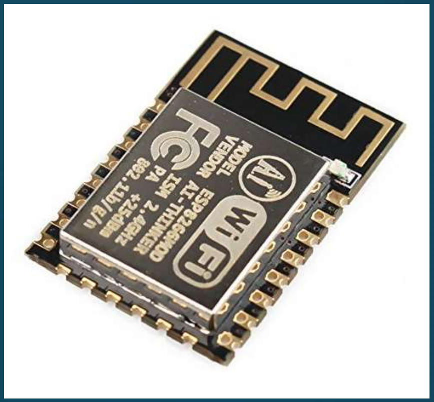
ESP8266 and IoT

Using the ESP8266 module for
Internet of Things (IoT) applications

Carl Sutter
carl@carlsutter.com
January 2018

ESP8266



- Produced by Shanghai-based Espressif Systems
- Feels like a powerful Arduino, and has WiFi built in!
- Chip is often sold on a carrier board with a WiFi antenna
- Basic module is very cheap: ~$5 on Amazon , ~$2 on Ali Express
- Many variants: https://en.wikipedia.org/wiki/ESP8266

Compared to Arduino

	Arduino UNO R3	Arduino Mega	ESP-12E
Power	5V	5V	3.3V
Clock Frequency	16MHz	16MHz	26-52MHz
Flash Memory	32KB	128 KB	4MB
SRAM	2 KB	8 KB	64 KB SRAM / 96KB DRAM

For Hobbyist Use

- Bare Module Has No USB
  - Can Be Programmed With an Arduino etc. Using I/O Pins
- Usually Used With Dev or Carrier boards

Popular Dev Boards

Amazon: ~$9

Ali Express: ~$2.75

NodeMCU Development Board

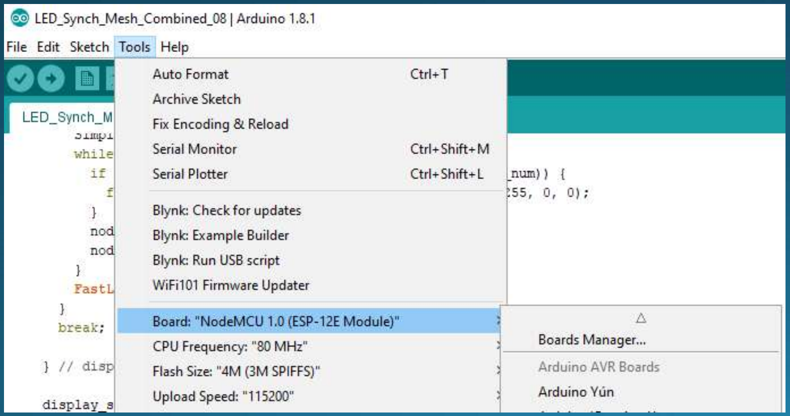
Wemos D1 Mini

NodeMCU & D1 Mini Boards

- NodeMCU Boards Come with LUA interpreter https://en.wikipedia.org/wiki/NodeMCU
- Both can be updated with Arduino firmware, and then behave very much like an Arduino
- There are cute little shields for the D1 Mini boards!

Adding Board Driver to Arduino

- Search Google for "Arduino ESP8266"
- Instructions are on GitHub page: https://github.com/esp8266/Arduino
- I am not sure if the USB Driver is separately required or not
- In the Arduino IDE, Select the NodeMCU board



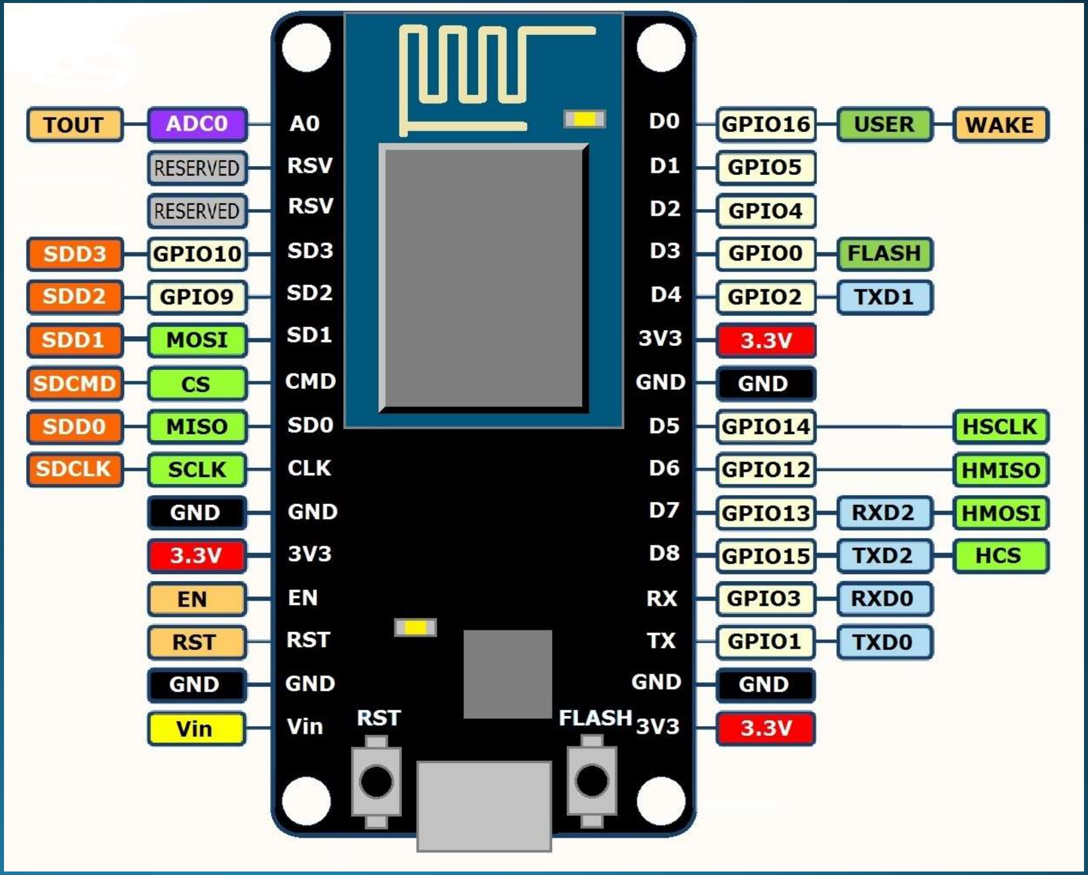
Board: "NodeMCU 1.0 (ESP-12E Module)"
CPU Frequency: "80 MHz"
Flash Size: "4M (3M SPIFFS)"
Upload Speed: "115200"

Using Arduino Libraries

- The Arduino IDE is compiling for a different chip set
- Libraries that use low level functions need to be ESP8266 specific
- Most network libraries have ESP8266 versions
- Current Neopixel library now works for both
- Generally, you can search for “Arduino ESP8266 XXX”

Arduino Pin Names vs ESP8266

- D2 = GPIO4, so in Neopixel etc. code:
- `#define PIN 2` changes to `#define PIN 4`
- `#define PIN 6` changes to `#define PIN 12`



Or Simply Use Dx Pin Constants in the Arduino ESP8266 System

```
static const uint8_t D0   = 16;
static const uint8_t D1   = 5;
static const uint8_t D2   = 4;
static const uint8_t D3   = 0;
static const uint8_t D4   = 2;
static const uint8_t D5   = 14;
static const uint8_t D6   = 12;
static const uint8_t D7   = 13;
static const uint8_t D8   = 15;
static const uint8_t D9   = 3;
static const uint8_t D10  = 1;
```

Connecting to WiFi

```
#include <ESP8266WiFi.h>

void setup() {
  WiFi.begin(“your-ssid”, “your-password”);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
  }
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}
```

Connecting to WiFi

- Hard coded SSID and Password
- Good for known location, or tethered to your phone
- How to make portable?

WiFi Manager

- https://github.com/tzapu/WiFiManager

- On bootup, looks for stored WiFi credentials
- If those fail, sets up a WiFi Access Point (AP) and web server
- You connect to that AP with your phone or PC
- Like a hotel, a browser may open to the config page
- If not, use IP address 192.168.4.1
- Add your local WiFi credentials, and it saves and resets

WiFi Manager – Simplest Form

```
#include <ESP8266WiFi.h>
#include <DNSServer.h>
#include <ESP8266WebServer.h>
#include <WiFiManager.h>

void setup() {
   WiFiManager wifiManager;
   wifiManager.autoConnect();
}
```

Web Clients

- Lots of libraries, including the old Arduino ones
- With the ESP8266, you get Arduino library examples in /ESP8266WiFi
- Do not need a fixed IP address

Web Clients

```
const char* host = "www.yoursite.com";
void loop() {
  WiFiClient client;
  if (!client.connect(host, 80)) return;
  client.print(String("GET /foo/bar") + " HTTP/1.1\r\n" +
               "Host: " + host + "\r\n" +
               "Connection: close\r\n\r\n");
  while(client.available()){
    String line = client.readStringUntil('\r');
    Serial.print(line);
  }
}
```

Web Servers

- You can hard-code an IP address as with any Arduino Ethernet Shield etc.
- To make it portable, you would need Dynamic DNS
- In a known location, you can set DDNS up on your router
- For a mobile app, there seem to be libraries like https://github.com/ayushsharma82/EasyDDNS

ESP8266WebServer Example

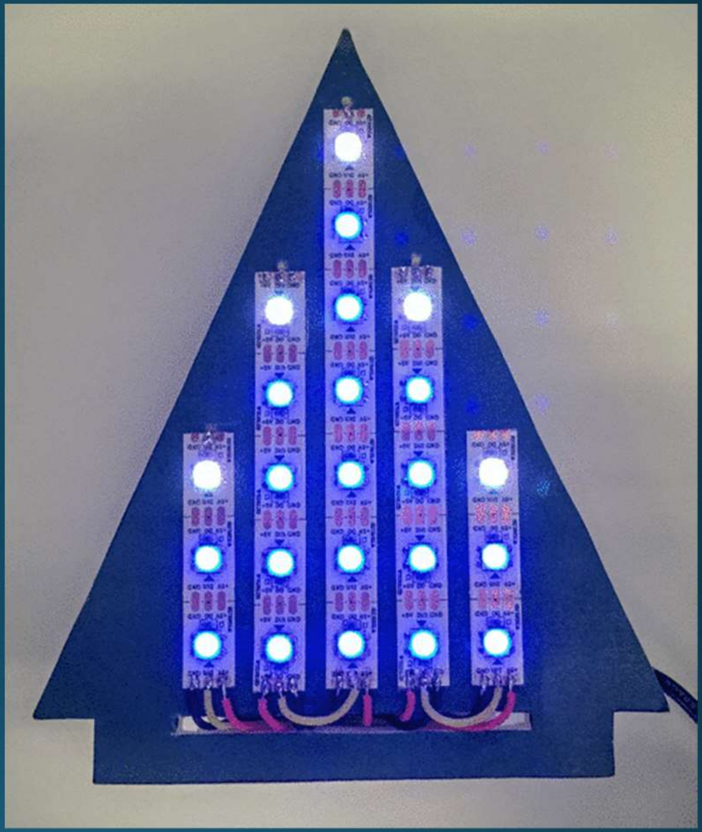
```
ESP8266WebServer server(80);
void handleOn() {
  server.send(200, "text/plain", “turning the device on");
}
void handleOff() {
  server.send(200, "text/plain", “turning the device off");
}
void setup(void){
  server.on("/on", handleOn);
  server.on("/off", handleOff;
  server.begin();
}
void loop(void) {
  server.handleClient();
}
```

Useful Libraries

- WiFiManager
- painlessMesh – mesh of ESP8266 nodes
- Ping – can check server response times
- MQTT – message queueing to a server (cloud or Pi)
- Neopixel, and FastLED

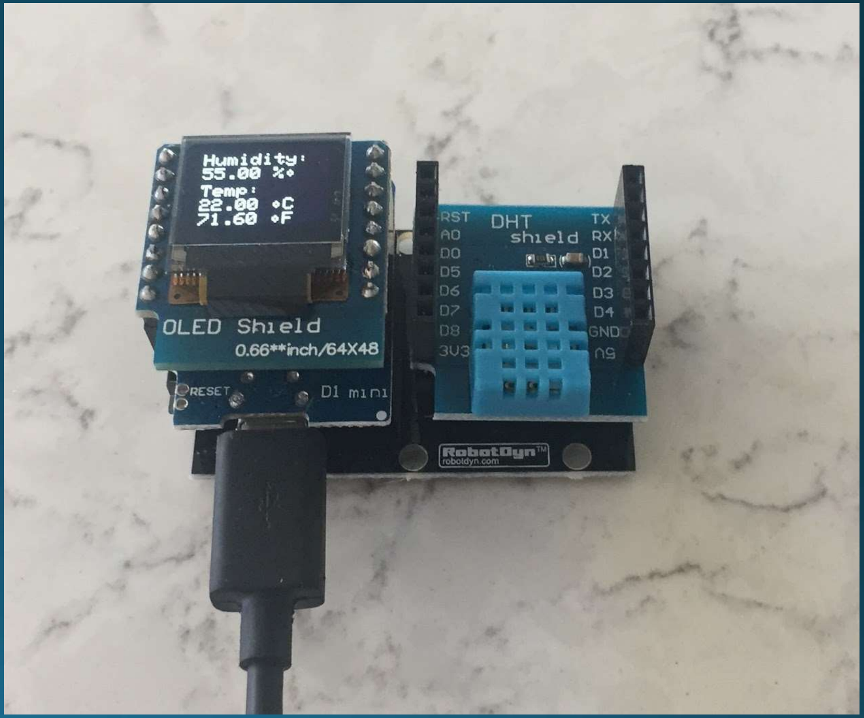
LED Tree Example

- Web site to set LED colors
  - teletoyland.com/Projects/LEDTree
- Server stores latest colors
- LED Tree has a NodeMCU
- Web Client polls the server every 5 seconds for new colors
- You can build one
  - http://www.instructables.com/id/LED-Holiday-Tree-With-Shared-Internet-Control



Humidity/Temp Example

- D1 Mini
- DHT22 shield – temp and humidity
- OLED shield (64x48 pixels)
- Optional dual base
  - to keep sensor further from CPU
- Total Cost from Ali Express < $10
- Could use WiFi to log data

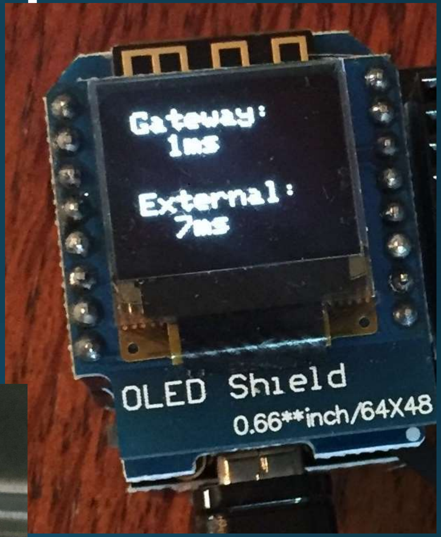


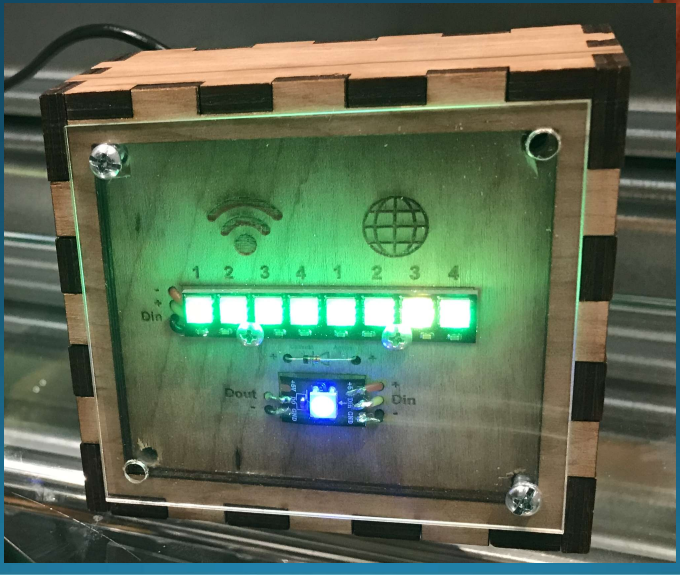
LED Bar Mesh Example

- Synchronized animation between LED bars
- Setup own Wifi APs and Mesh
- Mobile – off grid
- Original idea was batons for dual drum majors
- instructables.com/r/carl

Network Speed Monitor Example

- Brother-in-law does restaurant technology
- Restaurants have connectivity issues
- Device to monitor local gateway and remote server
- Reports Ping Latency
- D1 Mini, OLED
- Actual App uses RGB LEDs
- Easy with Ping Library



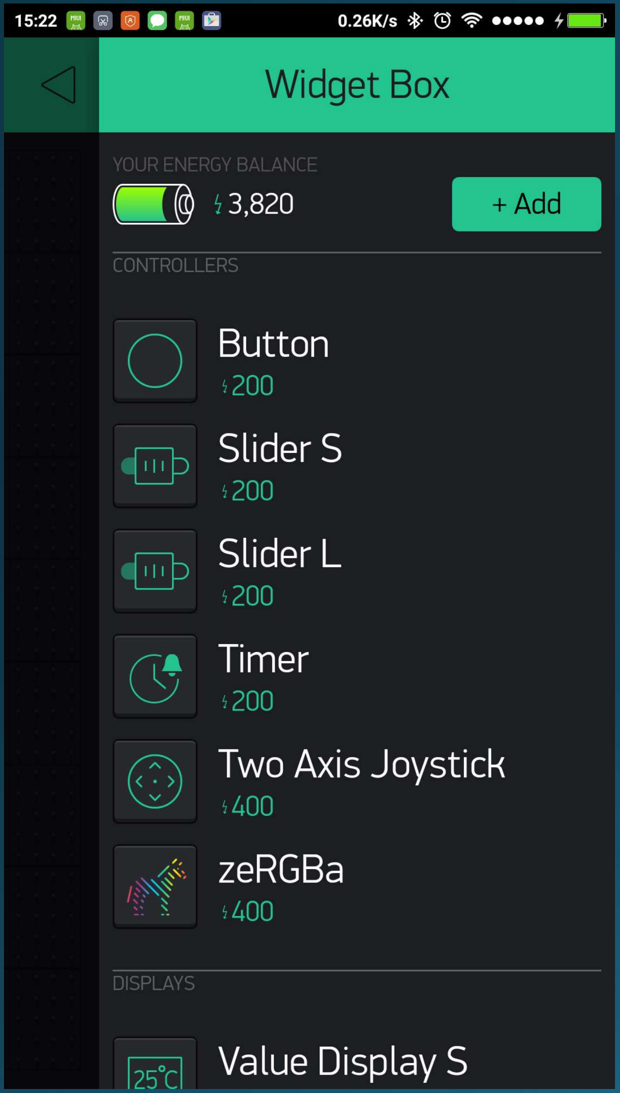


Neopixels & 3.3v Controllers

- WS2812b LEDs are ~5v devices
- ESP8266s are 3.3v – boards have a regulator
- So, 5v to each is fine
- But, data line is the issue – 3.3v, and WS2812b wants 70% of Vcc
- Can use Level Shifter ICs – some wiring and space usage
- LED Tree uses 4.5v wall adaptors
- Works, but hard to find > 1 amp, so site limits bright colors
- Sacrificial LED concept – first LED has a diode to drop the supply
- Both approaches are almost unnoticeable

Blynk – Phone Apps

- Phone UI to add controls
- Arduino sample code to connect
- Can read and write data
- Has a server handling all the message passing
- Account and auth codes free – they charge for controls after the first few
- www.blynk.cc



15:22 0.26K/s

Widget Box

YOUR ENERGY BALANCE

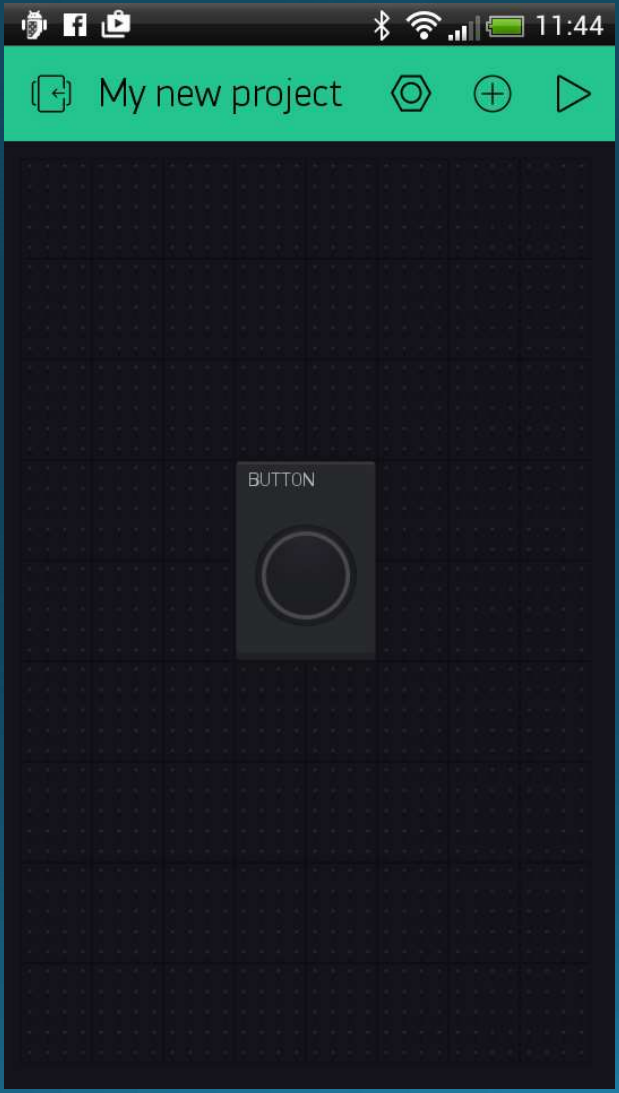
3,820 + Add

CONTROLLERS

Button	200
Slider S	200
Slider L	200
Timer	200
Two Axis Joystick	400
zeRGBa	400

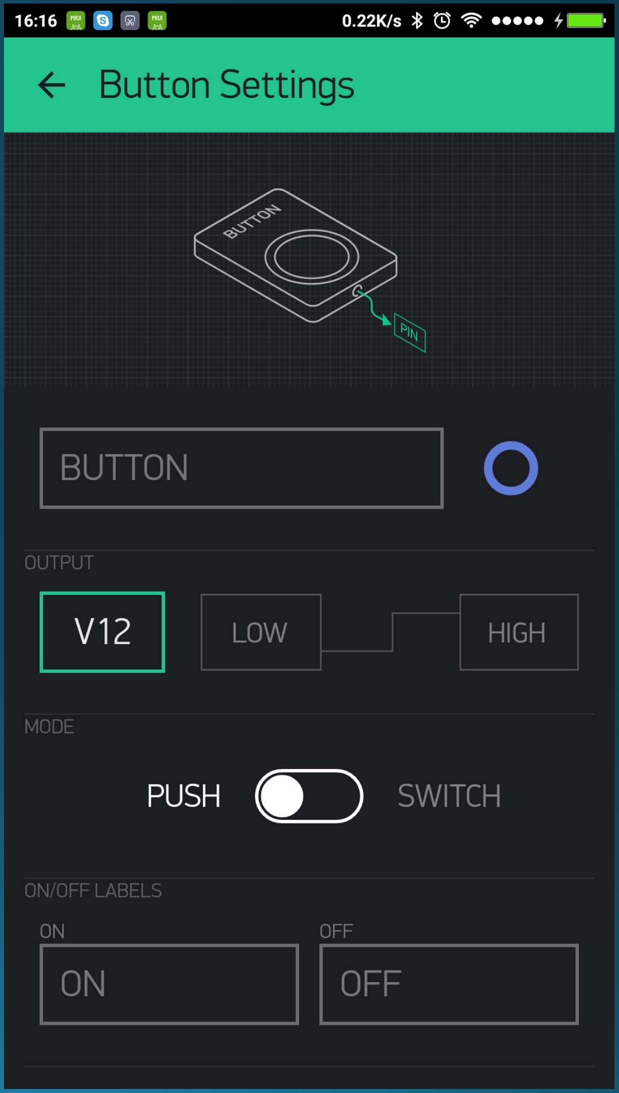
DISPLAYS

Value Display S



11:44

My new project



16:16 0.22K/s

Button Settings

BUTTON

OUTPUT

V12 LOW HIGH

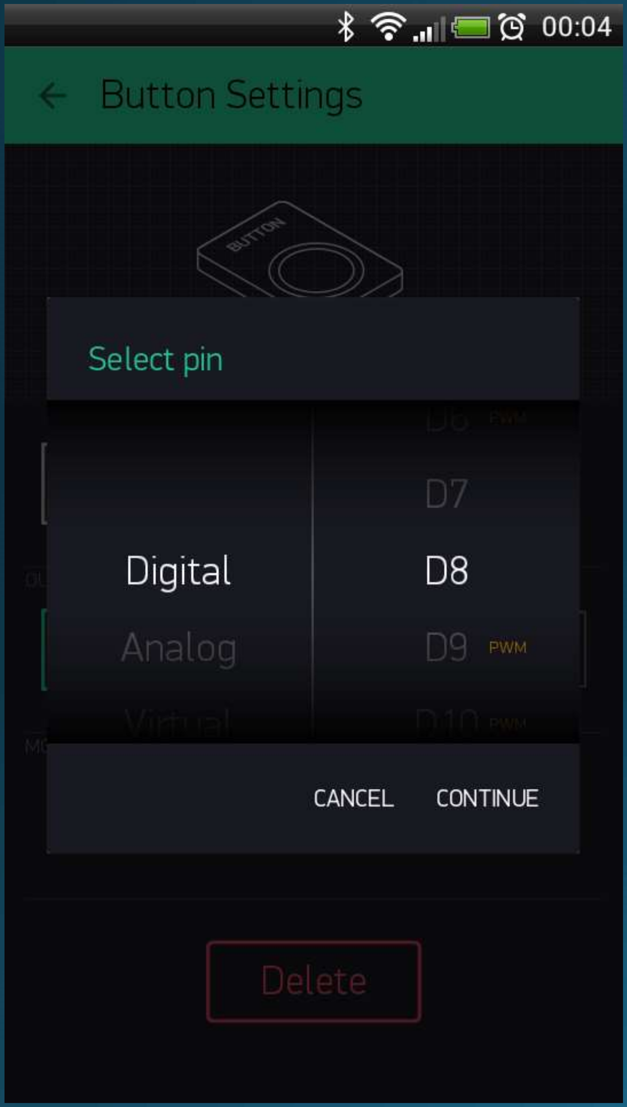
MODE

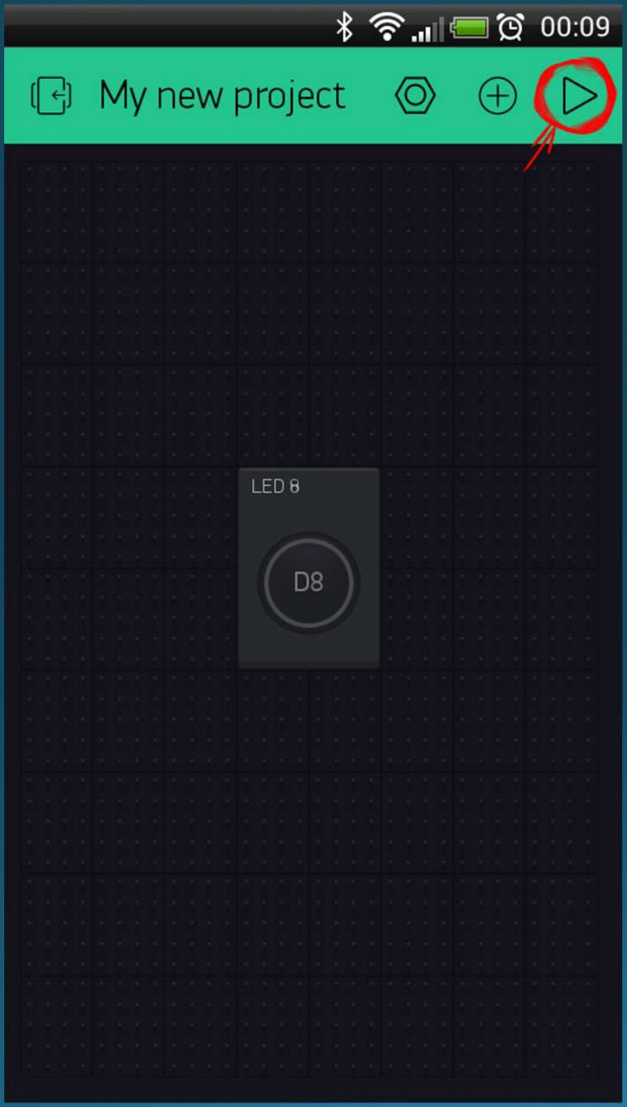
PUSH SWITCH

ON/OFF LABELS

ON: ON

OFF: OFF


00:04
Button Settings
BUTTON
Select pin
D6 PWM
D7
Digital
D8
Analog
D9 PWM
Virtual
D10 PWM
CANCEL
CONTINUE
Delete

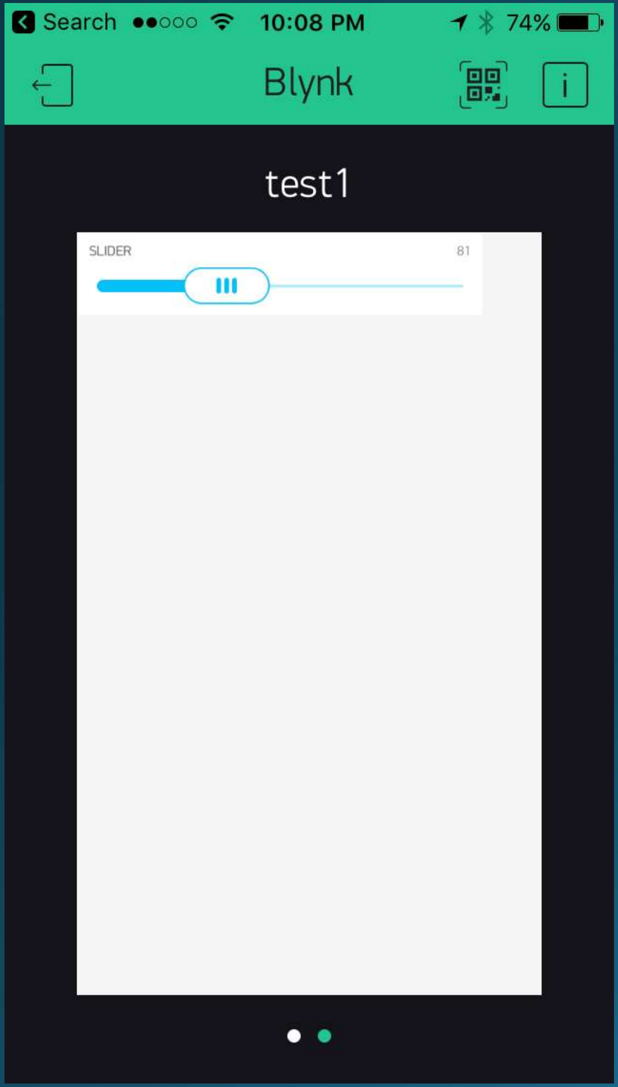

00:09
My new project
LED 8
D8

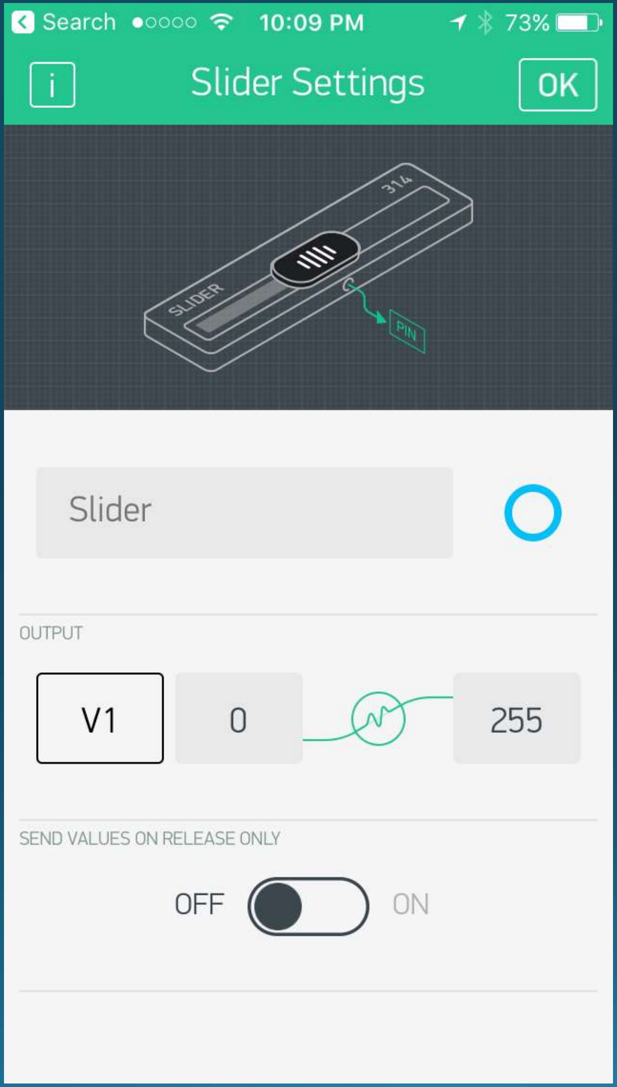
Blynk Code

```
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
char auth[] = "YourAuthToken";

void setup() {
  Blynk.begin(auth, "YourNetworkName", "YourPassword");
}

void loop() {
  Blynk.run();
}
```

Simple Blynk Neopixel

- Slider to change strip to rainbow color





Slider

OUTPUT

V1 0 255

SEND VALUES ON RELEASE ONLY

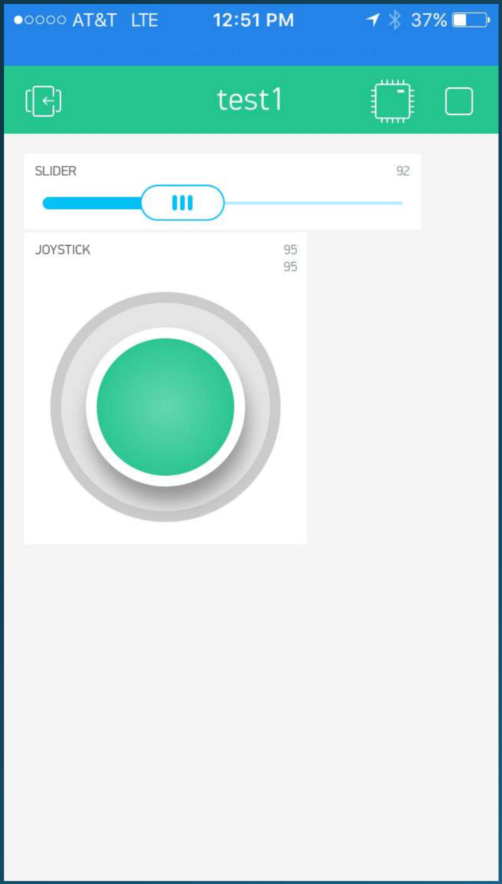
[✓ OFF] ON

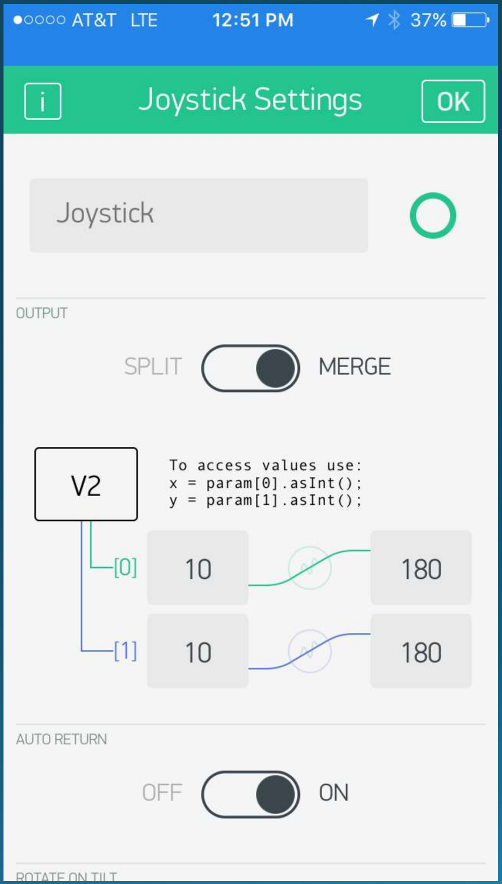
Blynk Additional Neopixel Code

In addition to the standard Neopixel and Blynk code…

```
BLYNK_WRITE(V1) {
  int shift = param.asInt();
  for (int i = 0; i < strip.numPixels(); i++) {
    strip.setPixelColor(i, Wheel(shift & 255));
  }
  strip.show();
}
```

Blynk: Adding Servos





Joystick

OUTPUT

SPLIT MERGE

V2

```
To access values use:
x = param[0].asInt();
y = param[1].asInt();
```

[0] 10 180

[1] 10 180

AUTO RETURN

OFF ON

Blynk Additional Servo Code

```
In addition to the standard Servo init code…
#include <Servo.h>
Servo servo_left, servo_right;

IN SETUP: servo_left.attach(2);
IN SETUP: servo_right.attach(4);

BLYNK_WRITE(V2) {
  int left = param[0].asInt();
  int right = param[1].asInt();
  servo_left.write(left);
  servo_right.write(right);
}
```

What’s Next

- So many comparable modules!
- Expressif ESP-32: WiFi and Bluetooth BLE!
- OTA – over the air updates like Spark/Particle Photon