GPIO Web Server for Raspberry Pi

January 18, 2021

Table of Contents

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-raspios-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 switches 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.

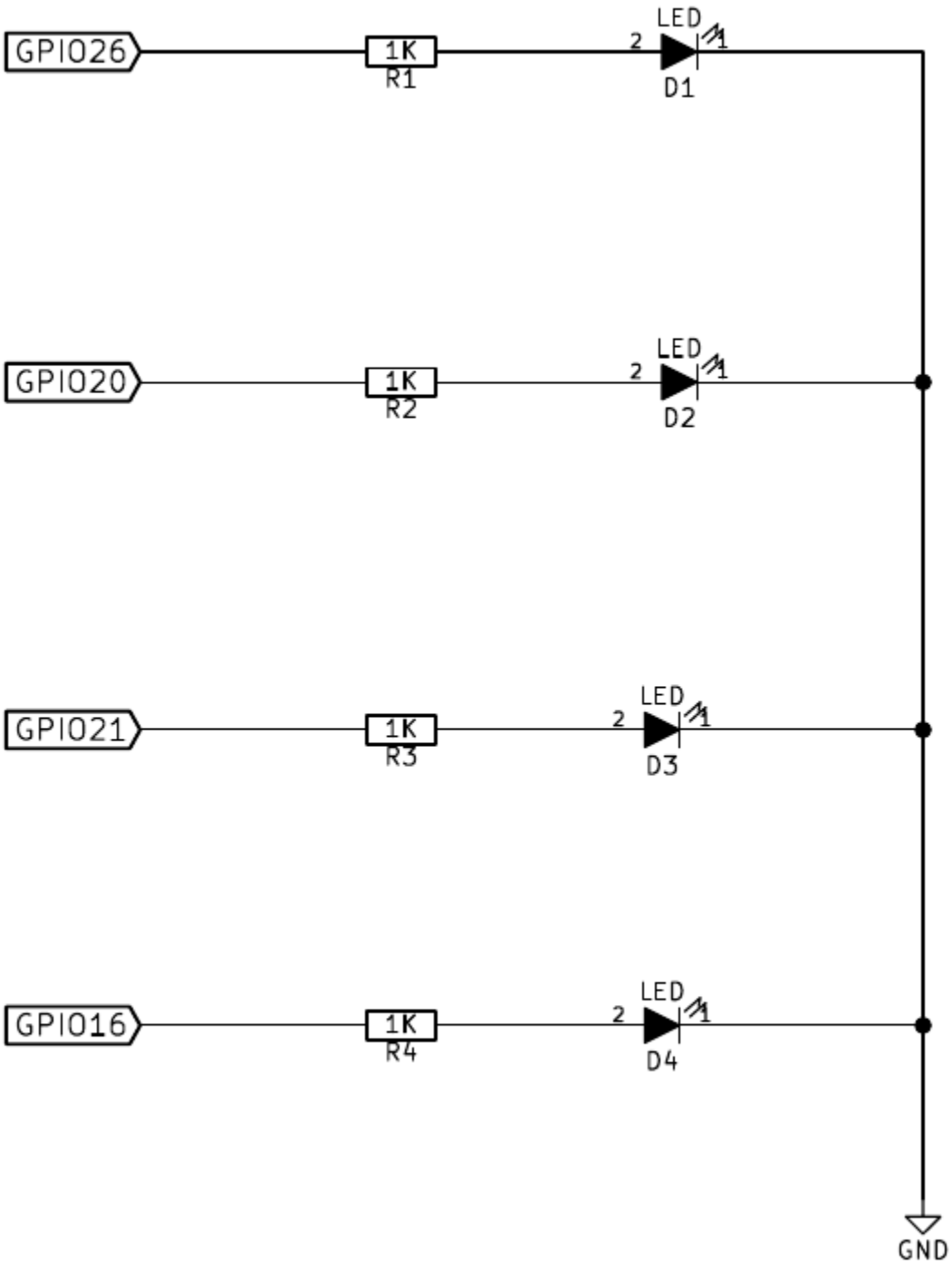
Useful resources

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=82hnvUYY6QA

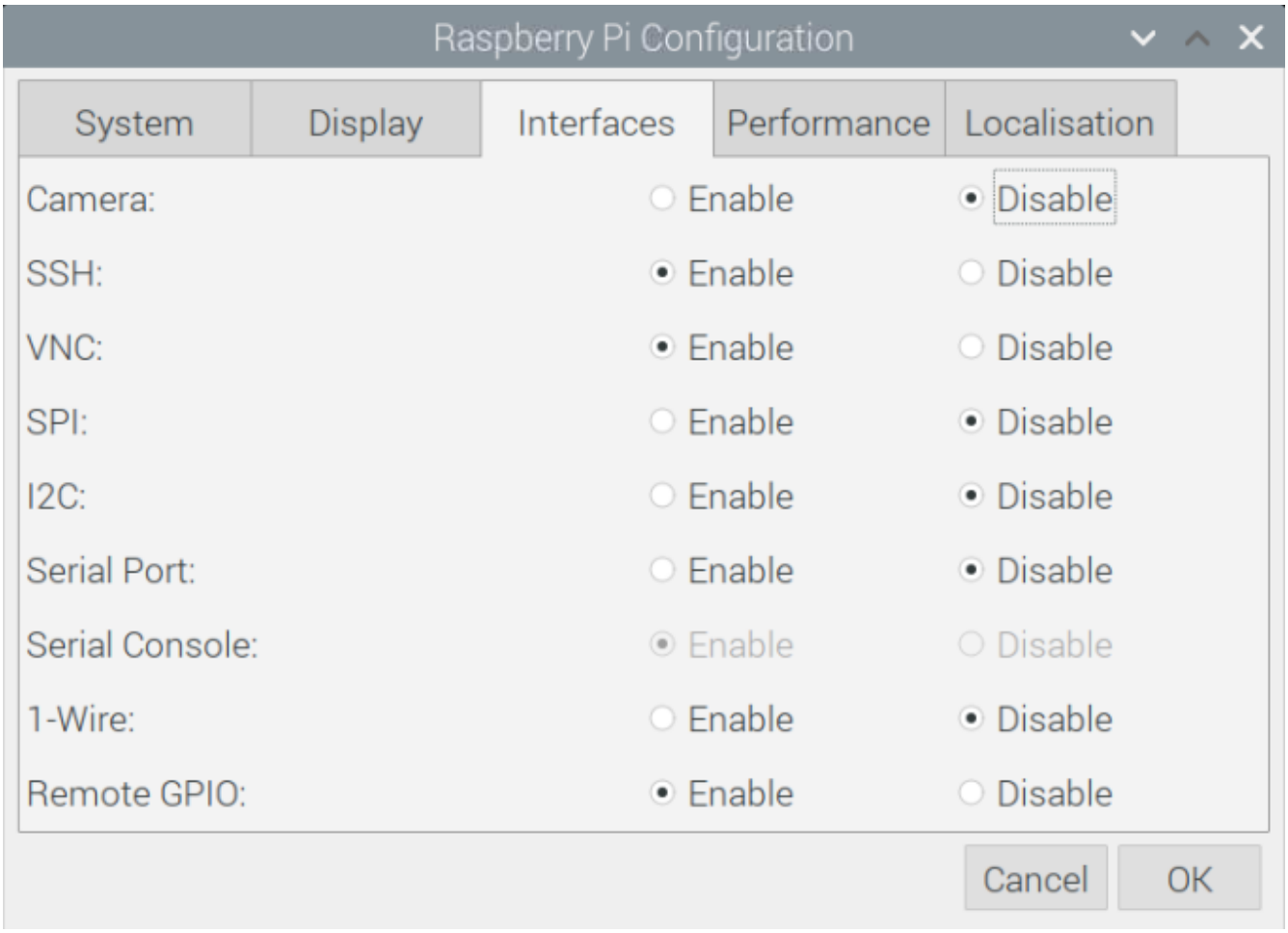
GPIO Wiring


GPIO26
1K
R1
LED
2
1
D1
GPIO20
1K
R2
LED
2
1
D2
GPIO21
1K
R3
LED
2
1
D3
GPIO16
1K
R4
LED
2
1
D4
GND

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.



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
        inet6 fe80::4746:e3b3:f904:e49a  prefixlen 64  scopeid 0x20<link>
        ether dc:a6:32:8f:50:99  txqueuelen 1000  (Ethernet)
        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 is 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. However, you do need to download the code from nodejs.org as the node.js package in the Raspberry Pi repository is usually too old to run the npm (node packet manager).

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 you 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 use it or you can go to https://nodejs.org/en/download/releases/ and get the same version as used in this demo.

cd ~/Downloads
wget https://nodejs.org/dist/latest-v12.x/node-v12.20.1-linux-armv7l.tar.xz

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
```

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 desription for you 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 down load 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 is 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
```

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 webserver.js ~/webserver/gpio/webserver.js
cp -R public ~/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 may case, that is:
http://192.168.18.105/

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