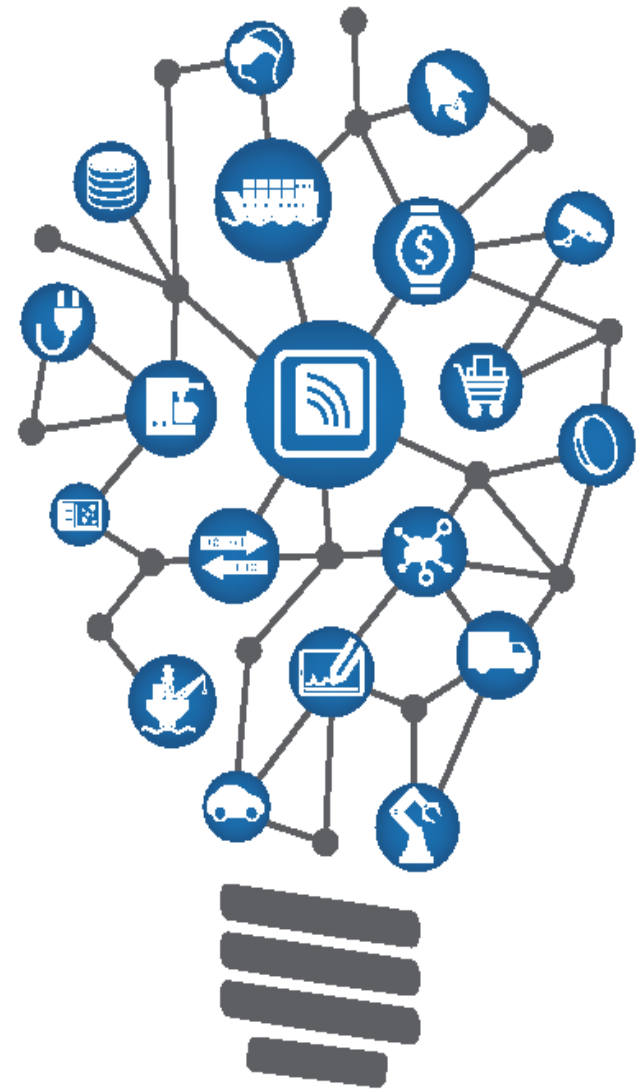


**NodeMCU**  
**ESP8266 dev board**  
**-Making Things Think**

# INTERNET OF THINGS



Internet of Things is connecting information, people, and things



“EVERY ANIMATE AND INANIMATE OBJECT ON EARTH  
WILL SOON BE **GENERATING DATA**, INCLUDING OUR  
HOMES, OUR CARS, AND YES, EVEN OUR BODIES.”

—Anthony D. Williams

“ ”

**IN A  
HUMBLE  
STATE,  
YOU LEARN  
BETTER**

**JOHN DOONER**





Always be  
willing to learn

# Development Boards



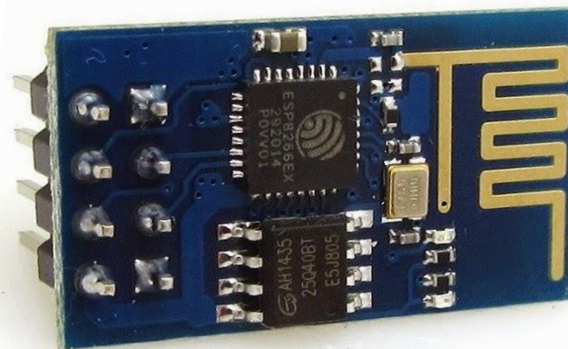
**Arduino Ethernet shield**



**Raspberry pi**



**Intel Edison**

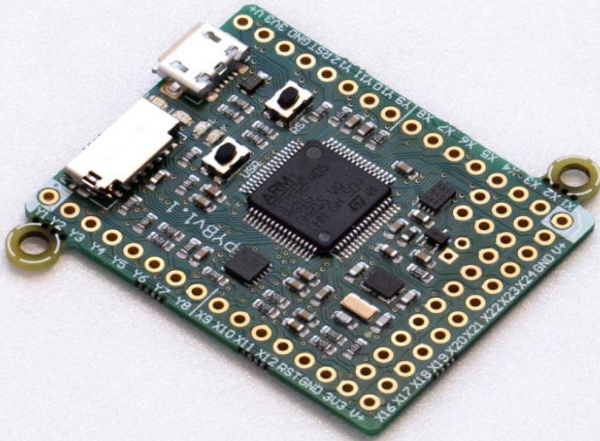


**ESP8266 WiFi Module**



**ESP8266 -12 node MCU**

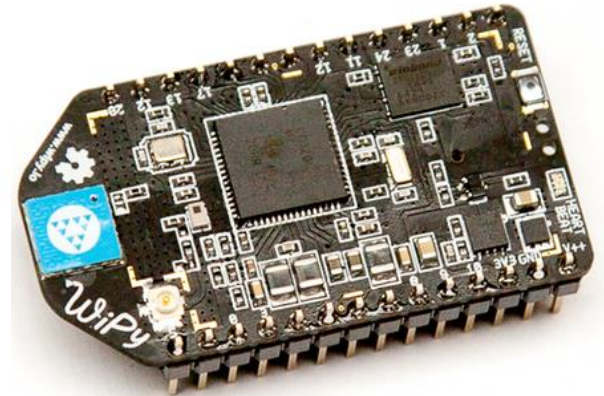
# Development Boards



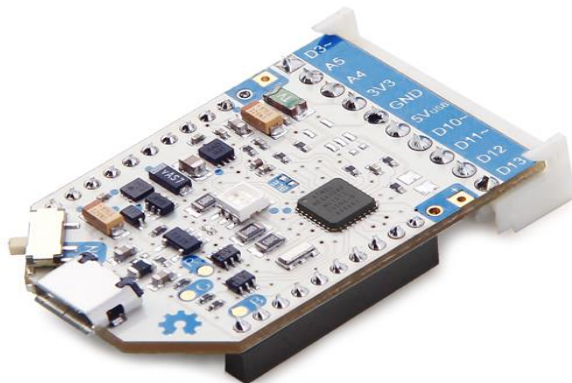
**Micropython**



**Partical Photon**

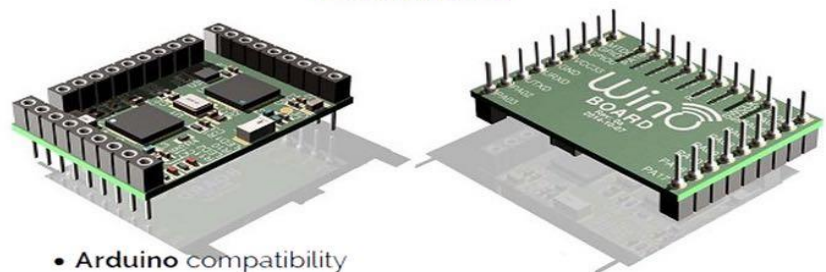


**WiPy Board**



**Air Board**

**Wino**  
BOARD



- Arduino compatibility
- Built-in WIFI

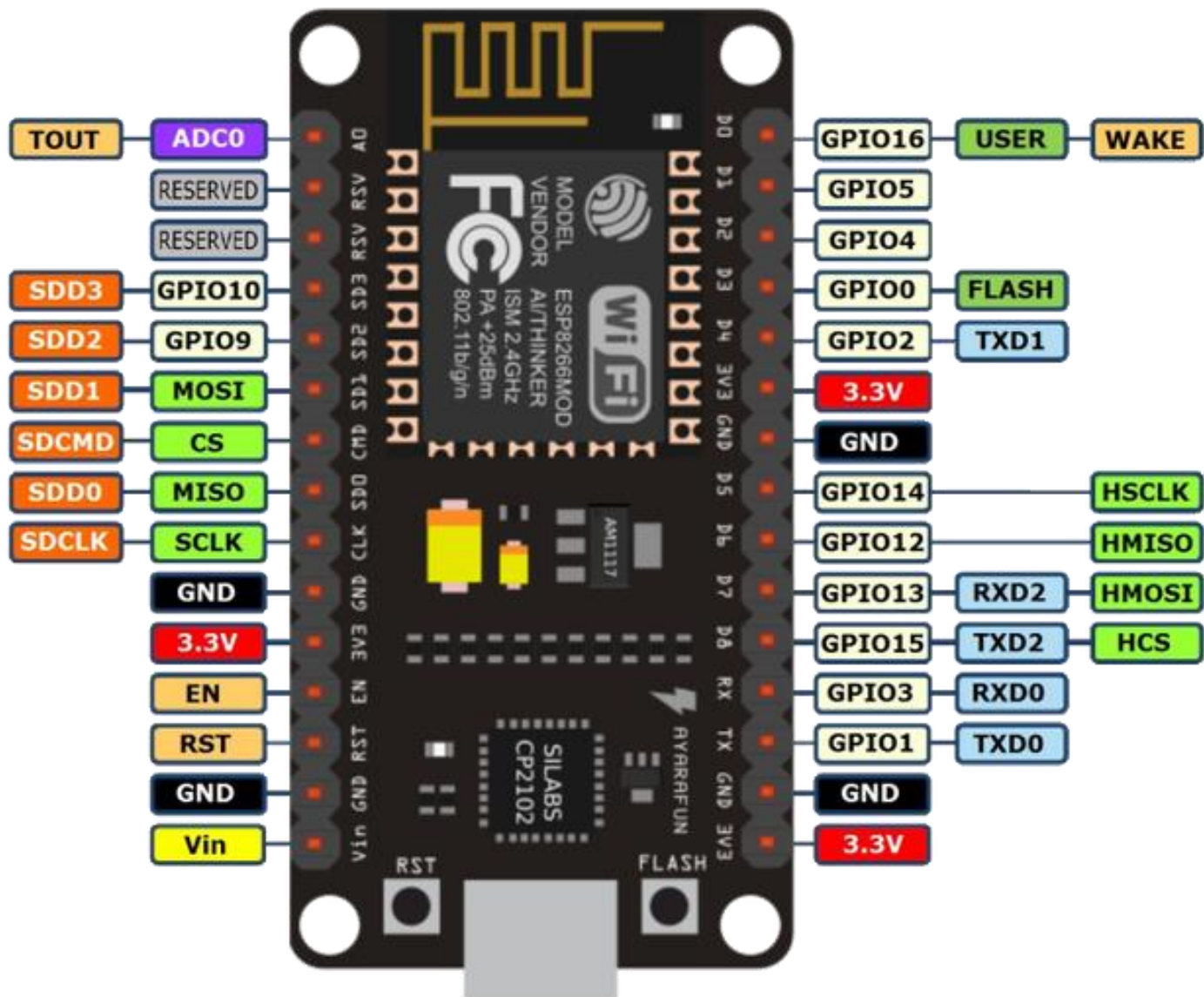


# Introduction to ESP8266

SMALL  
POWERFUL  
LOW POWER  
STANDALONE  
IOT







# Arduino

The name is an Italian masculine name, *meaning* "strong friend"



I hear and I forget. I see and I remember.  
I do and I understand.

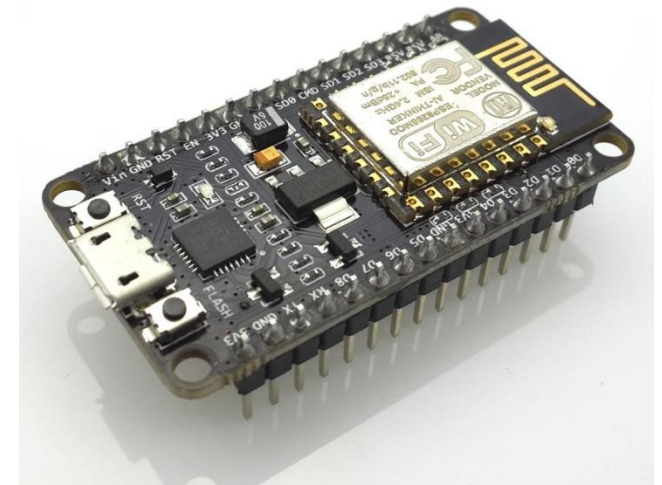
-Confucius



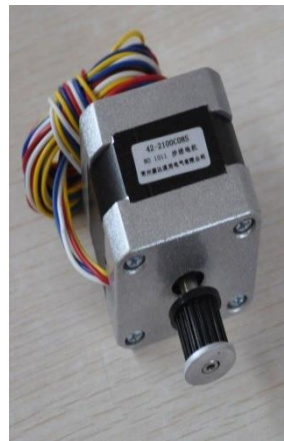
**'Reform, Perform And Transform Is Our Mantra'**

**-Shri Narendra Modi**

# What do we need?

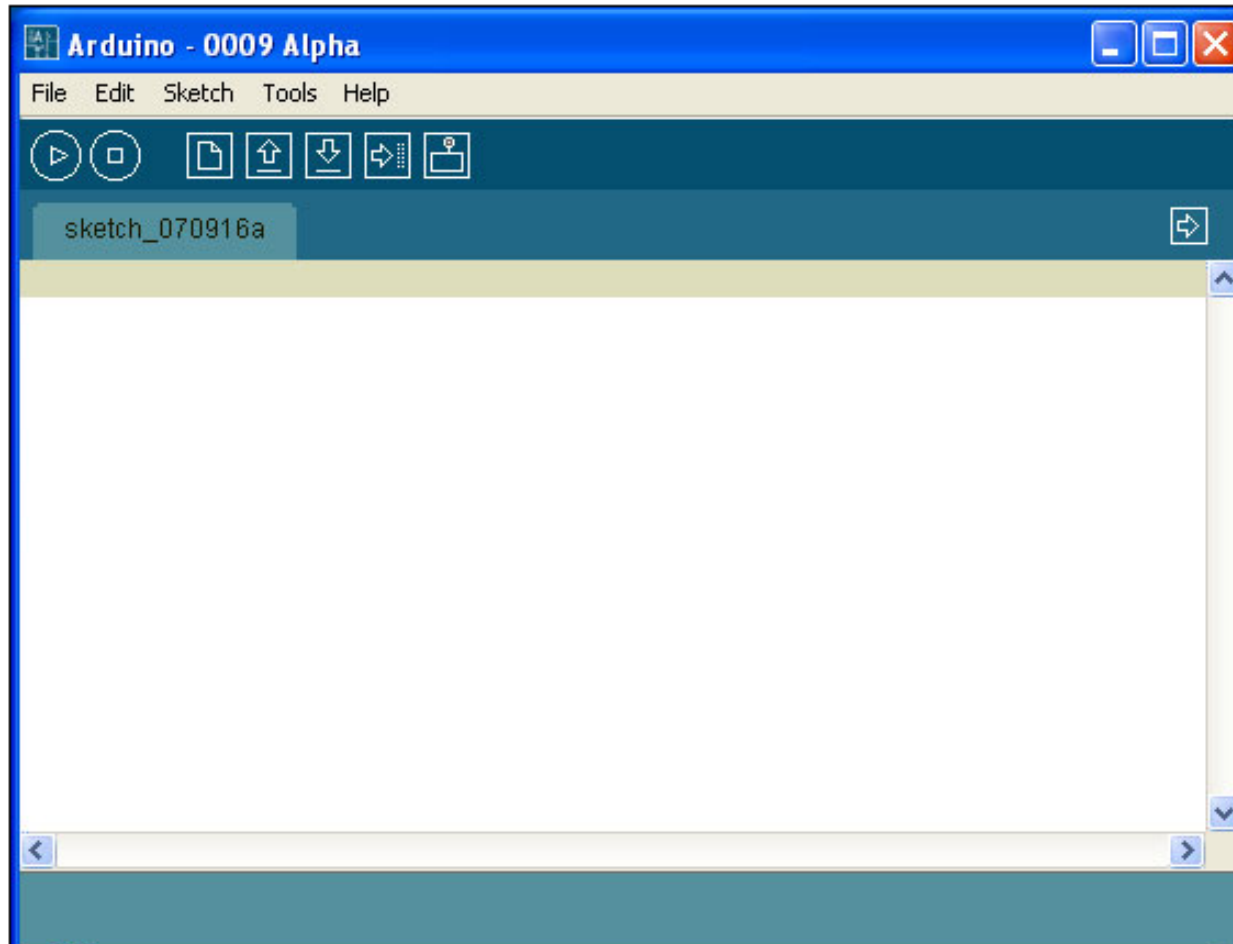


# Nodemcu - Periphery





# Download and install the Arduino Software



# Programming an Arduino

- The Arduino software consists of a development environment (IDE) and the core libraries.
- The IDE is written in Java and based on the processing environment.
- The core libraries are written in C and C++ and compiled using avr-gcc compiler.



The screenshot shows the Arduino IDE interface. The menu bar includes File, Edit, Sketch, Tools, and Help. The toolbar contains icons for opening, saving, and running sketches. The sketch is titled 'Blink'. The code in the editor is as follows:

```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.

  This example code is in the public domain.
  */

void setup() {
  // initialize the digital pin as an output.
  // Pin 13 has an LED connected on most Arduino boards:
  pinMode(13, OUTPUT);
}

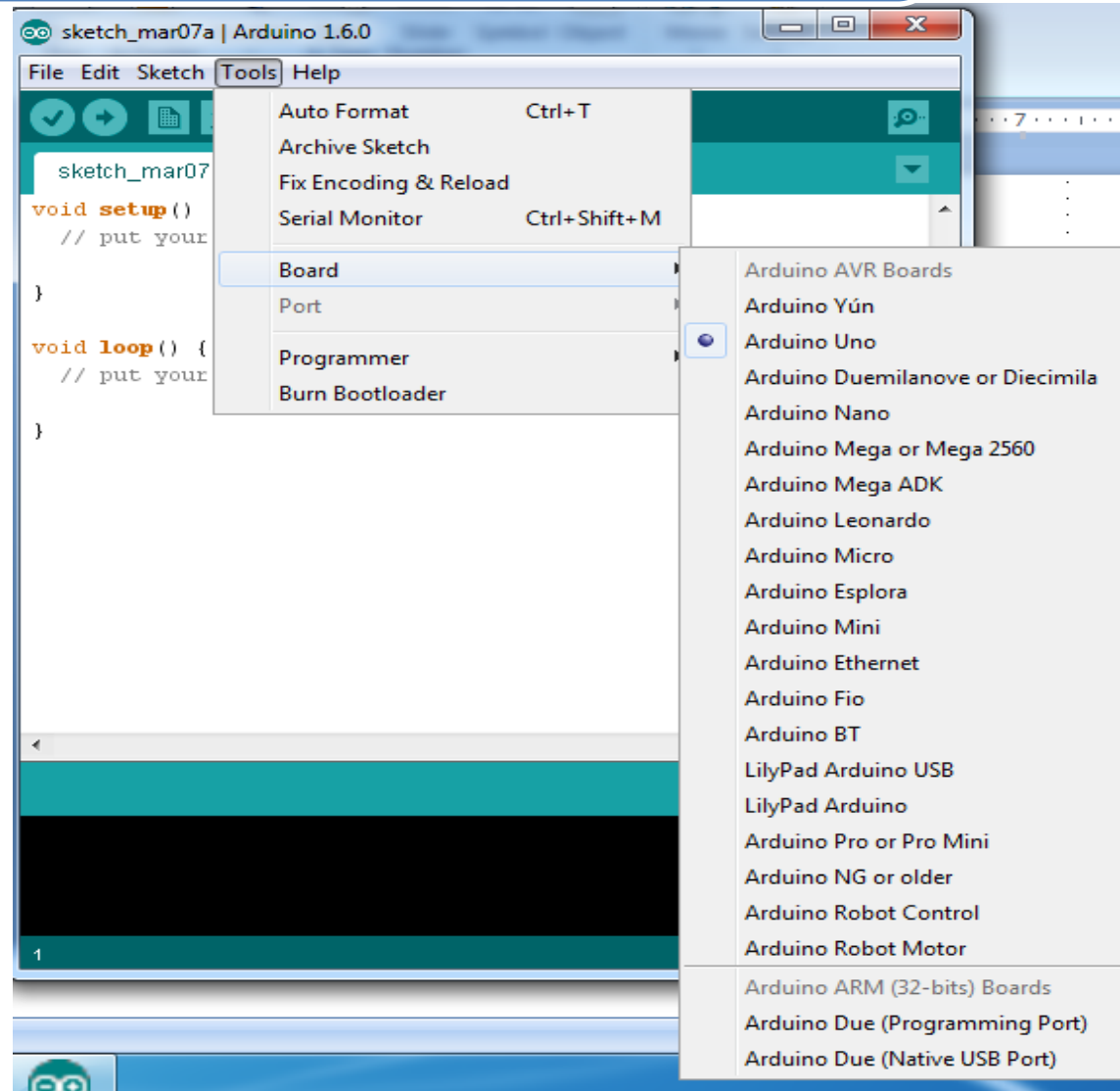
void loop() {
  digitalWrite(13, HIGH); // set the LED on
  delay(1000);            // wait for a second
  digitalWrite(13, LOW);  // set the LED off
  delay(1000);            // wait for a second
}
```

Below the code editor, a status bar indicates 'Done compiling.' and 'Binary sketch size: 1026 bytes (of a 32256 byte maximum)'. The bottom status bar shows '1' and 'Arduino Uno on COM40'.



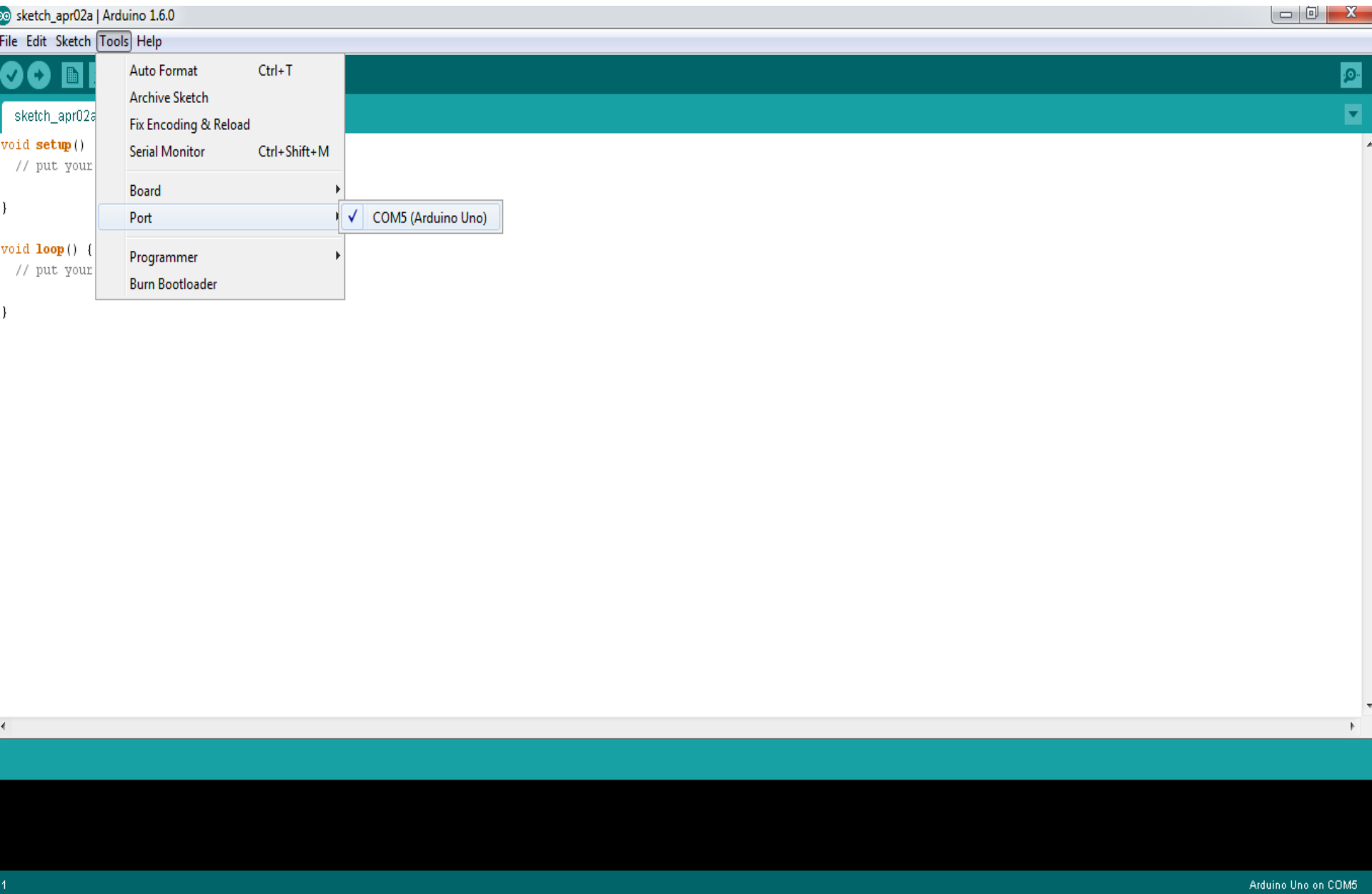
# FIRST PROGRAM ON IDE

- Click on arduino (∞) IDE
- Tools –Select board
- Serial port –COM port

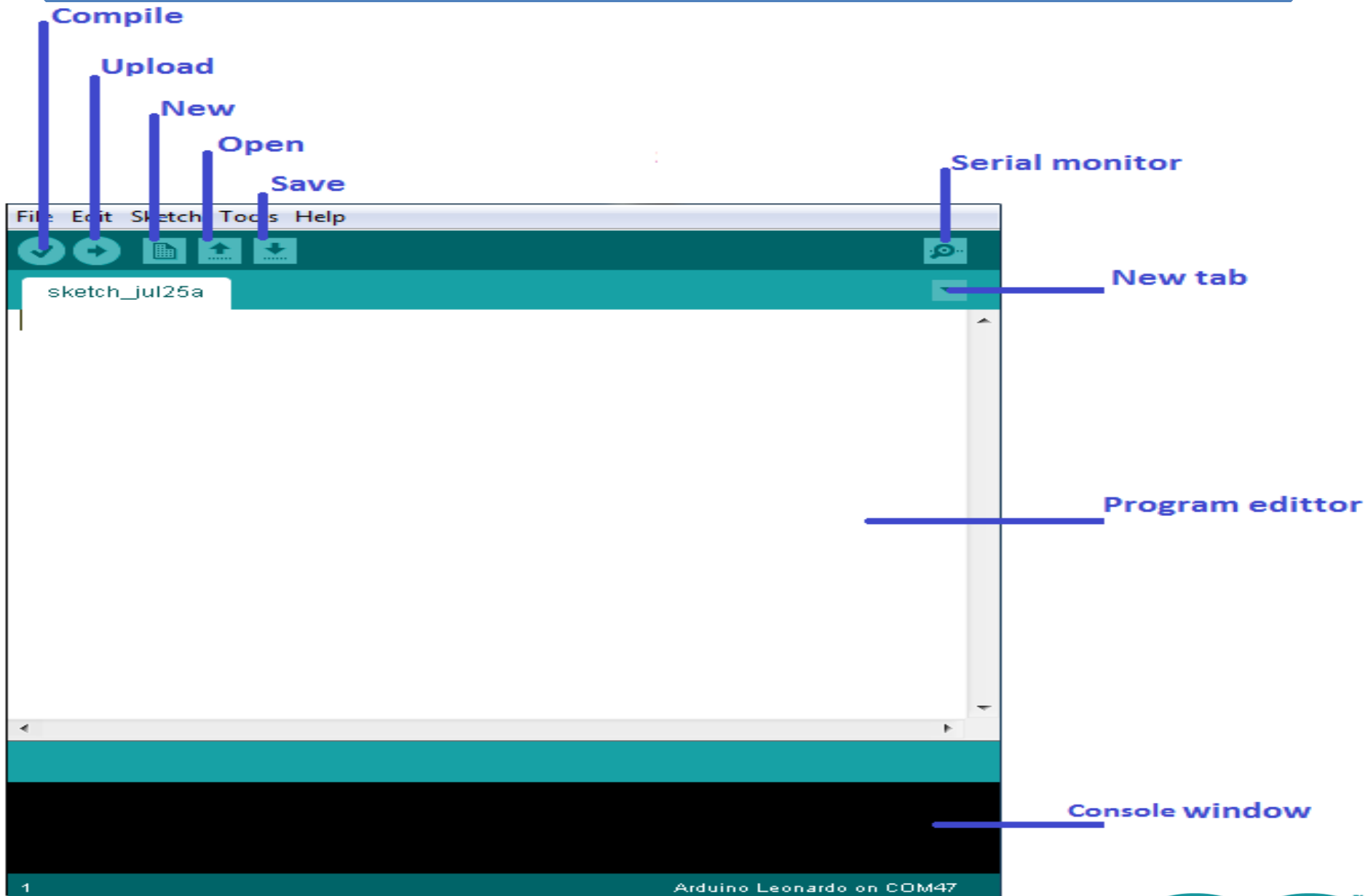




# Configure the Serial Port



# Arduino environment



# Structure of Arduino Uno code

```
sketch_mar12a
```

```
void setup() {  
    // put your setup code here, to run once:  
  
}  
  
void loop() {  
    // put your main code here, to run repeatedly:  
  
}
```





## Bit Rate Vs Baud Rate

*Bit rate is the number of bits per second. Baud rate is the number of signal units (symbols) per second. Baud rate is less than or equal to the bit rate.*

Bit

Baud rate =  $N$

Bit rate =  $N$

0	0	1	0	1	0	0	0	1	0	1	0	1	0	1	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Dibit

Baud rate =  $N$

Bit rate =  $2N$

0	0	1	0	1	0	0	0	1	0	1	0	1	1	1	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Tribit

Baud rate =  $N$

Bit rate =  $3N$

0	0	1	0	1	0	0	0	1	0	1	0	1	1	1	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Quadbit

Baud rate =  $N$

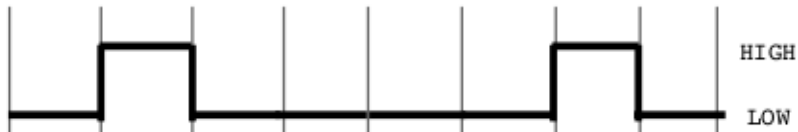
Bit rate =  $4N$

0	0	1	0	1	0	0	0	1	0	1	0	1	1	1	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

# Serial Communications

- “Serial” because data is broken down into bits, each sent one after the other down a single wire.

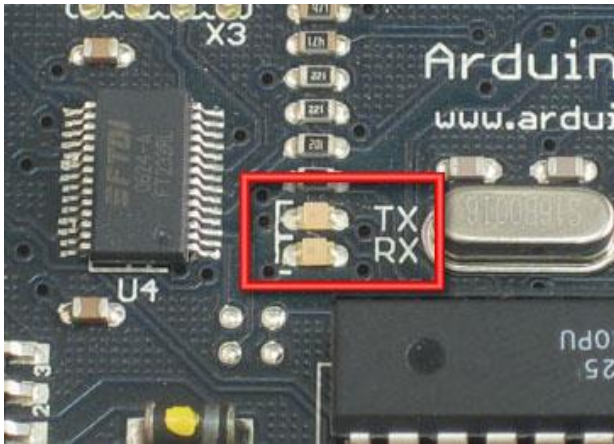
- The single ASCII character ‘B’ is sent as:

‘B’ = 0 1 0 0 0 0 1 0  
= L H L L L L H L  
=  HIGH  
LOW

The diagram shows a digital signal for the character 'B'. It consists of eight bits: 0, 1, 0, 0, 0, 0, 1, 0. Each bit is represented by a horizontal line segment. The signal is LOW (0) for the first, third, fourth, fifth, sixth, and eighth bits, and HIGH (1) for the second and seventh bits. The signal transitions from LOW to HIGH at the start of the second bit and from HIGH to LOW at the end of the seventh bit. The signal is labeled 'HIGH' and 'LOW' on the right side.

- Toggle a pin to send data, just like blinking an LED
- You could implement sending serial data with `digitalWrite()` and `delay()`
- A single data wire needed to send data. One other to receive.

# Serial Communication



- **Compiling** turns your program into binary data (ones and zeros)
- **Uploading** sends the bits through USB cable to the Arduino
- The two LEDs near the USB connector blink when data is transmitted
  - **RX** blinks when the Arduino is receiving data
  - **TX** blinks when the Arduino is transmitting data

# BIG 6 CONCEPTS



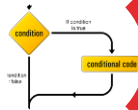
digitalWrite()



analogWrite()



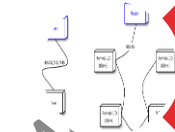
digitalRead()



if() statements / Boolean



analogRead()



Serial communication



Delay()



# pinMode()

Syntax:

**pinMode(pin, mode)**

Parameters

pin: the number of the pin whose mode you wish to set

mode: INPUT, OUTPUT



# commands to know...

- **pinMode** (pin, INPUT/OUTPUT) ;
- ex: **pinMode** (13, OUTPUT) ;
- **digitalWrite** (pin, HIGH/LOW) ;
- ex: **digitalWrite** (13, HIGH) ;
- **delay** (time\_ms) ;
- ex: **delay** (2500) ; // delay of 2.5 sec.
- **// NOTE: -> commands are CASE-sensitive**

# Error

- ◆ **avrdude: stk500\_getsync(): not in sync: resp=0x00**

```
Uploading to I/O Board...  
Binary sketch size: 1108 bytes (of a 14336 byte maximum)  
  
avrdude: stk500_getsync(): not in sync: resp=0x00  
avrdude: stk500_disable(): protocol error, expect=0x14, resp=0x51  
  
20
```

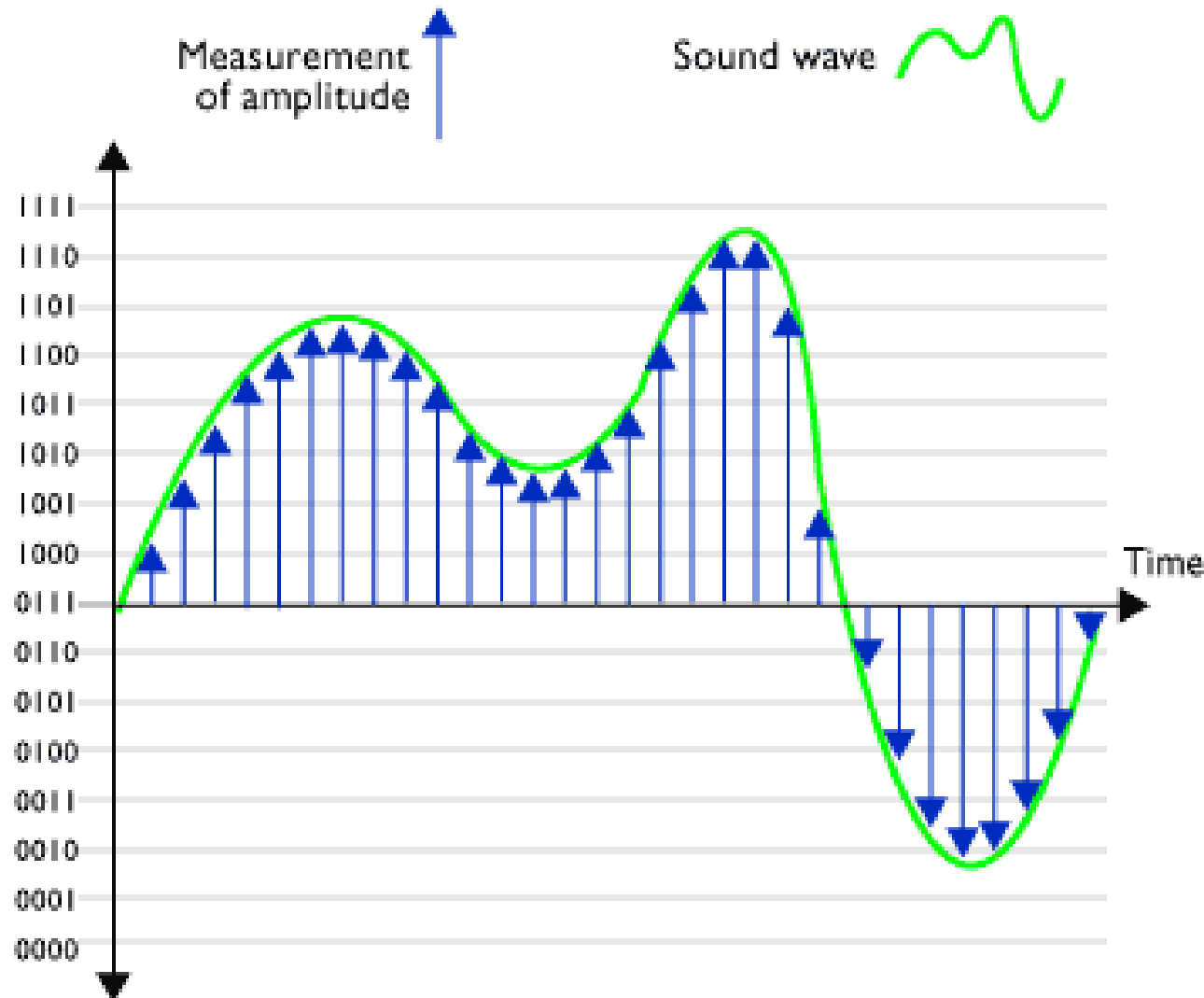
- ◆ **can't open device "COM10": The system cannot find the file specified**

```
Uploading to I/O Board...  
Binary sketch size: 1108 bytes (of a 14336 byte maximum)  
  
avrdude: ser_open(): can't open device "COM21": The system cannot find the  
file specified.  
  
22
```

# analogRead()(pin:A0-A5)

- Reads the value from the specified analog pin. The Arduino board contains a 6 channel.
- 10-bit ADC. This means that it will map input voltages between 0 and 5 volts into integer values between 0 and 1023.
- It takes about 100 microseconds to read an analog input
  - Syntax
    - `analogRead(pin)`
    - **pin:** the number of the analog input pin to read from (0 to 5 on most boards)

- Resolution of ADC..?
- Effect of resolution



Each measurement is assigned a number (byte) according to its amplitude. The end result is a file comprising a string of bytes, eg ...  
 1001 1110 0001 1010 0111 0100 1111 1101 etc



# digitalRead() (pin 0-13)

- Description
- Reads the value from a specified digital pin, either HIGH or LOW.

## Syntax

**digitalRead(pin)**

pin: the number of the digital pin you want to read (*int*)

## Returns

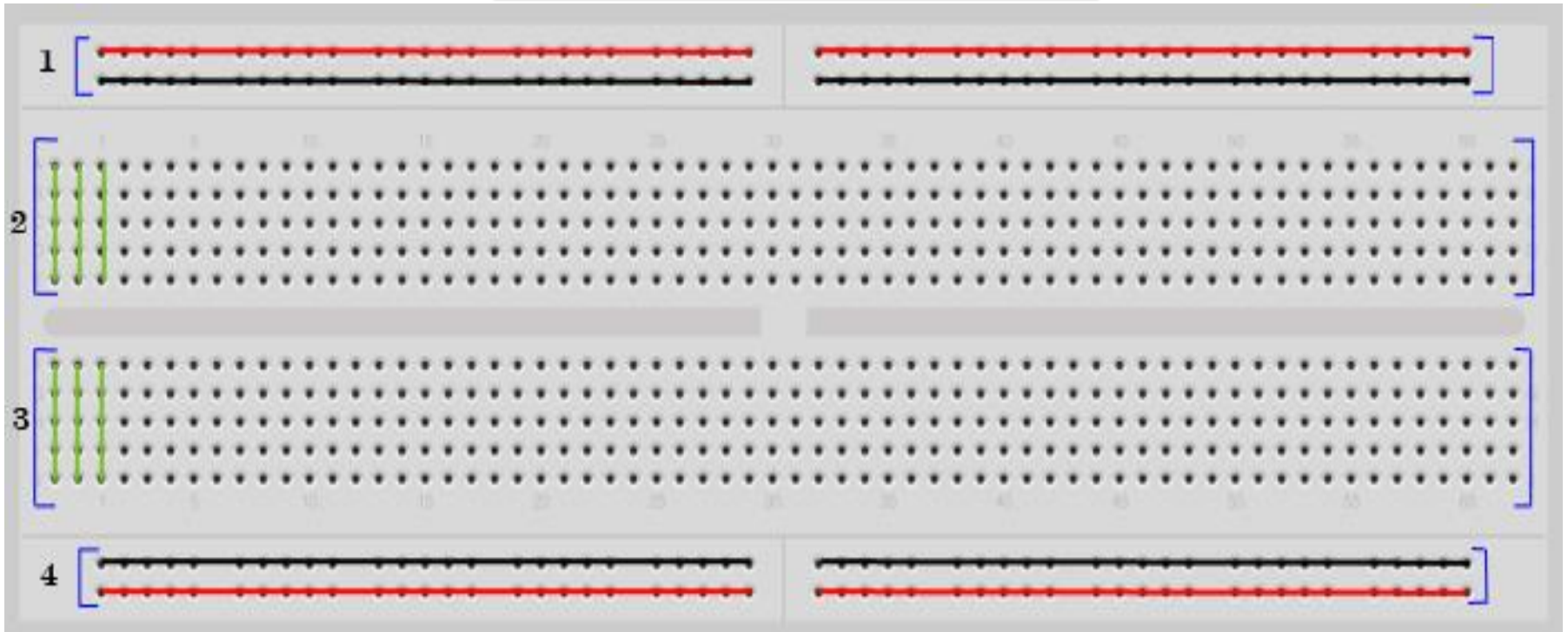
HIGH or LOW



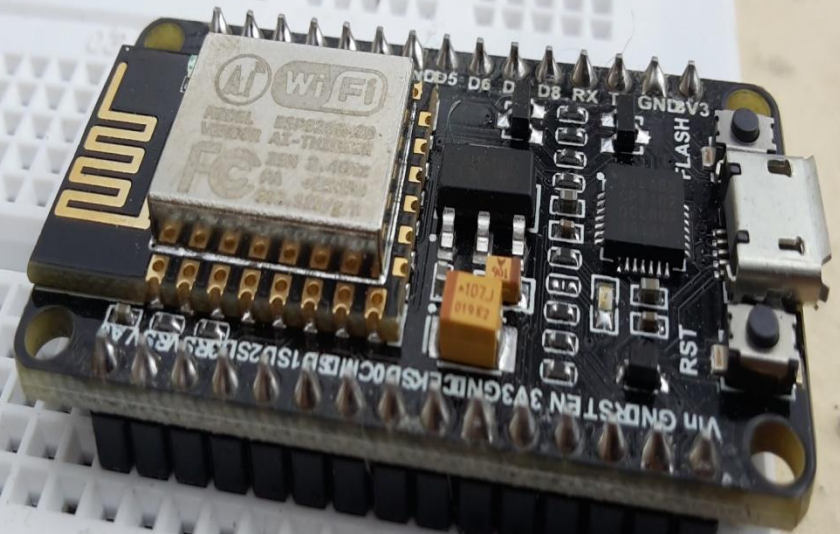
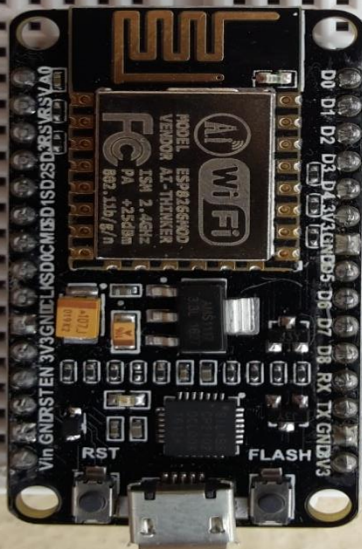
```
int analogPin = 3; connected to analog pin 3  
                // outside leads to ground and +5V  
int val = 0;      // variable to store  
the value read  
  
void setup()  
{  
    Serial.begin(9600);           // setup  
serial  
}  
  
void loop()  
{  
    val = analogRead(analogPin); // read  
the input pin  
    Serial.println(val);         // debug  
value  
}
```



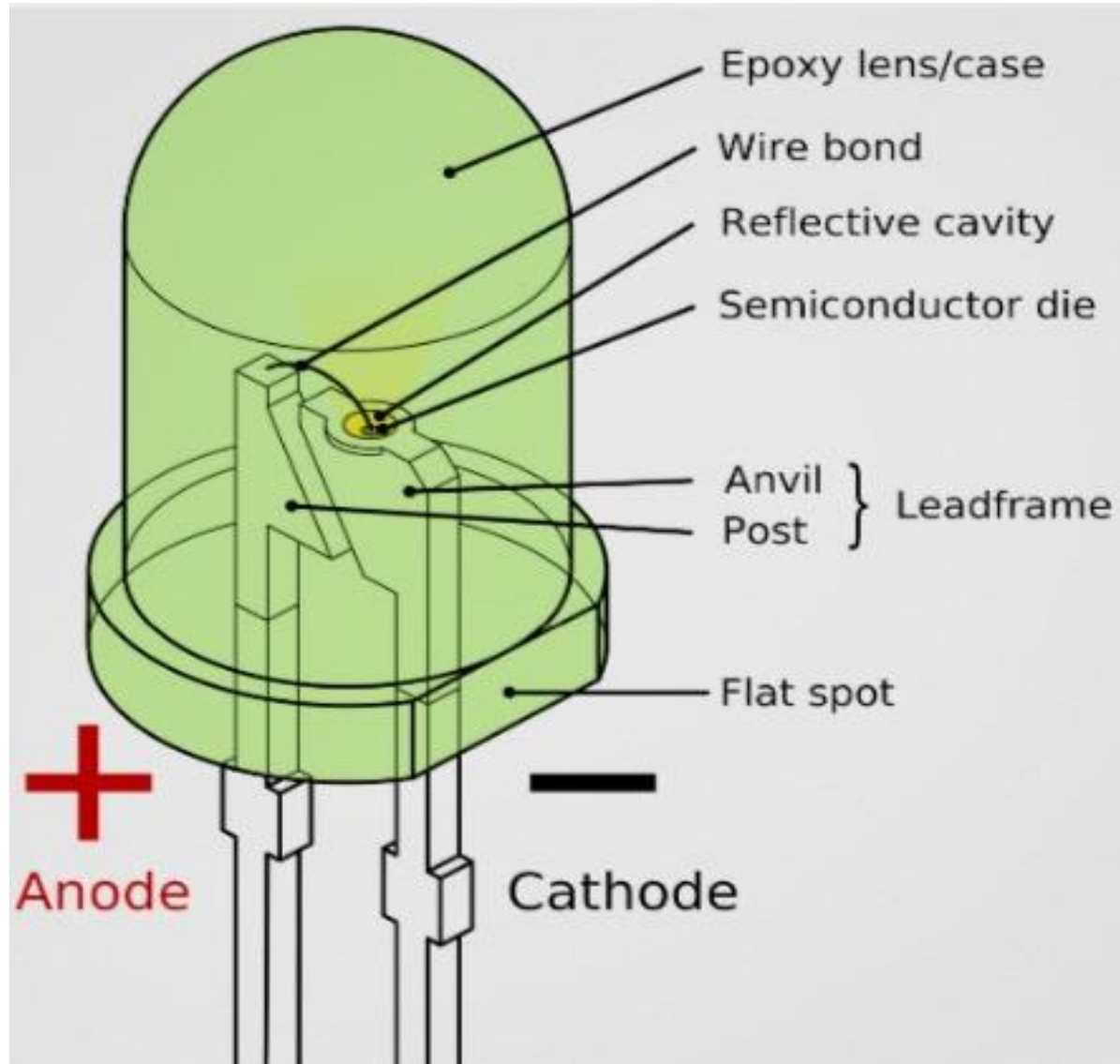
# BREAD BOARD



# Nodemcu assembling on Breadboard



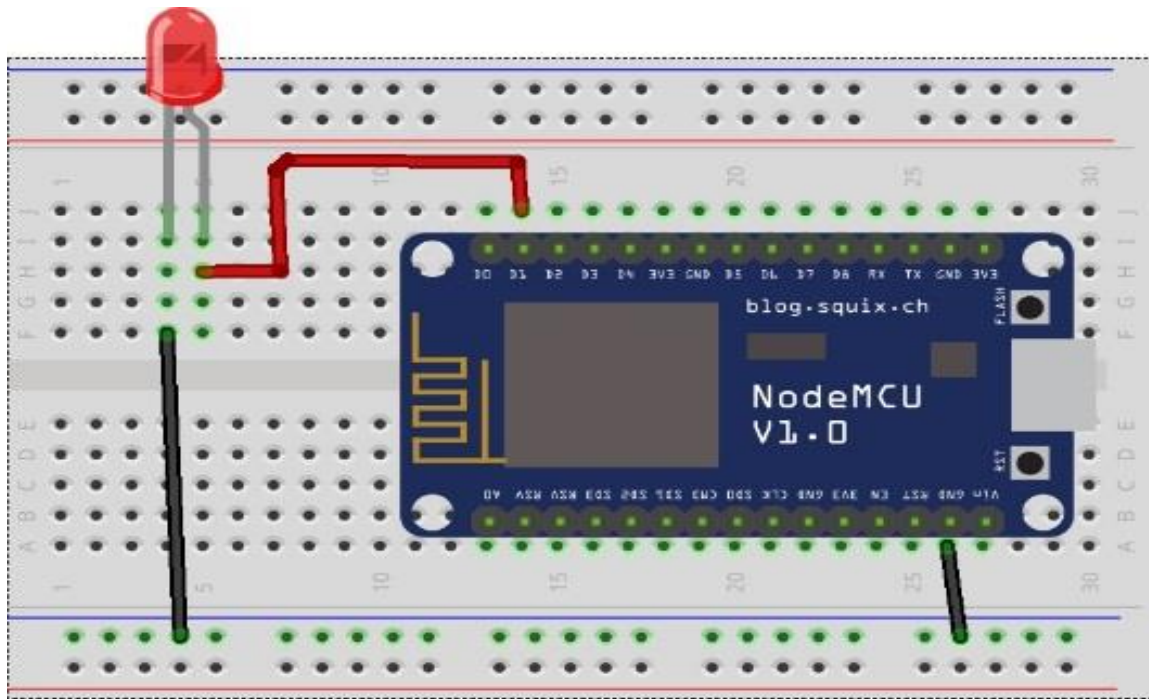
# LED (Light emitting diode)





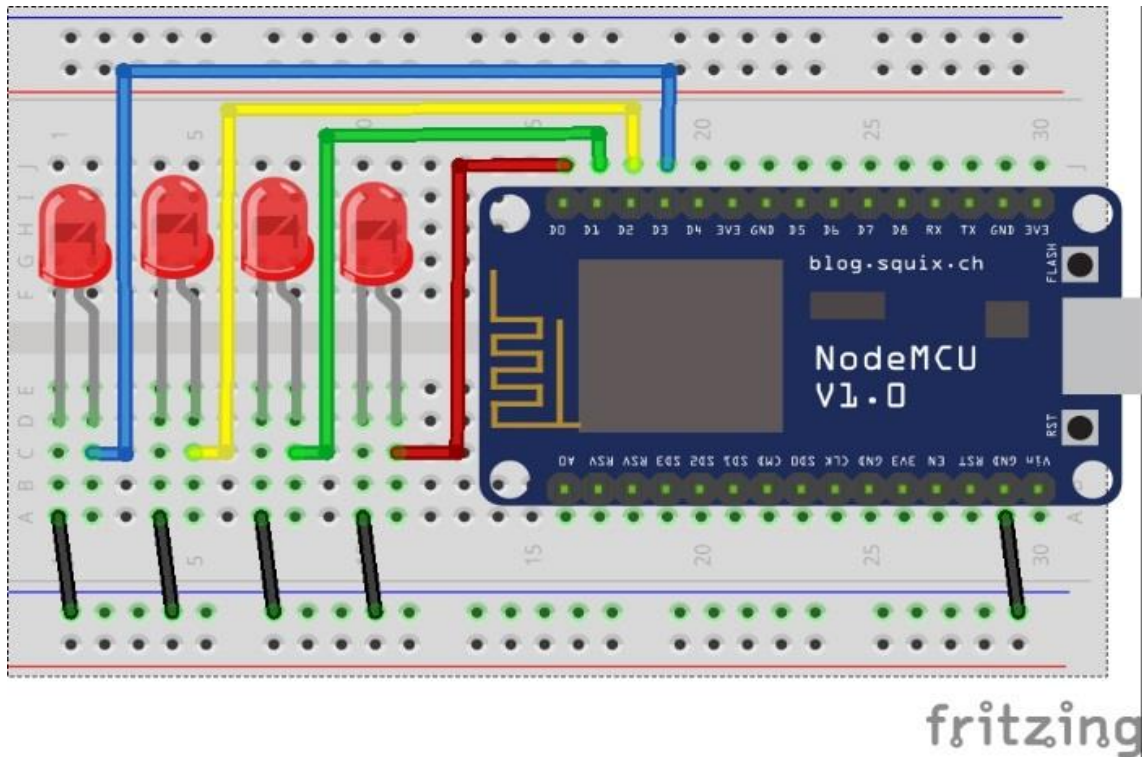
# BLINKING OF AN LED

# Interfacing with NodeMcu

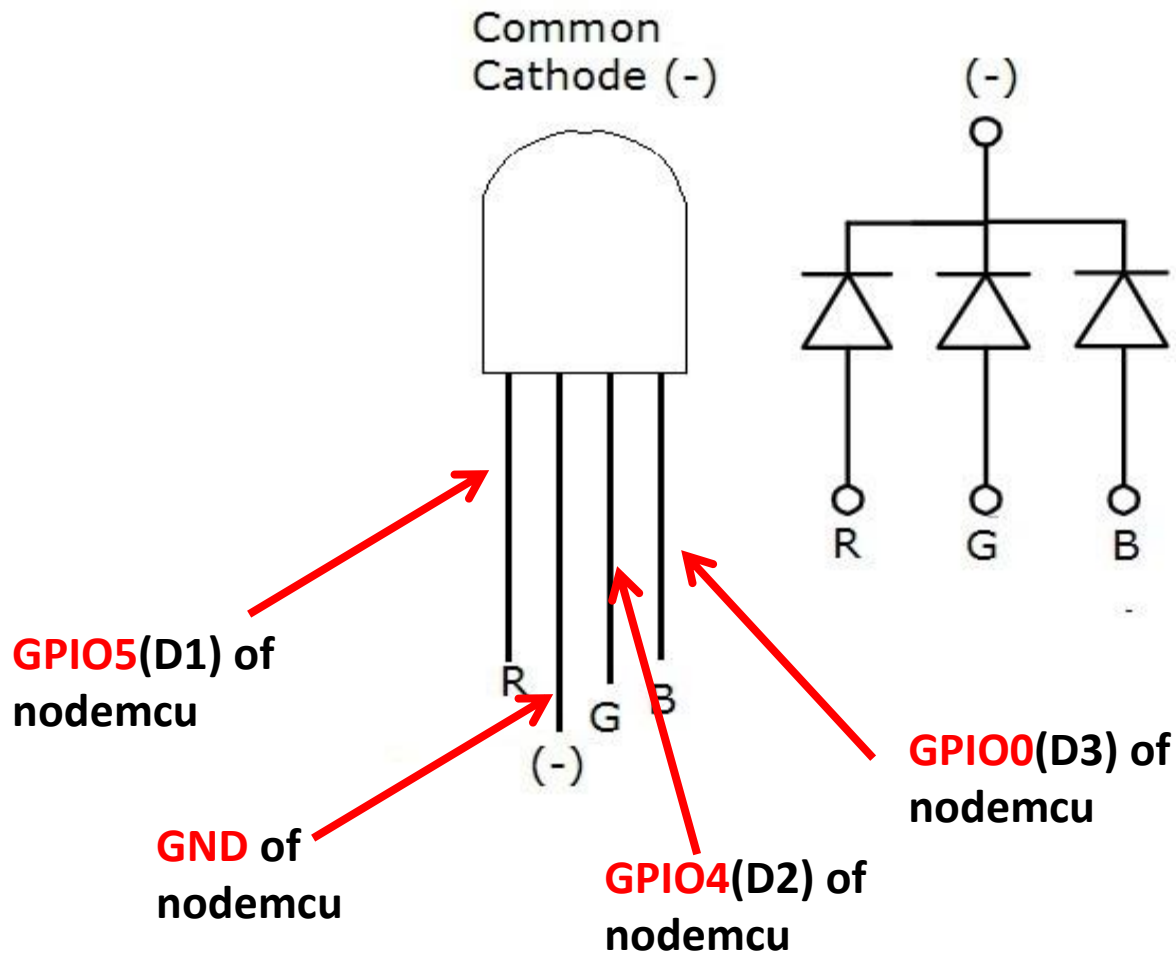


fritzing

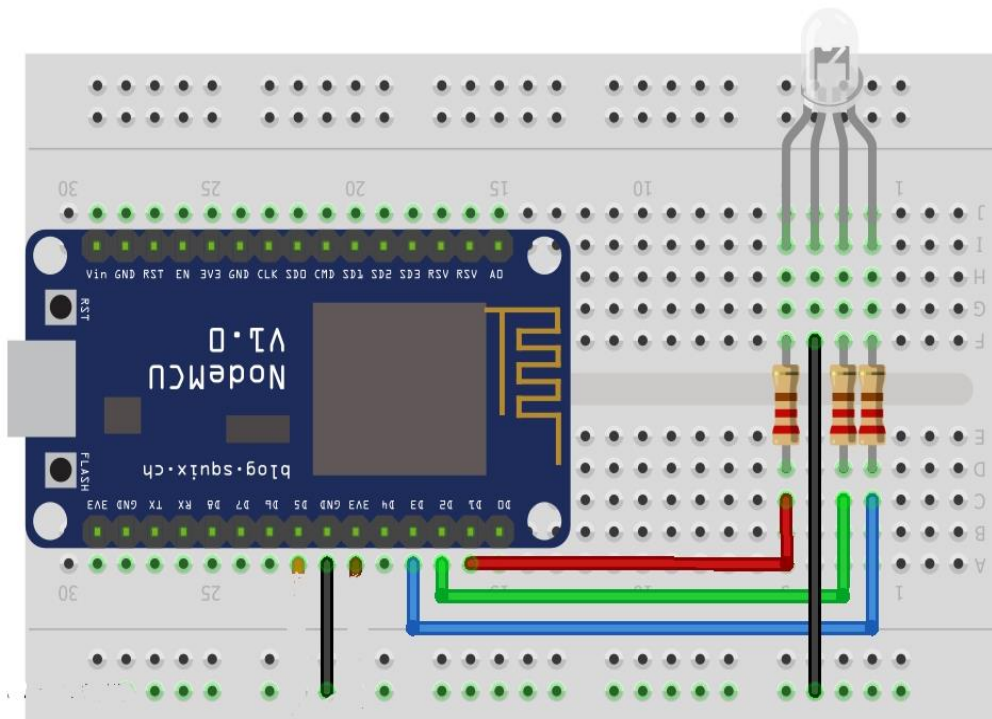
# LED Pattern using NodeMcu



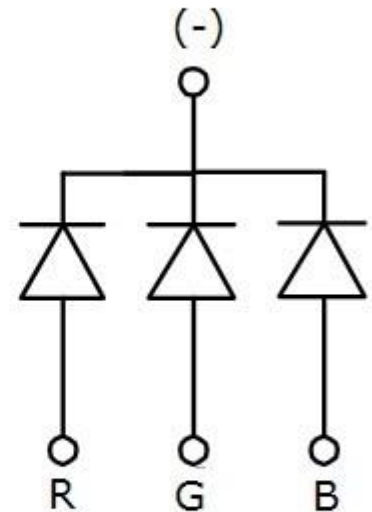
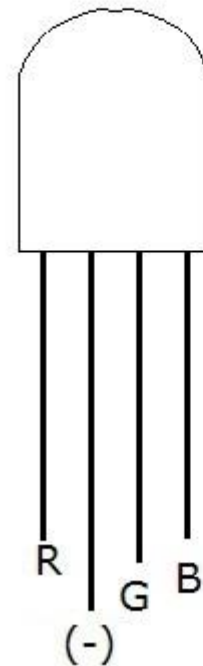
# RGB led Pinouts



# RGB led control



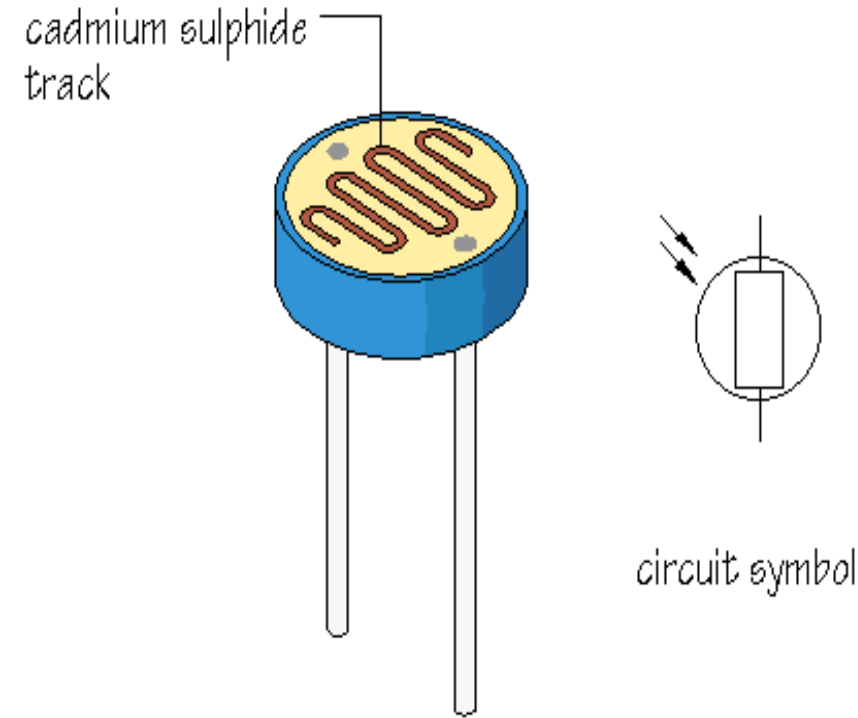
Common Cathode (-)





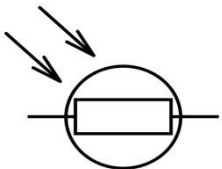
# LDR Sensor With NodeMcu

# LDR(LIGHT DEPENDENT RESISTOR)



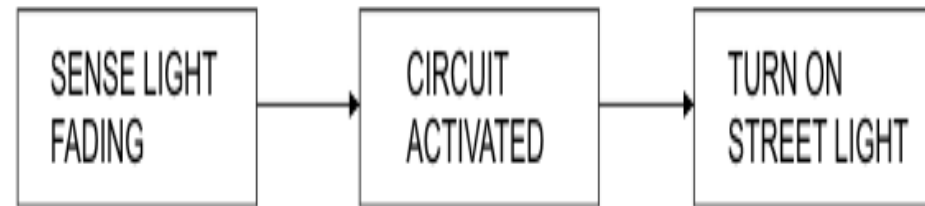
- **LDR stands for Light dependant resistor. An LDR is usually made of a semiconductor material(Normally Silicon) doped with a small percentage of a valence 5 material (commonly Arsenic), to make it an "N" material.**
- **Another word for LDR is photoresistor.**
- **The resistance of LDR decreases with increase in the intensity of light. An LDR works in the similar manner as any other analog device would work.**

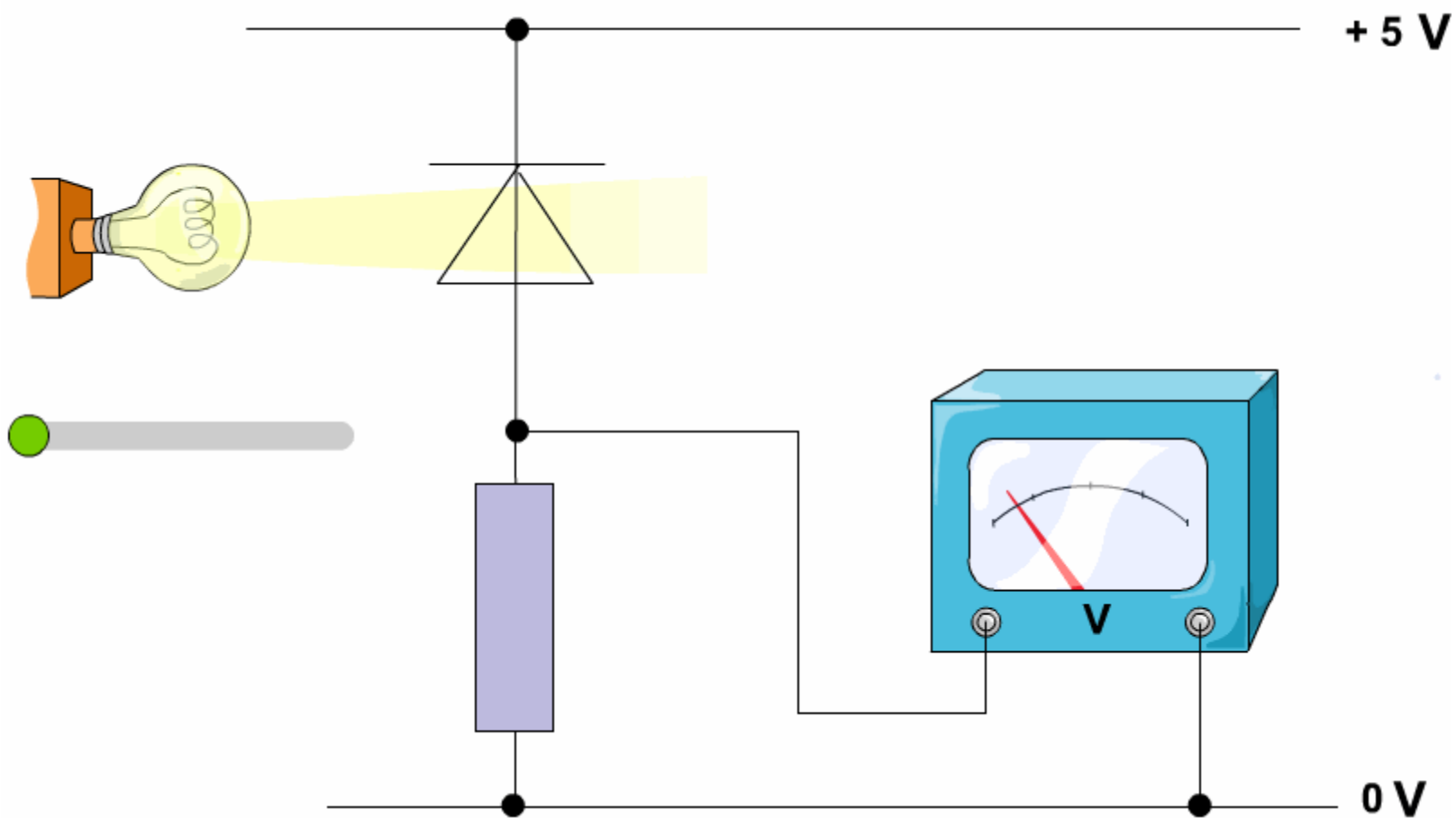
**Symbol of LDR:**



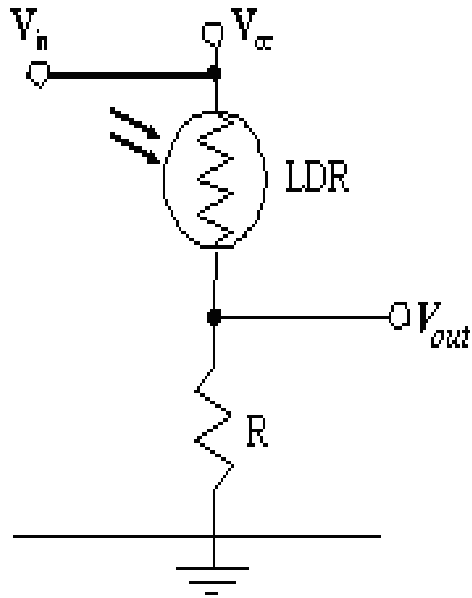
# APPLICATIONS

- They can be used to respond to events such as the transition from daytime to night-time (and vice versa) for home automation
- Gardening applications, and are often used to control street lighting.

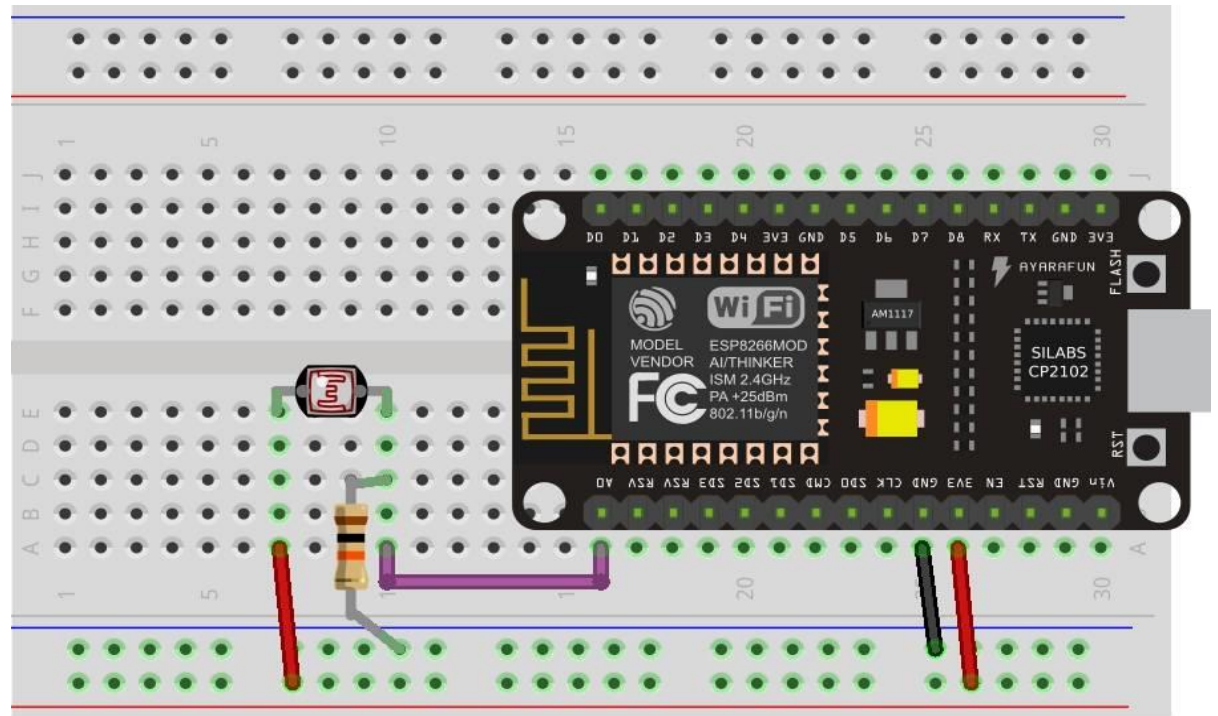




# INTERFACING LDR WITH NodeMcu



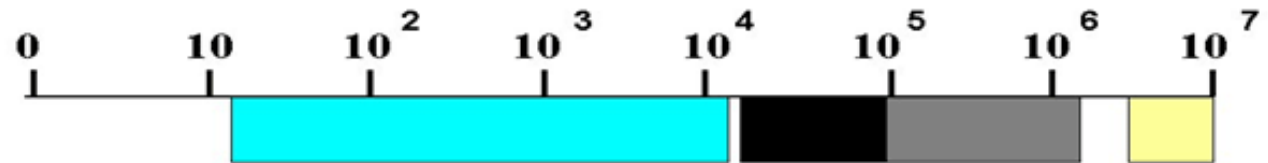
Darkness  $\Rightarrow$  LDR high  $\Rightarrow$   $V_{out}$  is low  
Bright  $\Rightarrow$  LDR low  $\Rightarrow$   $V_{out}$  is high



fritzing

# Ultrasonic

## The Frequency Ranges of the Sound



Human hearing



16Hz - 18kHz

Conventional power ultrasound



20kHz - 100kHz

Extended range for sonochemistry

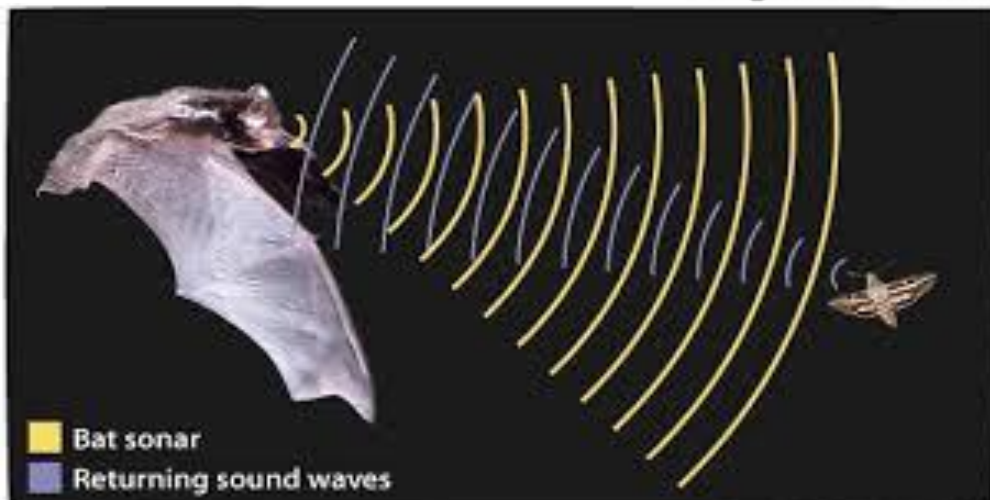


20kHz - 2MHz

Diagnostic ultrasound



5MHz - 10MHz

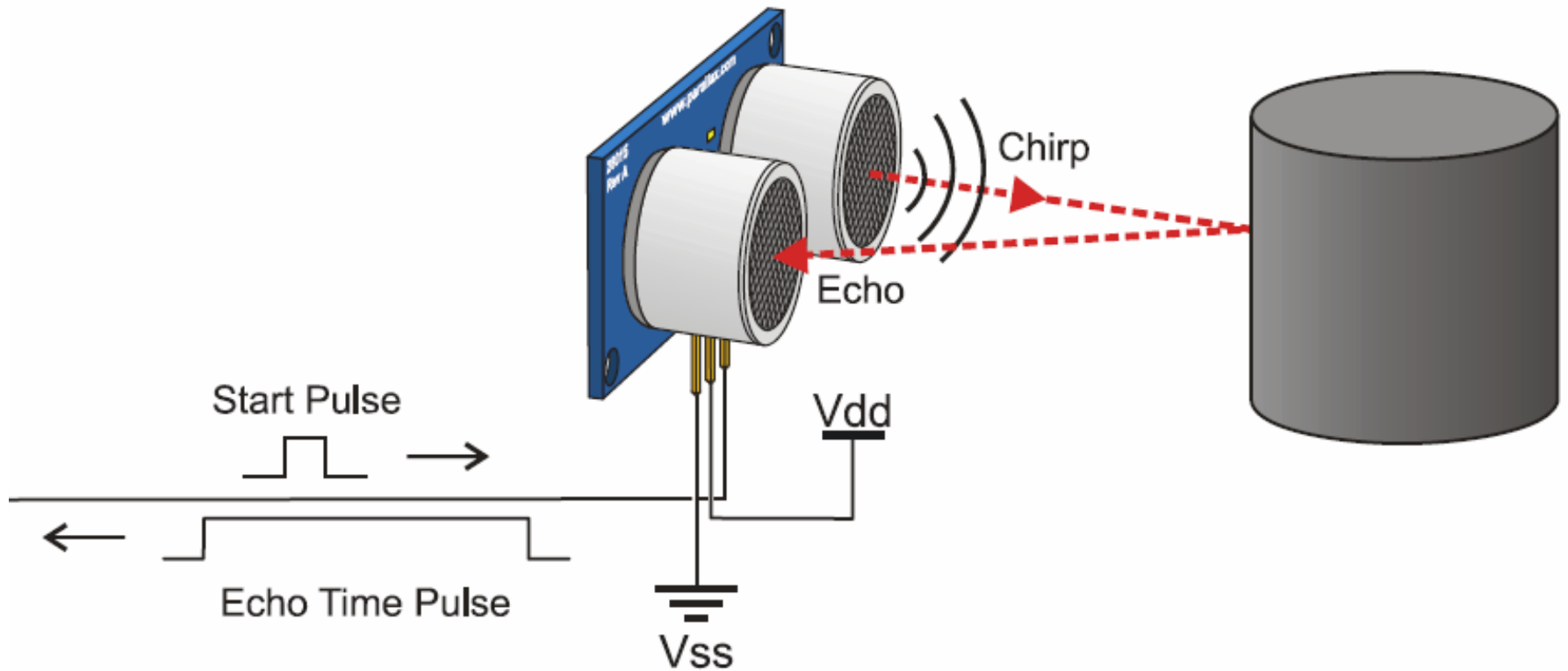




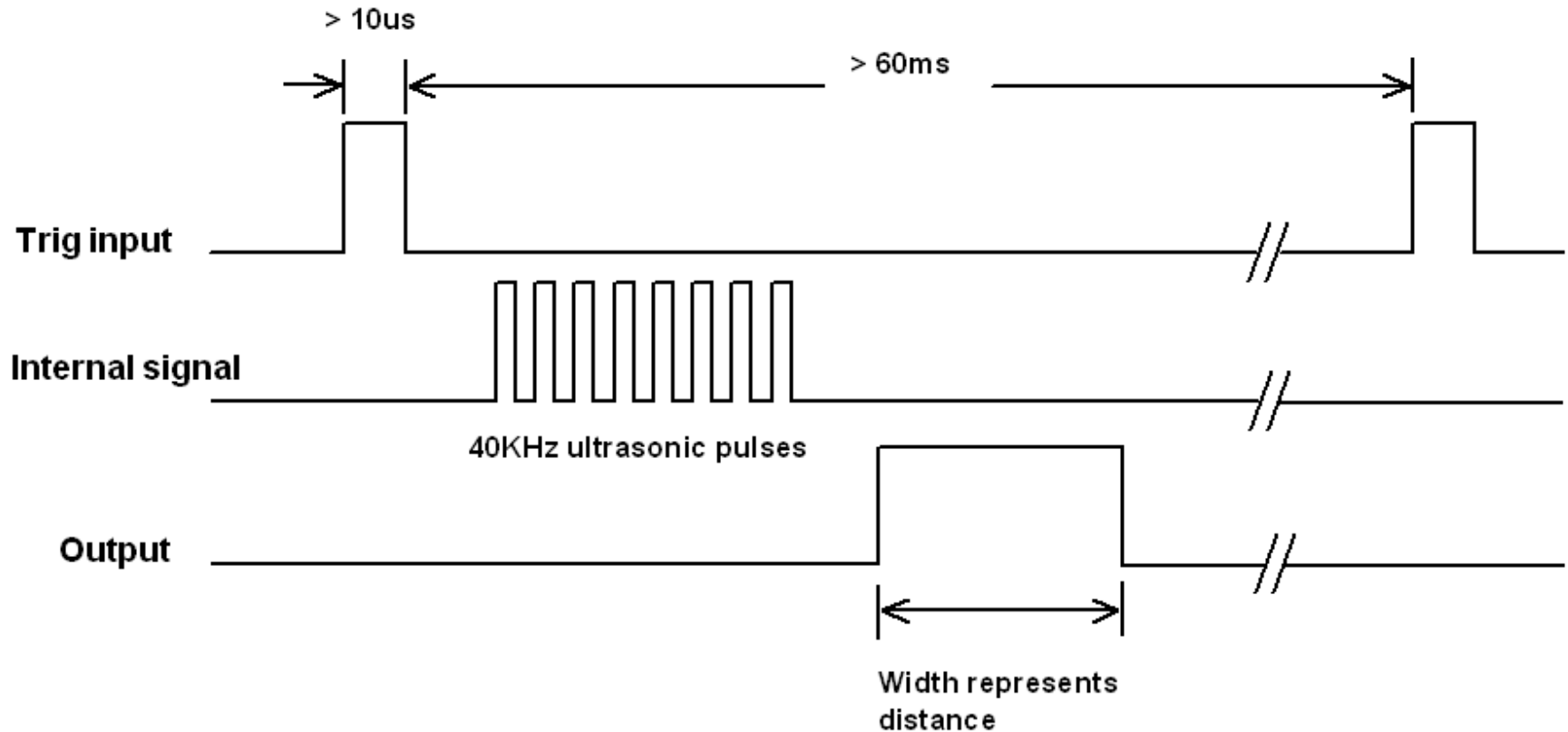
# Pin Outs



# Working



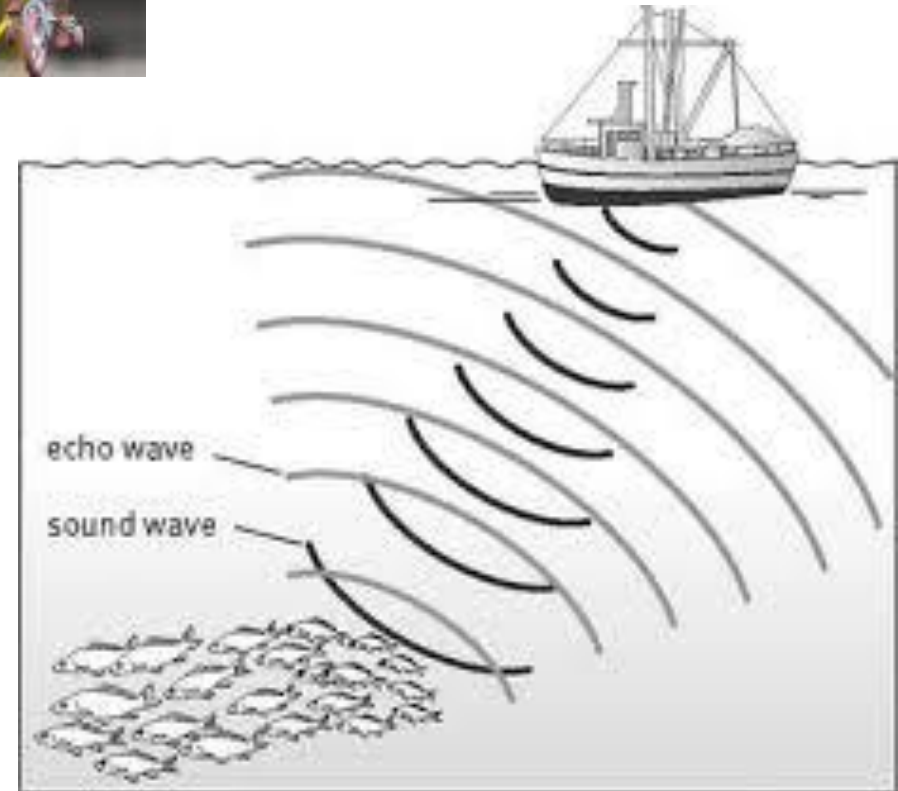
# Timing Diagram

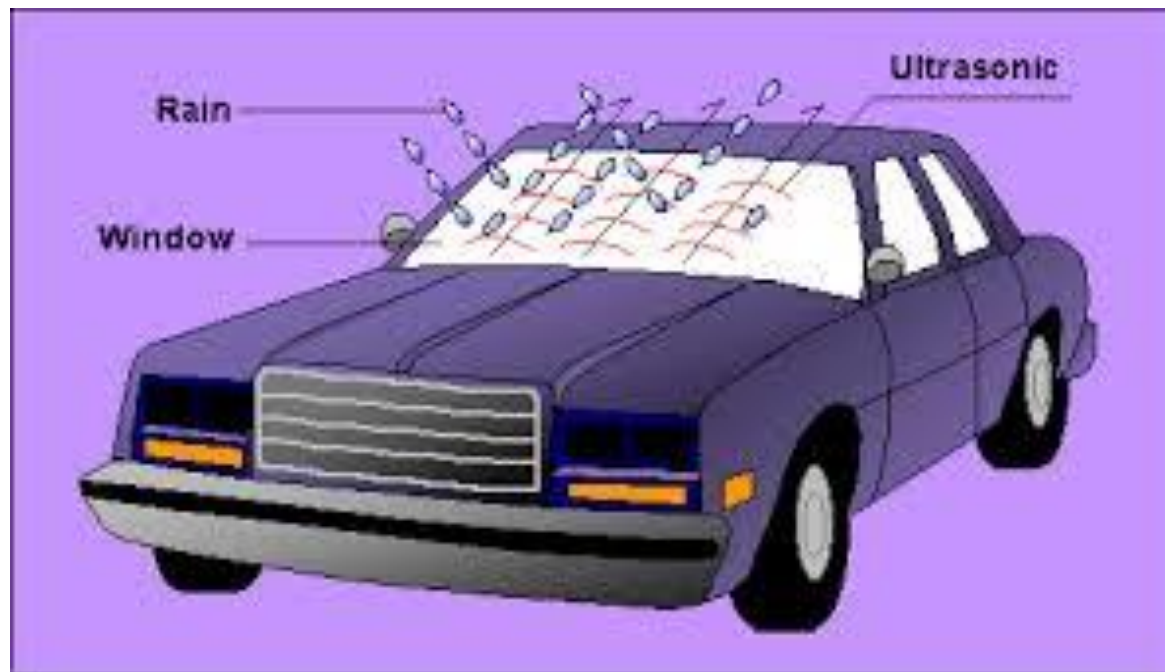
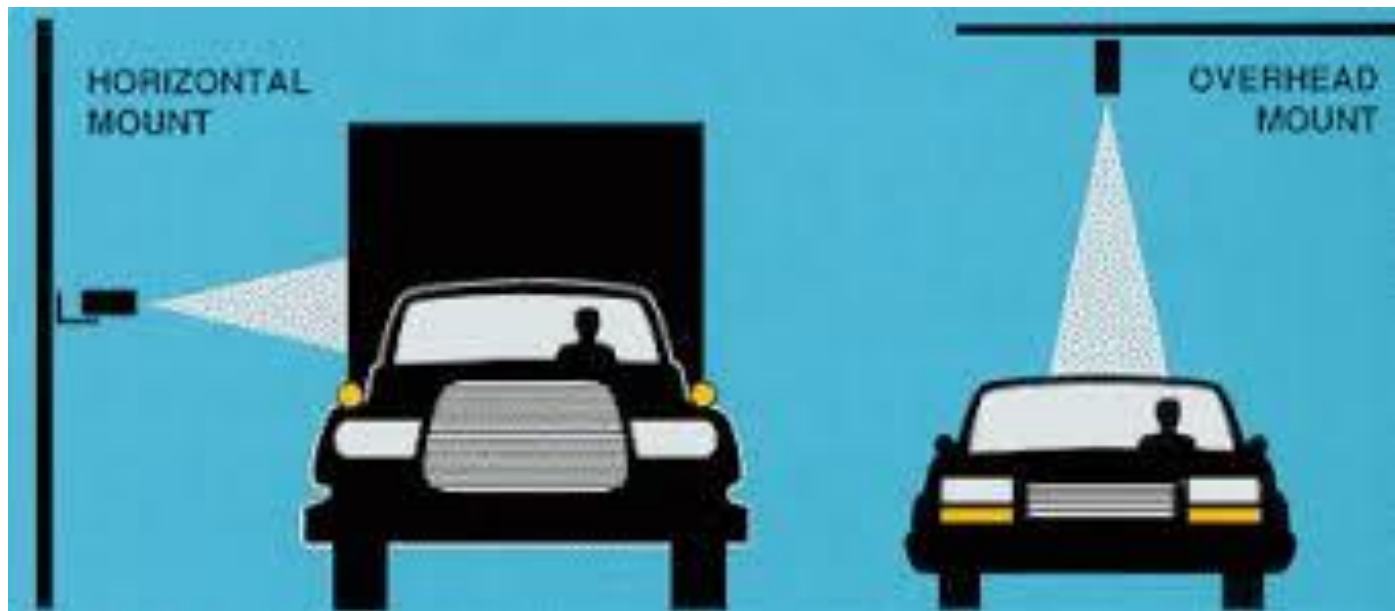


# Applications



- Park assistance
- Distance measuring device
- SONAR
- Fishing





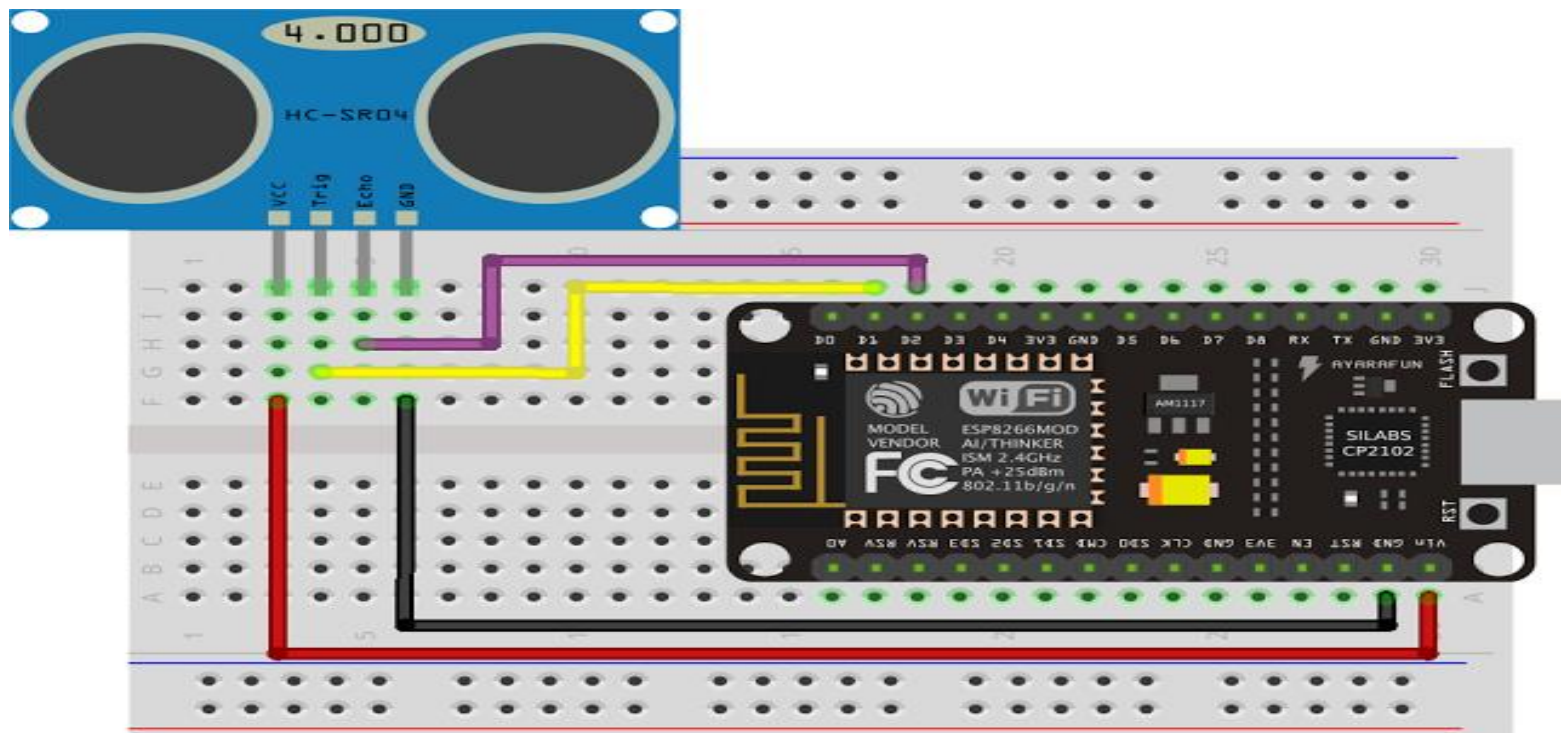
# Interfacing Ultrasonic Sensor WITH NodeMcu

Trig pin → GPIO5( D1)

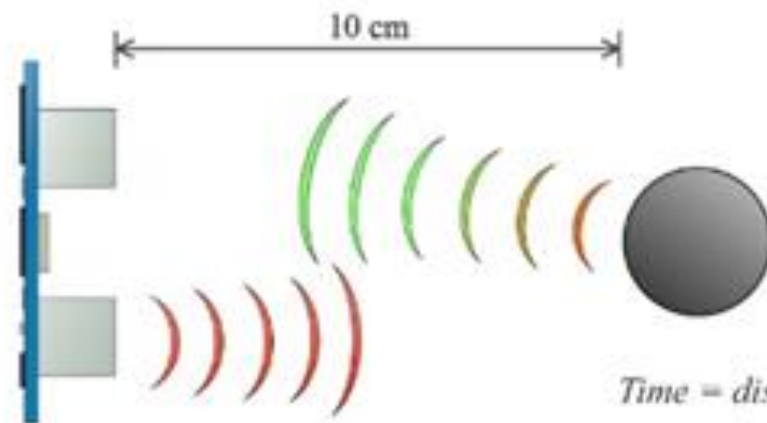
Echo pin → GPIO4 D2

VCC → 3.3v or Vin

Gnd → Gnd







*speed of sound:*

$$v = 340 \text{ m/s}$$

$$v = 0,034 \text{ cm}/\mu\text{s}$$

*Time = distance / speed:*

$$t = s / v = 10 / 0,034 = 294 \mu\text{s}$$

*Distance:*

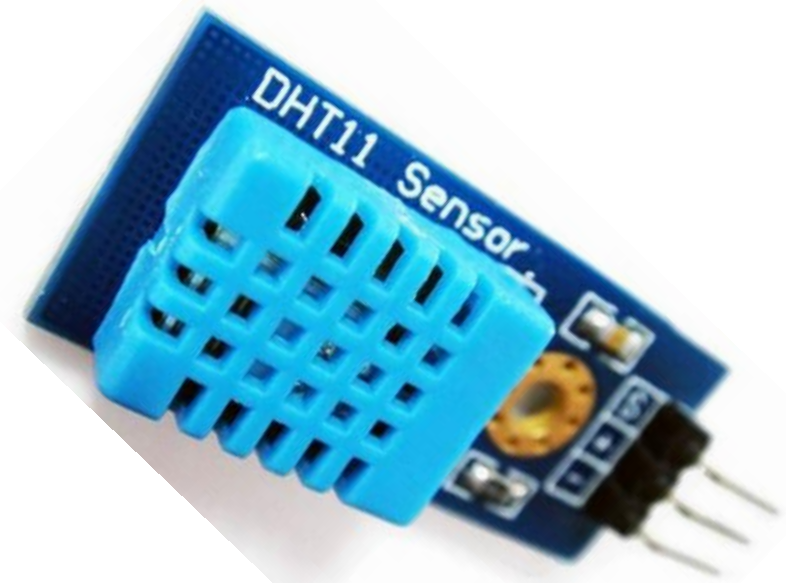
$$s = t \cdot 0,034 / 2$$

# DHT11 Temperature and Humidity Sensor

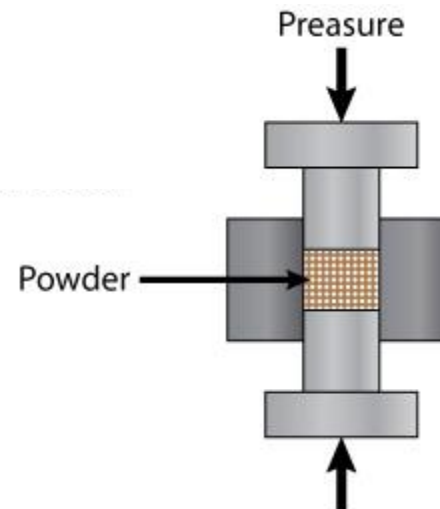
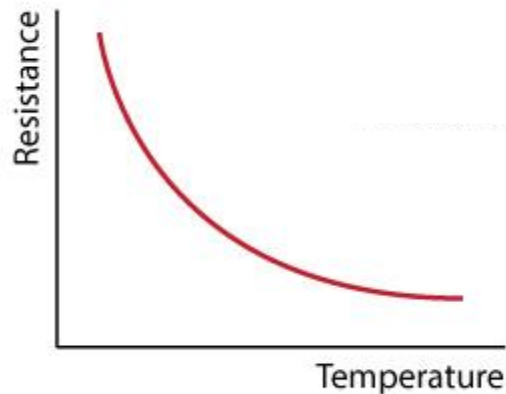
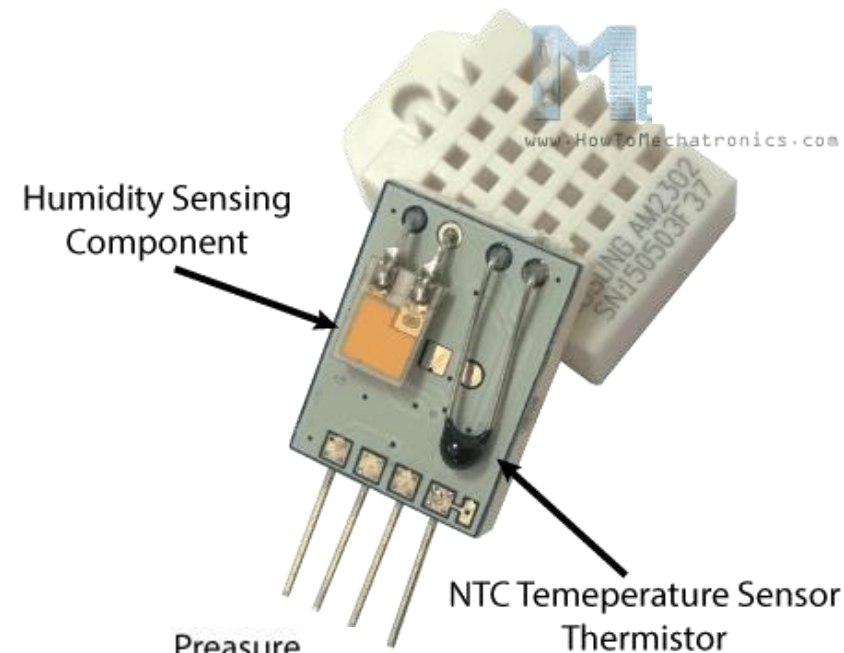
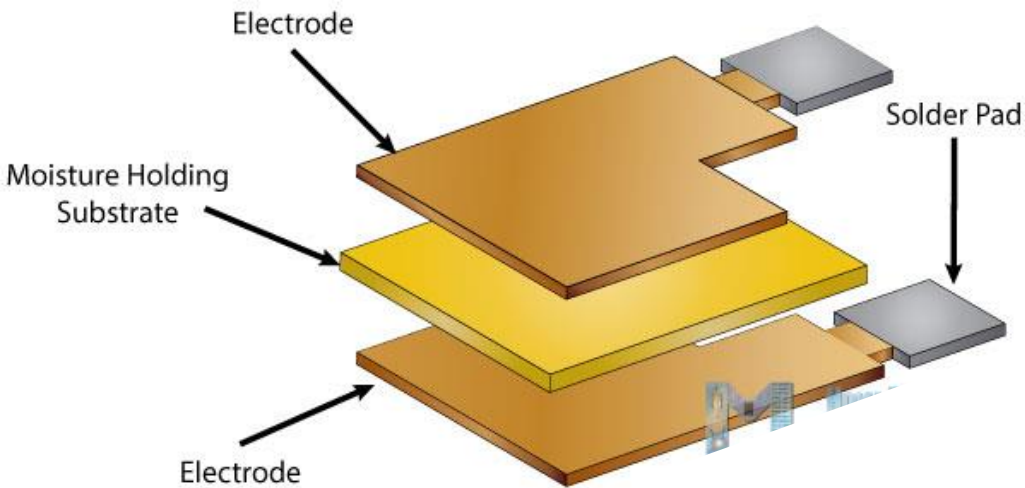
- DHT11 is a basic, ultra low-cost digital temperature and humidity sensor
- Capacitive humidity sensor and a thermistor to measure the surrounding air
- Detects water vapor by measuring the electrical resistance between two electrodes
- Humidity sensing component is a moisture holding substrate with electrodes applied to the surface

## Technical Specification:

- Humidity Range: 20-90% RH
- Humidity Accuracy:  $\pm 5\%$  RH
- Temperature Range: 0-50 °C
- Temperature Accuracy:  $\pm 2\%$  °C
- Operating Voltage: 3V to 5.5V



# DHT11 Temperature and Humidity Sensor

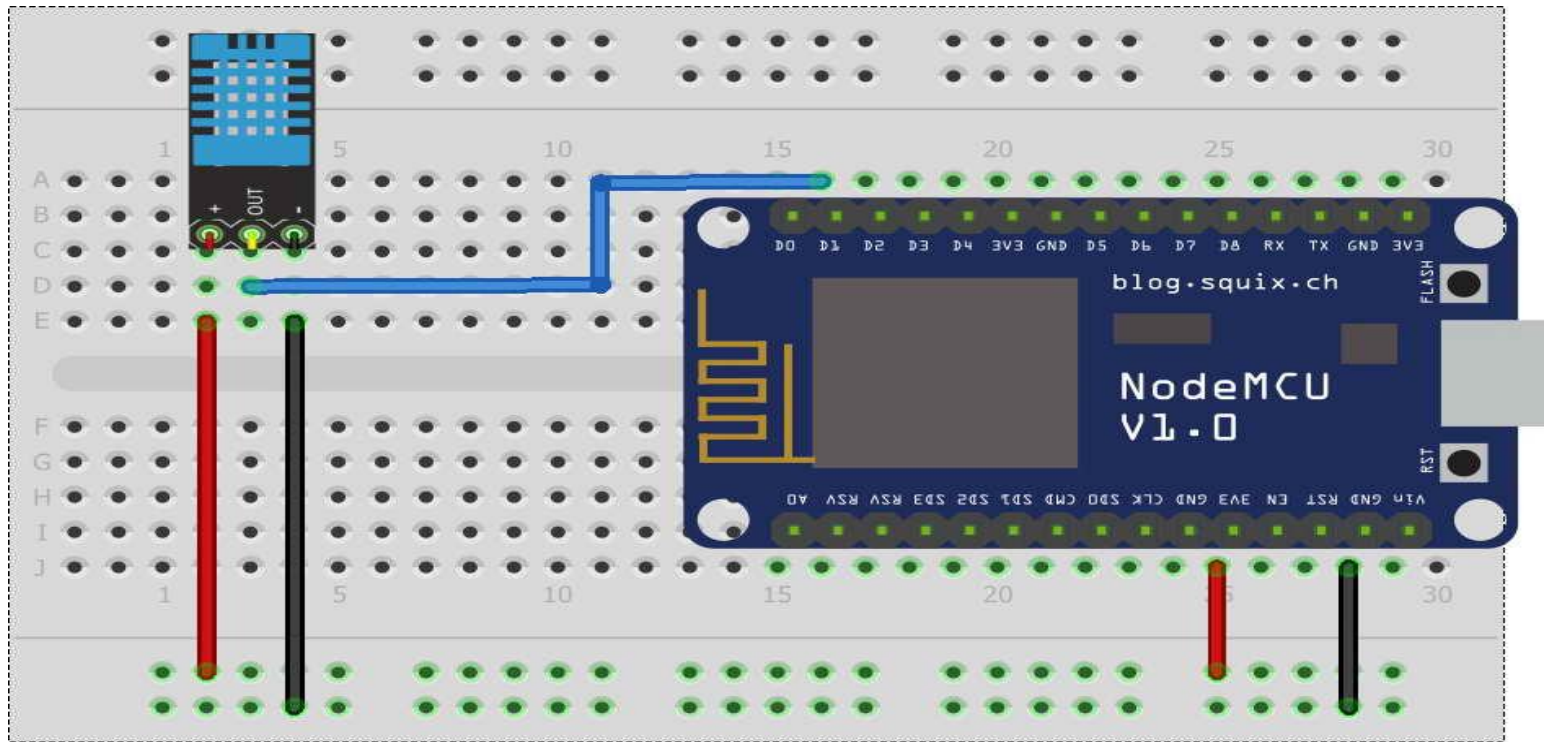


# INTERFACING DHT11 WITH NodeMcu

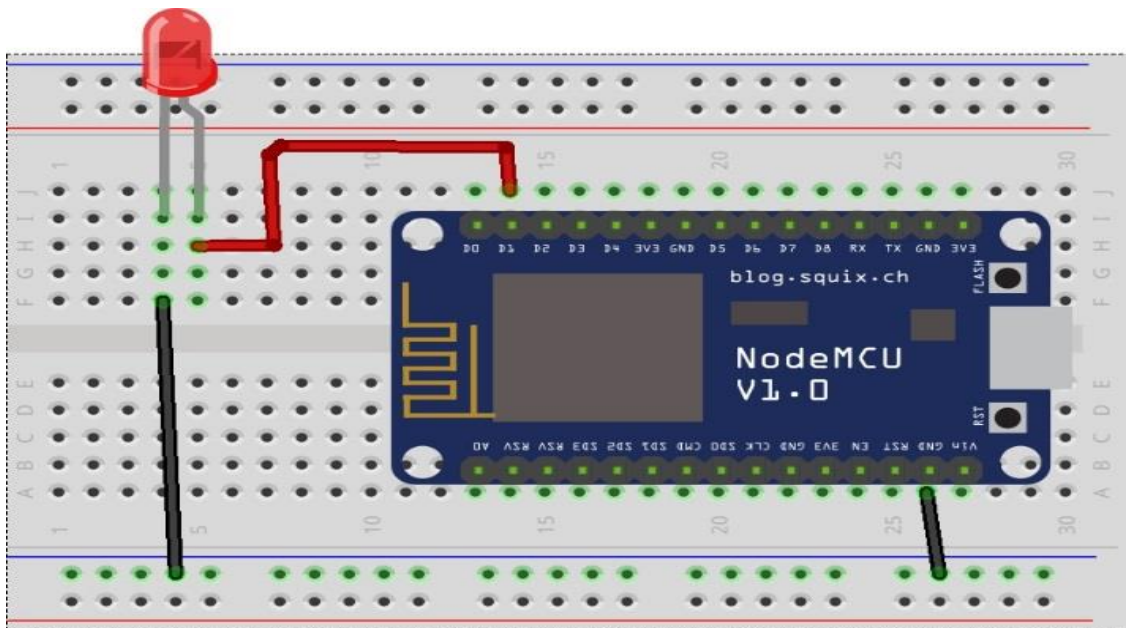
**+ve of DHT to 3.3v**

**-ve of DHT to GND**

**DATA of DHT to D1(GPIO5)**

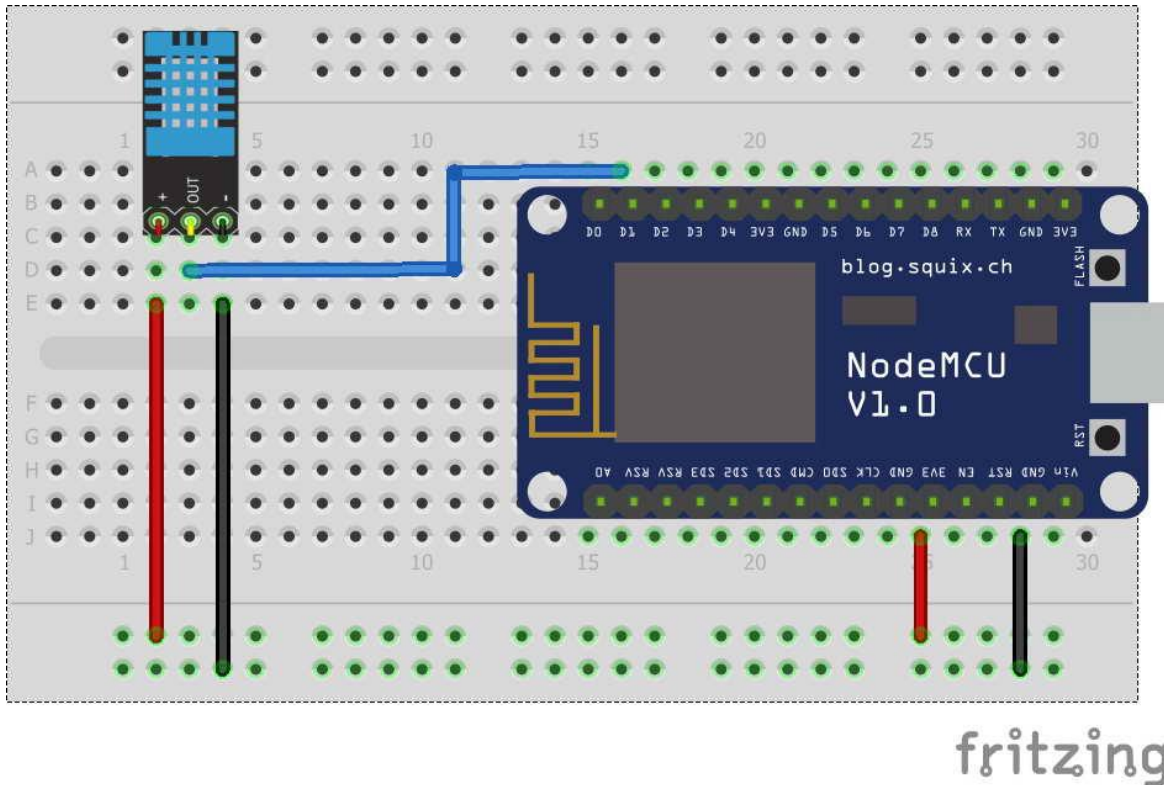


# LED Blinking Using Web Server Programme



fritzing

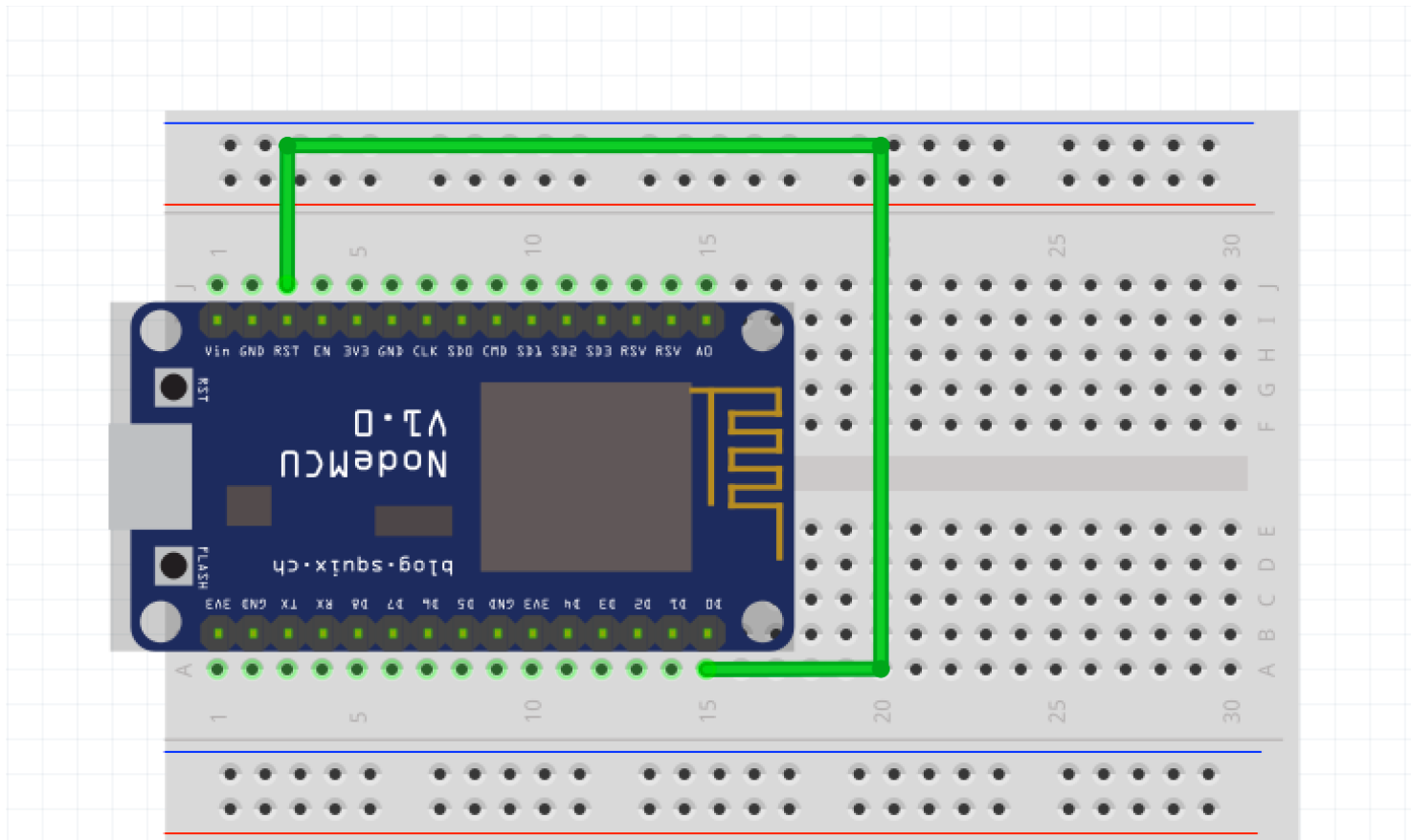
# Temperature and Humidity display using webserver program





# Deep Sleep Mode

**Short Reset(RST) and D1(GPIO16) of nodemcu**



**If you're not prepared to be  
wrong, you'll never come  
up with anything original.**

—Sir Ken Robinson

