PL9823 Web Controlled LEDs

Introduction

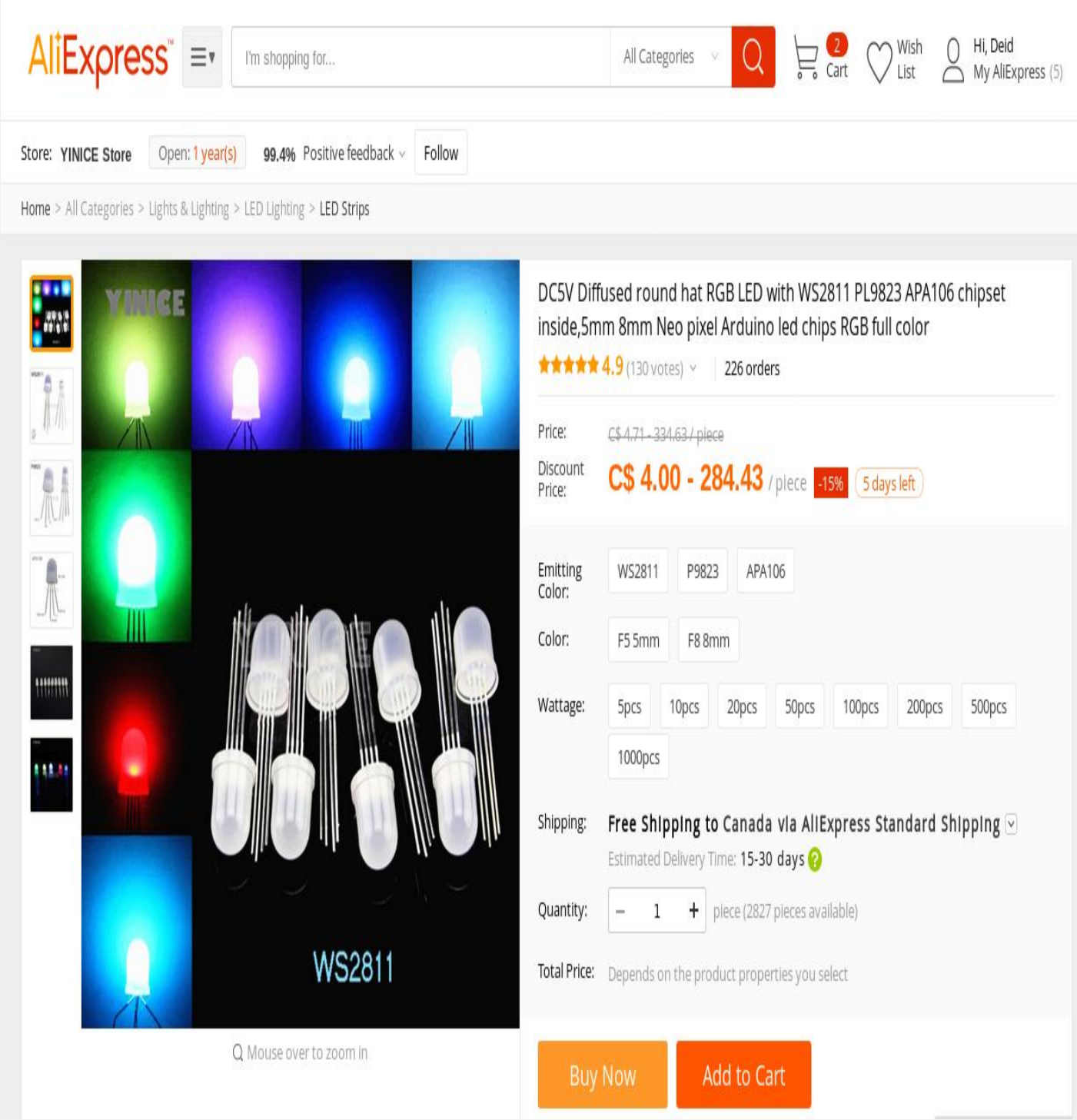
We thought we would start with something a bit whimsical.

> **Since we are early in the academic year**

- With pole
- With another pole
- With LEDs

This all started when my my wife decided that we should do something with the 60 or so telephone insulators she has been collected for the past 30+ years. Perhaps put a led in them and stick them in the garden she suggested.

So ... I went looking on Ali Express for cheap LEDs. Thinking a bunch of different coloured LEDs would be fun. And I found:



- Not only are the LEDs any colour I want, they can be any of 16 million or so colours. And they are individually

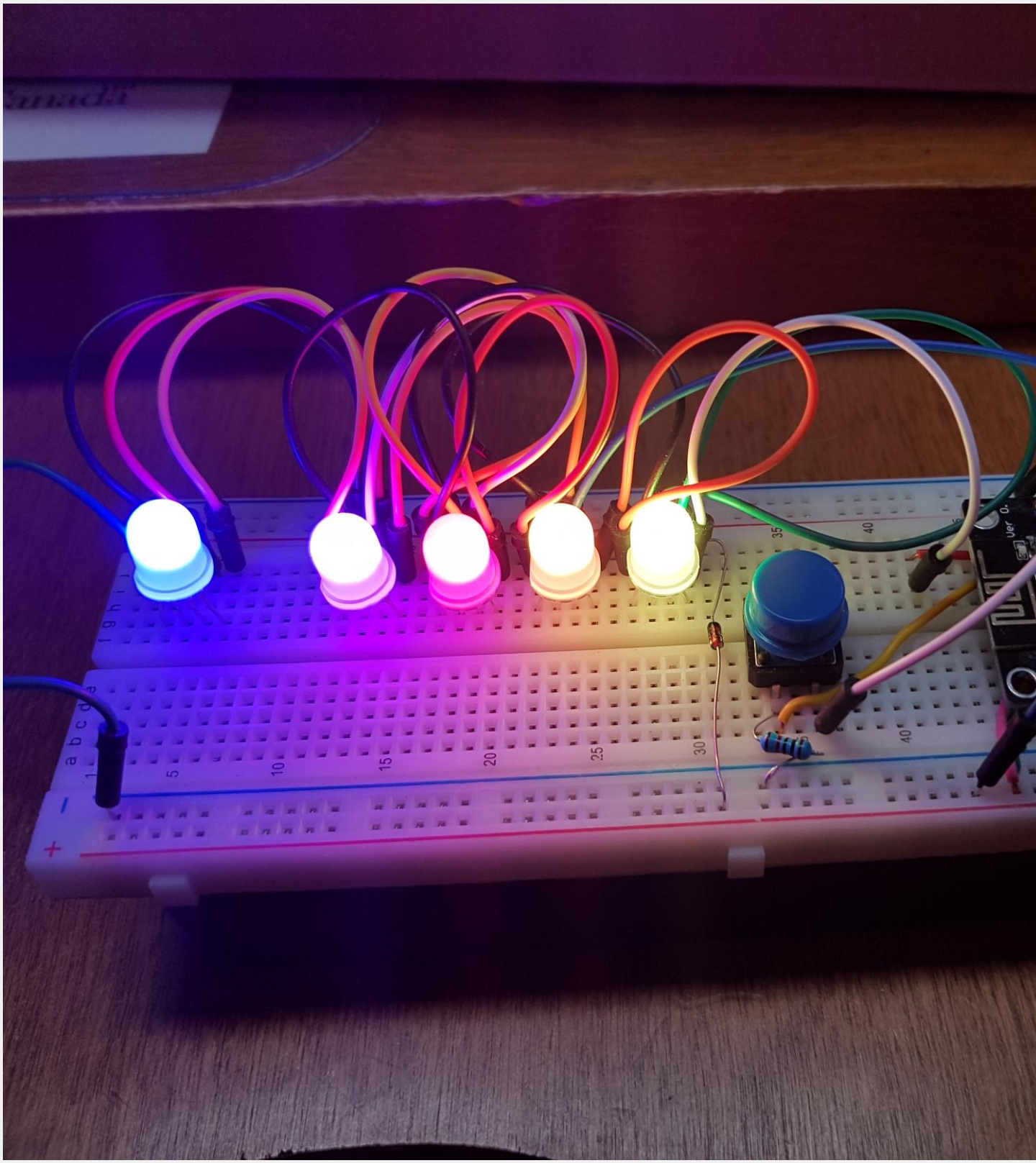
addressable. Each led in the circuit can be a different colour and the colours can change.

- So I bought 100 of them!
- Then over the next several days - with much reading of web pages and tutorials and libraries - I managed to get them to work.

Thanks to:

- http://fastled.io/
- The guy with the Swiss accent: https://www.youtube.com/watch?v=YJQG9JnDemM&t=7s
- https://hackaday.com/2017/01/20/cheating-at-5v-ws2812-control-to-use-a-3-3v-data-line/

And I built the following. The real thing is wandering about here somewhere with a battery attached to it.



- The NodeMCU ESP8266 micro-controller runs the show

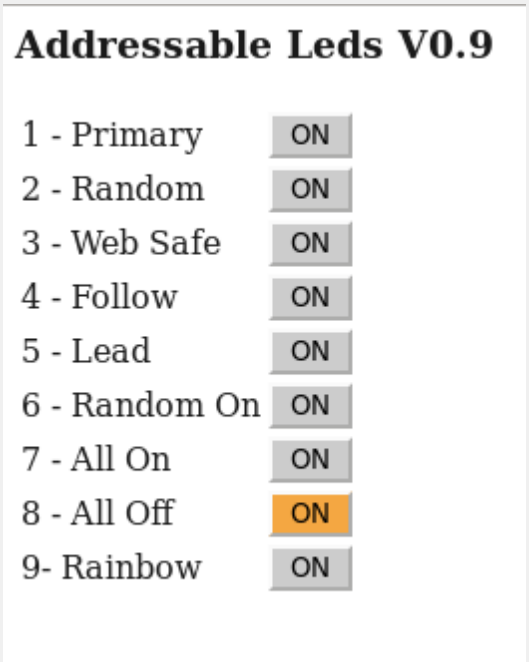
- The push button cycles through the various led programs that I wrote
- The LEDs, of course, light up
- And the prototype board holds it all together and provides electrical connections

I then spent the next month off and on writing led "programs" so that pushing the button does something.

- I currently have 8 programs that flash the lights in various ways,
- obviously, there are an infinite number of possibilities.
- If you changed the layout of the LEDs to say - a grid - you could show pictures ...

It then occurred to me that use a web page could web control the selection of light program.

- Thanks to the ESP8266s built in wifi and web server.
- So I did



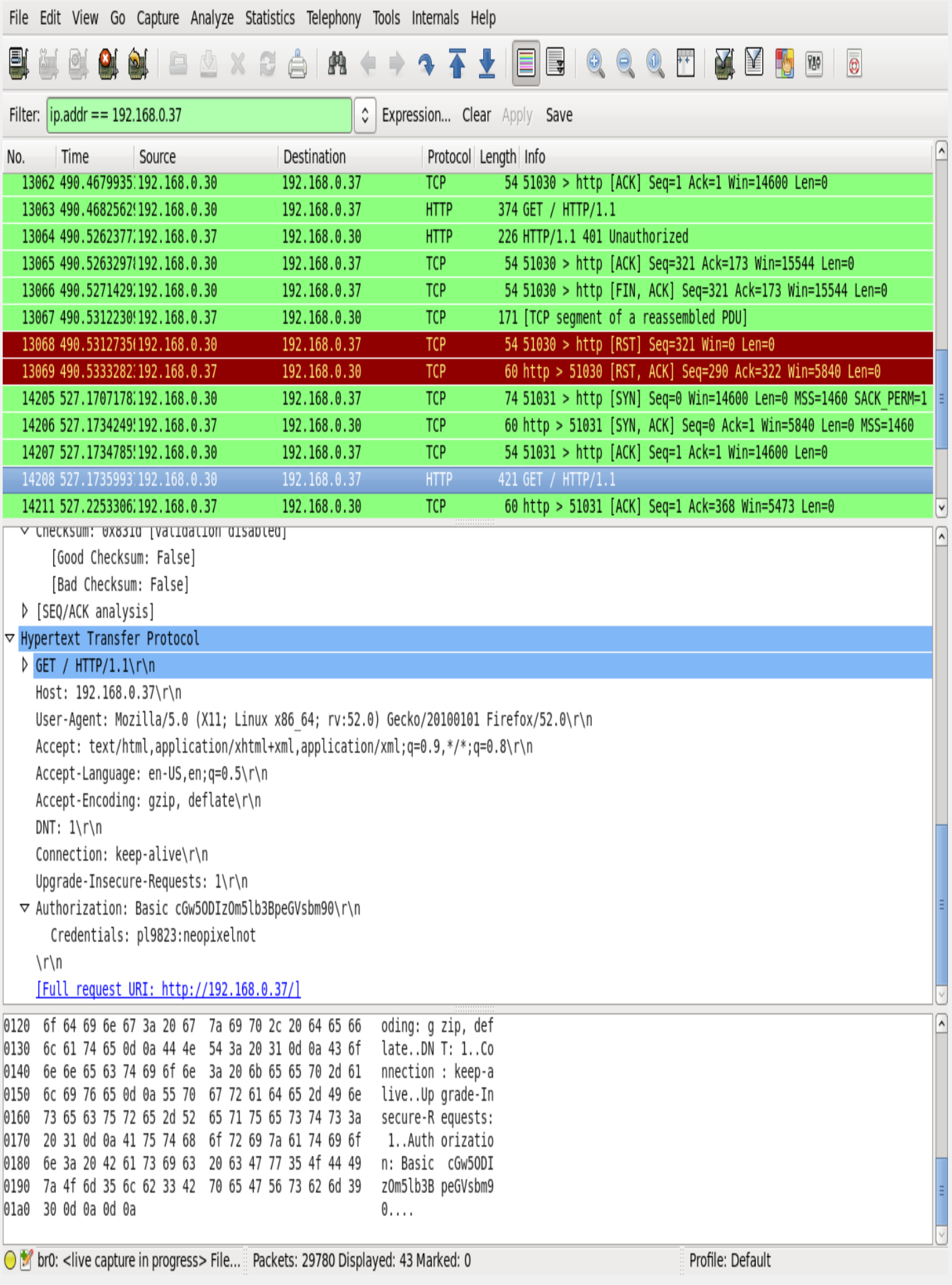
Details

Code

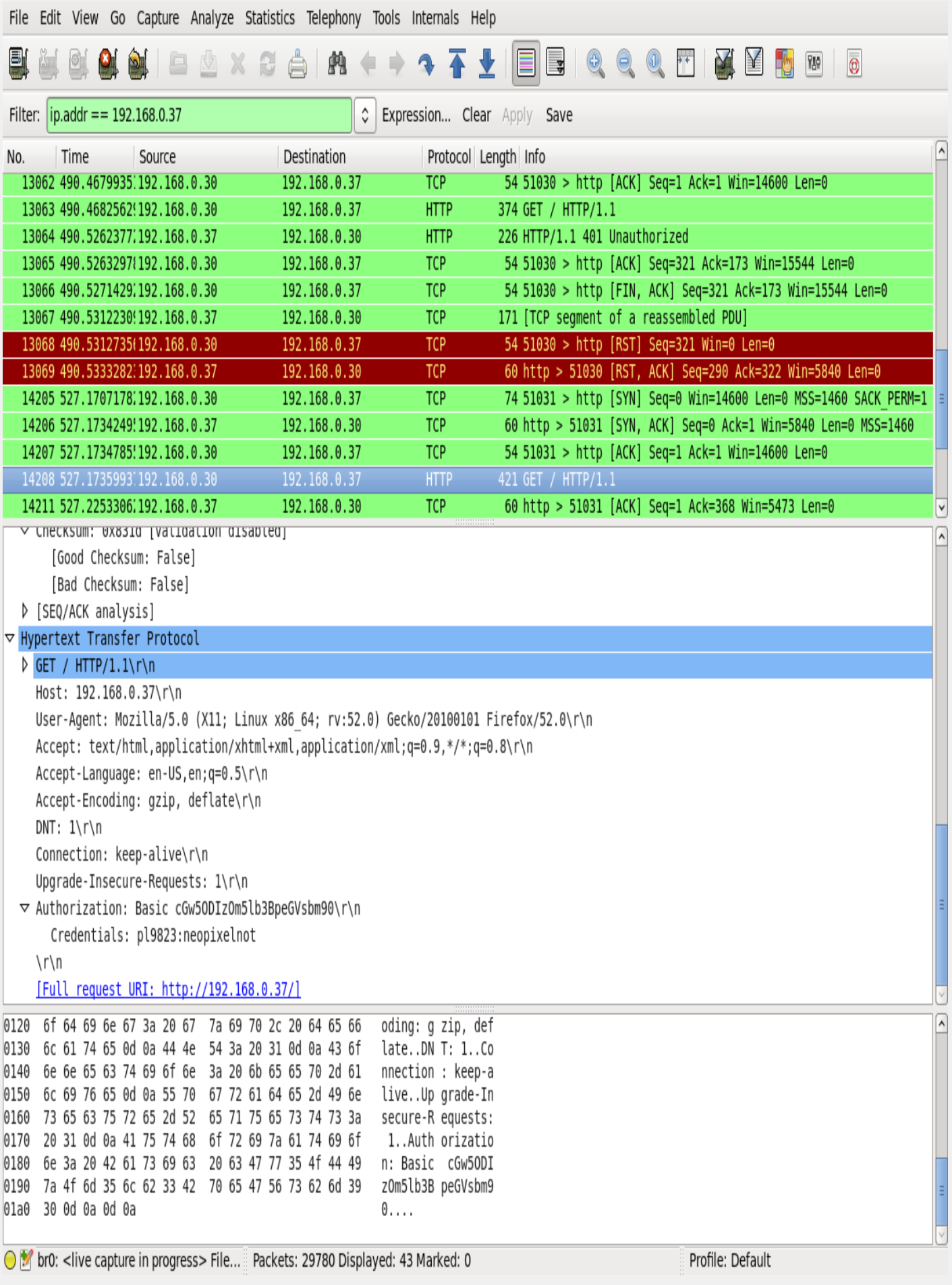
- The code was written in the Arduino IDE and compiled and uploaded to the ESP8266 with the Arduino IDE.
- The IDE is running on a Raspberry Pi.
- Click here for information on Installing the Arduino IDE on the Raspberry Pi

And there is a problem. We put a password on the request to get the web page to change the lights but look:

Anyone with network packet analyzer like wireshark can see the ID and Password!


File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter: ip.addr == 192.168.0.37
Expression... Clear Apply Save
No. Time Source Destination Protocol Length Info
13062 490.4679935 192.168.0.30 192.168.0.37 TCP 54 51030 > http [ACK] Seq=1 Ack=1 Win=14600 Len=0
13063 490.4682562 192.168.0.30 192.168.0.37 HTTP 374 GET / HTTP/1.1
13064 490.5262377 192.168.0.37 192.168.0.30 HTTP 226 HTTP/1.1 401 Unauthorized
13065 490.5263297 192.168.0.30 192.168.0.37 TCP 54 51030 > http [ACK] Seq=321 Ack=173 Win=15544 Len=0
13066 490.5271429 192.168.0.30 192.168.0.37 TCP 54 51030 > http [FIN, ACK] Seq=321 Ack=173 Win=15544 Len=0
13067 490.5312230 192.168.0.37 192.168.0.30 TCP 171 [TCP segment of a reassembled PDU]
13068 490.5312735 192.168.0.30 192.168.0.37 TCP 54 51030 > http [RST] Seq=321 Win=0 Len=0
13069 490.5333282 192.168.0.37 192.168.0.30 TCP 60 http > 51030 [RST, ACK] Seq=290 Ack=322 Win=5840 Len=0
14205 527.1707178 192.168.0.30 192.168.0.37 TCP 74 51031 > http [SYN] Seq=0 Win=14600 Len=0 MSS=1460 SACK_PERM=1
14206 527.1734249 192.168.0.37 192.168.0.30 TCP 60 http > 51031 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460
14207 527.1734785 192.168.0.30 192.168.0.37 TCP 54 51031 > http [ACK] Seq=1 Ack=1 Win=14600 Len=0
14208 527.1735993 192.168.0.30 192.168.0.37 HTTP 421 GET / HTTP/1.1
14211 527.2253306 192.168.0.37 192.168.0.30 TCP 60 http > 51031 [ACK] Seq=1 Ack=368 Win=5473 Len=0
Checksum: 0x831d [validation disabled]
[Good Checksum: False]
[Bad Checksum: False]
[SEQ/ACK analysis]
Hypertext Transfer Protocol
GET / HTTP/1.1\r\n
Host: 192.168.0.37\r\n
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:52.0) Gecko/20100101 Firefox/52.0\r\n
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n
Accept-Language: en-US,en;q=0.5\r\n
Accept-Encoding: gzip, deflate\r\n
DNT: 1\r\n
Connection: keep-alive\r\n
Upgrade-Insecure-Requests: 1\r\n
Authorization: Basic cGw5ODIzOm5lb3BpeGVsbm90\r\n
Credentials: pl9823:neopixelnot
\r\n
[Full request URI: http://192.168.0.37/]
0120 6f 64 69 6e 67 3a 20 67 7a 69 70 2c 20 64 65 66 oding: g zip, def
0130 6c 61 74 65 0d 0a 44 4e 54 3a 20 31 0d 0a 43 6f late..DN T: 1..Co
0140 6e 6e 65 63 74 69 6f 6e 3a 20 6b 65 65 70 2d 61 nnection : keep-a
0150 6c 69 76 65 0d 0a 55 70 67 72 61 64 65 2d 49 6e live..Up grade-In
0160 73 65 63 75 72 65 2d 52 65 71 75 65 73 74 73 3a secure-R equests:
0170 20 31 0d 0a 41 75 74 68 6f 72 69 7a 61 74 69 6f 1..Auth orizatio
0180 6e 3a 20 42 61 73 69 63 20 63 47 77 35 4f 44 49 n: Basic cGw5ODI
0190 7a 4f 6d 35 6c 62 33 42 70 65 47 56 73 62 6d 39 zOm5lb3B peGVsbm9
01a0 30 0d 0a 0d 0a 0....
br0: <live capture in progress> File...
Packets: 29780 Displayed: 43 Marked: 0
Profile: Default

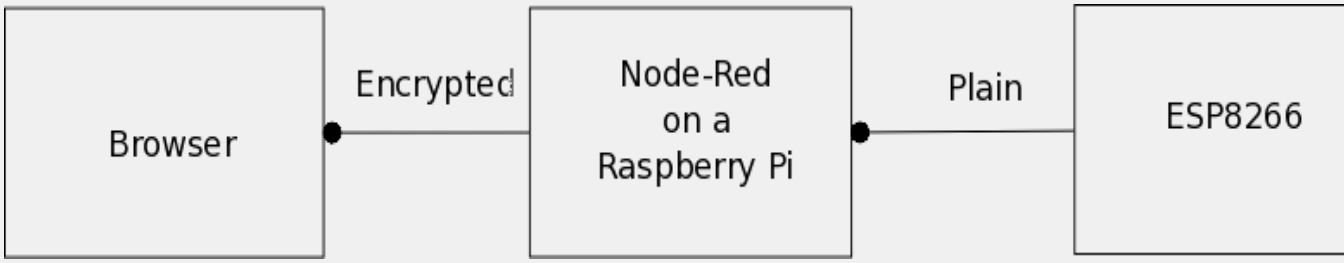
And then the web page is displayed


File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter: ip.addr == 192.168.0.37
Expression... Clear Apply Save
No. Time Source Destination Protocol Length Info
13062 490.4679935 192.168.0.30 192.168.0.37 TCP 54 51030 > http [ACK] Seq=1 Ack=1 Win=14600 Len=0
13063 490.4682562 192.168.0.30 192.168.0.37 HTTP 374 GET / HTTP/1.1
13064 490.5262377 192.168.0.37 192.168.0.30 HTTP 226 HTTP/1.1 401 Unauthorized
13065 490.5263297 192.168.0.30 192.168.0.37 TCP 54 51030 > http [ACK] Seq=321 Ack=173 Win=15544 Len=0
13066 490.5271429 192.168.0.30 192.168.0.37 TCP 54 51030 > http [FIN, ACK] Seq=321 Ack=173 Win=15544 Len=0
13067 490.5312230 192.168.0.37 192.168.0.30 TCP 171 [TCP segment of a reassembled PDU]
13068 490.5312735 192.168.0.30 192.168.0.37 TCP 54 51030 > http [RST] Seq=321 Win=0 Len=0
13069 490.5333282 192.168.0.37 192.168.0.30 TCP 60 http > 51030 [RST, ACK] Seq=290 Ack=322 Win=5840 Len=0
14205 527.1707178 192.168.0.30 192.168.0.37 TCP 74 51031 > http [SYN] Seq=0 Win=14600 Len=0 MSS=1460 SACK_PERM=1
14206 527.1734249 192.168.0.37 192.168.0.30 TCP 60 http > 51031 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460
14207 527.1734785 192.168.0.30 192.168.0.37 TCP 54 51031 > http [ACK] Seq=1 Ack=1 Win=14600 Len=0
14208 527.1735993 192.168.0.30 192.168.0.37 HTTP 421 GET / HTTP/1.1
14211 527.2253306 192.168.0.37 192.168.0.30 TCP 60 http > 51031 [ACK] Seq=1 Ack=368 Win=5473 Len=0
Checksum: 0x831d [validation disabled]
[Good Checksum: False]
[Bad Checksum: False]
[SEQ/ACK analysis]
Hypertext Transfer Protocol
GET / HTTP/1.1\r\n
Host: 192.168.0.37\r\n
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:52.0) Gecko/20100101 Firefox/52.0\r\n
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n
Accept-Language: en-US,en;q=0.5\r\n
Accept-Encoding: gzip, deflate\r\n
DNT: 1\r\n
Connection: keep-alive\r\n
Upgrade-Insecure-Requests: 1\r\n
Authorization: Basic cGw5ODIzOm5lb3BpeGVsbm90\r\n
Credentials: pl9823:neopixelnot
\r\n
[Full request URI: http://192.168.0.37/]
0120 6f 64 69 6e 67 3a 20 67 7a 69 70 2c 20 64 65 66 oding: g zip, def
0130 6c 61 74 65 0d 0a 44 4e 54 3a 20 31 0d 0a 43 6f late..DN T: 1..Co
0140 6e 6e 65 63 74 69 6f 6e 3a 20 6b 65 65 70 2d 61 nnection : keep-a
0150 6c 69 76 65 0d 0a 55 70 67 72 61 64 65 2d 49 6e live..Up grade-In
0160 73 65 63 75 72 65 2d 52 65 71 75 65 73 74 73 3a secure-R equests:
0170 20 31 0d 0a 41 75 74 68 6f 72 69 7a 61 74 69 6f 1..Auth orizatio
0180 6e 3a 20 42 61 73 69 63 20 63 47 77 35 4f 44 49 n: Basic cGw5ODI
0190 7a 4f 6d 35 6c 62 33 42 70 65 47 56 73 62 6d 39 zOm5lb3B peGVsbm9
01a0 30 0d 0a 0d 0a 0....
br0: <live capture in progress> File...
Packets: 29780 Displayed: 43 Marked: 0
Profile: Default

HTTPS vs HTTP

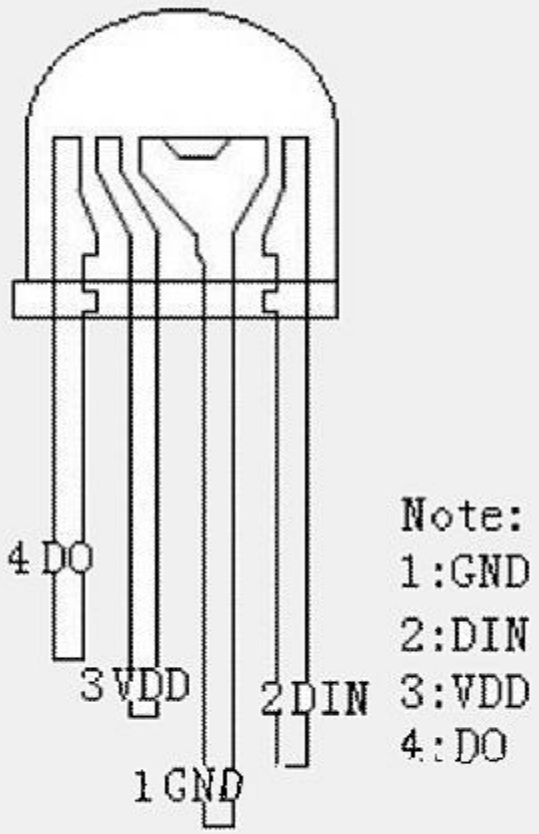
- The simple answer to this problem is instead of doing http:// We do https:// ... But we can't because the ESP8266 does not do https.
- Then, of course you ask the question - who cares if my lights get change by some hacker. And the answer is probably nobody. But this is an example of Internet of Things (IOT) and the Things are often much more critical that whimsical flashy lights Things like:
  - Live Billboards - you might not get paid if someone draws a mustache on the local politician
  - Pond pumps - could flood your yard and kill your goldfish (real story, but probably not a hacker).
  - Room lights
  - Greenhouse control
  - Pacemakers 
  - Predator Drones  Just sayin...
  - **...**

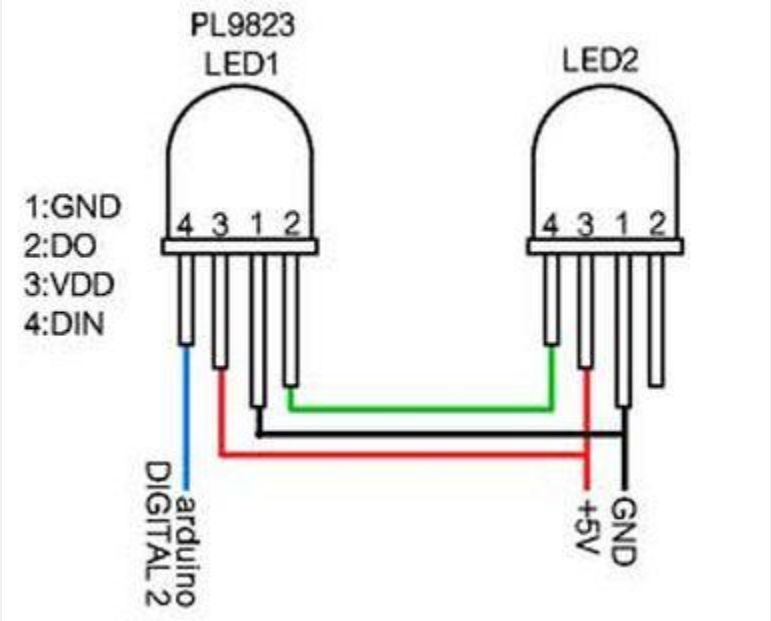
Our proposed solution is to create a middle man to handle https. So the http is only on a LAN, not the less forgiving WAN.

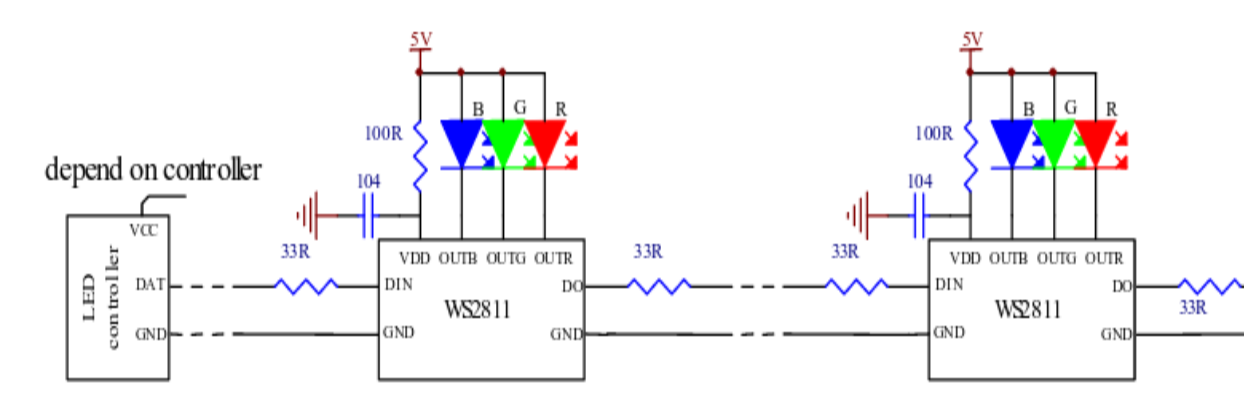


More on security next time.

Hardware


4 DO
3 VDD
2 DIN
1 GND
Note:
1:GND
2:DIN
3:VDD
4:DO


PL9823
LED1
LED2
1:GND
2:DO
3:VDD
4:DIN
4 3 1 2
4 3 1 2
arduino
DIGITAL 2
+5V
GND

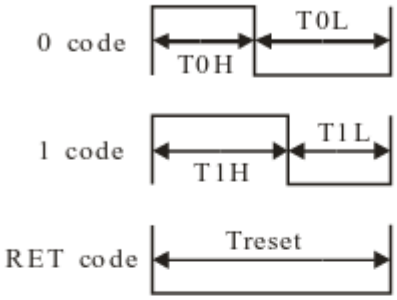


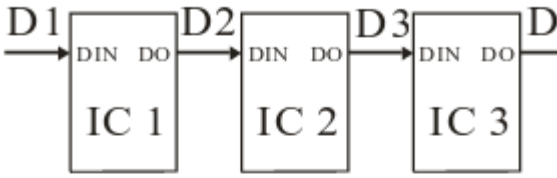
Block Diagram showing controller chip, LEDs and how they are daisy chained.

WS2811

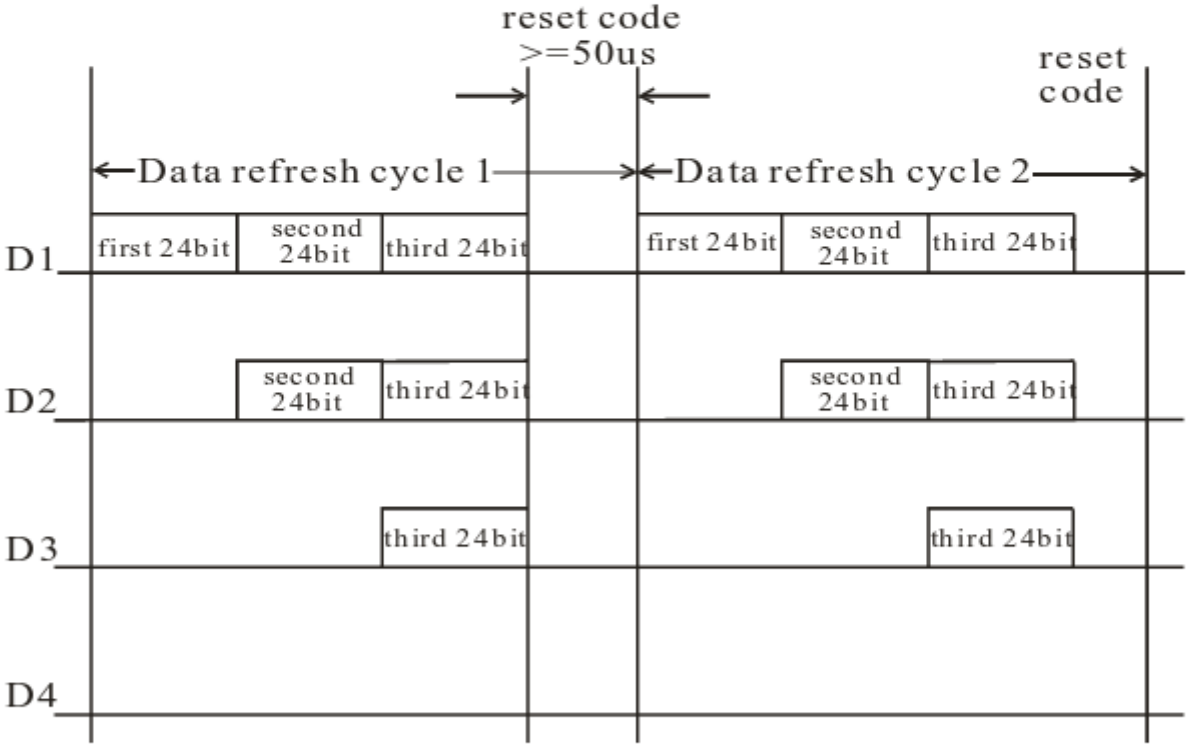
Signal line 256 Gray level 3 channa
Constant current LED drive IC

Sequence chart:





Cascade method:



Data transmission method:

Note: The data of D1 is send by MCU,and D2, D3, D4 through IC internal reshaping amplification to transmit.

Composition of 24bit data:

R7	R6	R5	R4	R3	R2	R1	R0	G7	G6	G5	G4	G3	G2	G1	G0	B7	B6	B5	B4	B3	B2

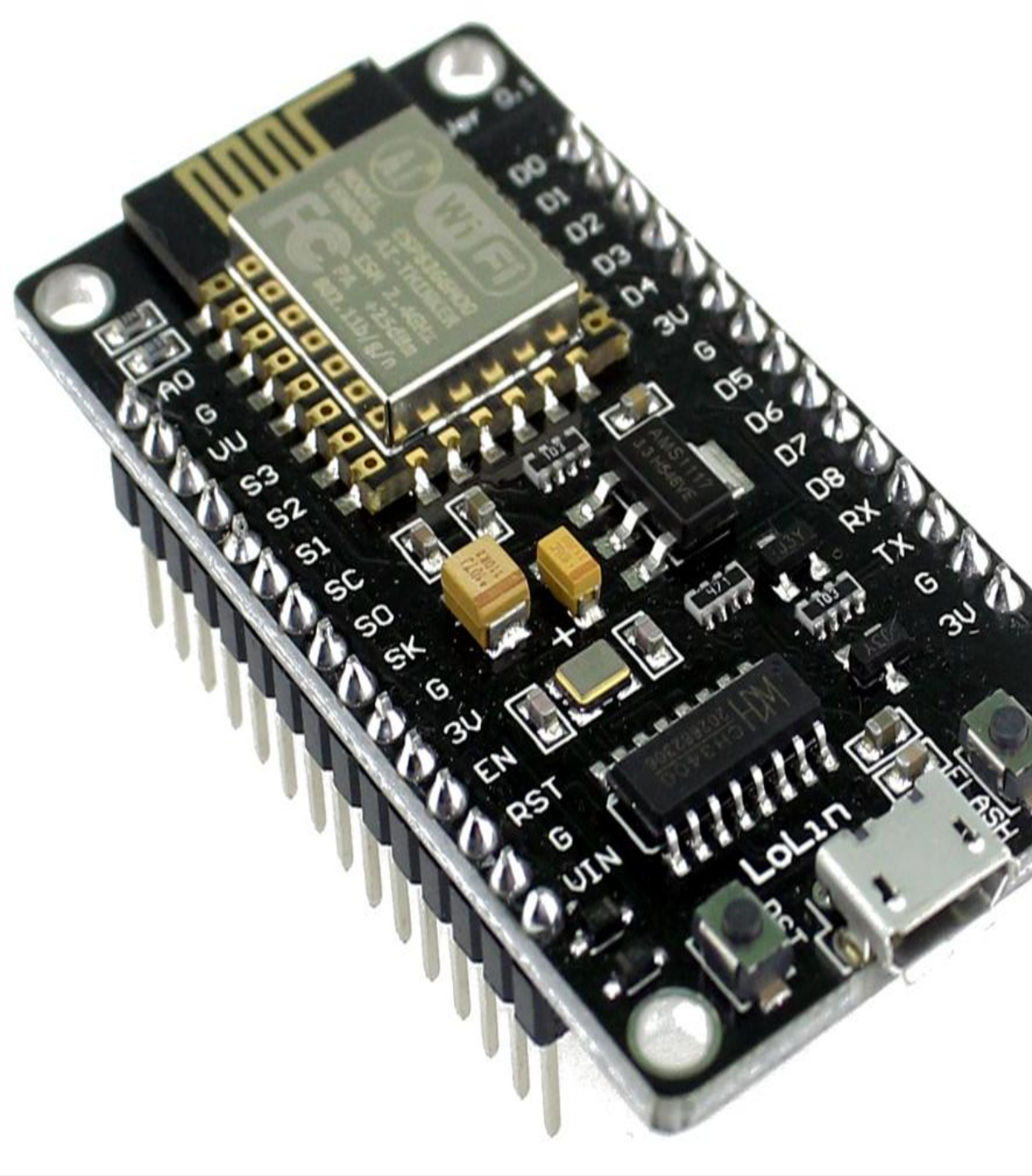
Note: Follow the order of RGB to sent data and the high bit sent at first.

Data flow. The way these WS2811 led devices work is that 24 bits for each of the led RGB colours are sent down the data wire. So in the case of 5 LEDs, 24 x 5 = 120 bits are sent from the ESP8266.

1. The first led grabs the first 24 and latches them. The remaining 96 are sent to the second led
2. The second grabs the next 24 and the remaining 72 are sent on.
3. Etc. for all the LEDs in the chain.

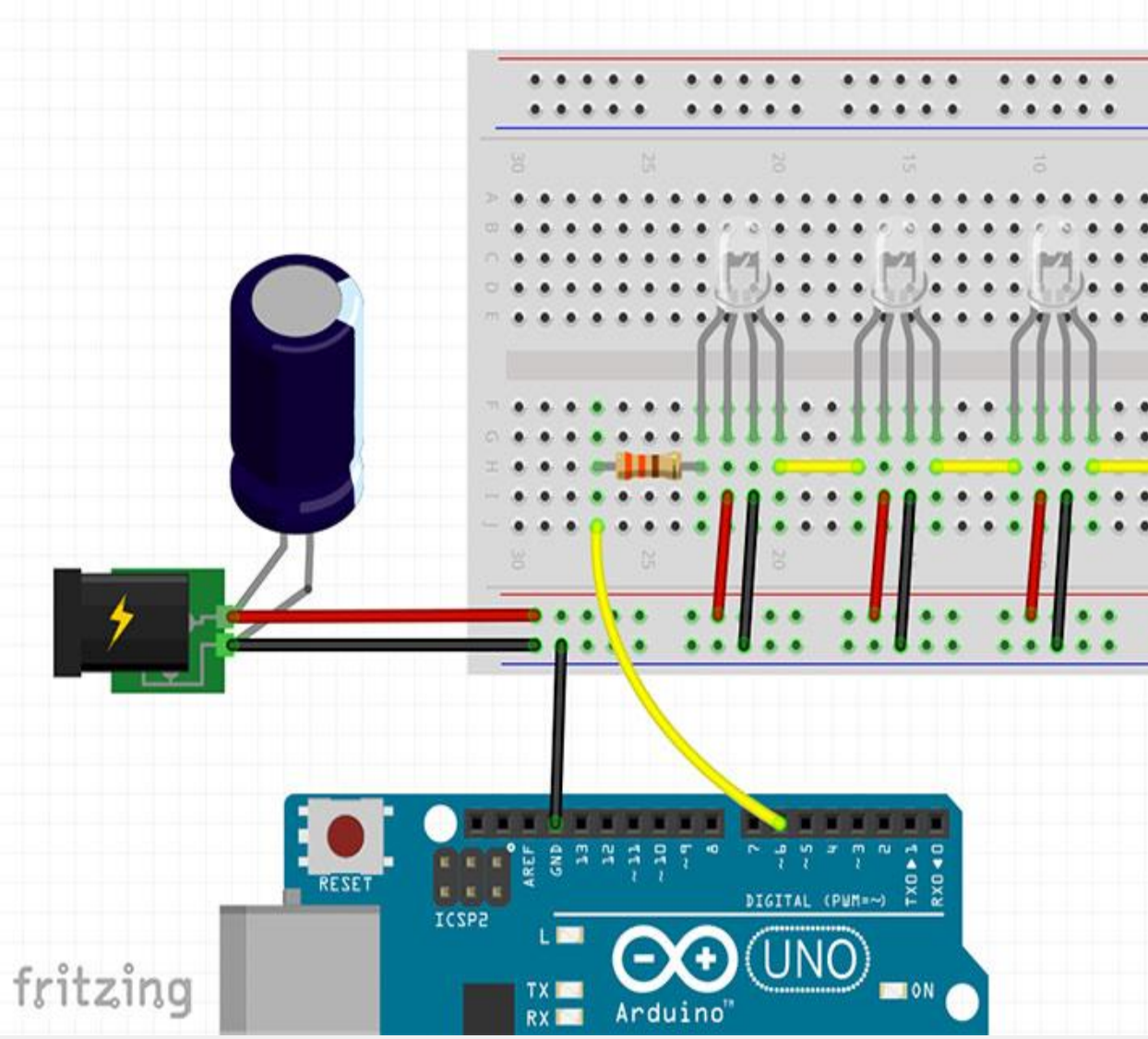
NodeMCU ESP8266 Development Board

- **NodeMCU** is an open source IoT platform.[4][5] It includes firmware which runs on the ESP8266 Wi-Fi SoC from Espressif Systems, and hardware which is based on the ESP-12 module.[6][7] The term "NodeMCU" by default refers to the firmware rather than the development kits. (Wikipedia)



NodeMCU ESP8266

Schematic



Controlling using a 5V Arduino

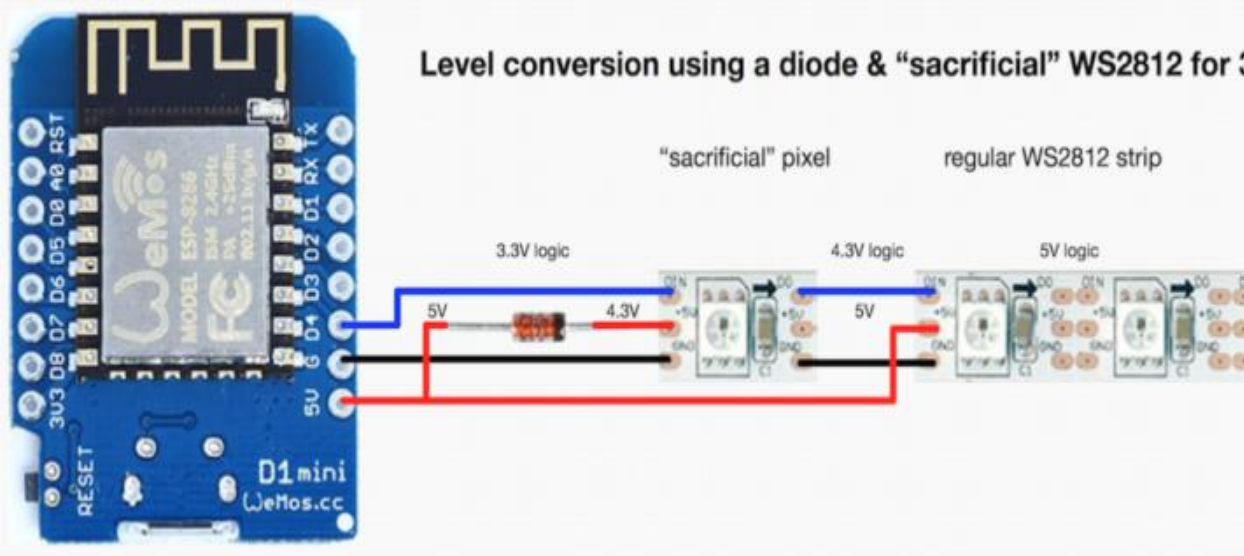
3.3 V ESP 3.3V vs 5V PL9823

To control the 5 Volt WS2811 (and others) with an ESP8266 at 3.3 volts you need to shift the level.

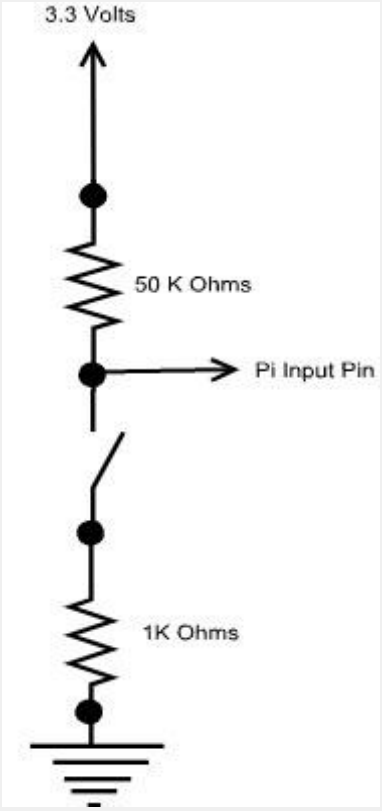
The data sheet states that a logic high input will be detected at a minimum voltage of 0.7 * Vcc. If you're running the LED at 5V, this means 5 V * 0.7 = 3.5 V will be needed for the WS2811 to detect a '1' on the data line. While you might get away with using 3.3 V, after all the specification in the data sheet is meant to be a worst case, it's possible that you'll run into reliability issues.

To perform the level shift, a signal diode is placed in series with the power

supply of the first LED. This drops the first LED to 4.3 V, which means a 4.3 V * 0.7 = 3.01 V signal can be used to control it. The logic out of this LED will be at 4.3 V, which is enough to power the rest of the LEDs running at 5 V. This information came from hackaday.com.



Controlling using a 3.3V ESP8266



Button

- The 50K Ohm resistor is called a pull up resistor. it ensures that the Pi input pin is normally connected to 3.3 Volts. This ensures that the input is not floating. If it were left floating then random environmental electrical noise could cause the input to go from 0 to 1, like, randomly. Because the resistance is so high no significant current is flowing. 0.066 MilliAmps
- 1K Ohm resistor between the switch and the ground is in case we accidentally set the pin to output rather than input. This will limit the output current in case the pin is set to output and the switch closed.
- You may need to debounce the input from a switch with some logic or delays. As the switch closes there is a period when it goes from open to closed a few times before it closes solidly. The easiest way to do this is to delay a few milliseconds before using a switch value. (https://en.wikipedia.org/wiki/Switch#Contact_bounce)

Are there topics you would like to see?