

GPIO Web Server for Raspberry Pi

July 15, 2023

Table of Contents

Introduction.....	2
Useful resources.....	3
GPIO Wiring.....	4
Initial setup of Raspberry Pi.....	5
Find the IP address of your Pi.....	6
Install Node JS.....	7
Overview.....	7
Installing node.js.....	8
Installing the onoff Module.....	10
Installing socket.io.....	11
socket.io.js client side javascript file.....	11
Allow web server to run on a port below 1024.....	11
RaspberryPi Zero W considerations.....	12
Retrieve and running the webserver code.....	13
Troubleshooting.....	14
Auto Start webserver at boot.....	15

Introduction

This demo project shows you how to create a web server on your Raspberry Pi so you can control your GPIO pins on your Raspberry Pi using a web browser on your computer or mobile device. Multiple users can access the webpage at the same time and updated GPIO status is push to all clients as it occurs.

This demo project assume you have a basic knowledge of the Raspberry Pi and are well familiar with the command line, installing the Raspberry Pi OS on a Raspberry Pi, using GPIOs on the Raspberry Pi. So we will not be covering those topics in this demo project. If you are unfamiliar with these basic concepts, refer to the literally thousands of tutorials and videos that others have made on getting started with the Raspberry Pi.

For this demo, we will be using the following hardware and software.

- Raspberry Pi OS 2020-12-02-raspbian-buster-armhf-full.zip
- Node.JS v12.20.1 LTS
- A Raspberry Pi 4 Model B with 4GB of memory and a 16GB or larger SD card
- A HDMI display, keyboard, and mouse is advisable at least for the initial setup.
- LEDs, resistors, breadboard, and interconnecting cables to display whether your web browser is actually controlling the GPIO pins on the Pi.
- Either a WiFi or LAN network connection to your Pi

Other version of software and hardware will probably work however you may have to make some adjustments to this procedure. In particular, I had issues with Node.js v14.15.4 and had to revert back to v12.20.1 to resolve those issue. If you have issues, I found that <https://stackoverflow.com> to be a valuable resource in resolving issues.

This project is a compilation of many google searches to figure out how to work through all the various little issues. I started with virtually no knowledge of html, javascript, css, and node.js and there was much trial and error before I came upon a method that work. I started off with an nginx webserver, then switched to an Apache2 webserver with PHP and WordPress, before I settled on node.js as my webserver. Node.js solved many of the issues I was having particular with regard to the use of websockets.

There are literally thousands of ways that you could implement this project. There are packages such a JQuery, Express, React, Angular, PHP, and Wordpress that can aid in web development. However, I found that just sticking to plain vanilla html, css, javascript, and node.js was the easiest to learn and implement.

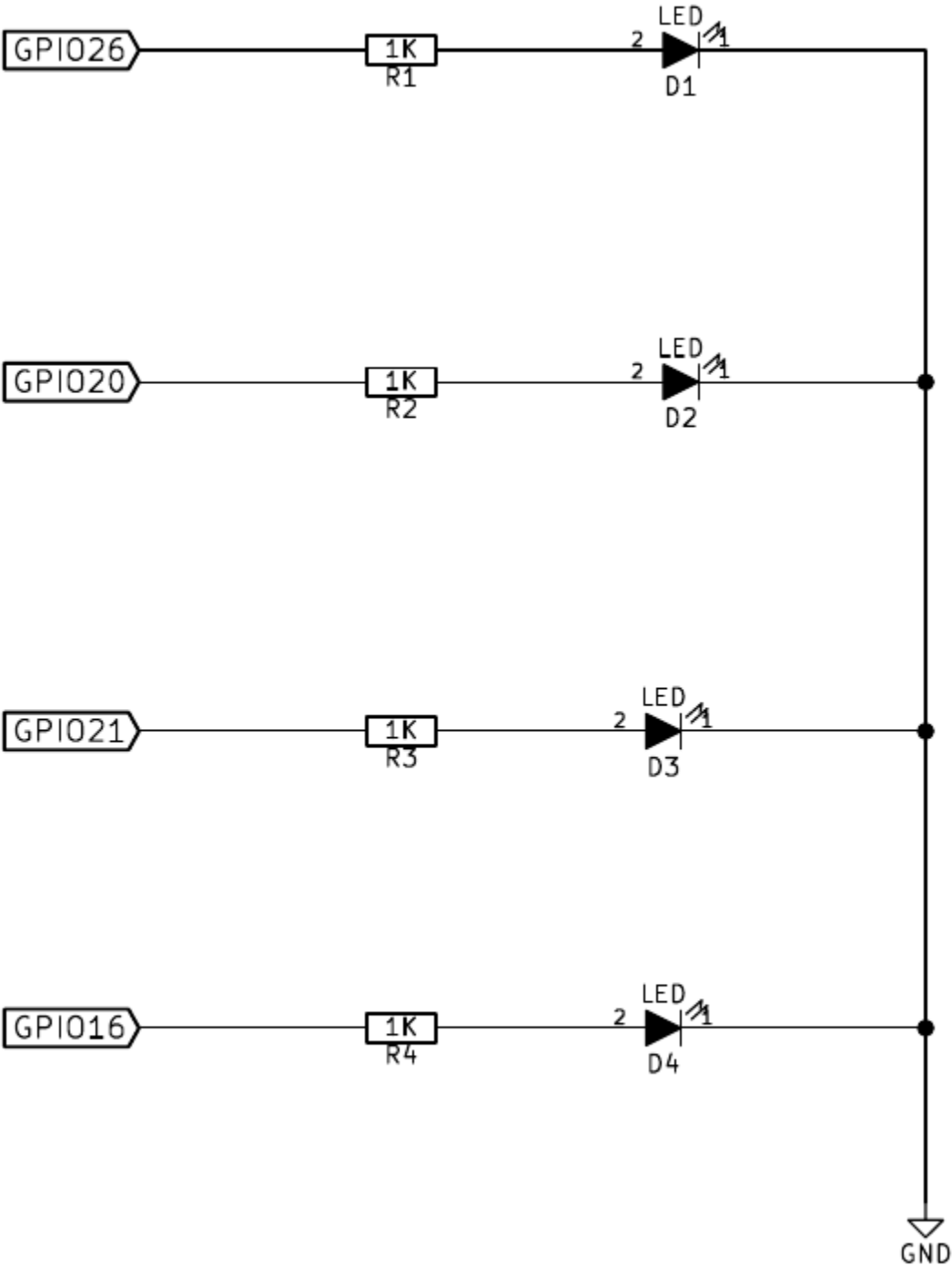
Useful resources

<https://nodejs.org/en/>
<https://www.w3schools.com/nodejs/default.asp>
<https://socket.io/docs/#What-Socket-IO-is-not>
<https://socket.io/get-started/chat>
<https://stackoverflow.com>

Videos from Traversy Media where particularly useful when learning web development. Here are just some of the videos I found useful:

HTML: <https://www.youtube.com/watch?v=UB1O30fR-EE>
JavaScript: <https://www.youtube.com/watch?v=hdI2bqOjy3c>
Node.js: <https://www.youtube.com/watch?v=fBNz5xF-Kx4>
JavaScript DOM: <https://www.youtube.com/watch?v=0ik6X4DJKCc>
CSS: <https://www.youtube.com/watch?v=yfoY53QXEnI>
AJAX: <https://www.youtube.com/watch?v=82hmvUYY6QA>

GPIO Wiring

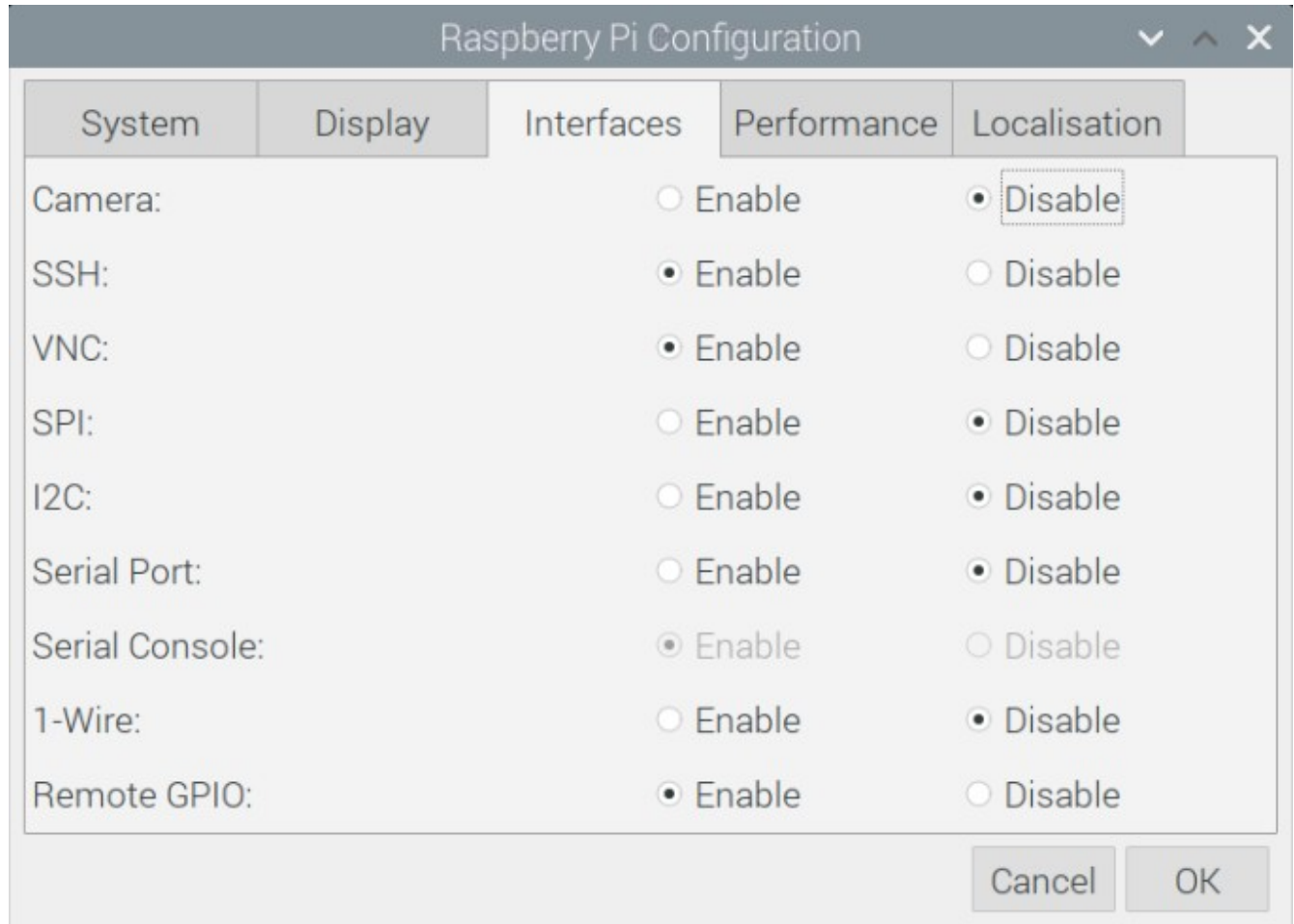


Initial setup of Raspberry Pi

Install a clean version Raspberry Pi OS on your Pi. I recommend you use the desktop with recommended software version. Refer to the introduction for the version I used. You can download the latest version of software at:

<https://www.raspberrypi.org/software/operating-systems/>

Once you go through the initial installation and update the Pi, click the menu button and go to "Preferences/Raspberry Pi Configuration" and enable "SSH" and "Remote GPIO". If you wish to have a remote Desktop to the Pi using RealVNC software, enable "VNC" too.



The screenshot shows the "Raspberry Pi Configuration" window with the "Interfaces" tab selected. The window has a title bar with a close button (X) and a maximize button (up arrow). Below the title bar are five tabs: "System", "Display", "Interfaces", "Performance", and "Localisation". The "Interfaces" tab is active, showing a list of interfaces with "Enable" and "Disable" radio buttons. The "Camera" interface is currently set to "Disable", which is highlighted with a dashed box. The "SSH" and "Remote GPIO" interfaces are set to "Enable".

Interface	Enable	Disable
Camera:	<input type="radio"/>	<input checked="" type="radio"/>
SSH:	<input checked="" type="radio"/>	<input type="radio"/>
VNC:	<input checked="" type="radio"/>	<input type="radio"/>
SPI:	<input type="radio"/>	<input checked="" type="radio"/>
I2C:	<input type="radio"/>	<input checked="" type="radio"/>
Serial Port:	<input type="radio"/>	<input checked="" type="radio"/>
Serial Console:	<input checked="" type="radio"/>	<input type="radio"/>
1-Wire:	<input type="radio"/>	<input checked="" type="radio"/>
Remote GPIO:	<input checked="" type="radio"/>	<input type="radio"/>

At the bottom right of the window are "Cancel" and "OK" buttons.

Find the IP address of your Pi

Open a terminal on the Pi and execute "ifconfig" to get the IP address of your pi. You will use that later to SSH into the Pi and to access the web server. If you wish, you can assign a static IP to you Pi however I usually prefer to give the Pi a reserved DHCP in my router's DHCP server.

ifconfig

```
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.18.105 netmask 255.255.255.0 broadcast 192.168.18.255
    RX packets 5779 bytes 523040 (510.7 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 7188 bytes 4134507 (3.9 MiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

In this case, we are connected by the LAN port and our IP address is 192.168.18.105. So we will use that IP address for the rest on this demo. So where you see 192.168.18.105, you want to replace it with the IP address of your Pi.

Install Node JS

Overview

Node.js provides a method to run Javascript on your web server. Normally when people refer to javascript, they are talking about javascript that runs in the client's web browser. We will use node.js to implement our web server using javascript that runs on the server side instead of the client side. This has an advantage in that we don't have to learn a different programming language on the server side code. Both the client and server side uses javascript programming.

There is really no installation required for node other than unzipping the archive and placing it in your /usr/local folder. Your webserver files can be located in any other folder and usually in your home folder.

Node.js also supports websockets natively which is a big plus for it over an Apache2/PHP implementation. Do not install node.js via the "apt" package manager. This project will not work if you don't install node manually by downloading from node.js.org instead. Following the instructions in the video exactly except for:

- 1) Use the 32bit Raspberry Pi OS Desktop WITHOUT recommend software. The full desktop version of OS now includes a version of node preinstalled that will not work with this project. When I originally made this video, the full desktop version did not include node preinstalled.
- 2) Use a newer version of node. See the video descriptions for which versions of node worked for me. As of August 2022, Node JS v16.16.0 with npm v8.11.0 works fine for me. For the most part, the newest version of node work although occasionally you may need to revert to an earlier package if bugs are introduced.

We will install Socket.io as that provides an easy interface to web sockets. Generally you want to include a reference socket.io.js in your html file. When your web client first connects to the pi, socket.io establishes a long-polling connection for the client, then socket.io tries to upgrade to better transports on the client side, like WebSockets.

Installing node.js

Blue text refers to commands that you execute in a terminal on the pi and green text is the response you get back from the pi.

To install node.js on a raspberry pi, open a terminal and execute:

```
sudo apt update
sudo apt upgrade
sudo apt dist-upgrade
```

Next we need to determine what ARM processor the Pi is using. Execute:

```
uname -m
armv7l
```

So we are using an ARM7 processor. Go to <https://nodejs.org/en/download/> and download ARMv7 version of Node.js. In this case, we will be installing node-v12.20.1-linux-armv7l. If there is a newer version of node, you can try it or you can go to <https://nodejs.org/en/download/releases/> and get the same version as used in this demo. As of August 1, 2021, the latest version of Node (v14.15.4) does work on the Pi 4. Node v12.x still the works too for the Raspberry Pi 4B.

Note: Node JS will also work on 64 bit versions of Raspberry Pi OS however you will need to use the ARMv8 version of Node JS instead.

If you are using a Raspberry Pi Zero, see the "RaspberryPi Zero W considerations" section of this document. I have successfully run this project on a Raspberry Pi Zero W however you need to install a different version of NodeJS, use an older version of Raspberry Pi OS, and install a couple of extra packages.

```
cd ~/Downloads
wget https://nodejs.org/dist/latest-v12.x/node-v12.20.1-linux-armv7l.tar.xz
(You will need to modify the above line for what ever version of node you are using).
```

```
ls -l
node-v12.20.1-linux-armv7l.tar.xz
```

Next extract the files from the archive:

```
tar -xvf node-v12.20.1-linux-armv7l.tar.xz
```

```
ls -l
total 13556
drwxr-xr-x 6 pi pi      4096 Jul 22 10:00 node-v14.15.4-linux-armv7l
-rw-r--r-- 1 pi pi 12505784 Jul 22 10:01 node-v14.15.4-linux-armv7l.tar.xz
```

Then open the directory with the node files and confirm they have been extracted.

```
cd node-v12.20.1-linux-armv7l
ls -l
total 180
drwxr-xr-x 2 pi pi   4096 Jul 22 10:00 bin
-rw-r--r-- 1 pi pi  53543 Jul 22 10:00 CHANGELOG.md
```



```
drwxr-xr-x 3 pi pi 4096 Jul 22 10:00 include
drwxr-xr-x 3 pi pi 4096 Jul 22 10:00 lib
-rw-r--r-- 1 pi pi 80055 Jul 22 10:00 LICENSE
-rw-r--r-- 1 pi pi 27288 Jul 22 10:00 README.md
drwxr-xr-x 5 pi pi 4096 Jul 22 10:00 share
```

Copy files to your /usr/local directory

```
sudo cp -R * /usr/local/
```

Check if node and npm have been installed correctly.

```
node -v
```

```
v12.20.1
```

```
npm -v
```

```
6.14.10
```

IMPORTANT: When you check the versions of node and npm, if they are not the same versions as you just installed, reboot the pi to activate the new version before installing the npm modules. Newer versions of the full desktop version of Raspberry Pi OS may already have an older version of Node installed. If you install npm packages with that older version, things may not work correctly.

Next we need to install a few modules using the node package manager (npm). But first we need to create a directory for our node files and initialize the npm. That will create a json file that will be needed when we install packages.

```
mkdir ~/webserver/
```

```
cd ~/webserver
```

```
npm init
```

For the most part, you can use the defaults when asked a question by npm init. Add a description for your project. It is okay to leave the repository blank.

Installing the onoff Module

To interface with the GPIO on the Raspberry Pi using Node.js, we will use a Module called "onoff".

Install the onoff module using npm:

```
npm install onoff
```

```
> epoll@4.0.0 install /home/pi/node/gpio/node_modules/epoll
```

```
> node-gyp rebuild
```

```
make: Entering directory '/home/pi/node/gpio/node_modules/epoll/build'
```

```
  CXX(target) Release/obj.target/epoll/src/epoll.o
```

```
  SOLINK_MODULE(target) Release/obj.target/epoll.node
```

```
  COPY Release/epoll.node
```

```
make: Leaving directory '/home/pi/node/gpio/node_modules/epoll/build'
```

```
npm notice created a lockfile as package-lock.json. You should commit this file.
```

```
npm WARN gpio@1.0.0 No repository field.
```

```
+ onoff@6.0.1
```

```
added 6 packages from 12 contributors and audited 6 packages in 7.689s
```

```
found 0 vulnerabilities
```

Now onoff should be installed and we can interact with the GPIO of the Raspberry Pi.

Installing socket.io

To install socket.io, open a terminal on the Raspberry Pi and execute:

```
npm install socket.io --save
+ socket.io@3.0.5
added 23 packages from 74 contributors and audited 29 packages in 3.547s
found 0 vulnerabilities
```

socket.io.js client side javascript file

In your html, include a reference to:

```
<script src="/socket.io/socket.io.js"></script>
```

While `"/socket.io/socket.io.js"` looks like an invalid url, it works because you wrap your HTTP server in Socket.IO and it intercepts requests for `/socket.io/socket.io.js` and sends the appropriate response automatically. If you want to inspect socket.io.js, you can download it to your current directory (assuming your web server is running on port 80) by executing the following:

```
wget 192.168.18.105:80/socket.io/socket.io.js
```

Allow web server to run on a port below 1024

If you want to run the node web server on a port lower than 1024 without running node as root (generally a bad idea to run something as root), you need to run following from a terminal on the pi to give node access to the lower ports without running it as root. You only need to execute this one.

```
sudo apt update
sudo apt install libcap2-bin
sudo setcap cap_net_bind_service=+ep /usr/local/bin/node
```

You can later check if a program has extra privileges by executing:

```
getcap /usr/local/bin/node
/usr/local/bin/node = cap_net_bind_service+ep
```

To see all files in a directory and subdirectories that have elevated privileges, execute:

```
getcap -r /usr/
/usr/local/bin/node = cap_net_bind_service+ep
```

RaspberryPi Zero W considerations

If you are using a RaspberryPi zero-W, it will report:

```
uname -m  
armv6l
```

Support for armv6l has been dropped in later version of nodes. If you wish to use a Raspberry Pi zero, you will have to use node v11.x instead. You can download this version at:

<https://nodejs.org/dist/latest-v11.x/>

However, this version of node hasn't been supported for years -- so with the latest versions of Raspberry Pi OS, node no longer works. You could revert to the same versions of raspberry Pi OS that I specified in the video (Pi OS 2020-12-02-raspbian-buster-armhf-full) and then it will work. But it will not have any security updates so use at your own risk.

Before you install npm packages, make sure the version of node that you installed is running on the pi. If not, reboot the pi. Full desktop versions of the Pi OS may have an older version of node installed.

When you run

```
npm install socket.io --save
```

You will likely get errors like:

```
npm WARN ws@7.4.6 requires a peer of bufferutil@^4.0.1 but none is installed. You must install peer dependencies yourself.  
npm WARN ws@7.4.6 requires a peer of utf-8-validate@^5.0.2 but none is installed. You must install peer dependencies yourself.
```

If you do, run the following commands:

```
npm install --save-optional utf-8-validate  
npm install --save-optional bufferutil  
npm install socket.io --save
```

Otherwise, the setup is exactly the same for a Raspberry Pi Zero W and this example program will run on it too. I have successfully run this project on the Zero W using the following versions

RaspberryPi OS: 2021-05-07-raspbian-buster-armhf-lite

Node.JS: node-v11.15.0-linux-armv6l

If you are using the headless lite version of RaspberryPi OS like I did on the Zero W, you will need to do everything from the command line which requires a little more skill.

Retrieve and running the webserver code

Open a terminal or ssh connection on your Pi.

```
cd ~/Downloads
ls -l
total 12080
drwxr-xr-x 6 pi pi      4096 Jan  4 05:27 node-v12.20.1-linux-armv7l
-rw-r--r-- 1 pi pi 12362220 Jan  4 05:27 node-v12.20.1-linux-armv7l.tar.xz

rm -r node-v12.20.1-linux-armv7l
wget https://github.com/StevesRandomProjects/PiGpioWebServer/archive/main.zip
ls -l
total 12088
-rw-r--r-- 1 pi pi      8314 Jan 17 18:28 main.zip
-rw-r--r-- 1 pi pi 12362220 Jan  4 05:27 node-v12.20.1-linux-armv7l.tar.xz

unzip main.zip
ls -l
total 12092
-rw-r--r-- 1 pi pi      8314 Jan 17 18:28 main.zip
-rw-r--r-- 1 pi pi 12362220 Jan  4 05:27 node-v12.20.1-linux-armv7l.tar.xz
drwxr-xr-x 3 pi pi      4096 Jan 17 18:03 PiGpioWebServer-main

cd PiGpioWebServer-main
ls -l
total 20
-rw-r--r-- 1 pi pi 1077 Jan 17 18:03 LICENSE
drwxr-xr-x 4 pi pi 4096 Jan 17 18:03 public
-rw-r--r-- 1 pi pi  71 Jan 17 18:03 README.md
-rw-r--r-- 1 pi pi 6943 Jan 17 18:03 webserver.js

mkdir ~/webserver/gpio
cp -R * ~/webserver/gpio/
cd ~/webserver/gpio
```

And finally, this command will start our webserver
`node webserver.js`

You webserver should now be running. Open a webserver and go to the ipaddress of your Pi. In my case, that is:
<http://192.168.18.105/>

To stop the webserver, press Ctrl-C in the terminal window.

Troubleshooting

- 1) Be sure to start with a clean install of Raspberry Pi OS. Many people have had issues due to conflicts from other services that they have installed on their Pi. In particular, if you have another web server running on port 80, you need to use a different port for Node JS. Two services cannot use the same port.
- 2) The newer full versions of Raspberry Pi OS may already have an older version of node JS installed. So when you check the version after installing node, if it is different than the version you just installed, reboot the Pi to activate the new version. Then and only then, install the npm packages.
- 3) This demo project does not read the current state of the GPIO pins. So if you modify the GPIO from another program, it will not show up on the page. If you want to integrate the web service with another program such as Python, see my follow up video <https://youtu.be/YvsOUkTUxsg>

Auto Start webserver at boot

Sparkfun has excellent tutorial on auto starting a program. SystemD is one of the methods they discuss.
<https://learn.sparkfun.com/tutorials/how-to-run-a-raspberry-pi-program-on-startup/all>

We will use SystemD in this case as it has several advantages over other methods.

- 1) You can have system D automatically restart the program if it crashes.
- 2) You can specify to start the program only after certain processes, such as the network adapter, are available.
- 3) You can use systemD to check the status of the process and it will give you some error information if the process fails.

Create the .service file

From what I have read, you can put your .service file in one of the following directories:

- /usr/lib/systemd/system -- is for files installed by the system. So generally you don't want to use this directory for user .service files.
- /etc/systemd/system -- Most sources say to use this directory for services you create
- /lib/systemd/system -- some websites also say you can use this directory

```
sudo nano /etc/systemd/system/webserver.service
```

Then add the following lines to it

```
[Unit]
```

```
Description=webserver service
```

```
After=network.target
```

```
[Service]
```

```
ExecStart=node /home/pi/webserver/gpio/webserver.js
```

```
WorkingDirectory=/home/pi/webserver/gpio
```

```
StandardOutput=inherit
```

```
StandardError=inherit
```

```
Restart=always
```

```
User=pi
```

```
[Install]
```

```
WantedBy=multi-user.target
```

Save the file.

```
cd /etc/systemd/system
```

```
sudo chmod 644 webserver.service (this is normally not needed)
```

```
sudo systemctl daemon-reload
```

Before you enable this service at boot, you may want to test starting and stopping it first with the following commands

```
sudo systemctl start webserver.service
sudo systemctl stop webserver.service
```

To check the status, execute

```
sudo systemctl status webserver.service
```

To enable or disable the webserver at boot, use the following commands

```
sudo systemctl enable webserver.service
sudo systemctl disable webserver.service
```

Some things to be aware of: The order in which things are started is based on their dependencies — this particular script should start fairly late in the boot process, after a network is available (see the After section in [man systemctl](#)). You can configure different dependencies and orders based on your requirements.

To test the auto restart of the webserver, first find the process ID by executing

```
sudo systemctl status webserver.service
```

Then kill the process with

```
sudo kill <PID>
```

To create a log file for the webserver, change the ExecStart line in the .service file.

```
sudo nano /etc/systemd/system/webserver.service
```

Change the line

```
ExecStart=node /home/pi/webserver/gpio/webserver.js
```

To

```
ExecStart=/bin/bash -c 'node /home/pi/webserver/gpio/webserver.js > /home/pi/webserver/gpio/webserver.log 2>&1'
```

This starts a new bash shell, runs your program, and redirects the output (stdout) to a new webserver.log text file. The 2>&1 command says that any errors (stderr) should also be redirected (written to) the same log file. Any output from node or errors will then be saved to webserver.log. Note that you might need to stop the service and program before viewing the log). Also, the log file will get replaced when the process restarts.