


Google Home Controlled LEDs

By MatthewD40 (/member/MatthewD40/) in Circuits (/circuits/) > Raspberry Pi (/circuits/raspberry-pi/projects/) 1,266 7 6



[Download](#)

[Favorite](#)



Lately, I've had a lot of free time on my hands, so I've been working on a bunch of projects.

This project will allow you to control RGB LEDs via Google Home using a Raspberry Pi. Now there are 3 parts of the project, setting up a Raspberry Pi, setting up Google Home with a custom command using IFTTT, and then creating a circuit for the lights. I made the circuit myself, but I bet you could get something from Adafruit that does what is needed.

Materials for Raspberry Pi Part

- **Raspberry Pi** - any will work, but I'm using a Zero
- **Wireless Dongle** - if the Pi doesn't have built in Wifi
- **Google Home** - Optional if you have Google Assistant on your phone

Materials for Circuit Part- Don't let this deter you... It's fairly simple

- **Protoboard**
- **Wire**
- **LED Strip**
- **12V Power Supply** - Anything above 2 Amps should be fine
- **DC Barrel Jack** - Same size as your power supply's
- **NPN BJT Power Transistors (x3)** - I'm using TIP31C
- **Male & Female Pin Headers** - Optional, but highly recommended

Step 1: Downloading Express for the Pi

```
pi@raspberrypi:~ $ mkdir piWebpage
pi@raspberrypi:~ $ cd piWebpage/
pi@raspberrypi:~/piWebpage $ ls
pi@raspberrypi:~/piWebpage $ express --view=ejs webApp

  create : webApp/
  create : webApp/public/
  create : webApp/public/javascripts/
  create : webApp/public/images/
  create : webApp/public/stylesheets/
  create : webApp/public/stylesheets/style.css
  create : webApp/routes/
  create : webApp/routes/index.js
  create : webApp/routes/users.js
  create : webApp/views/
  create : webApp/views/error.ejs
  create : webApp/views/index.ejs
  create : webApp/app.js
  create : webApp/package.json
  create : webApp/bin/
  create : webApp/bin/www

  change directory:
    $ cd webApp

  install dependencies:
    $ npm install

  run the app:
    $ DEBUG=webapp:* npm start

pi@raspberrypi:~/piWebpage $
```

I'm not going to go into too much detail about setting up the Pi because there are so many tutorials out there for setting them up.

What you'll need to do that I'm not covering...

- Flashing Raspberry Pi with newest Raspbian
- Setup the network card so you can access the internet from the Pi
- Set a static IP on the Raspberry Pi

Now here is where the fun begins! We need to install nodeJS, npm, express, and express-generator.

```
sudo apt-get update
sudo apt-get install nodejs npm
```

Once those are installed, run the following

```
npm install express express-generator
```

Express allows you to make a very basic webserver for your Raspberry Pi to use. Express-generator just auto generates files for an express server.

Make a directory and cd into the directory. I named mine piWebpage. Now run the following (seen in the picture too)

```
mkdir piWebpage
cd piWebpage
express --view=ejs webApp
```

This will generate a folder named webApp with all the express files in it. If you plan to do more with this webpage later and you like PUG, replace `--view=ejs` with `--view=pug`. We won't be touching the webpage, so for this application, it doesn't matter what we use.



Add Tip



Ask Question



Comment

Download

Step 2: Pi Server Setup

Move into the new webApp directory.

```
cd webApp
npm install
```

npm install will take some time because it is installing all the dependencies for express.

Paste `setColor.py` into the webApp folder. This file has some presets in it for basic colors. Feel free to add more as you like. The range is 0 to 255 where 255 is full color. At some point, I will likely add the ability to dim the lights, but for now, they are full brightness.

Move into routes

```
cd routes
```

Now replace `index.js` with the file attached. This will add some lines to receive a POST command which is what the Google Home will send. From that POST, we will get the color selected and tell the Pi to run the **setColor python script** to adjust the lights.

One last thing... Go back to webApp folder.

```
cd ~/piwebpage/webApp
```

Using your favorite editor, paste and save the code below into your `webApp.js`. Anywhere is fine **as long as it is before** `module.exports = app;`

```
// Setup LED hardware driver
const {exec} = require('child_process');
exec('sudo pigpiod', (err, stdout, stderr) => {
  if(err){
    console.log('Error loading LED Driver');
    return;
  }else{
    console.log('LED Driver Successfully Loaded');
  }
});
```

Like the comment says, `pigpiod` is the hardware driver for PWM signals that we will use to adjust the LED colors. I believe it comes already installed in Raspbian, but if not...

```
sudo apt-get install pigpiod
```

Now for the real test! Starting the server!

```
DEBUG=webapp:* npm start
```

**index.js**<https://codepen.io/matthewd40/pen/0b0e3f711940d1f554065d19e0a34321?editors=0010> (JREU5S4G.js)**setColor.py**<https://codepen.io/matthewd40/pen/0b0e3f711940d1f554065d19e0a34321?editors=0010> (JREU5T0A.py)

Add Tip



Ask Question



Comment

Download

Step 3: IFTTT Setup (Trigger)



is the

LED \$

Enter a \$ where you'll say the text ingredient

What's another way to say it? (optional)

Set LEDs \$

Enter a \$ where you'll say the text ingredient

And another way? (optional)

Set LED color \$

Enter a \$ where you'll say the text ingredient

What do you want the Assistant to say in response?

Setting color to \$

You can enter a \$ where you want to hear the text ingredient in the response

Language

English

IFTTT can do a lot, and I would highly recommend looking around at some of the applications.

First, you'll need to make an account. Use the same Google account associated to your Google Home, otherwise they won't sync together. Once complete and logged in, click the upper right of the IFTTT page where it shows your name and avatar. Then click **New Applet** from the dropdown.

If you're curious, IFTTT stands for **IF This Then That** if you didn't notice by the screen that pops up. So what we want is **If Google Assistant, then Webhook** as our options.

Proceed by clicking **+this** which will load a search bar. In the search, type **Google Assistant** and click the icon below the search.

In **Choose a Trigger**, select the 3rd option called *Say a phrase with a text ingredient*. Now this allows you to have 3 commands that will do the same action. You add the **\$** into phrase where you would mention the color. For example, if I naturally would say **Hey Google, Set LEDs Blue** (as natural as yelling at a device can be), then I would type into the field **Set LEDs \$**. Do that for all 3 fields with different versions of the command.

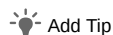
The 3 I used were

LEDs \$

Set LED color \$

The last field is what you would like your Google Home to reply after saying your command. It can be anything you want, but I used **Setting color to \$**. The **\$** means she will repeat the color back.

Click **Create Trigger**



Add Tip



Ask Question



Comment

Download

Step 4: IFTTT Setup (Action)

en

This action will make a web request to a publicly accessible URL. NOTE: Requests may be rate limited.

URL

http://xxx.xxx.xxx.xxx:3000/

TextField

Surround any text with "<<>>" to escape the content

Add ingredient

Method

POST



The method of the request e.g. GET, POST, DELETE

Content Type

text/plain



Optional

Body

TextField

After clicking **Create Trigger**, you'll load back to the **if this then that** view, but **this** has been replaced with the Google Assistant logo. Proceed by clicking **+that**

Same thing as before where it brings you to the search bar. Type in **Webhooks** and click the webhook icon below the search bar. Under **Choose Action** for Webhooks, there is only one option, so click **Make a web request**.

Here is where things get a little tricky. Because Google isn't another computer in your house, you'll need your external IP address. This will require some port forwarding, but we will touch on that later. To get your external IP address, go to <https://canyouseeme.org/>

In the **URL** field, type **http://xxx.xxx.xxx.xxx:3000/{{TextField}}** (with the x's being your external IP address). In case you're curious, TextField will have the color you selected when you make a command. The reason we use 3000 is because that is the port the Raspberry Pi Express server is running on. (You can change the port in the code, but we are just using the default settings for Express)

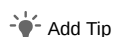
For **Method**, select **POST**.

For **Content Type**, select **text/plain**.

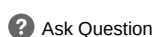
For **Body**, type in **{{TextField}}**

Those of you that know how a POST command works, you would think that if you parse the body property of the request that you would get the color. For some reason, nothing is ever put into the body field of the request, so I'm actually parsing the URL for the color. Hope that gets fixed soon, as that would simplify my code in the index.js route. But I digress.

Lastly, click **Create Action** and then **Finish** on the next page. (I turned off notifications, but that's preference)



Add Tip



Ask Question



Comment

Download

Step 5: Port Forwarding

Common Ports	
FTP	21
SSH	22
Telnet	23
SMTP	25
DNS	53
HTTP	80
POP3	110
IMAP	143
Other Applications	
Remote Desktop	3389
PC Anywhere	5631


This is where things get difficult to explain because all routers are different...

Now we have Google sending a command to our house using port 3000, but it doesn't know which device on the LAN it needs to go to. To remedy this, we need to forward port 3000 to the local IP address of your Raspberry Pi.

Go into your router using either 10.0.0.1 or 192.168.1.1 (I've also seen it where the last digit is 254) and find port forwarding. In port forwarding, similar to the image, you'll name a new device (**IFTTT**) and forward the port (**3000**) to the Pi's IP address (in my case **10.0.0.11**).

Save your new setting, reboot your router, and check to make sure your Raspberry Pi server is still running. If it isn't running, start it up again.

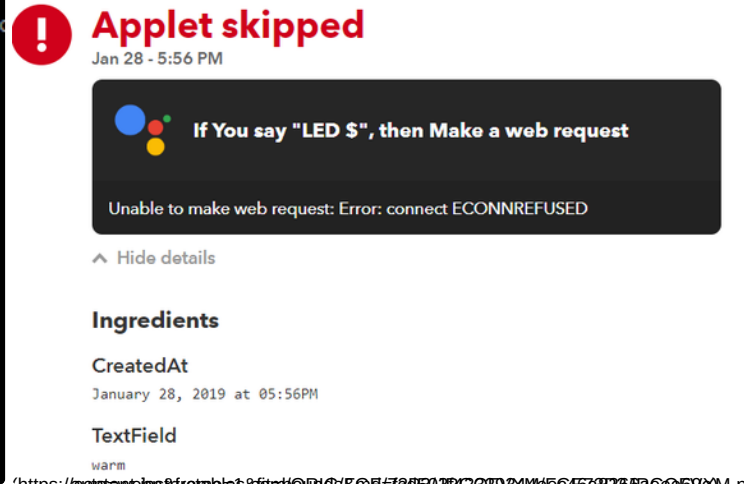
your IP address, it has a port checker. Assuming your port forwarding is correct, type in 3000 and hit **check port**. It should come back with a **Success**.

 Ask Question

Download

Step 6: Checking Your Work So Far

```
[piWebpage/webApp $ DEBUG=webapp:* npm start
90022] DeprecationWarning: os.tmpDir() is depre
start /home/pi/piWebpage/webApp
Listening on port 3000 +0ms
Successfully Loaded
piWebpage/webApp/setColor.py blue
200 250.458 ms - 2
```



Now... the moment you've been waiting for... Tell Google a command such as **LEDs blue** (if you followed my example).

Assuming all went correctly, you'll get the output seen in the picture. We don't have a circuit yet, so all you'll see is text on a screen. There is usually a second or 2 delay before it is processed from Google and appears on the Pi.

(Skip to next step if this came out similarly to the picture)

Now there are a few things to look at if it didn't work...

In the picture, there is a line that says

```
POST /color/blue 200 250.458 ms - 2
```

The 200 is the important part. If you are not seeing a 200, then there was a bad POST meaning your server didn't know what to do with the data. Go back to Step 2 and check your index.js file.

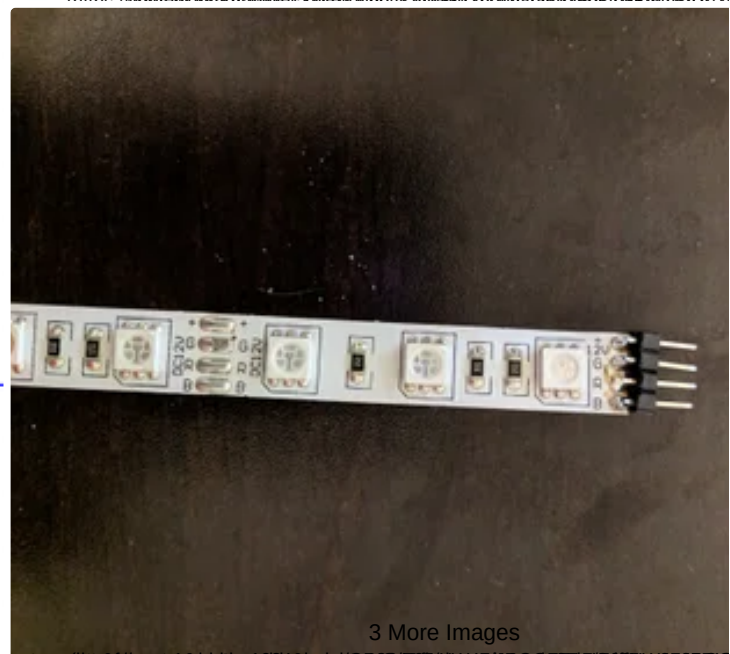
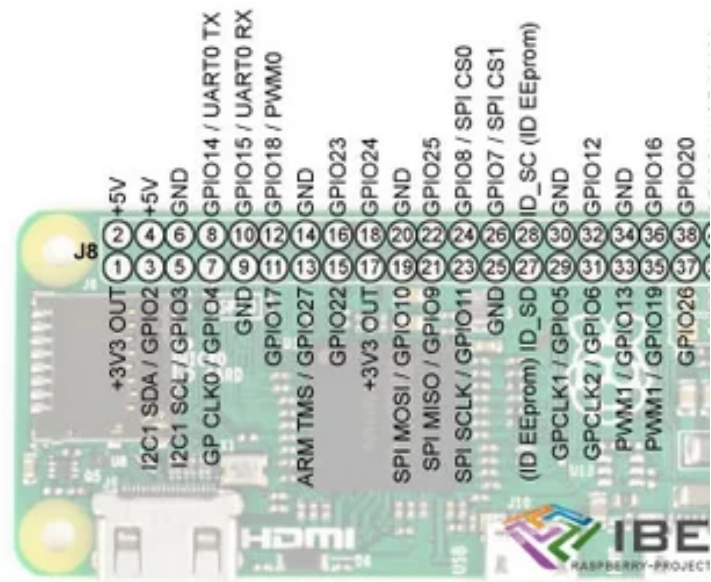
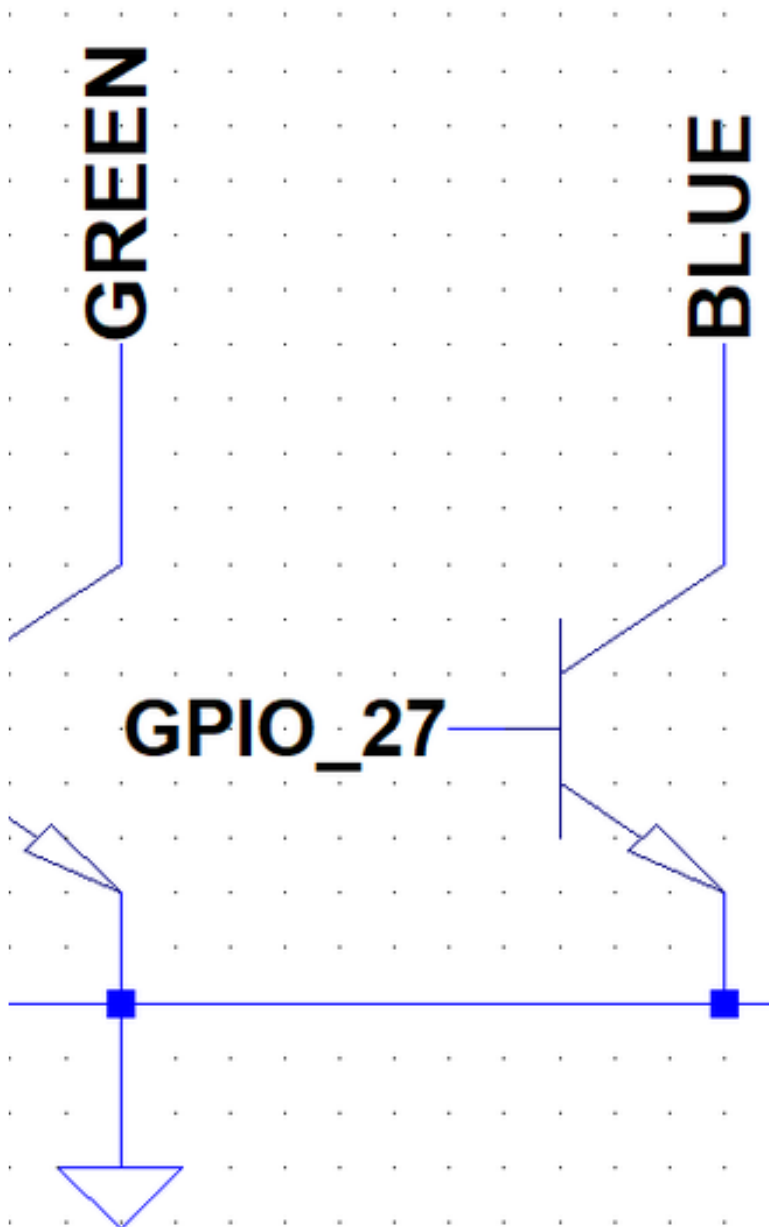
Also in the picture

```
stdout: blue
stderr:
```

This is the output from the python file that runs the LEDs. If you see an error there, it might be that you don't have the driver installed on Step 2.

Lastly, if nothing showed up at all... your IFTTT might not have been setup correctly or failed to connect to the server. Go back to the IFTTT page, and in the top nav bar, click **Activity**. In there, you can see every time your app has run, and if there was an error, you can see what it was. I made a Google command with the Pi server off and got the error in the picture.

Step 7: Building the Circuit



3 More Images

The reason why we need to do this is because the Raspberry Pi don't have enough power... So the solution is... **MORE POWER** (Tim the tool man Taylor grunts in the distance). AKA another power supply (12V 2A)

Materials for Circuit Part

- **Protoboard**
- **Wire**
- **RGB LED Strip**
- **12V Power Supply** - Anything above 2 Amps should be fine
- **DC Barrel Jack** - Same size as your power supply's
- **NPN BJT Power Transistors (x3)** - I'm using TIP31C
- **Male & Female Pin Headers**

Using the handy picture I stole from the interwebs with the GPIO of the Pi Zero, you can see

GPIO17, GPIO18, and GPIO 27 are all right next to each other with a **GND**. We will be using that square of 4 pins (Pins 11, 12, 13, 14).

First off, I would recommend soldering male headers onto your LED strip as seen in the image (not my best work). This allows for an easy disconnect if you ever need to. I used female headers for the connection of the LED strip to my protoboard and male headers from the protoboard to the Raspberry Pi. (Always use female connections for power/signal source). **You don't need to move all of the Pi's pins all next to each other** like I did... I just wanted it to look clean, but it was a lot of work.

Circuit Explanation

For those of you unfamiliar with transistors, transistors are basically a digital switch. The GPIO pins from the Pi trigger the three switches (red, green, and blue). Looking specifically at RED in the circuit diagram, when GPIO_17 is on, the switch "closes" connecting RED to GND causing the red lights to turn on. When GPIO_17 turns off, then the switch is open and thus the lights are off.

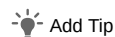
Base - GPIOs

Collector - Colors (RED, GREEN, BLUE)

Emitter - Ground (of both the Power Supply and Pi)

Make sure to connect the Pi's ground to the ground of the power supply. The lights will still work, but they will appear very dim until the ground is connected.

I have a 4th transistor looking device on my protoboard. It is a L7805CV which is used to convert 12V to 5V so I could power the Pi on the same circuit. It worked but kept over heating, so I removed it's connections.

[Download](#)

Step 8: Test It Out!

Google Home Controlled LEDs



Once complete with the circuit, *restart your Pi before making any connections*. This is because the pins are probably still active from the server test. Alternatively, you can kill the server and the **pigpiod** service.

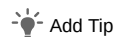
Plug in the LEDs and the jumpers from the protoboard to the Pi. Double check all connections before providing power. If you wired it wrong, you might fry your Pi (no pressure).

Checklist

- Check wires
- Power Pi
- Power Circuit
- Start Server (**DEBUG=webapp:* npm start** while in the ~/piWebpage/webApp directory)
- Tell Google to do your bidding!

CONGRATULATIONS you didn't blow anything up, and you can now control your LEDs from Google Home.

Leave comments if you're having issues, and I'll do my best to get back to you!



Add Tip



Ask Question



Comment

Download

Be the First to Share

Did you make this project? Share it with us!

I Made It!

Recommendations



(/Automated-Overhead-Camera-Assistant-for-Instructional-Videos/)

Automated Overhead Camera Assistant for Instructional Videos (/Automated-Overhead-Camera-Assistant-for-

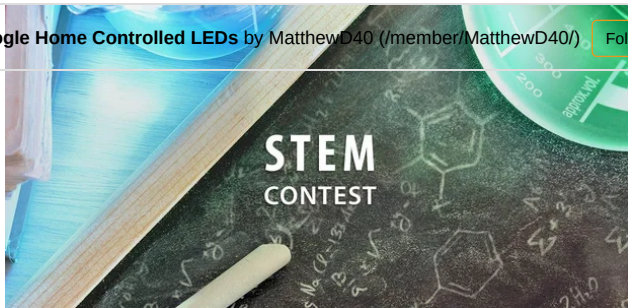
♥ 30 👁 2.5K



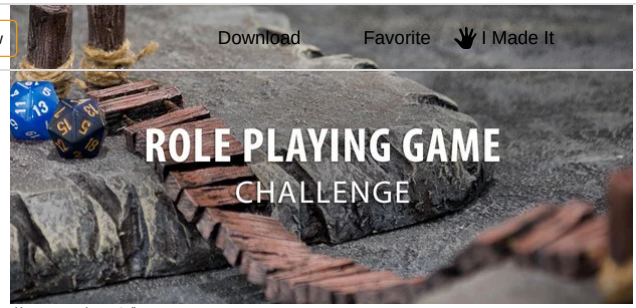
(/Tape-Measure-Yagi-Antenna-With-3D-Printed-Couplers/)

Tape Measure Yagi Antenna With 3D Printed Couplers (/Tape-Measure-Yagi-Antenna-With-3D-Printed-Couplers/) by

♥ 58 👁 7.7K



(/contest/STEM/)



(/contest/RPG/)



Add Tip



Ask Question



Post Comment

We have a **be nice** policy.
Please be positive and constructive.

Add Images

Post

6 Comments



(/member/VoicelessProximity1701/)

VoicelessProximity1701 (/member/VoicelessProximity1701/)

Question 5 months ago

Answer

▲ Upvote

Hi,
Please could you adapt this to single-colour LEDs, as I am quite new and unsure of how to wire things up?



(/member/william.siffer/)

william.siffer (/member/william.siffer/)

Question 10 months ago

Answer

▲ Upvote

I have two problems, I get a 404 error whenever I get webhooks, and I also never get the message that says that it is starting the setColor.py (<http://setcolor.py>) script. I went through you guide a couple times but just cannot seem to find my error... Any ideas?

EDIT: When changed to a Get hook, i get the 200 code, but it still doesn't seem to be running the setColor.py (<http://setcolor.py>) script

1 answer ▼

(/member/william.siffer/)

william.siffer (/member/william.siffer/)

10 months ago

Reply

▲ Upvote

Hi there! I am looking to modify this project to combine it with the hackster.io (<http://hackster.io>) robotic bartender.. I think google home control would be crazy cool. How does your index.js work? Does it just output a string to setColor.py (<http://setcolor.py>)? I assume i could output to any python script?

1 reply ▼

(/member/DIY+Hacks+and+How+Tos/)








DIY Hacks and How Tos

(/member/DIY+Hacks+and+How+Tos/) 2 years ago

This would also be great for setting up Christmas decoration lighting.

Post Comment

Categories

-  Circuits
(/circuits/)
-  Living
(/living/)
-  Workshop
(/workshop/)
-  Outside
(/outside/)
-  Craft
(/craft/)
-  Teachers
(/teachers/)
-  Cooking
(/cooking/)
- Find Us

About Us

- Who We Are
(/about/)
- Why Publish?
(/create/)
- Jobs (/jobs/)

Resources

- Sitemap (/sitemap/)
- Help (/how-to-write-a-great-instructable/)
- Contact (/contact/)

(<https://www.instagram.com/instructables/>) (<https://www.pinterest.com/instructables/>) (<https://www.facebook.com/instructables/>) (<https://www.twitter.com/instructables/>)