



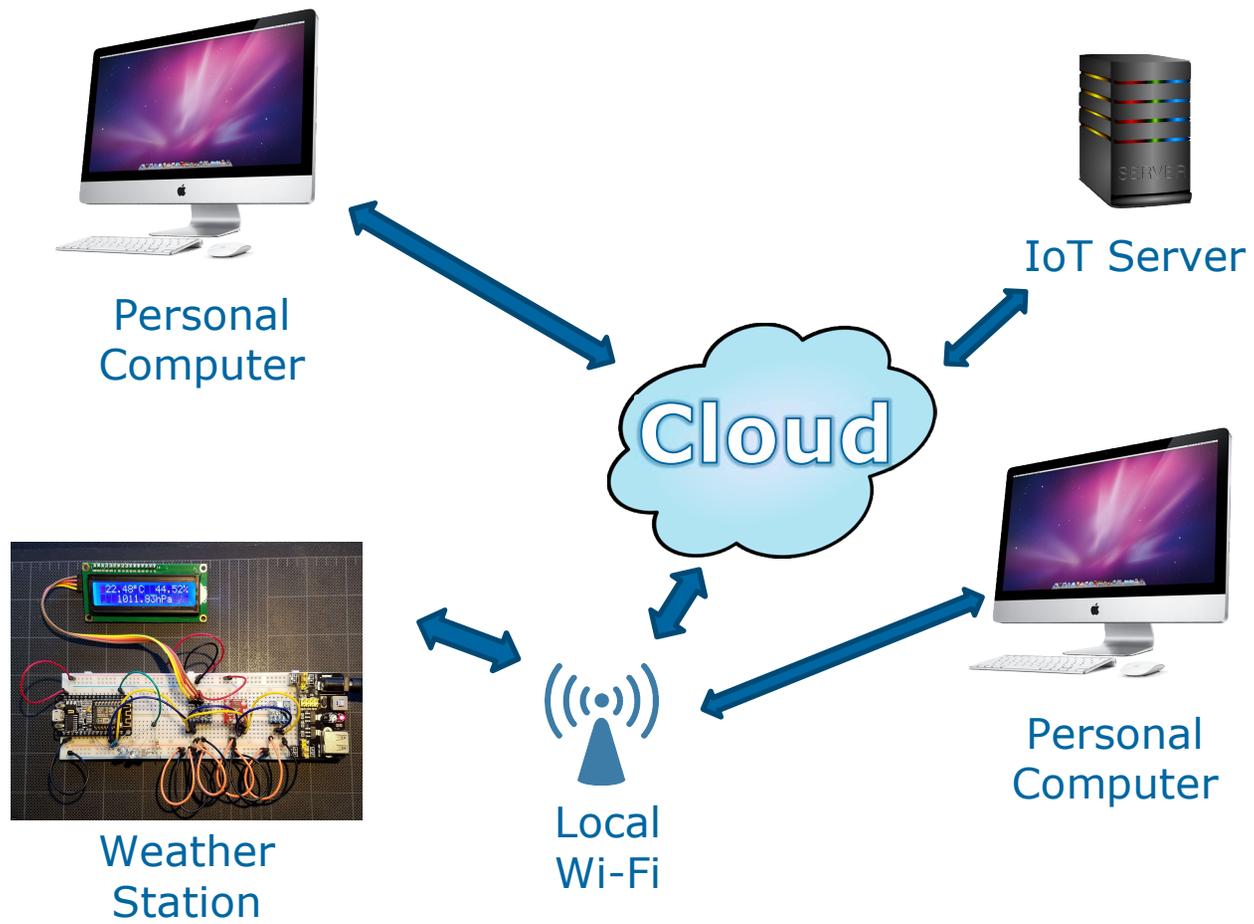
# Internet of Things Weather Station

IEEE Northern Virginia Section

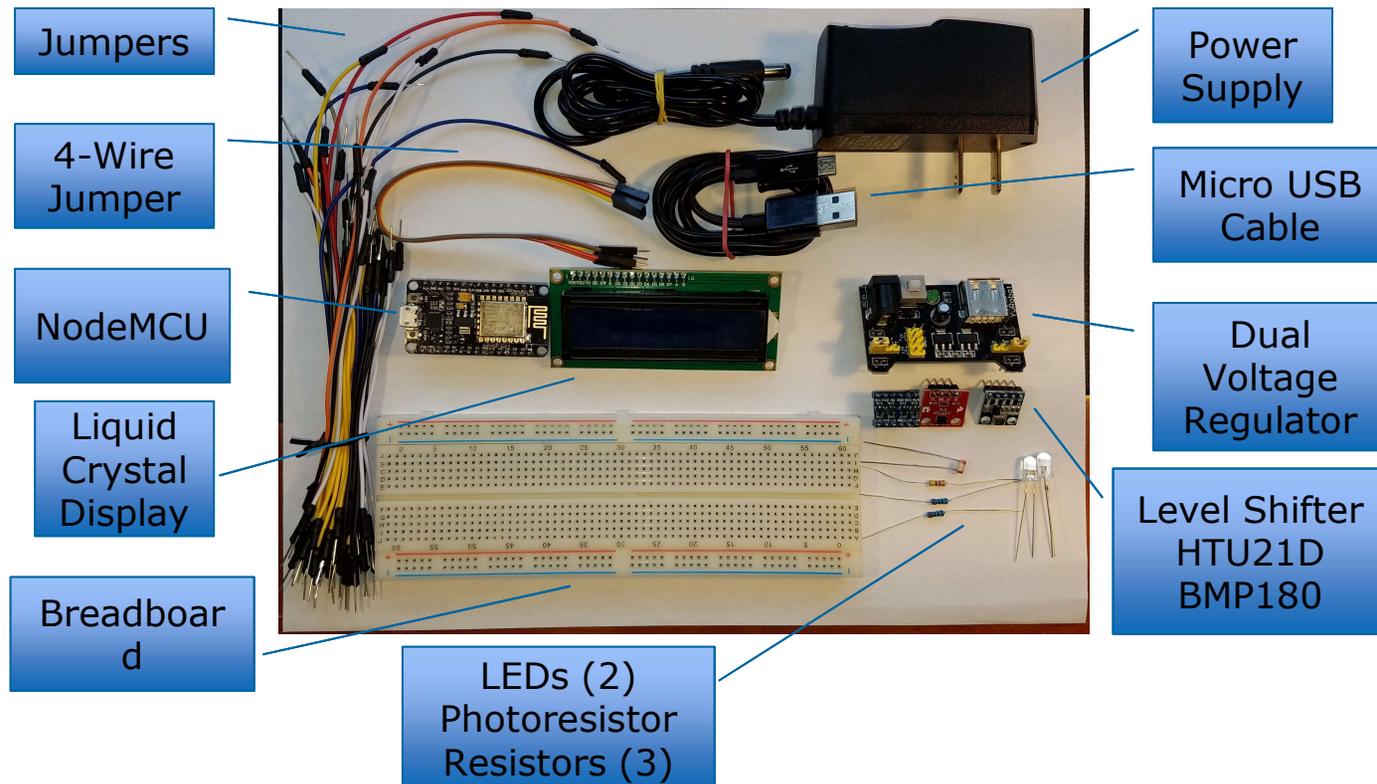
Hands-On Professional Development Series

June 4, 2016

# Unboxing & Sketch 01-Blink



# Parts List

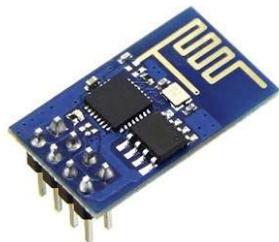


# Microcontroller

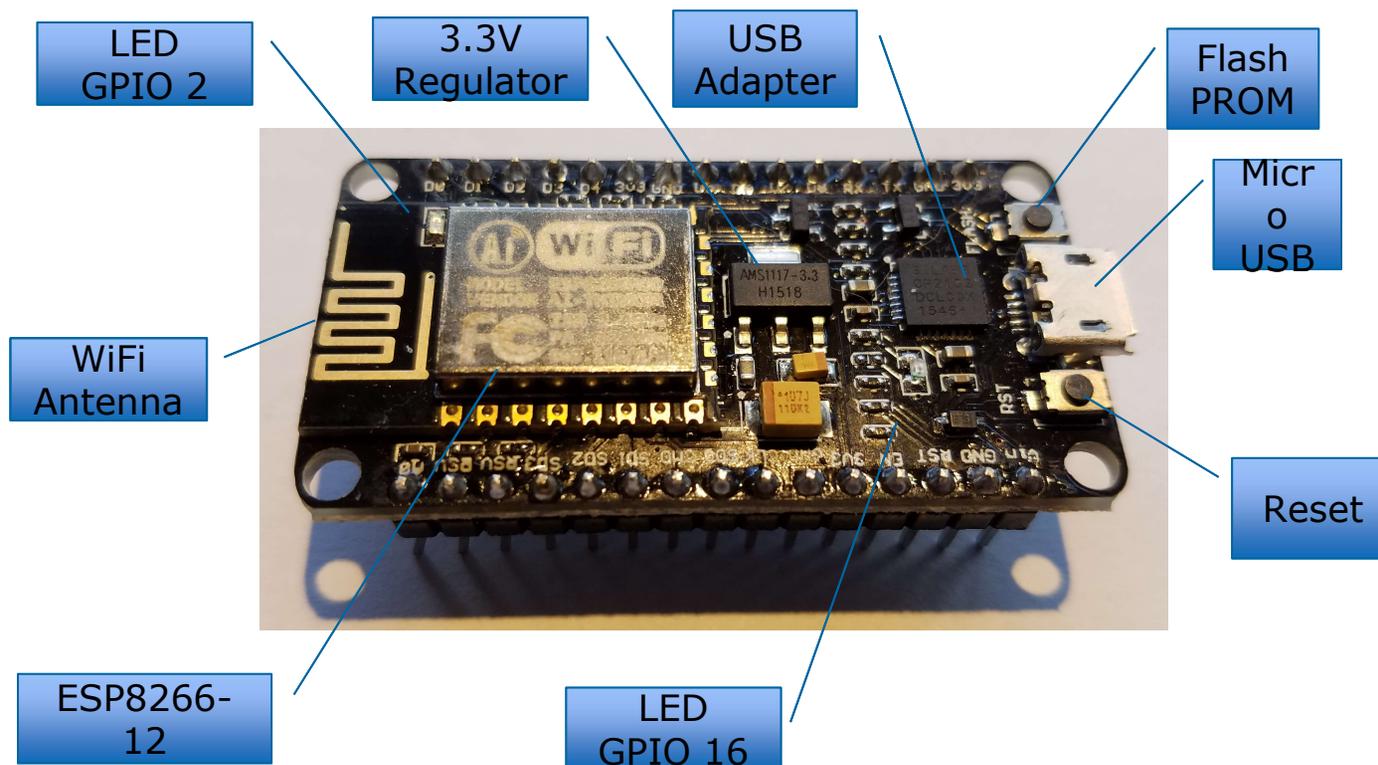
- ▶ A microcontroller is a **System on Chip** computer
  - Processor, memory, analog, digital, and radio frequency circuits
- ▶ **Embedded** in a device with a dedicated purpose
- ▶ Generally low power and often battery powered
- ▶ Program is stored in firmware & does not change
- ▶ Has multiple digital **General Purpose Input / Output**, analog-to-digital conversion & pulse width modulation

# ESP8266 Timeline

- ▶ January 2014 - Introduced by Expressif of Shanghai as a Wi-Fi chip
- ▶ Mid-2014 – Early adopters used it as a Wi-Fi modem with Hayes “AT” commands generated by an Arduino or Raspberry Pi.
- ▶ October 2014 - Expressif released SDK
- ▶ March 2015 - Arduino core released
- ▶ December 2015 – Breadboard formats introduced



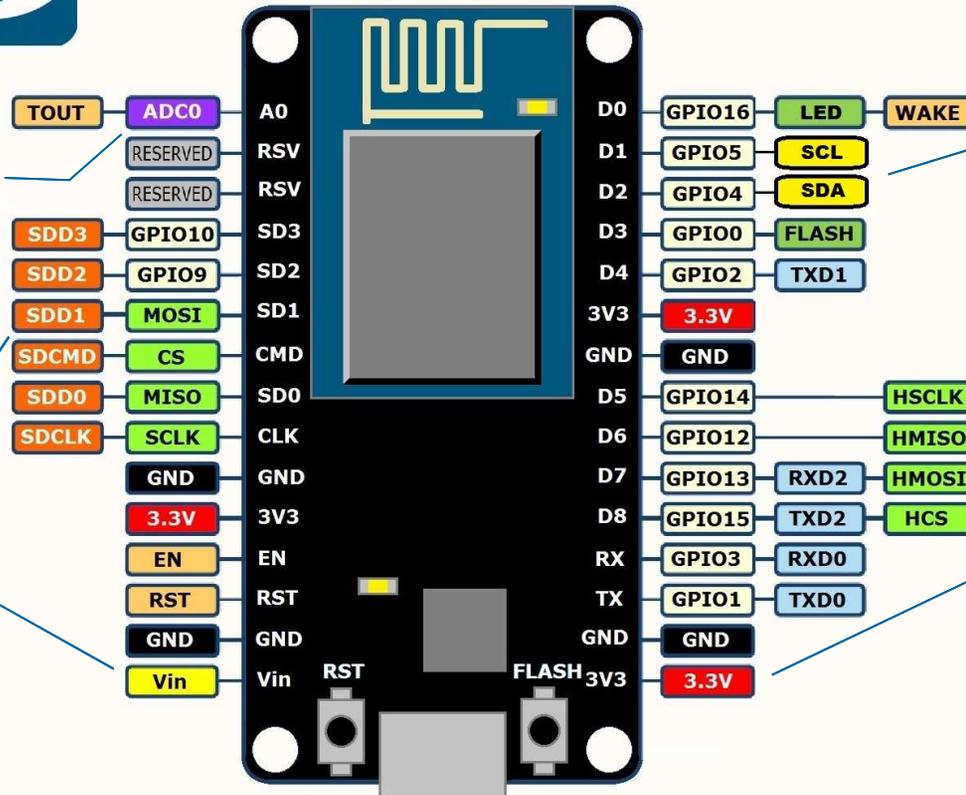
# NodeMCU





# NodeMCU ESP-12 development kit V1.0

## PIN DEFINITION



10-bit Analog to digital

SD Memory

Input Power Supply 7 – 10 Vdc

I2C Bus

Limited 3.3V Source

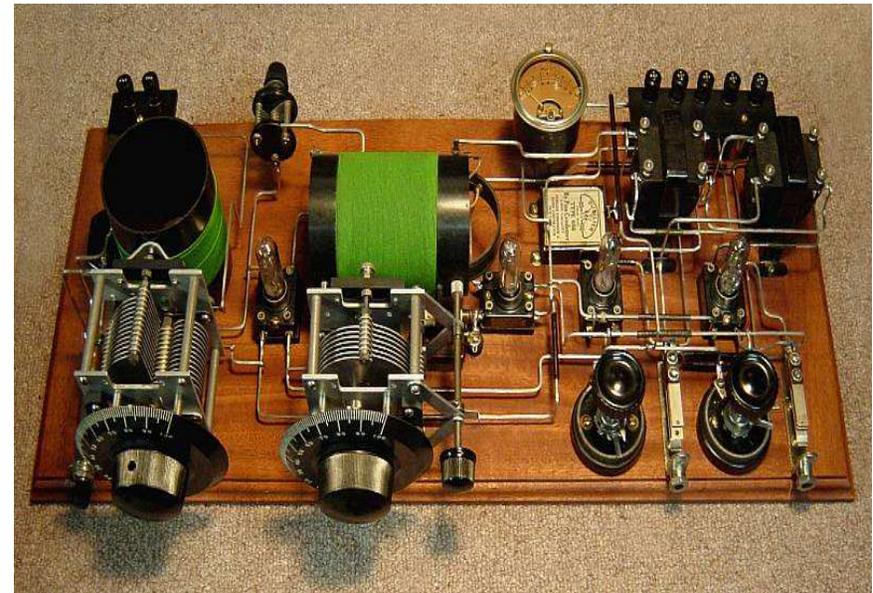
Arduining.com

# The Original Breadboard

In the Kitchen

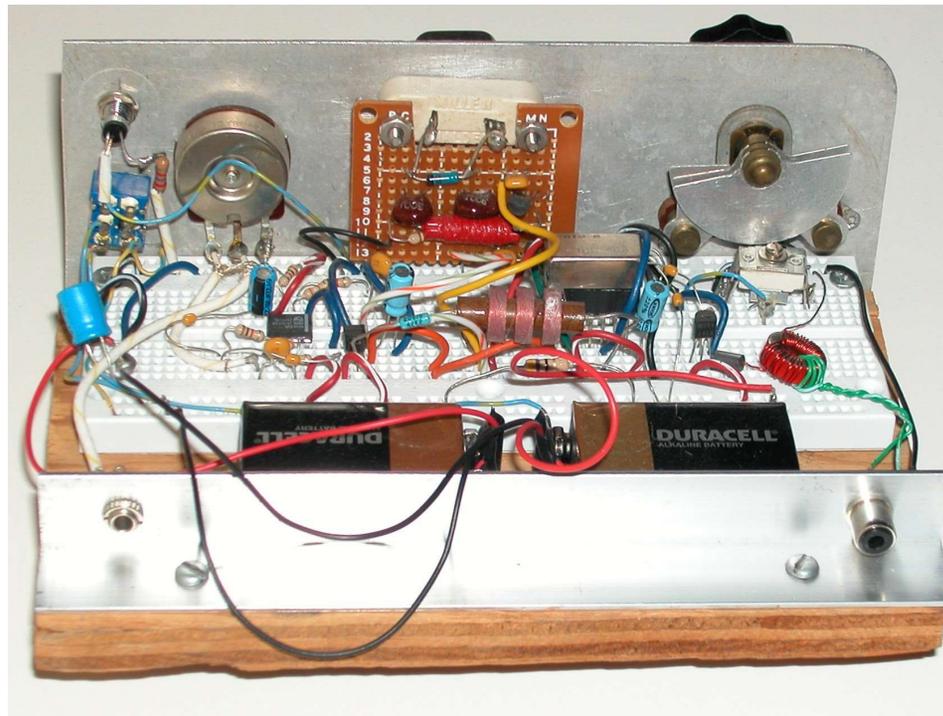


On the Air



# Modern Solderless Breadboard

## K4KRE Direct Conversion SDR Receiver





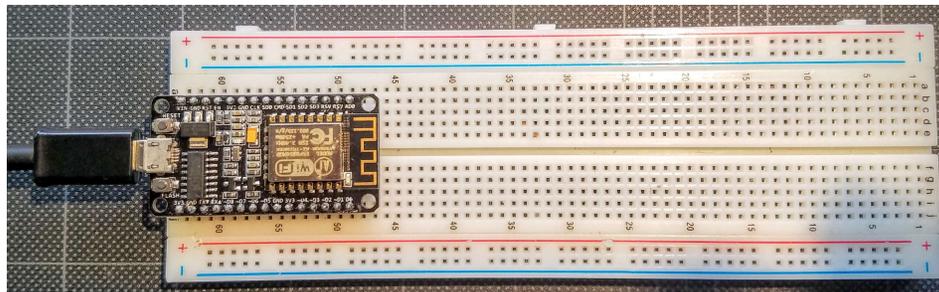
# Breadboard Usage

- ▶ Use solid (not stranded) wire AWG 22-26
- ▶ Use "Dupont" jumpers or precut wires
- ▶ Use 1/4Watt or 1/2Watt resistors
- ▶ Do Not force a wire into the board
- ▶ Do Not pass more than 1Amp through one point
- ▶ Do Not use more than 50Volts between columns



# Prepare the breadboard

1. Orient the protoboard with the + (Red) bus at the top.
2. Use any jumper to **open up 15 points in rows B and I.**
3. Carefully insert the NodeMCU on the left end with the USB connector pointing left.
4. **Apply pressure over the pins** – not in middle of board.

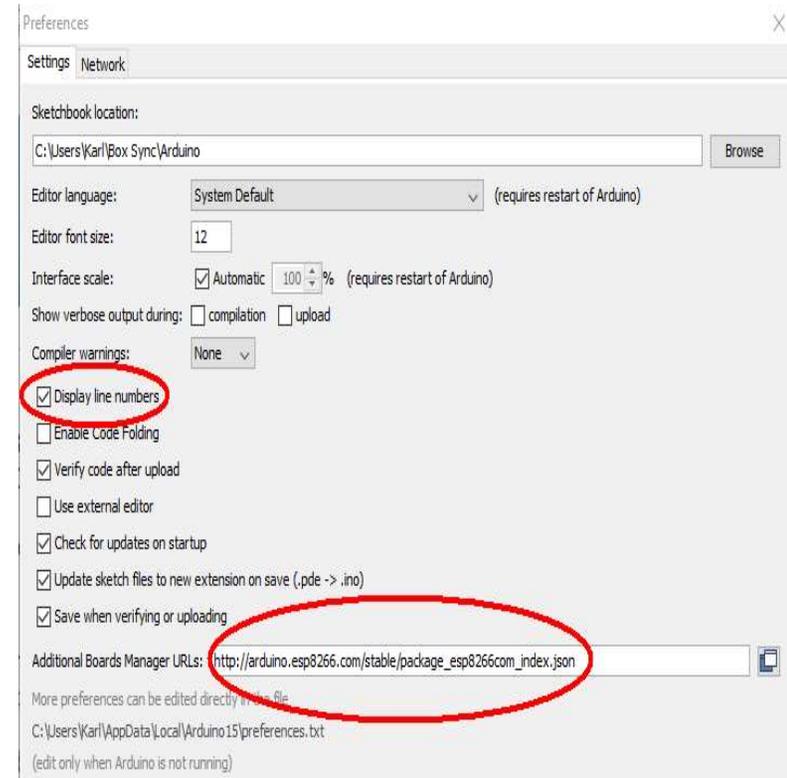


# Course Files

- ▶ We will have a local server set up during the course to increase the speed of downloads.
- ▶ Instructions will be provided.
- ▶ After the course, all course materials, programs, and links to resources are available at:
  - <http://w4krl.com/projects/ieee-iot/>

# Install Arduino IDE

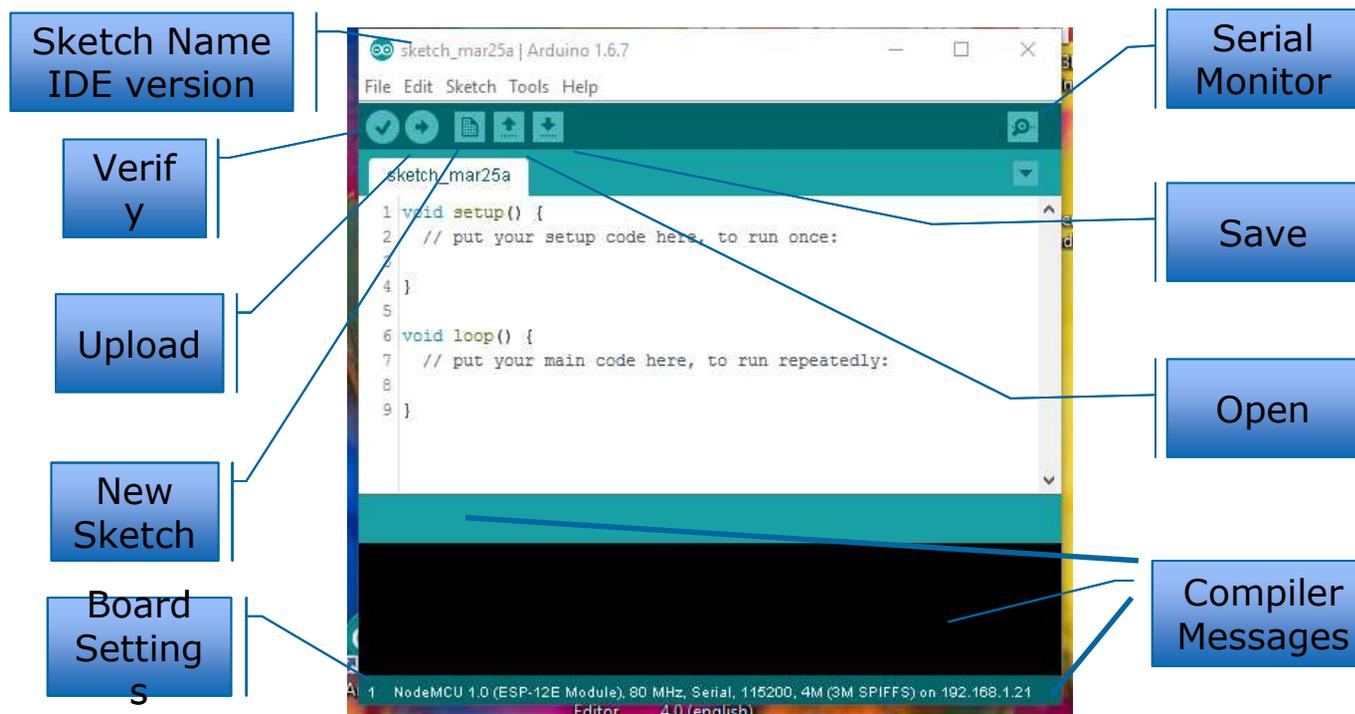
1. Open Windows Explorer
2. Navigate to Documents / Arduino
3. Run Arduino-1.6.9-windows
4. Click the new Arduino icon on your desktop
5. Open menu item File | Preferences
  - Check **Display line numbers**
6. Type in Additional Boards Manager URLs:  
**[http://arduino.esp8266.com/stable/package\\_esp8266com\\_index.json](http://arduino.esp8266.com/stable/package_esp8266com_index.json)**
7. Click OK



## Install ESP8266 Core

1. Open menu item Tools | Boards | Boards Manager...
2. Type in the search box "esp8266"
3. Click on "More info." Install latest version from dropdown box that appears on the right
4. Click Close
5. Open menu item Tools | Boards and select **NodeMCU 1.0 (ESP-12 E Module)**

# Integrated Development Environment (IDE)



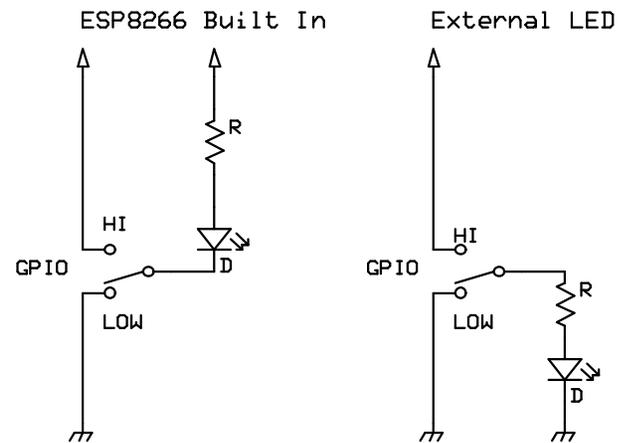
# First Sketch - Blink

1. Open Arduino IDE
2. Set COM port in menu Tools | Port
3. Set Board to **NodeMCU 1.0 (ESP-12E Module)**
4. File | Sketchbook | **IEEE\_IoT\_Sketch01\_Blink**
5. Click  to verify the sketch – compiles without uploading
6. Click  to upload the sketch to the NodeMCU

# IEEE\_IoT\_Sketch01\_Blink

```
11 // Global constants
12 const int LED_PIN = 16;           // the built in LED is on GPIO16
13 const int DURATION_ON = 100;      // ON duration in milliseconds
14 const int DURATION_OFF = 1000;    // OFF duration in milliseconds
15
16 // Every Arduino sketch must have a setup() function
17 // setup() runs once
18 void setup()
19 {
20   // Initialize the LED_PIN as a digital output
21   pinMode(LED_PIN, OUTPUT);       // Initialize the LEDPIN as an output
22 } // setup()
23
24 // Every Arduino sketch must have a loop() function
25 // loop() runs forever
26 void loop()
27 {
28   // pull the output to ground to turn LED on
29   digitalWrite(LED_PIN, LOW);
30
31   // pause for the ON duration
32   delay(DURATION_ON);
33
34   // pull the output to V+ to turn the LED on
35   digitalWrite(LED_PIN, HIGH);
36
37   // Pause for the OFF duration
38   delay(DURATION_OFF);
39 } // repeat loop()
```

# LED Connections



# Make changes, upload, observe, repeat

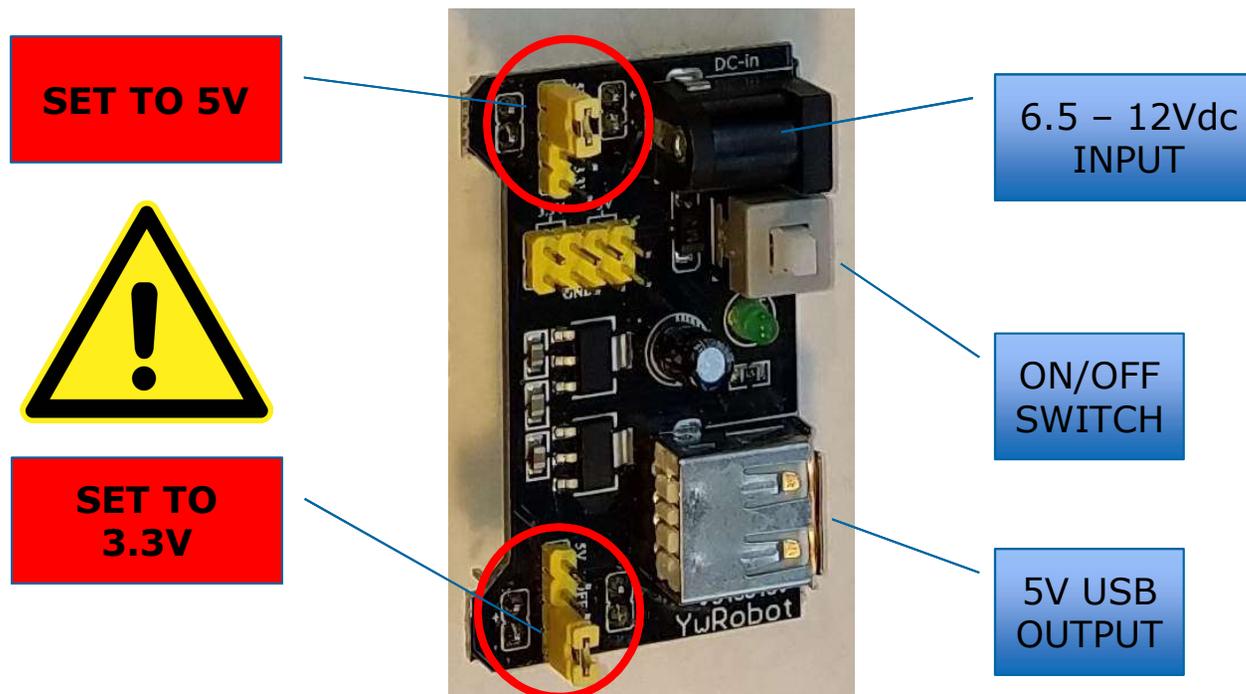
- ▶ Change **DURATION\_ON** to 500 (milliseconds)
- ▶ Change **DURATION\_OFF** to 500 (milliseconds)
- ▶ Upload the modified code to the NodeMCU
- ▶ Try other values and upload
- ▶ Return to some reasonable values and upload to prepare for the next hardware step

# Color Code Guide

- ▶ Electrons don't care but we do.
- ▶ Try to use one color for one signal, another color for a different signal.
- ▶ For our use, 120mm (4-1/2-in) jumpers are sufficient.

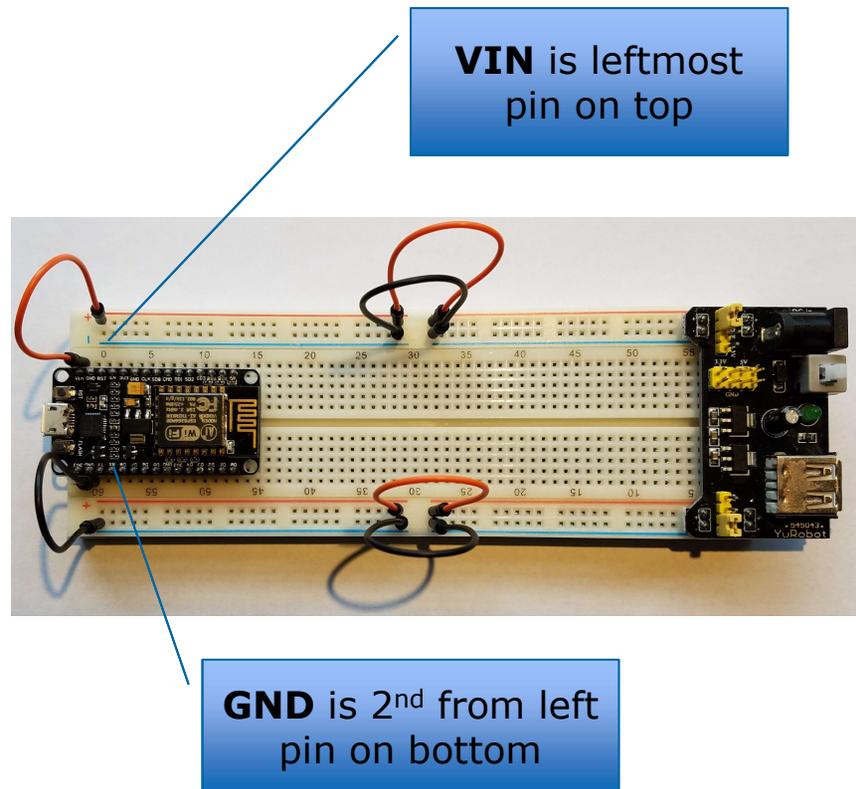
Signal	Quantity	Ideal	Plan B	Generic
+	7	RED	ORANGE	A BRIGHT color
-	6	BLACK	GREEN	A DARK COLOR
SCL	3	YELLOW	WHITE	A different BRIGHT color
SDA	3	BLUE	BLUE	A different DARK color
Other	3	Any	Any	Any

# Dual Voltage Regulator



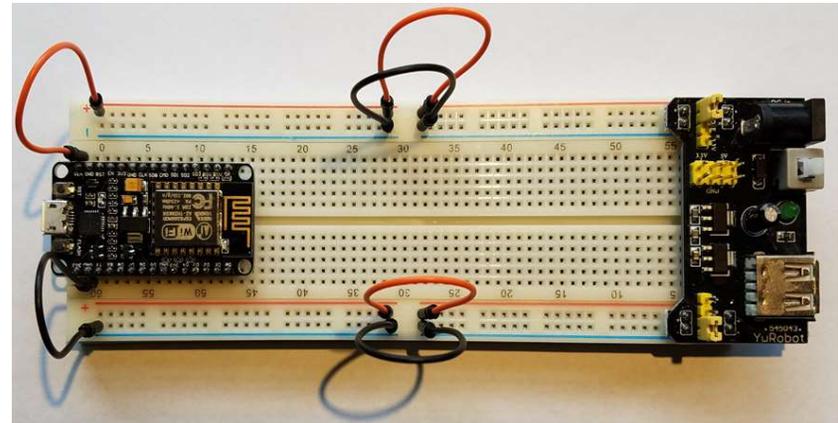
# Wire the power supply circuit

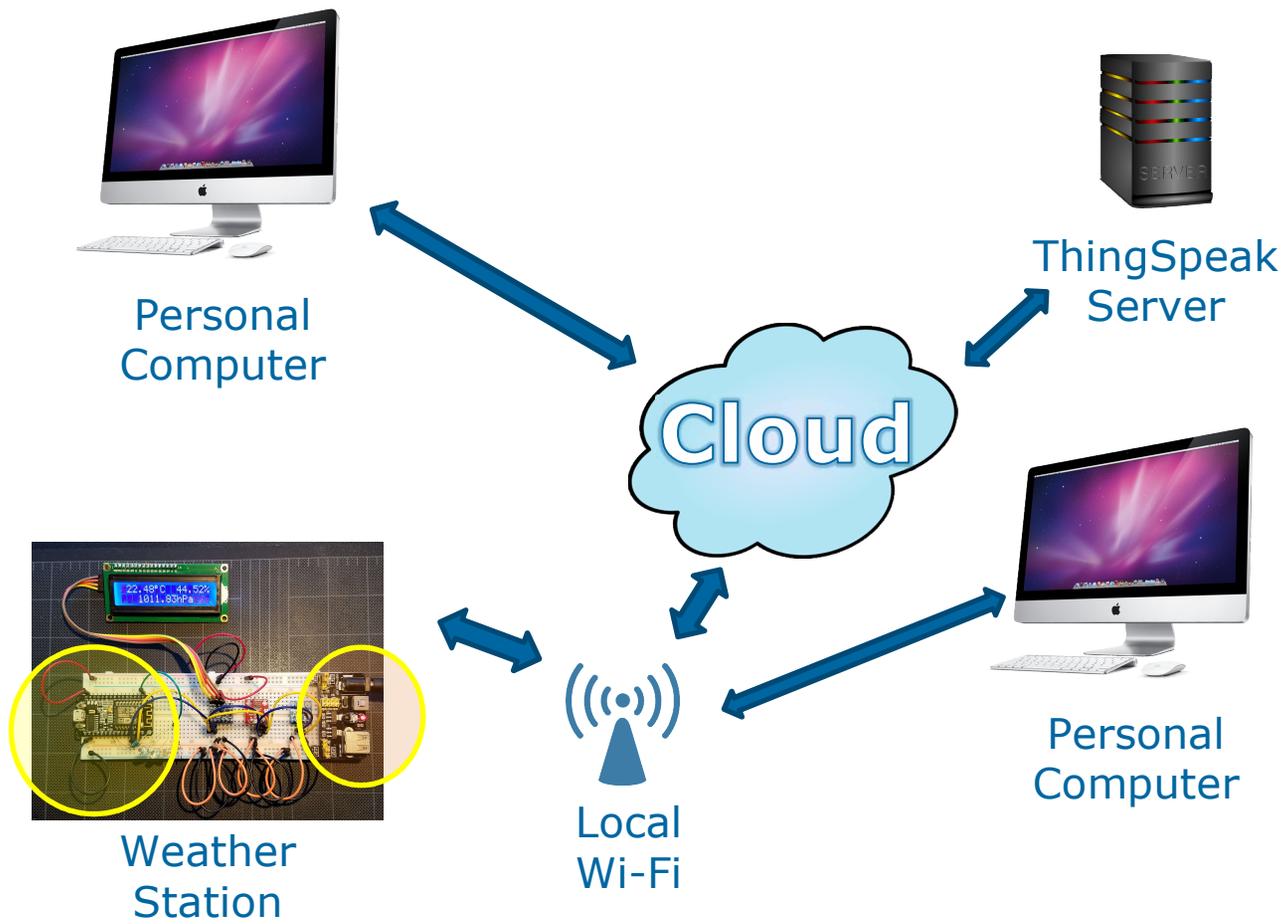
1. Unplug the USB cable from the NodeMCU.
2. Insert 4 jumpers to bridge the bus gaps.
3. Insert a jumper from the top + rail to **VIN**.
4. Insert a jumper from the bottom – rail to **GND**.



# Running Firmware

- ▶ Plug in power supply to wall socket.
- ▶ Plug in power cord to regulator.
- ▶ Turn on regulator.
- ▶ LED should blink.





# Questions?