo raspi-config
```

For additional details on this step please see my how to Enable the I2C Interface on the Raspberry Pi post.

The following libraries may already be installed but run these commands anyway to make sure :

```
sudo apt install -y python3-dev
sudo apt install -y python-imaging python-smbus i2c-tools
sudo apt install -y python3-pil
sudo apt install -y python3-pip
sudo apt install -y python3-setuptools
sudo apt install -y python3-rpi.gpio
```

If you are using Python 2 then use these commands instead :

Subscribe on YouTube 1.4K

Like on Facebook

Follow on Instagram

TAGS

3D Printing audio battery berryclip Birthday bluetooth cambridge camera CamJam DigiMakers display games GPIO I2C interface Kickstarter LCD LED Linux media Minecraft Model A Model B motionEyeOS PCB photography photos Pi-Lite portable power python Raspberry Jam Raspberry Pi Bootcamp raspbian Retrogaming retroPie screen SD card security sensor SPI temperature ultrasonic video WiFi

RASPBERRY PI RELATED

- Adafruit Blog
- Average Maker
- Official RaspBerry Pi Site
- Raspberry Pi Pod
- RasPi.tv
- RaspTut
- Stuff About Code

TECH RESOURCES

- MattsBits – Pi Resources
- Microbit Spy
- Technology Spy

ARCHIVES

Select Month ▼

```
sudo apt install -y python-dev
sudo apt install -y python-imaging python-smbus i2c-tools
sudo apt install -y python-pil
sudo apt install -y python-pip
sudo apt install -y python-setuptools
```

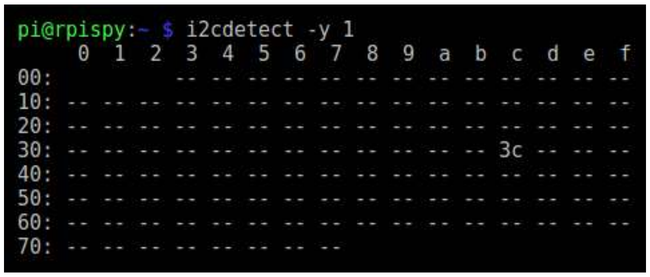
I would recommend using Python 3 unless you have a really good reason for using Python 2.

Finding the OLED Display Module's Address

With the I2C libraries installed I used the i2cdetect command to find the module on the I2C bus.

```
i2cdetect -y 1
```

and I got the following result :



```
pi@rpispy:~ $ i2cdetect -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- 3c -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

This was good news as it showed the device had been detected with an address of "0x3c". This is the default hex address for this type of device. I've got no idea why the device PCB suggests the address is "0x78" when it is clearly "0x3c".

If you've got an original Model B Rev 1 Pi then type the following command instead :

```
i2cdetect -y 0
```

Install OLED Python Library

In order to display text, shapes and images you can use the Adafruit Python library. It should work with all SSD1306 based displays including their own 128×32 and 128×64 devices.

To install the library we will clone the Adafruit git repository. Ensure git is installed by running :

```
sudo apt install -y git
```

Then clone the repository using the following command :

```
git clone https://github.com/adafruit/Adafruit_Python_SSD1306.git
```

Once that completes navigate to the library's directory :

```
cd Adafruit_Python_SSD1306
```

and install the library for Python 2 :

```
sudo python setup.py install
```

and/or Python 3 :

```
sudo python3 setup.py install
```

This process will give you ability to include the library within your own Python scripts.

Example Python Scripts

Now we are ready to test some examples scripts. Navigate into the “examples” directory :

```
cd examples
```

In there you should find a number of example scripts such as :

- animate.py
- buttons.py
- image.py
- shapes.py
- stats.py

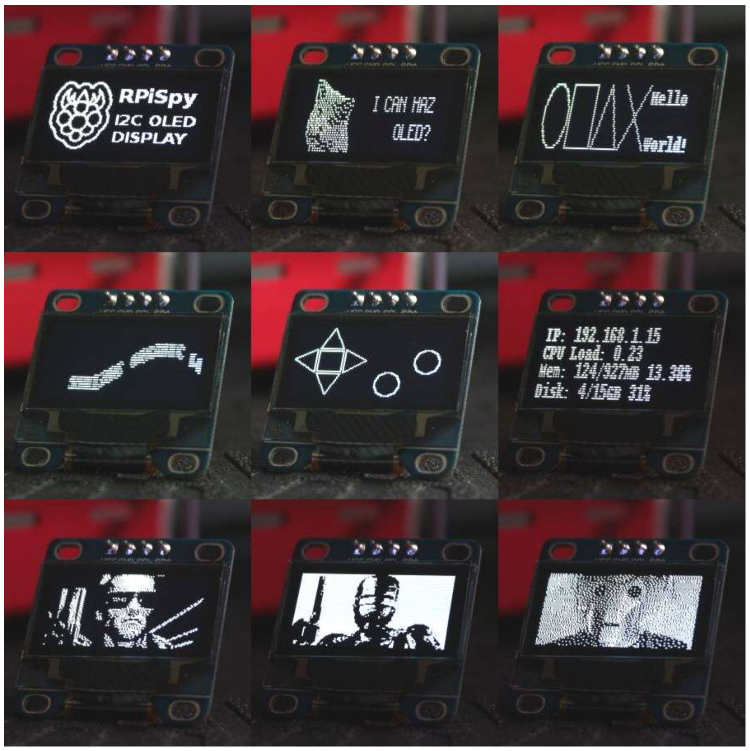
These examples can be run using :

```
python shapes.py
```

or using Python 3 :

```
python3 shapes.py
```

The examples should give you screens that appear in the examples below :

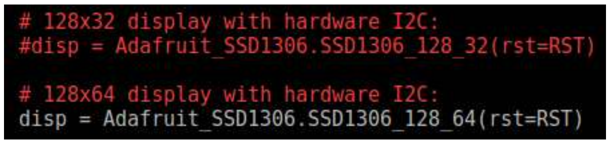


By modifying these scripts you can create your own graphics with shapes, images and text depending on your project. See if you can guess which ones were photos I downloaded to my Pi from Google Images!

Screen Size Adjustment

The Adafruit examples assume you have a 128×32 screen. They still run with a 128×64 pixel screen but it is better to change them before you move onto anything more complicated. To do this simply edit the scripts and disable the

128×32 config line by placing a # character at the front, and enable the 128×64 line by deleting the # character from the front. The section in the script should now look like this :



```
# 128x32 display with hardware I2C:
#disp = Adafruit_SSD1306.SSD1306_128_32(rst=RST)

# 128x64 display with hardware I2C:
disp = Adafruit_SSD1306.SSD1306_128_64(rst=RST)
```

This step becomes essential if you want to start creating your own images to display on the screen.

Creating New Images

So you tried image.py example and wondered how you can create your own images? It's fairly easy if you have an image editing application such as Photoshop or GIMP. I prefer using GIMP because it is free.

Ideally you want the images to be :

- 128×64 resolution
- 1-bit colour (i.e. black and white)

By default the image.py example will convert the image to 1-bit but it assumes the resolution is correct.

You'll notice in the script an alternative line which resizes and converts an image so you can load images without worrying about their size and colour.

Load image and convert to 1-bit color :

```
image = Image.open('happycat_oled_64.ppm').convert('1')
```

Load an image, resize and convert to 1-bit :

```
image = Image.open('example.png').resize((disp.width, disp.height), Image.ANTIALIAS).convert('1')
```

Which technique you use is up to you. Resizing and converting takes extra processing time so in high performance applications you are better feeding the script images that have already been resized.

Resizing and Converting Images

If you want to load an image or photo then load it into your graphics application and perform the following steps :

- Load image
- Resize/scale to 128×64
- Convert to 1-bit colour (monchrome)
- Export as a ".pbm" or ".png" file
- Copy to your Pi in the same location as your Python script
- Update the Python script to use your new file

The Adafruit example image is a "ppm" file because it is colour although it is converted to monochrome at the point it is displayed on the screen. Adafruit use ppm as the library also supports their colour OLED modules. If you don't have a colour screen you can switch to pbm or png.

I prefer creating "pbm" files as they are black and white and much smaller files. It also means your Python script doesn't need to convert them. The library can handle both just make sure you use the correct filename and extension in your scripts.

Increasing I2C Bus Speed

If you are displaying multiple images per second it is worth increasing the bus speed of the interface as it can improve performance. Please see the Change Raspberry Pi I2C Bus Speed post .

Troubleshooting

If your screen isn't working you should start at the beginning of this tutorial and work through it. Here are some thing to consider :

- Did you enable I2C and instal "python-smbus" and "i2c-tools"?
- Are the four module connections correct? Did you get SDA and SCL mixed up?
- Did "i2cdetect -y 1" give you the address of the display on the I2C bus?
- If your screen is using an address other than 0x3c did you adjust the Python script?

Buy a Miniature OLED Screen

These screens are available from a number of retailers so take a look and pick one that is convenient for your location :

- Adafruit
- The PiHut
- ModMyPi
- eBay
- Amazon

Read the descriptions carefully as some OLED display modules use the SPI interface rather than I2C. Those are fine but you'll need to follow a different tutorial to use that style.

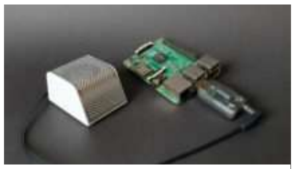
SHARE.

PREVIOUS ARTICLE
Raspberry Pi RetroPie Shutdown Button

NEXT ARTICLE
Cheap SD Cards from eBay are Fake

RELATED POSTS

OCTOBER 21, 2019 2
Pi-Hole OLED Status Screen


JUNE 4, 2019 3
Using a USB Audio Device with the Raspberry Pi

FEBRUARY 5, 2019 1
Setting up SSH Keys on the Raspberry Pi

DECEMBER 15, 2018 3
Running Flask under NGINX on the Raspberry Pi

DECEMBER 8, 2018 3
Remote Access to a Raspberry Pi using MobaXterm

NOVEMBER 5, 2018 0
Raspberry Pi 7-Segment LED Display Module using Python

SEPTEMBER 4, 2018 0
Using a Level Shifter With the Raspberry Pi GPIO

JUNE 16, 2018 0
Create an I2C OLED Display Slideshow with Python

MAY 13, 2018 1
Creating a Simple FTP Server with a Raspberry Pi

27 COMMENTS

JES WOODLAND on APRIL 20, 2018 7:53 AM

Thanks Matt. This has been a really helpful tutorial. There is very little info regarding these little screens beyond setting them up. This is far and away the most concise and well explained tutorial I have read about displaying your own images.

REPLY ›

PIETRO on MAY 15, 2018 5:07 PM

Hello,
Thank you for the helpful tutorial.
I have a problem while installing the python library. When i execute the command

```
sudo python setup.py install
```

i get the following output:

```
Extracting in /tmp/tmpoaljab
Traceback (most recent call last):
File “setup.py”, line 4, in
use_setuptools()
File “/home/pi/Adafruit_Python_SSD1306/ez_setup.py”, line 128, in use_setuptools
return _do_download(version, download_base, to_dir, download_delay)
File “/home/pi/Adafruit_Python_SSD1306/ez_setup.py”, line 108, in _do_download
_build_egg(egg, archive, to_dir)
File “/home/pi/Adafruit_Python_SSD1306/ez_setup.py”, line 57, in _build_egg
with archive_context(archive_filename):
File “/usr/lib/python2.7/contextlib.py”, line 17, in __enter__
return self.gen.next()
File “/home/pi/Adafruit_Python_SSD1306/ez_setup.py”, line 88, in archive_context
with get_zip_class()(filename) as archive:
File “/usr/lib/python2.7/zipfile.py”, line 770, in __init__
self._RealGetContents()
File “/usr/lib/python2.7/zipfile.py”, line 813, in _RealGetContents
raise BadZipfile, “File is not a zip file”
zipfile.BadZipfile: File is not a zip file
```

and library does not install. How can I solve this? Thank you!

REPLY ›

MATT on MAY 15, 2018 5:38 PM

There seems to be an issue with a missing file. Someone has raised this on the Adafruit github page and they have assigned the issue for someone to investigate. https://github.com/adafruit/Adafruit_Python_SSD1306/issues/22. Hopefully they will resolve it soon.

REPLY ›

IVAN KUZEV on JULY 17, 2018 4:00 PM

Is it possible to connect two screens?

REPLY ›

MATT on AUGUST 3, 2018 10:43 AM

It should be possible, but you would need a screen that allowed you to change the default I2C address. Some allow this by soldering two contacts together.

REPLY ›

KEVIN ACKLAND on SEPTEMBER 13, 2018 9:08 PM

Hi Ivan, you can get an I2C multiplexer to add 8 screens. I think you can get 2 x 8 channel multiplexers to 16 screens working

REPLY ›

ROBIN HOOD on AUGUST 7, 2018 8:16 PM

Matt- The reason why the PCB said 0x78 and you're reading 0x3C from i2c-tools is:

I2c-tools shifts the entire address to the left by 1 (adding the last bit for R/W). Shift 0x3C to left by 1 gives you 0x78 (and obviously vise versa).

All the best!

REPLY ›

MATT on AUGUST 7, 2018 8:30 PM

Thanks for explaining that. I've just Googled it and understand now. 0x3c is these 7 bits 111100 but once the R/W bit is included it becomes these 8 bits 1111000 which is 0x78.

REPLY ›

FRANK on SEPTEMBER 6, 2018 5:43 PM

I had to install python3-dev before the setup script ran ok. *sudo apt-get install python3-dev*

REPLY ›

MATT on SEPTEMBER 10, 2018 8:48 PM

python3-dev should already be installed in the latest Raspbian image.

REPLY ›

JEZ MCKEAN on JUNE 8, 2019 9:44 PM

Not in the `-lite` image.

Also needed `setuptools`, Image/PIL/pillow (?!), and RPi.GPIO, but I couldn't get them all installed at this time, so I gave up for now.

REPLY ›

MATT on JUNE 17, 2019 10:16 PM

I've just used this guide on a project and have made some updates. I think you have to install more libraries if using Raspbian Lite.

REPLY ›

DON KINGDON on OCTOBER 21, 2018 6:47 PM

Thanks for the tutorial – had my screen up and running in about 30 seconds with your excellent direction

REPLY ›

BOB RANDOM on OCTOBER 31, 2018 3:55 PM

THANK YOU SOOOO MUCH!
I tried to get it work for hours with the official tutorial of the screen, but it was far to complicated and did'nt work.
Luckily i found this Tutorial, it saved me from a lot of wasted time <3

REPLY ›

MUCHO on FEBRUARY 11, 2019 1:04 PM

Hello! This is really great and I got it to work perfectly. I was wondering where I would be able to change the GPIO pins as I want to set my relais there (respectively the setup forsees so).
I couldn't find the Pins 1,3,5,14 anywhere assigned in the code.
Thanks for your help.

REPLY ›

MATT on FEBRUARY 13, 2019 1:16 PM

1 and 14 are 3.3V and Ground. You can use any of the other power pins on the header :

There is only 1 other 3.3V pin and that is pin 17. There are plenty of Ground pins.

The I2C pins are the defaults. You can configure a new software I2C interface by editing the /boot/config.txt file and adding :

```
dtoverlay=i2c-gpio,i2c_gpio_sda=5,i2c_gpio_scl=6
```

This would set up a new interface at /dev/i2c-3. However I've never tried this myself.

To detect the device on the new interface you can use :

```
i2cdetect -y 3
```

REPLY ›

MUCHO on FEBRUARY 25, 2019 4:15 PM

thank you MATT, I edited to config.txt and got an output from the i2cdetect -y 3 command, but the result was empty… meaning no detection.

I am assuming that with i2c_gpio_sda=5 and i2c_gpio_scl=6 pins 5 and 6 are meant, I would change these to let's say 19 and 21?

Then again, I read something about a BUS and that I could control different interfaces with sam pins… ughhh. difficult & clueless I am

REPLY ›

MATT on FEBRUARY 27, 2019 1:31 PM

With the dtoverlay the numbers are GPIO references not physical pin numbers. So i2c_gpio_sda=5 means GPIO5, physical pin 29.

REPLY ›

G. ADAM on MARCH 2, 2019 1:41 PM

minor typo:

it should spell "sudo apt-get upgrade" and not "ugrade"

REPLY ›

MATT on MARCH 2, 2019 6:35 PM

Well spotted. I have updated the text 🙂

REPLY ›

IVÁN on APRIL 18, 2019 10:47 PM

Is the screen compatible with the Raspberry Pi Zero?

REPLY ›

MATT on MAY 2, 2019 12:47 PM

It works via I2C so should work just fine on the Pi Zero.

REPLY ›

JAIMYS on MAY 11, 2019 3:57 AM

Hi, fantastic tutorial – very clear!

I was wondering if you would know why when using this i2c screen it seems to effect my ability to also use a DS18B20 temperature sensor? (DS18b20 connects to BCM 4 using the One-Wire (w1) protocol).

Before detecting the OLED, when I run the following in cmd:

cd /sys/bus/w1/devices
ls

It returns a serial number (28-xxxx).
After I connect the i2c screen it seems to replace the temp sensor – replaced with a set of DS18b20 unrelated serial numbers… Any thoughts on what's going on would be greatly appreciated!

REPLY ›

TC on JUNE 18, 2019 8:33 PM

I've got an OLED display that displays in blue/yellow – it's also based on the 1306. How would I make an image that makes use of both of those colours? I'm guessing use two colours in the image file, but which ones?

Silly question, but I'm new to all this!

REPLY ›

MATT on JUNE 26, 2019 6:16 PM

I haven't got one of those blue/yellow boards but I suspect the colour is fixed in particular zones of the screen. I've only ever seen the different colours at the top and bottom of the screens.

REPLY ›

BEN on JULY 17, 2019 9:36 PM

how do you chang the screen size as you stated from 128×32 to 128×64 i cant figure how to do it please help

REPLY ›

MATT on SEPTEMBER 1, 2019 11:16 PM

Edit the example python script using a text editor and make the change as described in the "Screen Size Adjustment" section. You can edit the example scripts on the command line using :

```
sudo nano shapes.py
```

Make the change and then save/quit using CTRL-X, Y then ENTER.

REPLY ›

LEAVE A REPLY

Your Comment

Your Name

Your Email

Your Website

Save my name, email, and website in this browser for the next time I comment.

POST COMMENT

This site uses Akismet to reduce spam. Learn how your comment data is processed.

ABOUT

Unofficial site devoted to the Raspberry Pi credit card sized computer offering tutorials, guides, resources,scripts and downloads. We hope to help everyone get the most out of their Pi by providing clear, simple articles on configuring, programming and operating it.

POPULAR POSTS

JULY 27, 2012 100
16×2 LCD Module Control Using Python

OCTOBER 20, 2013 99
Analogue Sensors On The Raspberry Pi Using An MCP3008

SEPTEMBER 19, 2014 96
Top 5 Reasons The Raspberry Pi Sucks

RECENT POSTS

NOVEMBER 13, 2019 1
Ntablet an Open-source Tablet

OCTOBER 21, 2019 2
Pi-Hole OLED Status Screen

SEPTEMBER 22, 2019 0
KKSB Raspberry Pi 4 Steel Case

Entries RSS | Comments RSS