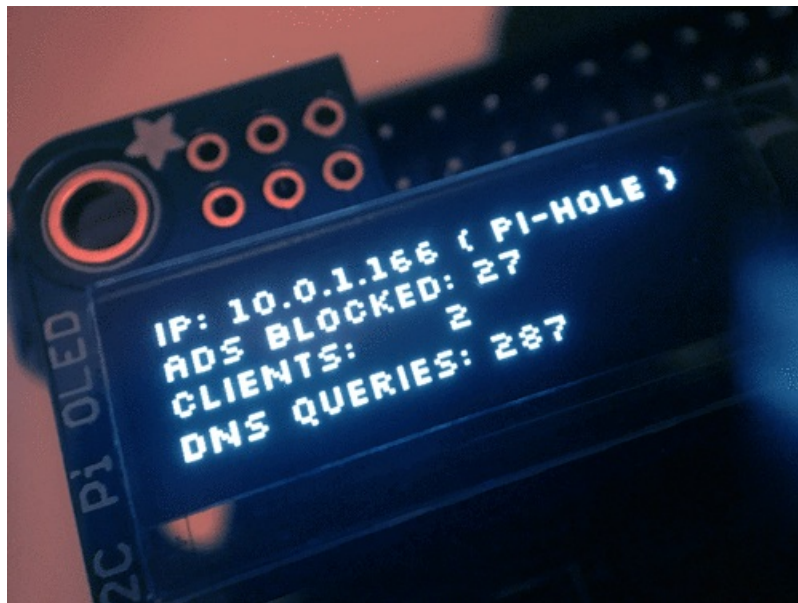


Pi Hole Ad Blocker with Pi Zero W

Created by lady ada



Last updated on 2019-12-13 08:49:59 PM UTC

Overview



A long time ago we made a Pi into a WiFi gateway that also blocked ads but the Pi Hole project does a *way* better job!

This project will make your Pi Zero W act as a DNS (**Domain Name Server**) The kind of device that tells you that **adafruit.com** is known as IP address 104.20.38.240.

Except Pi Hole DNS will do a special trick, when it is asked for the IP address of ads.adserver.com (for example) it will return nothing! So you will never even connect to the ad server and get the ad. Your connection will be faster, less data, and no intrusive ads. It works great on computers, tablets, phones, etc. Even if you cannot run an ad-blocker plugin on your phone or tablet, this will work and ad-blocker-detectors can't tell you're running it.

Unlike our WiFi gateway demo, you do not have to set up the Pi as your access point, you will only use it as a DNS ad blocker so it will not act as a bottleneck

The screenshot shows the EE Times website interface. At the top, there is a Rackspace logo and a banner that says "WITH MORE THAN 250+ DEDICATED EXPERTS.". The main header includes the EE Times logo, the tagline "Connecting the Global Electronics Community", a search bar, and navigation links for "REGISTER | LOGIN" and "About Us | Subscribe". Below the header is a horizontal menu with categories like "designlines", "Android", "Automotive", "Embedded", "Industrial Control", "Internet of Things", "MCU", "Medical", "Memory", "Open Source", and "PCB". A "BREAKING NEWS" banner is visible, followed by a "NEWS & ANALYSIS: Gen-Z Points to New Memories" section.

The main content area features a "designlines TEST & MEASUREMENT" section with a "Design How-To" article titled "A Current Sensing Tutorial—Part II: Devices" by Pete Semig and Collin Wells, Texas Instruments. The article is dated 2/15/2012 03:44 PM EST and has 3 comments. It includes social media sharing options (Like 11, Tweet, Share, 1, G+). The article text discusses the fundamentals of current sensing devices and the use of operational amplifiers for current sensing.

On the left side, there is a sidebar with a "There Are Plenty of Uses for Op-Amps" article and a "Get in touch with the new R&S® RTB2000 series oscilloscopes." advertisement. On the right side, there is a large advertisement for the "EXECUTIVE CONFERENCE" on October 22-24, 2017, with the slogan "SHIFT Not the Same Old Shift" and the ECIA logo.

We upgraded our Pi Zero Pi Hole with a little display, that makes setting up clients easy and also gives you some nifty stats!

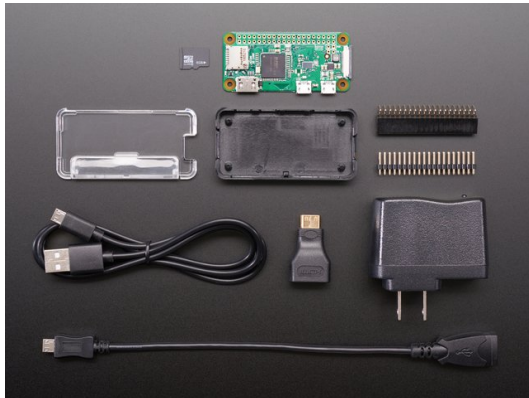
Follow along with this guide to DIY your own

Project Parts

This project can be done with any Raspberry Pi, but for the most adorably compact version we're using a Pi Zero W - this has enough power to do what we want, and has built in WiFi too!

Pi Zero W base parts

Its easiest if you pick up a Pi Zero W budget pack as it contains most everything you need



Raspberry Pi Zero W Budget Pack - Includes Pi Zero W

OUT OF STOCK

Out Of Stock

But you can also just DIY with the minimum requirements:

1 x [Pi Zero W](#)

the type of low cost game-changing product Raspberry Pi's known for - the super light, super lean microcomputer we've come to know and love, but now with built-in WiFi.

Add To Cart

1 x [4G or larger SD Card](#)

You will be burning this card with Raspbian Jessie Lite so its ok if its blank or pre-burned

Out Of Stock

1 x [Adafruit Pi Zero Enclosure](#)

Adafruit's classic, sturdy plastic enclosure. Keeps your Pi Zero safe and sleek.

Add To Cart

1 x [5V 1A USB wall adapter](#)

This one is plenty good and you can use any Micro USB cable with it

Out Of Stock

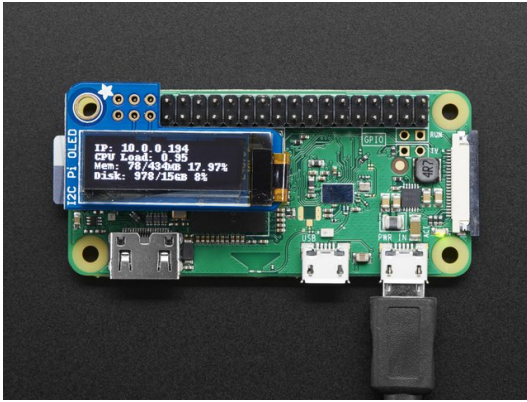
1 x [5V 2.4A USB wall adapter](#)

Super powerful for any uses, and comes with a built in MicroUSB cable

Add To Cart

Pi OLED Display Addition

If you want to add an OLED display (which is suggested!) you'll also need:



Adafruit PiOLED - 128x32 Monochrome OLED Add-on for Raspberry Pi

\$14.95
IN STOCK

Add To Cart

If you are using a Pi Zero W you'll need to add 2x20 headers too

1 x [2x20 Male Header](#)

Solder this in to plug in Pi HATs, GPIO cables, etc as you would into a normal Pi. Requires soldering

Add To Cart

or

1 x [2x20 No-Solder Hammer Headers](#)

If your soldering isn't quite up to scratch, or you just don't own a soldering iron yet, then these nifty hammer headers from Pimoroni might be just what you need.

Add To Cart

Other things you may need...

You also need a way to burn that SD card!



USB MicroSD Card Reader/Writer - microSD / microSDHC / microSDXC

\$5.95
IN STOCK

Add To Cart

Using a Pi 3 Instead of Pi Zero W

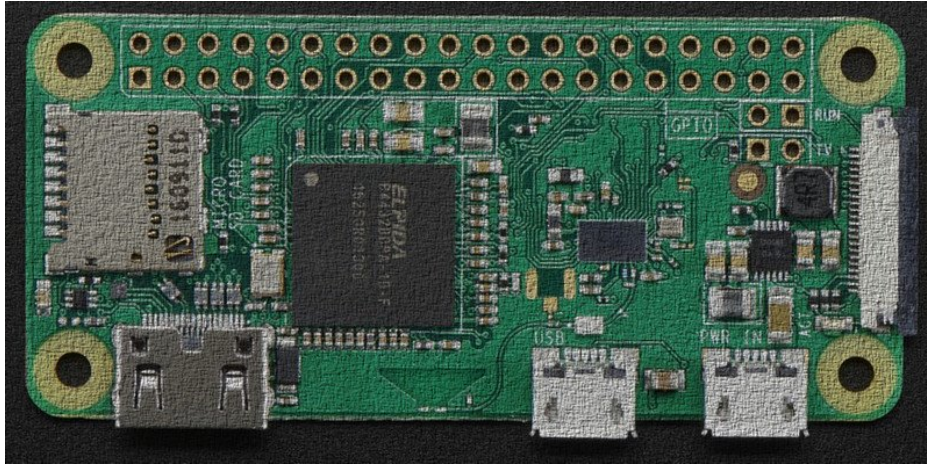
Instead of a Pi Zero W you can directly substitute in a Pi 3 which also has built in WiFi, you won't need the 2x20 header in that case

1 x **Raspberry Pi 3**

Raspberry Pi 3 with WiFi built in!

Add To Cart

Prepare the Pi



The Pi Zero W is a very minimal computer, so it requires a little work to get it up and running.

We have a guide on how to set up your Pi Zero W 'headless' which is how we recommend you get started. Check out the guide for how to do that!

<https://adafru.it/youC>

<https://adafru.it/youC>

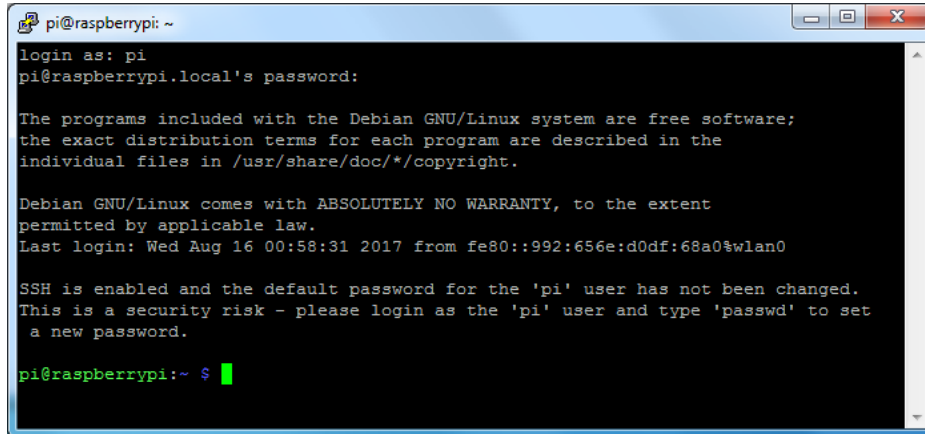
Here's the quick-start for people with some experience:

1. Download the [latest 'Lite' Raspbian](https://adafru.it/fQi) to your computer
2. [Burn the Lite Raspbian to your micro SD card](https://adafru.it/dDL) using your computer
3. [Re-plug the SD card into your computer \(don't use your Pi yet!\) and set up your wifi connection by editing supplicant.conf](https://adafru.it/youD)
4. [Activate SSH support](https://adafru.it/youD)
5. Plug the SD card into the Pi Zero W
6. If you have an HDMI monitor we recommend connecting it up via the mini HDMI adapter we provide in the budget pack - so you can see that it's booting OK
7. Plug in power to the Pi Zero W - you will see the green LED flicker a little. The Pi Zero will reboot while it sets up so wait a good 10 minutes
8. [If you are running Windows on your computer, install Bonjour support so you can use .local names, you'll need to reboot Windows after installation](https://adafru.it/IPE)
9. [You can then ssh into raspberrypi.local](https://adafru.it/jvB)

Install Pi Hole

Pre-Check

OK once you have set your Pi up and the WiFi is connecting to your home or office network, and you can `ssh` into it, continue with these easy steps! If you cannot connect via `ssh` yet, go back and read some of our guides until you are able to log into your Pi.



```

pi@raspberrypi: ~
login as: pi
pi@raspberrypi.local's password:

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Wed Aug 16 00:58:31 2017 from fe80::992:656e:d0df:68a0%wlan0

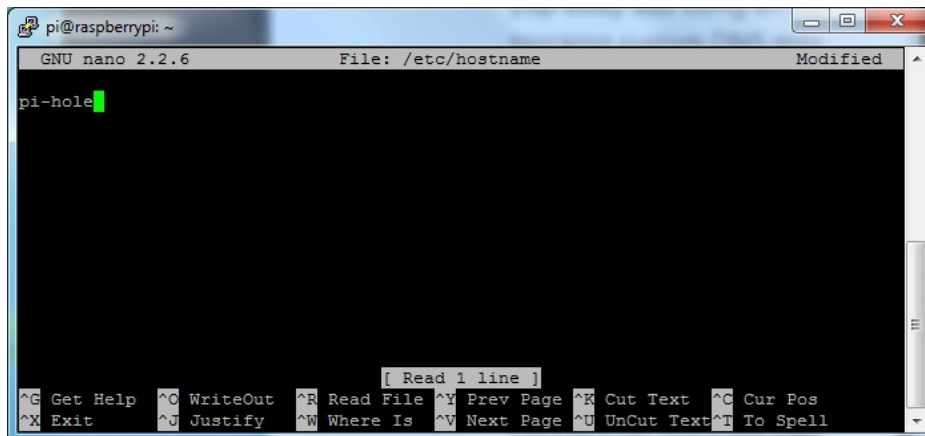
SSH is enabled and the default password for the 'pi' user has not been changed.
This is a security risk - please login as the 'pi' user and type 'passwd' to set
a new password.

pi@raspberrypi:~ $
  
```

Change Hostname

I like to do this first so I don't get confused between all the different Pi's in the house

Edit the hostname with `sudo nano /etc/hostname` and put something else on that first line, like `pi-hole`

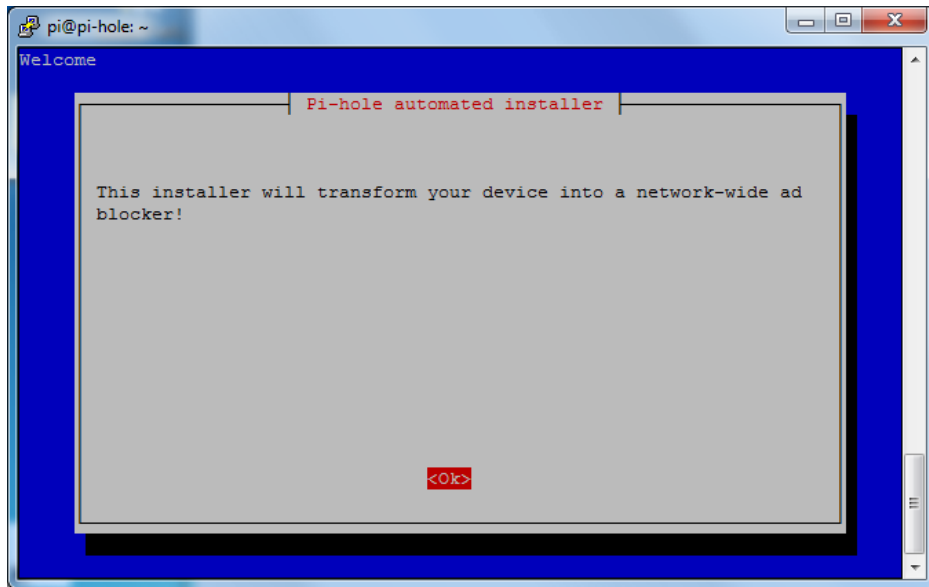


```

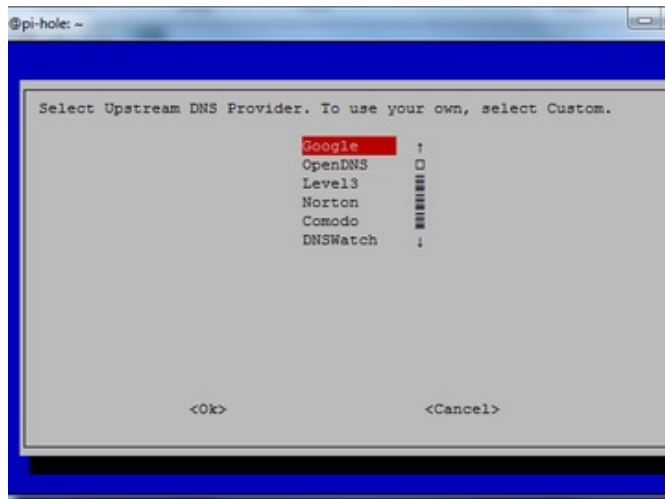
pi@raspberrypi: ~
GNU nano 2.2.6      File: /etc/hostname      Modified
pi-hole
[ Read 1 line ]
^G Get Help      ^O WriteOut      ^R Read File    ^Y Prev Page    ^X Cut Text     ^C Cur Pos
^X Exit          ^U Justify       ^W Where Is    ^V Next Page    ^U UnCut Text  ^T To Spell
  
```

Also change it in the hosts file with `sudo nano /etc/hosts` to match the same name. It's probably the last line:

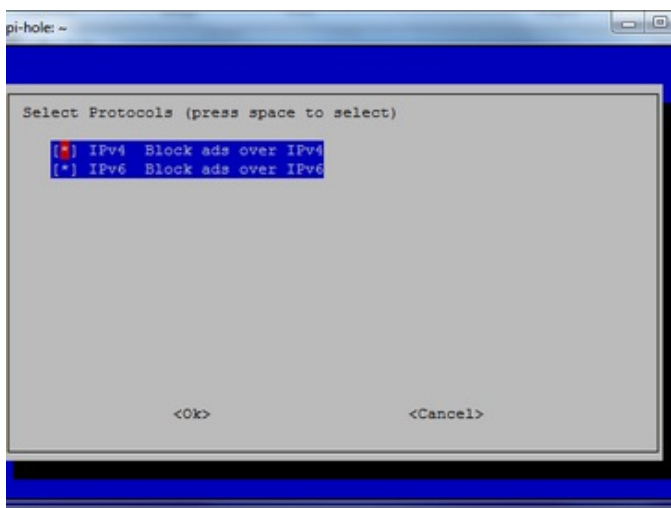

```
pi@pi-hole: ~  
..'''.  
:::  
::: You are root.  
::: Verifying free disk space...  
:::  
::: Updating local cache of available packages...█
```



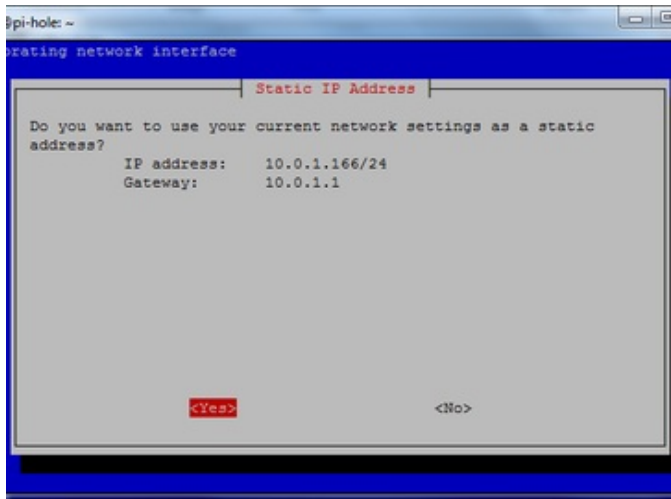
Configuration



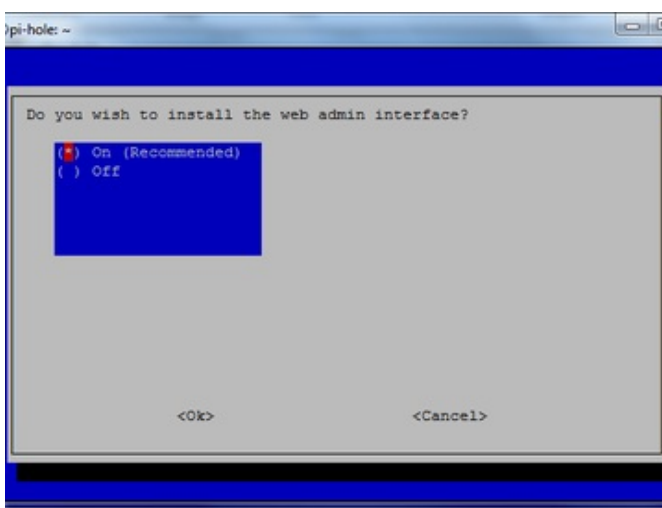
Pick who will be the upstream DNS (for non-ad blocked sites) - Google is fine and will probably be up all the time



99% of people will use IPv4 - if you need IPv6 you'd know!



The installer will automatically try to set the *dynamic* IP address it got from your router to be fixed. This works well enough, if you have an advanced network set up, you can configure a custom IP address

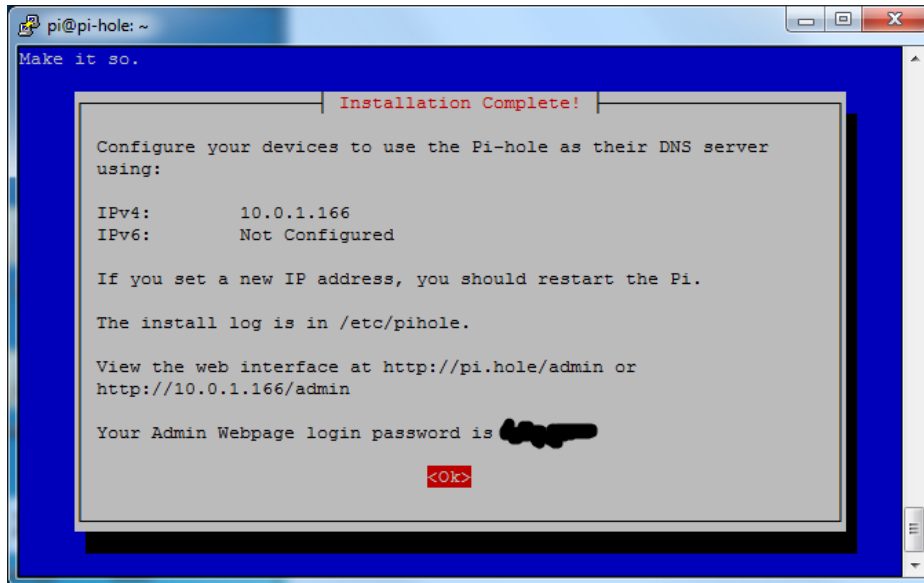


The Web Interface is kinda cool, and is password protected. We'll be showing most of the stats on the little OLED but we still need the API to be running so keep this on

It will keep installing! Just hold tight...

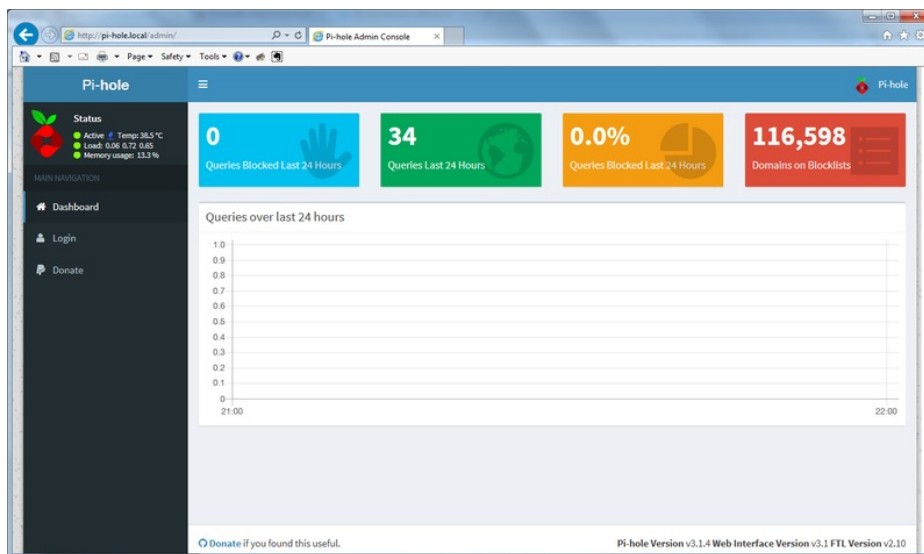
```
pi@pi-hole: ~
:::
::: Aggregating list of domains... done!
::: Formatting list of domains to remove comments.... done!
::: 141326 domains being pulled in by gravity...
::: Removing duplicate domains... done!
::: 116598 unique domains trapped in the event horizon.
:::
::: Adding adlist sources to the whitelist... done!
rm: cannot remove '/etc/pihole/local.list': No such file or directory
::: Whitelisting 6 domains... done!
::: Nothing to blacklist!
::: No wildcards used!
::: Formatting domains into a HOSTS file...::: Nothing to blacklist!
done!
:::
::: Cleaning up un-needed files... done!
:::
::: Refresh lists in dnsmasq... done!
::: DNS service is running
::: Pi-hole blocking is Enabled
:::
::: Starting pihole-FTL service... done.
:::
::: Enabling pihole-FTL service to start on reboot... █
```

When its done you'll get this final config screen! Copy & paste the password into another window for now



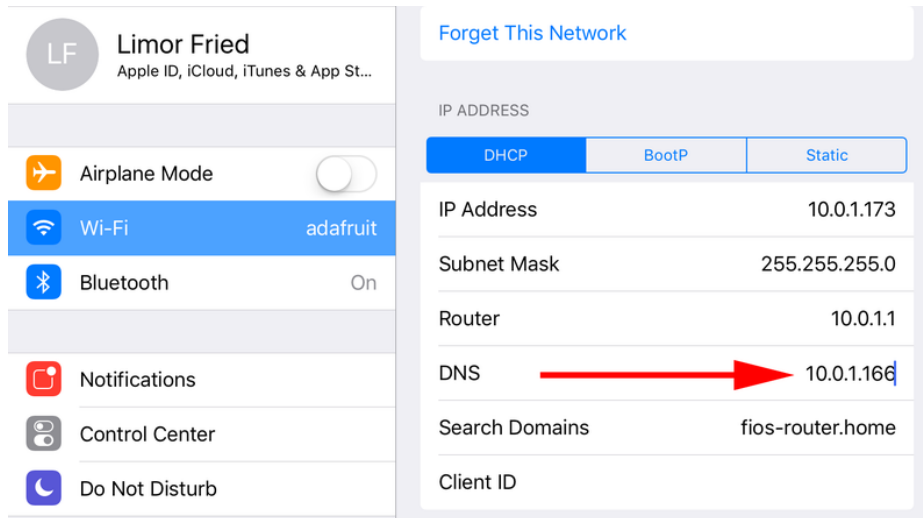
Test Admin Page

On your desktop computer or tablet, visit <http://pi-hole.local/admin/> (<https://adafruit.it/yluF>) And you should see an administration panel!



Test Blocking

On your tablet, phone, computer, etc - Set up your **DNS** server in the network settings to be the IP address of the Pi



You *may* need to restart your network or browser to have it kick in, also there may be some cached ads so don't worry if not everything is blocked. Visit your favorite site with ads (not adafruit.com cuz we don't have any! :) and see the difference!



Now that you've got that done, lets continue and install the display!

Install PiOLED

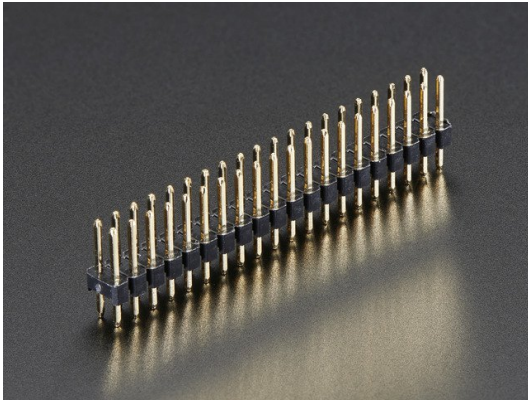
Our little PiOLED add on makes a very cute and easy way to display the Pi Hole stats. We were inspired to add this when we saw this tweet!



What a perfect use! Here's how to add it on for some nice stats. It also displays the hostname and IP address so if you forget it you can just look at the display. It will also tick up when its in use so you can tell its working.

Install 2x20 Header

If you are using a Pi Zero you'll need to solder in or somehow attach a 2x20 header so you can plug in the Pi OLED. Use either a plain 2x20 header and [solder it in using an iron + some solder...](https://adafru.it/drl) (<https://adafru.it/drl>)



Break-away 0.1" 2x20-pin Strip Dual Male Header

\$0.95
IN STOCK

Add To Cart

Or you can use Hammer headers which do not need soldering!

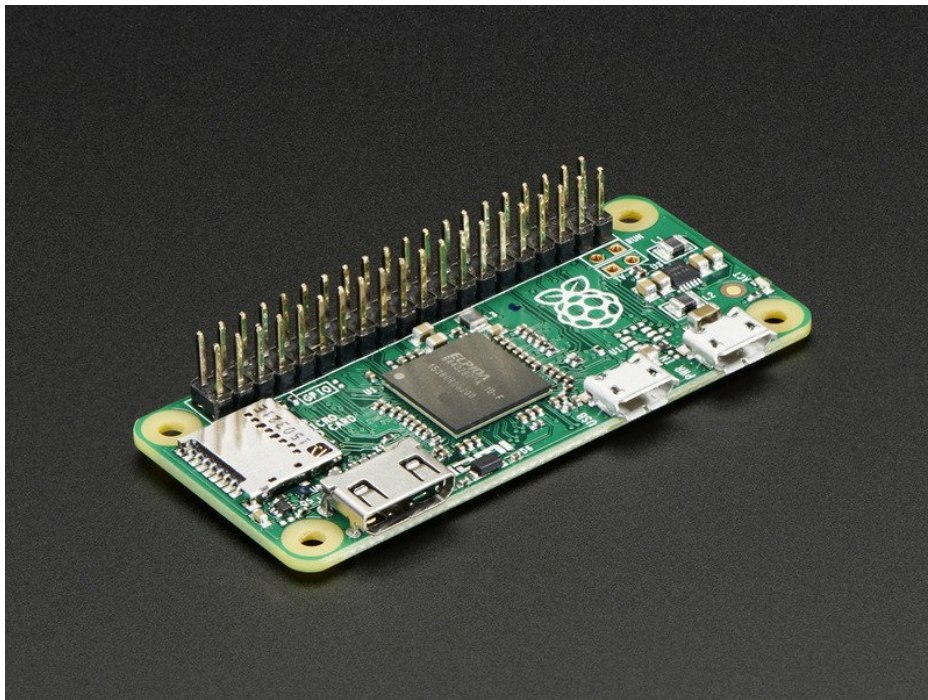


GPIO Hammer Headers - Solderless Raspberry Pi Connectors

\$6.50
IN STOCK

Add To Cart

Either way, you'll want to end up with something like this:



Install CircuitPython Libraries

This guide assumes that you've gotten your Raspberry Pi up and running, and have CircuitPython installed. If not, check out the guide:

<https://adafru.it/Deo>

<https://adafru.it/Deo>

To [install the library for the Pi OLED \(https://adafru.it/u1f\)](https://adafru.it/u1f), enter the following into the terminal:

```
sudo pip3 install adafruit-circuitpython-ssd1306
```

If that complains about pip3 not being installed, then run this first to install it:

```
sudo apt-get install python3-pip
```

We also need PIL to allow using text with custom fonts. There are several system libraries that PIL relies on, so installing via a package manager is the easiest way to bring in everything:

```
sudo apt-get install python3-pil
```

And let's also make sure the requests module is installed:

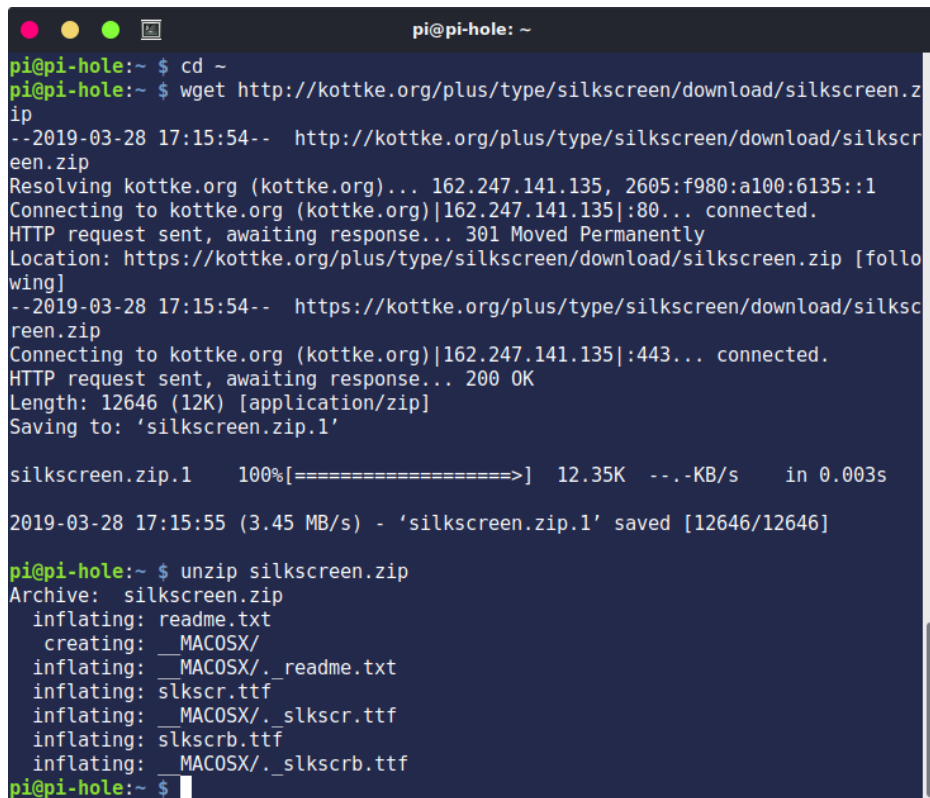
```
sudo pip3 install requests
```



A nice font really helps with this little OLED display, something other than the default PIL font. First thing I did is update the font so its a little clearer. [I used Kottke's free Silkscreen font which looks great on small screens. \(https://adafru.it/yvb\)](https://adafru.it/yvb)

It's easy to install on your Pi, run

```
cd ~
wget http://kottke.org/plus/type/silkscreen/download/silkscreen.zip
unzip silkscreen.zip
```



```
pi@pi-hole: ~
pi@pi-hole:~ $ cd ~
pi@pi-hole:~ $ wget http://kottke.org/plus/type/silkscreen/download/silkscreen.zip
--2019-03-28 17:15:54-- http://kottke.org/plus/type/silkscreen/download/silkscreen.zip
Resolving kottke.org (kottke.org)... 162.247.141.135, 2605:f980:a100:6135::1
Connecting to kottke.org (kottke.org)|162.247.141.135|:80... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: https://kottke.org/plus/type/silkscreen/download/silkscreen.zip [following]
--2019-03-28 17:15:54-- https://kottke.org/plus/type/silkscreen/download/silkscreen.zip
Connecting to kottke.org (kottke.org)|162.247.141.135|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 12646 (12K) [application/zip]
Saving to: 'silkscreen.zip.1'

silkscreen.zip.1  100%[=====>] 12.35K  --.-KB/s   in 0.003s

2019-03-28 17:15:55 (3.45 MB/s) - 'silkscreen.zip.1' saved [12646/12646]

pi@pi-hole:~ $ unzip silkscreen.zip
Archive: silkscreen.zip
  inflating: readme.txt
   creating: _MACOSX/
  inflating: _MACOSX/.readme.txt
  inflating: slkscr.ttf
  inflating: _MACOSX/.slkscr.ttf
  inflating: slkscrb.ttf
  inflating: _MACOSX/.slkscrb.ttf
pi@pi-hole:~ $
```

Update stats.py program

Here's the new stats.py code which uses the PiOLED.

Create a new file with `nano ~/pi/stats.py` and paste this code below in! Then save it.

```
# Copyright (c) 2017 Adafruit Industries
# Author: Ladyada, Tony DiCola & James DeVito
#
# Permission is hereby granted, free of charge, to any person obtaining a copy
# of this software and associated documentation files (the "Software"), to deal
# in the Software without restriction, including without limitation the rights
# to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
# copies of the Software, and to permit persons to whom the Software is
# furnished to do so, subject to the following conditions:
#
# The above copyright notice and this permission notice shall be included in
# all copies or substantial portions of the Software.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
# IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
# FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
# AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
# LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
# OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
# THE SOFTWARE.
```

```

# This example is for use on (Linux) computers that are using CPython with
# Adafruit Blinka to support CircuitPython libraries. CircuitPython does
# not support PIL/pillow (python imaging library)!

# Import Python System Libraries
import json
import subprocess
import time

# Import Requests Library
import requests

# Import Blinka
from board import SCL, SDA
import busio
import adafruit_ssd1306

# Import Python Imaging Library
from PIL import Image, ImageDraw, ImageFont

api_url = 'http://localhost/admin/api.php'

# Create the I2C interface.
i2c = busio.I2C(SCL, SDA)

# Create the SSD1306 OLED class.
# The first two parameters are the pixel width and pixel height.  Change these
# to the right size for your display!
disp = adafruit_ssd1306.SSD1306_I2C(128, 32, i2c)

# Leaving the OLED on for a long period of time can damage it
# Set these to prevent OLED burn in
DISPLAY_ON = 10 # on time in seconds
DISPLAY_OFF = 50 # off time in seconds

# Clear display.
disp.fill(0)
disp.show()

# Create blank image for drawing.
# Make sure to create image with mode '1' for 1-bit color.
width = disp.width
height = disp.height
image = Image.new('1', (width, height))

# Get drawing object to draw on image.
draw = ImageDraw.Draw(image)

# Draw a black filled box to clear the image.
draw.rectangle((0, 0, width, height), outline=0, fill=0)

# Draw some shapes.
# First define some constants to allow easy resizing of shapes.
padding = -2
top = padding
bottom = height - padding
# Move left to right keeping track of the current x position
# for drawing shapes.
x = 0

```

```

x = 0

# Load nice silkscreen font
font = ImageFont.truetype('/home/pi/slkscr.ttf', 8)

while True:
    # Draw a black filled box to clear the image.
    draw.rectangle((0, 0, width, height), outline=0, fill=0)

    # Shell scripts for system monitoring from here :
    # https://unix.stackexchange.com/questions/119126/command-to-display-memory-usage-disk-usage-and-
cpu-load
    cmd = "hostname -I | cut -d\ ' \ ' -f1 | tr -d '\ \ \ \ n \'"
    IP = subprocess.check_output(cmd, shell=True).decode("utf-8")
    cmd = "hostname | tr -d '\ \ \ \ n \'"
    HOST = subprocess.check_output(cmd, shell=True).decode("utf-8")
    cmd = "top -bn1 | grep load | awk ' \
        '\{printf \"CPU Load: %.2f\ ", $(NF-2)}\ '"
    CPU = subprocess.check_output(cmd, shell=True).decode("utf-8")
    cmd = "free -m | awk 'NR==2{printf \" \
        \"Mem: %s/%sMB %.2f%\ ", $3,$2,$3*100/$2 } \'"
    MemUsage = subprocess.check_output(cmd, shell=True).decode("utf-8")
    cmd = "df -h | awk '$NF==\"/\ \"\{printf \" \
        \"Disk: %d/%dGB %s\ ", $3,$2,$5}\ '"
    Disk = subprocess.check_output(cmd, shell=True).decode("utf-8")

    # Pi Hole data!
    try:
        r = requests.get(api_url)
        data = json.loads(r.text)
        DNSQUERIES = data['dns_queries_today']
        ADSBLOCKED = data['ads_blocked_today']
        CLIENTS = data['unique_clients']
    except KeyError:
        time.sleep(1)
        continue

    draw.text((x, top), "IP: " + str(IP) +
        "( " + HOST + ")", font=font, fill=255)
    draw.text((x, top + 8), "Ads Blocked: " +
        str(ADSBLOCKED), font=font, fill=255)
    draw.text((x, top + 16), "Clients: " +
        str(CLIENTS), font=font, fill=255)
    draw.text((x, top + 24), "DNS Queries: " +
        str(DNSQUERIES), font=font, fill=255)

    # skip over original stats
    # draw.text((x, top+8), str(CPU), font=font, fill=255)
    # draw.text((x, top+16), str(MemUsage), font=font, fill=255)
    # draw.text((x, top+25), str(Disk), font=font, fill=255)

    # Display image.
    disp.image(image)
    disp.show()
    time.sleep(DISPLAY_ON)
    disp.fill(0)
    disp.show()
    time.sleep(DISPLAY_OFF)

```



Running the OLED for a long period of time without changing the display can cause burn in. For that reason, the version of stats.py above does not show the information constantly.

You'll notice its very similar to the original **stats.py** but we've added PiHole API support. Here's how we did that!

First up, Pi Hole stats are available through the web server, in json format, so we need to add web requests and json parsing to python. Then set the URL for the API access, which is localhost (the same computer) and through the admin page:

```
import subprocess
import json
import requests

api_url = 'http://localhost/admin/api.php'
```

We load up the nice Silkscreen font here, in 8 point type. Note that we have to have the full path of the file.

```
# Load nice silkscreen font
font = ImageFont.truetype("/home/pi/slkscr.ttf", 8)
```

This is where we grab the API data. I put it in a **try** block, so it would retry in case the API access failed for some reason

```
# Pi Hole data!
try:
    r = requests.get(api_url)
    data = json.loads(r.text)
    DNSQUERIES = data['dns_queries_today']
    ADSBLOCKED = data['ads_blocked_today']
    CLIENTS = data['unique_clients']
except:
    time.sleep(1)
    continue
```

If you want to print out different info, run this small script in python to see what is available:

```
import json
import requests

api_url = 'http://localhost/admin/api.php'
r = requests.get(api_url)
data = json.loads(r.text)
print(data)
```

```
pi@pi-hole: ~
python3
Python 3.5.3 (default, Sep 27 2018, 17:25:39)
[GCC 6.3.0 20170516] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import json
>>> import requests
>>> api_url = 'http://localhost/admin/api.php'
>>> r = requests.get(api_url)
>>> data = json.loads(r.text)
>>> print(data)
{'privacy_level': 0, 'domains_being_blocked': 112361, 'reply_CNAME': 126, 'unique_domains': 89, 'queries_cached': 101, 'status': 'enabled', 'dns_queries_all_types': 405, 'ads_percentage_today': 8.888889, 'reply_IP': 60, 'reply_NXDOMAIN': 0, 'clients_ever_seen': 2, 'ads_blocked_today': 36, 'unique_clients': 2, 'queries_forwarded': 268, 'reply_NODATA': 18, 'dns_queries_today': 405, 'gravity_last_updated': {'relative': {'hours': '00', 'minutes': '48', 'days': '0'}, 'absolute': 1553790178, 'file_exists': True}}
>>>
```

You can also customize the display printout, but i liked having the IP first, then the pi hole stats below:

```
draw.text(x, top),          "IP: " + str(IP) + "( " + HOST + ")", font=font, fill=255)
draw.text(x, top+8),        "Ads Blocked: " + str(ADSBLOCKED), font=font, fill=255)
draw.text(x, top+16),       "Clients:    " + str(CLIENTS), font=font, fill=255)
draw.text(x, top+24),       "DNS Queries: " + str(DNSQUERIES), font=font, fill=255)
```

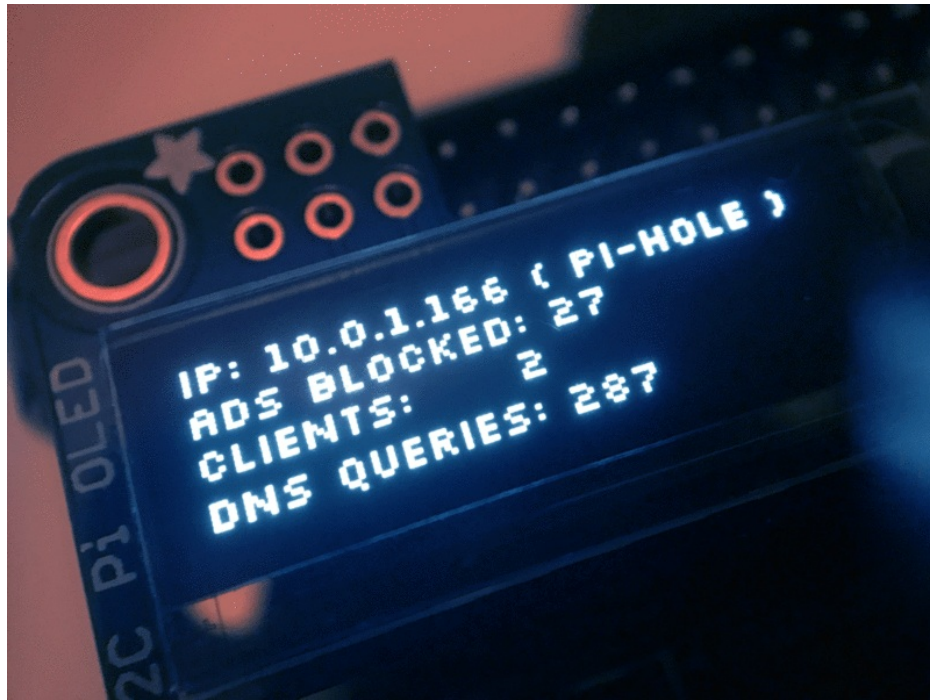
Test & Stats at Startup

Once you have the script saved, you can run it with `sudo python3 ~pi/stats.py` and look on the OLED to make sure you see your IP address and such!

Lastly we just want to make this run at boot. We'll do that the easy way by editing `/etc/rc.local` with `sudo nano /etc/rc.local` and adding `sudo python3 ~pi/stats.py &` before `exit 0`

```
pi@pi-hole: ~
GNU nano 2.7.4      File: /etc/rc.local      Modified
#
# By default this script does nothing.
# Print the IP address
_IP=$(hostname -I) || true
if [ "$_IP" ]; then
  printf "My IP address is %s\n" "$_IP"
fi
sudo python3 ~pi/stats.py &
exit 0
^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify  ^C Cur Pos
^X Exit      ^R Read File ^\ Replace  ^U Uncut Text ^I To Linter ^_ Go To Line
```

Then save and you can reboot to test it out

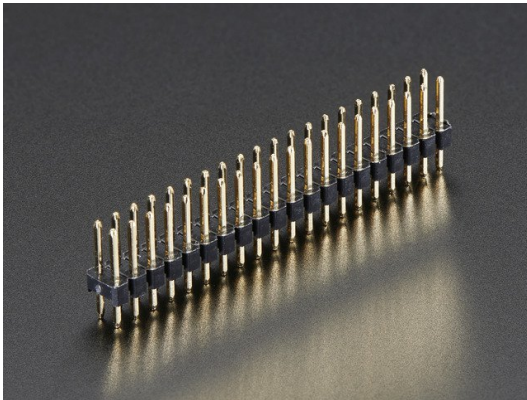


Install Mini PiTFT

We've updated our popular PiOLED script for use with the [Mini PiTFT \(https://adafru.it/HBK\)](https://adafru.it/HBK), a 135x240 Color TFT add-on for your Raspberry Pi. This cute little display has *two* tactile buttons on GPIO pins that we'll use to make a simple user-interface display for your Pi-Hole.

Install 2x20 Header

If you are using a Pi Zero, you'll need to solder on or somehow attach a 2x20 header so you can plug in the Pi OLED. Use either a plain 2x20 header and [solder it in using an iron + some solder... \(https://adafru.it/drl\)](https://adafru.it/drl)



Break-away 0.1" 2x20-pin Strip Dual Male Header

\$0.95
IN STOCK

Add To Cart

Or you can use Hammer headers which do not need soldering.

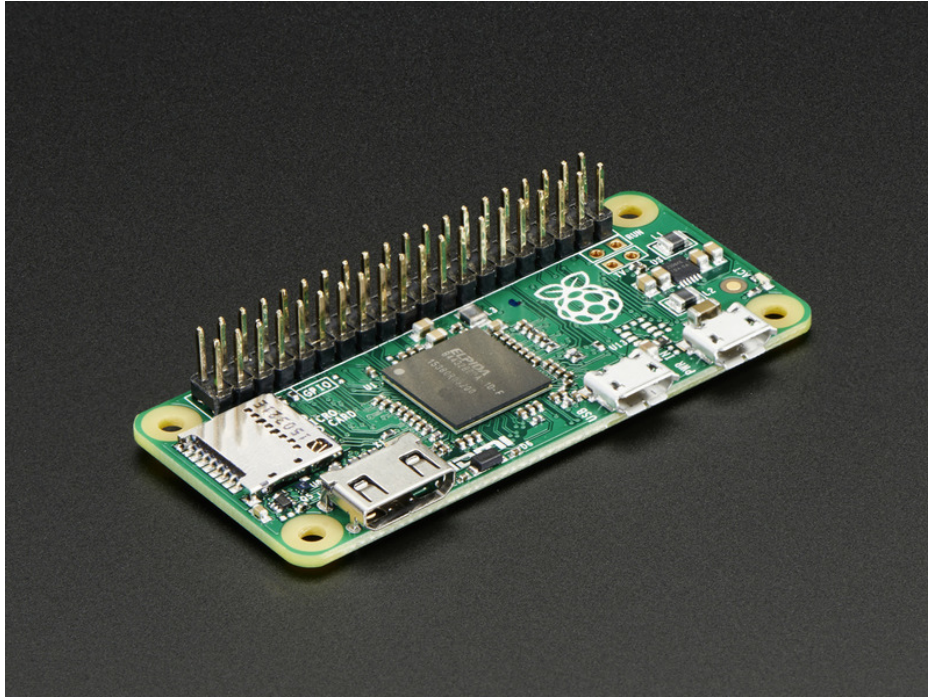


GPIO Hammer Headers - Solderless Raspberry Pi Connectors

\$6.50
IN STOCK

Add To Cart

Either way, you'll want to end up with something like this:



Python Setup

This guide assumes that you've gotten your Raspberry Pi up and running, have CircuitPython installed, and have installed CircuitPython libraries for the Mini PiTFT. If not, follow the steps in the guide below and come back to this page when you've completed them.

<https://adafruit.it/HBL>

<https://adafruit.it/HBL>

Update the stats.py program

Here's the new **stats.py** code which uses the Mini PiTFT.

Create a new file using nano `~pi/stats.py` and paste the code below in. Then, save the code.

Temporarily unable to load content:

You'll notice it's very similar to the original **stats.py**, but we've added PiHole API support. Here's how we did that:

First up, Pi Hole stats are available through the web server, in JSON format, so we need to add web requests and JSON parsing to Python. Then set the URL for the API access, which is localhost (the same computer) and through the admin page:

```
# Import Python System Libraries
import time
import json
import subprocess

# Import Requests Library
import requests

api_url = 'http://localhost/admin/api.php'
```

We load up the nice DejaVuSans font here. Note that we have to have the full path of the file.

```
# Alternatively load a TTF font. Make sure the .ttf font file is in the
# same directory as the python script!
# Some other nice fonts to try: http://www.dafont.com/bitmap.php
font = ImageFont.truetype('/usr/share/fonts/truetype/dejavu/DejaVuSans.ttf', 24)
```

This is where we grab the API data. I put it in a **try** block, so it would retry in case the API access failed for some reason

```
# Pi Hole data!
try:
    r = requests.get(api_url)
    data = json.loads(r.text)
    DNSQUERIES = data['dns_queries_today']
    ADSBLOCKED = data['ads_blocked_today']
    CLIENTS = data['unique_clients']
except KeyError:
    time.sleep(1)
    continue
```

If you want to print out different info, run this small script in python to see what is available:

```
import json
import requests

api_url = 'http://localhost/admin/api.php'
r = requests.get(api_url)
data = json.loads(r.text)
print(data)
```

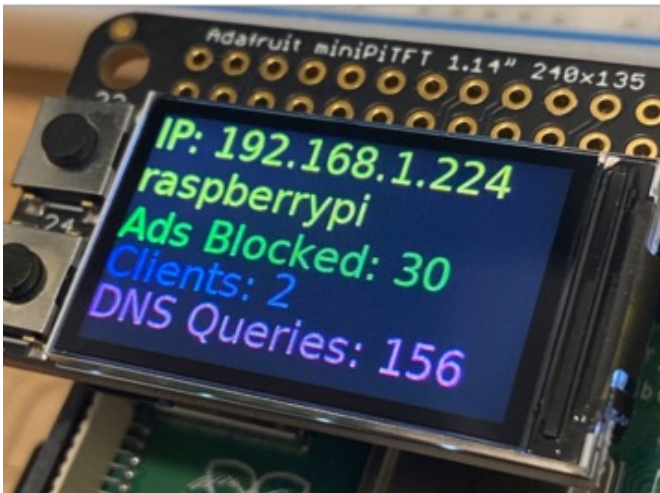
```
pi@pi-hole: ~
python3
Python 3.5.3 (default, Sep 27 2018, 17:25:39)
[GCC 6.3.0 20170516] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import json
>>> import requests
>>> api_url = 'http://localhost/admin/api.php'
>>> r = requests.get(api_url)
>>> data = json.loads(r.text)
>>> print(data)
{'privacy_level': 0, 'domains_being_blocked': 112361, 'reply_CNAME': 126, 'unique_domains': 89, 'queries_cached': 101, 'status': 'enabled', 'dns_queries_all_types': 405, 'ads_percentage_today': 8.888889, 'reply_IP': 60, 'reply_NXDOMAIN': 0, 'clients_ever_seen': 2, 'ads_blocked_today': 36, 'unique_clients': 2, 'queries_forwarded': 268, 'reply_NODATA': 18, 'dns_queries_today': 405, 'gravity_last_updated': {'relative': {'hours': '00', 'minutes': '48', 'days': '0'}, 'absolute': 1553790178, 'file_exists': True}}
>>>
```

Since the MiniTFT has *two* tactile push-buttons, we modified the script to print out extra information from the Raspberry Pi when you press the top button.

```
if not buttonA.value: # just button A pressed
    draw.text((x, y), IP, font=font, fill="#FFFF00")
    y += font.getsize(IP)[1]
    draw.text((x, y), CPU, font=font, fill="#FFFF00")
    y += font.getsize(CPU)[1]
    draw.text((x, y), MemUsage, font=font, fill="#00FF00")
    y += font.getsize(MemUsage)[1]
    draw.text((x, y), Disk, font=font, fill="#0000FF")
    y += font.getsize(Disk)[1]
    draw.text((x, y), "DNS Queries: {}".format(DNSQUERIES), font=font, fill="#FF00FF")
else:
    draw.text((x, y), IP, font=font, fill="#FFFF00")
    y += font.getsize(IP)[1]
    draw.text((x, y), HOST, font=font, fill="#FFFF00")
    y += font.getsize(HOST)[1]
    draw.text((x, y), "Ads Blocked: {}".format(str(ADSBLOCKED)), font=font, fill="#00FF00")
    y += font.getsize(str(ADSBLOCKED))[1]
    draw.text((x, y), "Clients: {}".format(str(CLIENTS)), font=font, fill="#0000FF")
    y += font.getsize(str(CLIENTS))[1]
    draw.text((x, y), "DNS Queries: {}".format(str(DNSQUERIES)), font=font, fill="#FF00FF")
    y += font.getsize(str(DNSQUERIES))[1]
```

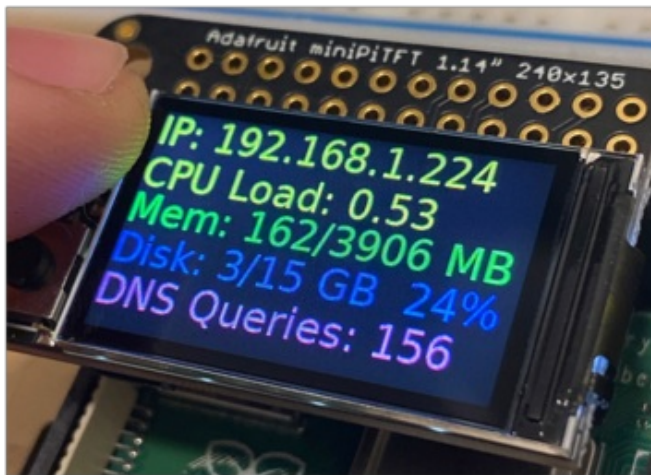
Test & Stats at Startup

Once you have the script saved, you can run it with `sudo python3 ~pi/stats.py`



Look on the Mini PiTFT to make sure you see your IP address along with some statistics from Pi-Hole.

Pushing the top button should display *extra* statistics about the Pi such as its hostname, CPU load, memory utilization, disk usage, and DNS queries.



Lastly we just want to make this run at boot. We'll do that the easy way by editing `/etc/rc.local` with `sudo nano /etc/rc.local` and adding `sudo python3 ~/pi/stats.py &` before `exit 0`

```
pi@pi-hole: ~
GNU nano 2.7.4 File: /etc/rc.local Modified
#
# By default this script does nothing.
# Print the IP address
_IP=$(hostname -I) || true
if [ "$_IP" ]; then
  printf "My IP address is %s\n" "$_IP"
fi

sudo python3 ~pi/stats.py &
exit 0

^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
^X Exit ^R Read File ^\ Replace ^U Uncut Text ^T To Linter ^_ Go To Line
```

Then save and you can reboot to test it out



