3/27/2020 

|  |
| --- |
| **When a color meets a character** | Ranjita |

|  |  |
| --- | --- |
| KEPL | RGB MESH CONTROL |

**Intelligent control LED Integrated light source**

WS2813B is an intelligent control LED light source that the control circuit and RGB chip are integrated in a package of 5050 components. Its internal includes intelligent digital port data latch and signal reshaping amplification drive circuit.

Also include a precision internal oscillator and a 12V voltage programmable constant current control part, which achieves highly consistent color effect.

**Dual-signal wires version,** signal break-point continuous transmission. Any pixel’s failure won't affect signal transfer and total emitting effect.

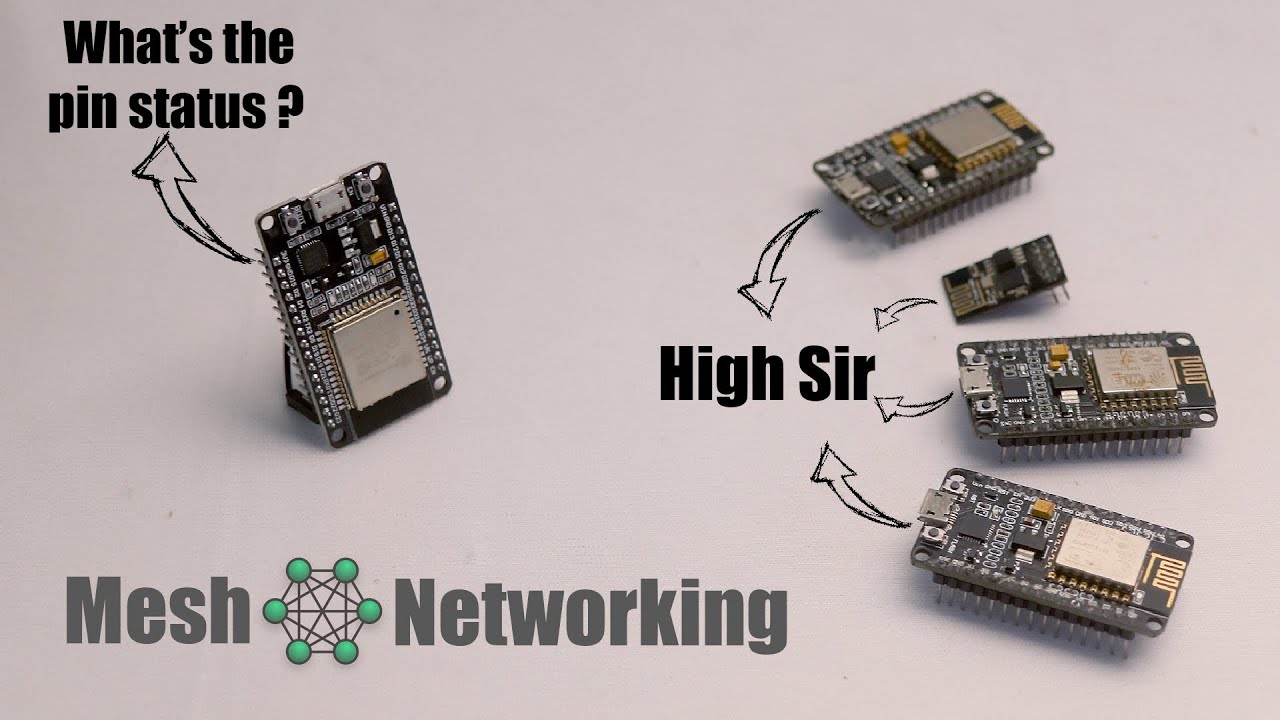
The data transfer protocol use single NZR communication mode. After the pixel power-on reset, the DIN port receive data from controller, the first pixel collect initial 24bit data then sent to the internal data latch, the other data which reshaping by the internal signal reshaping amplification circuit sent to the next cascade pixel through the DO port. After transmission for each pixel, the signal to reduce 24bit. Every pixel adopts auto-reshaping transmit technology, making the pixel cascade numbers are not limited to the signal transmission, only relate to the speed of signal transmission. The BIN receives the data signal, and then compare the data with the DIN side after phagocytosis of 24bit data, if DIN do NOT receive the signal, then switching to BIN for receiving the input signal, which ensure that any the IC's damage does not affect the signal cascade transmission and make the BIN in state of receiving signal until restart after power-off.

Refresh Frequency updates to **2 KHz**, Low Frame Frequency and no Flicker appear in HD Video Camera. RESET time>**280μs**, it won’t cause wrong reset while interruption, it supports the lower frequency and inexpensive

MCU. Integrated circuit chips enable the circuit control simpler, neater and more reliable while NO extra components needed.

**Introduction to PainlessMesh**

PainlessMesh is a library that takes care of the particulars of creating a simple mesh network using esp8266 and esp32 hardware.  The goal is to allow the programmer to work with a mesh network without having to worry about how the network is structured or managed.



**True ad-hoc networking**

PainlessMesh is a true ad-hoc network, meaning that no planning, central controller, or router is required.  Any system of 1 or more nodes will self-organize into fully functional mesh.  The maximum size of the mesh is limited (we think) by the amount of memory in the heap that can be allocated to the sub-connections buffer and so should be really quite high.

**JSON based**

PainlessMesh uses JSON objects for all its messaging.  There are a couple of reasons for this.  First, it makes the code and the messages human readable and painless to understand and second, it makes it painless to integrate painlessMesh with javascript front-ends, web applications, and other apps.  Some performance is lost, but I haven’t been running into performance issues yet.  Converting to binary messaging would be fairly straight forward if someone wants to contribute.

**Wi-Fi &amp; Networking**

PainlessMesh is designed to be used with Arduino, but it does not use the Arduino WiFi libraries, as we were running into performance issues (primarily latency) with them.  Rather the networking is all done using the native esp32 and esp8266 SDK libraries, which are available through the Arduino IDE.  Hopefully though, which networking libraries are used won’t matter to most users much as you can just include painlessMesh.h, run the init() and then work the library through the API.

**PainlessMesh is not IP networking**

PainlessMesh does not create a TCP/IP network of nodes. Rather each of the nodes is uniquely identified by its 32bit chipId which is retrieved from the esp8266/esp32 using the system\_get\_chip\_id() call in the SDK.  Every node will have a unique number.  Messages can either be broadcast to all of the nodes on the mesh, or sent specifically to an individual node which is identified by its nodeId.

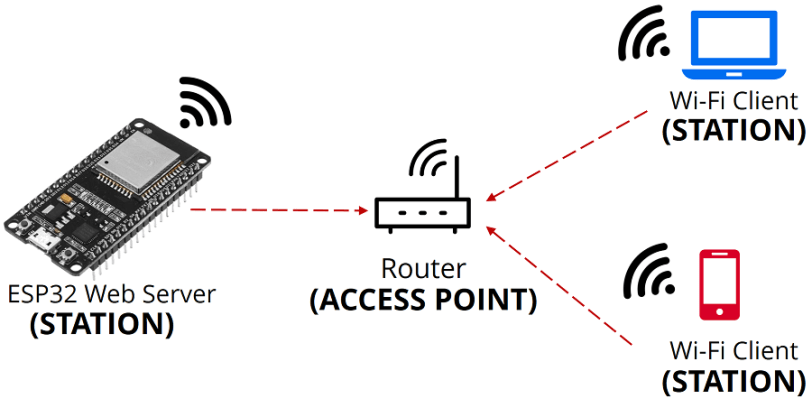
**Set an ESP32 Access Point (AP) for Web Server**

The ESP32 can act as a Wi-Fi station, as an access point, or both. In this tutorial we’ll show you how to set the ESP32 as an access point using Arduino IDE.



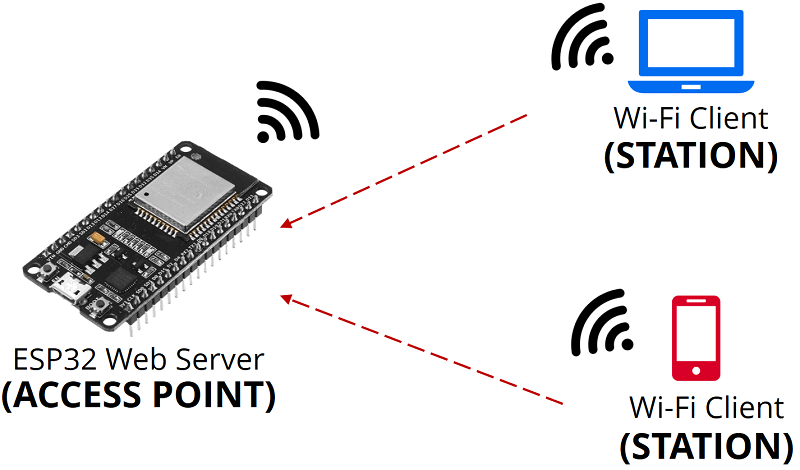
In most projects with the ESP32, we connect the ESP32 to a wireless router (see our [ESP32 web server tutorial](https://randomnerdtutorials.com/esp32-web-server-arduino-ide/)). This way we can access the ESP32 through the local network.

In this situation the router acts as an access point and the ESP32 is set as a station. In this scenario, you need to be connected to your router (local network) to control the ESP32.



But if you set the ESP32 as an access point (hotspot), you can be connected to the ESP32 using any device with Wi-Fi capabilities without the need to connect to your router.

In simple words, when you set the ESP32 as an access point you create its own Wi-Fi network and nearby Wi-Fi devices (stations) can connect to it (like your smartphone or your computer).

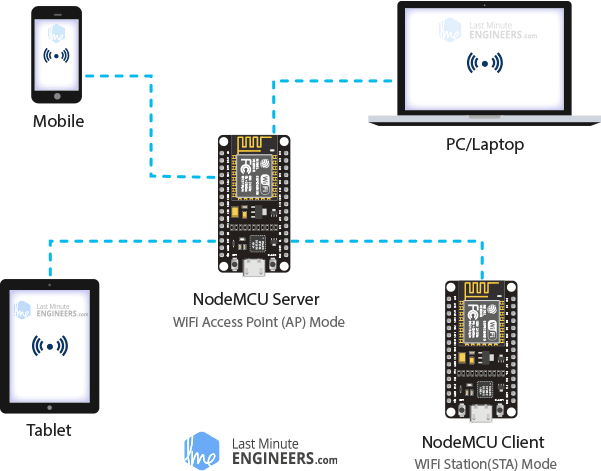


**Working:**

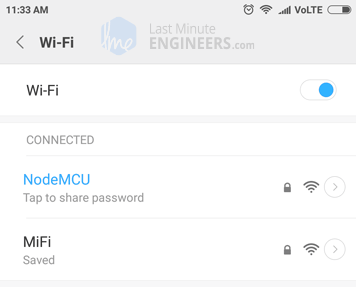
Smart lighting generally uses mesh networking, in this case we have one master bulb and remaining lights as clients. And master bulb acts as soft access point.

Web server is place which stores, processes and delivers web pages to web clients. Web client is nothing but a web browser on our laptops and smartphones. The communication between client and server takes place using a special protocol called Hypertext Transfer Protocol (HTTP).

The ESP8266 that creates its own Wi-Fi network and acts as a hub (Just like Wi-Fi router) for one or more stations is called **Access Point** (AP). Unlike Wi-Fi router, it does not have interface to a wired network. So, such mode of operation is called **Soft Access Point** (soft-AP). Also the maximum number of stations that can connect to it is limited to five.



Find any device that you can connect to a Wi-Fi network – phone, laptop, etc. And go to Network and Internet settings look for a network called **NodeMCU**. Join the network with password **12345678**.



After connecting to your NodeMCU AP network, load up a browser and point it to 192.168.1.1 The NodeMCU should serve up a web page showing ESP color picker. And send a HTTP request a desired color. According to the request the master bulb changes its color and the nodes connected to this node change accordingly.