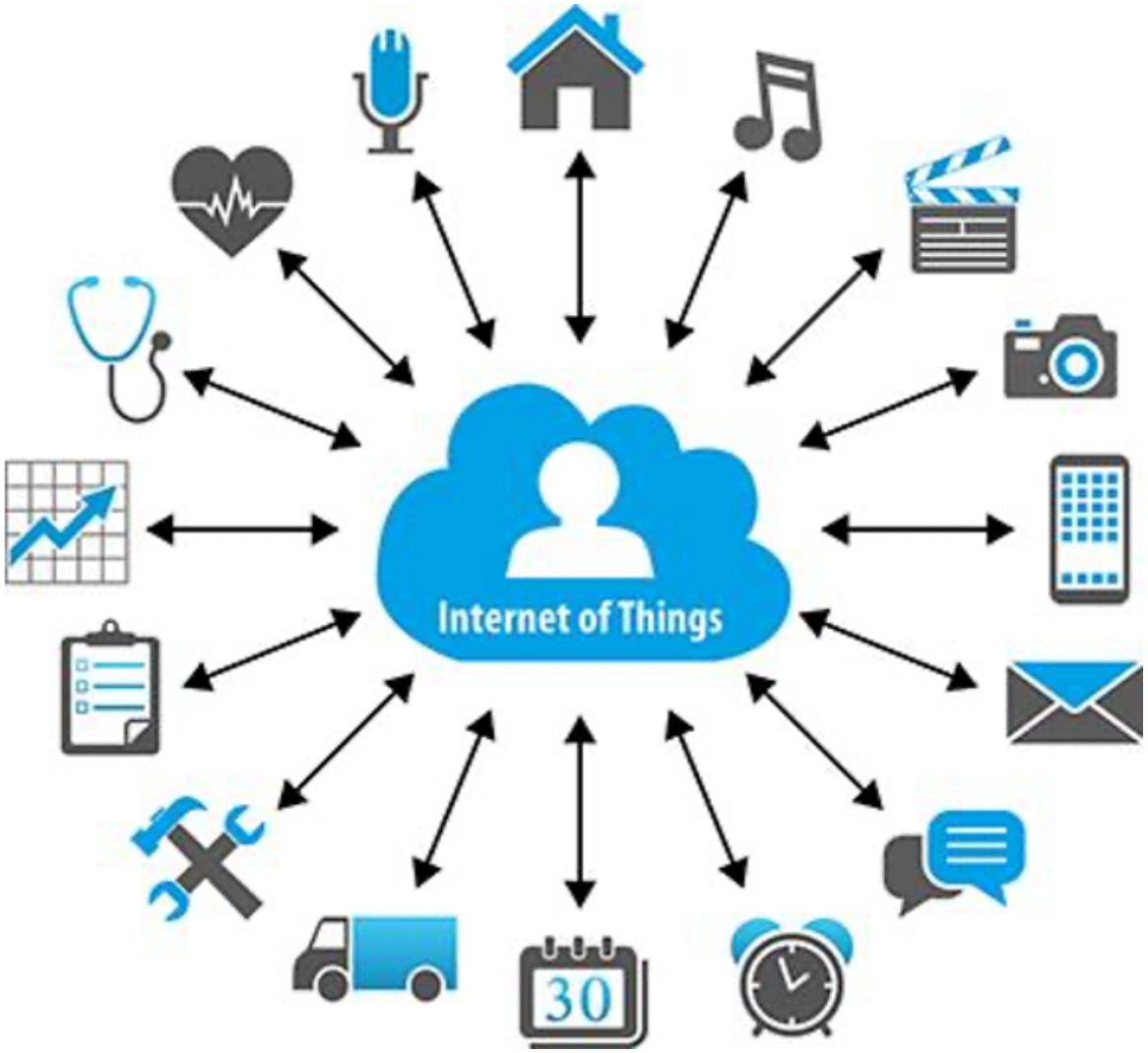
IoT Workshop

IoTPlex

19 September 2020

Alireza Abdeshah

What is The Internet of Things?

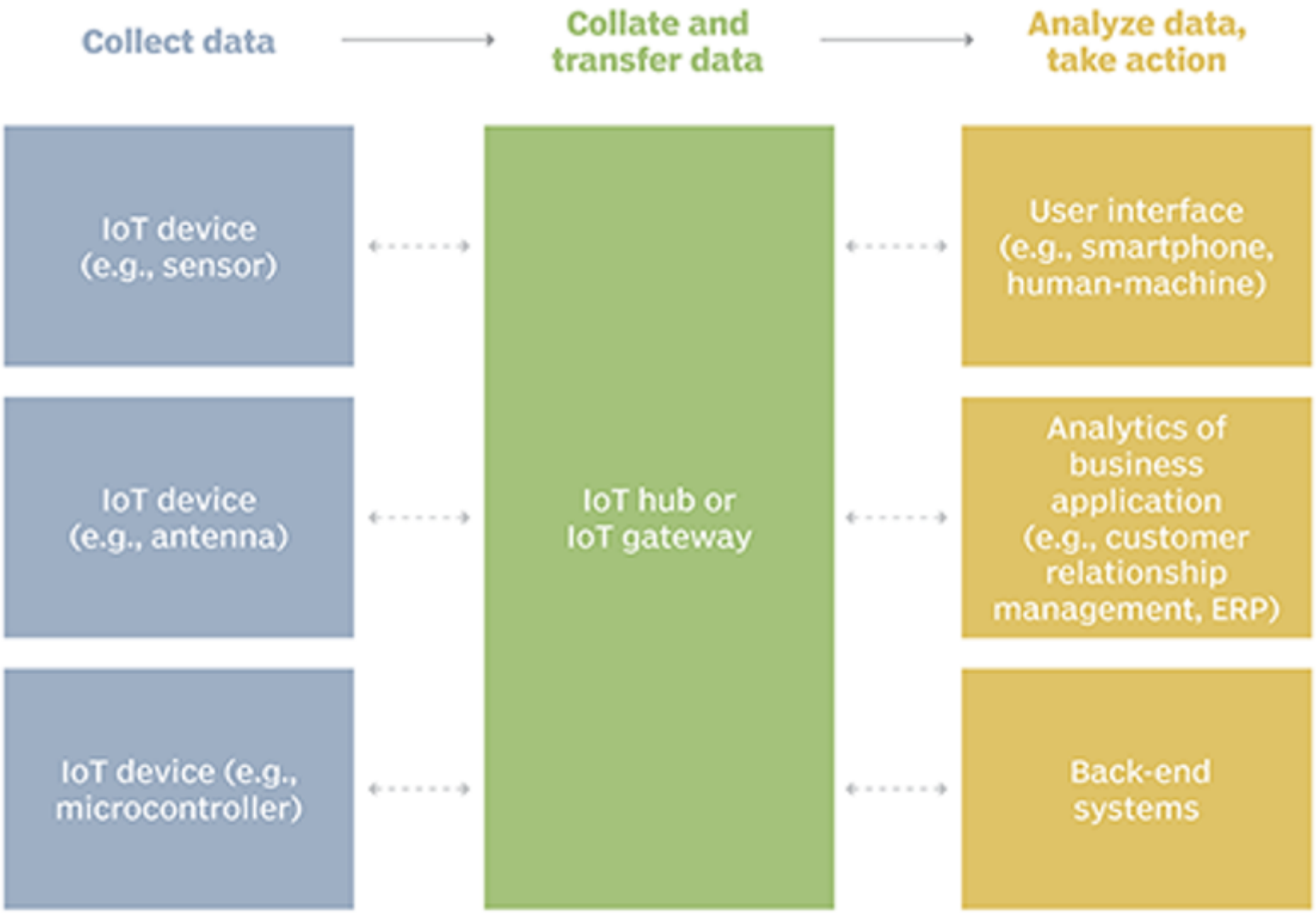


Application of the Internet of Things

According to Wikipedia Application of the Internet of Things:

- Environmental monitoring
- Infrastructure management
- Manufacturing
- Energy management
- Medical and healthcare
- Building and home automation
- Transportation
- Metropolitan scale deployments
- Consumer application

IoT Architecture



4

IoT Architecture

- IoT Device
- IoT Device Host (IoT Gateway)
- IoT Application

5

IoT Device

- ESP8266 & ESP32
- NRF5x
- Mediatek Linkit
- Samsung Artik
- Particle

6

IoT Gateway

- LoRa
- WiFi
- Bluetooth
- NB-IoT
- Cellular network

IoT Platform

- Mainflux
- Blynk
- Thingsboard
- Kaa Project
- Gobot
- ...

8

IoT Messaging Protocols

- Mqtt
- Ntas
- rest
- ...

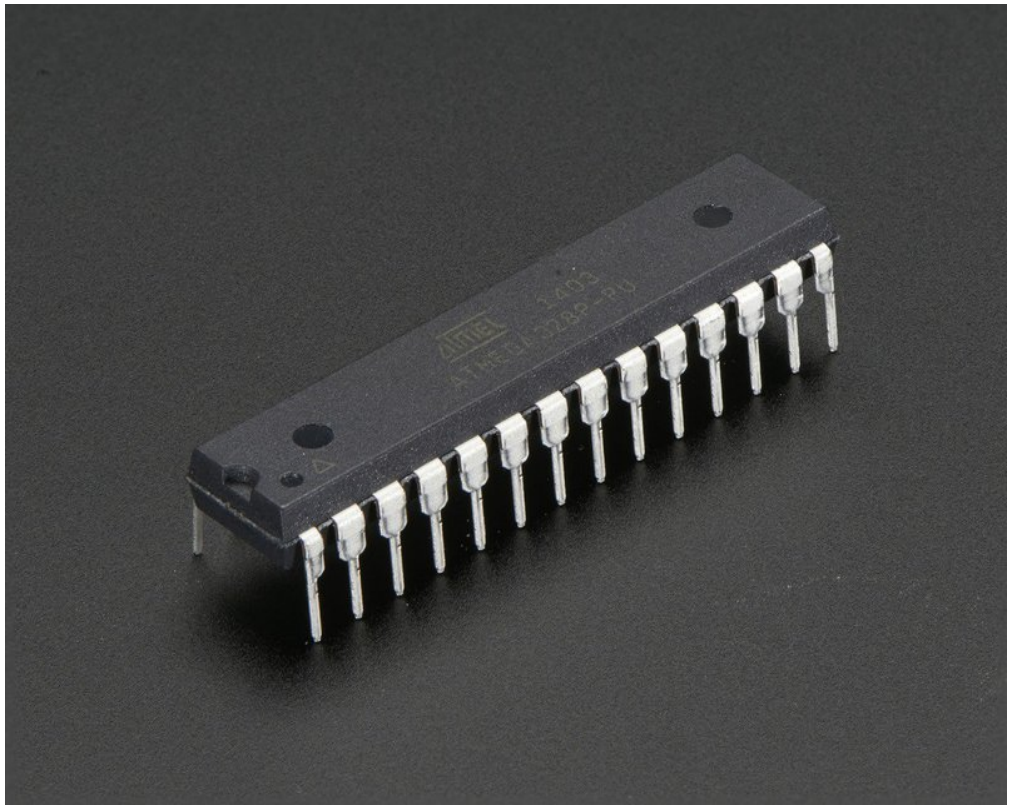
9

More Resource

https://github.com/HQarroum/awesome-iot
https://github.com/phodal/awesome-iot
https://github.com/nebgnahz/awesome-iot-hacks

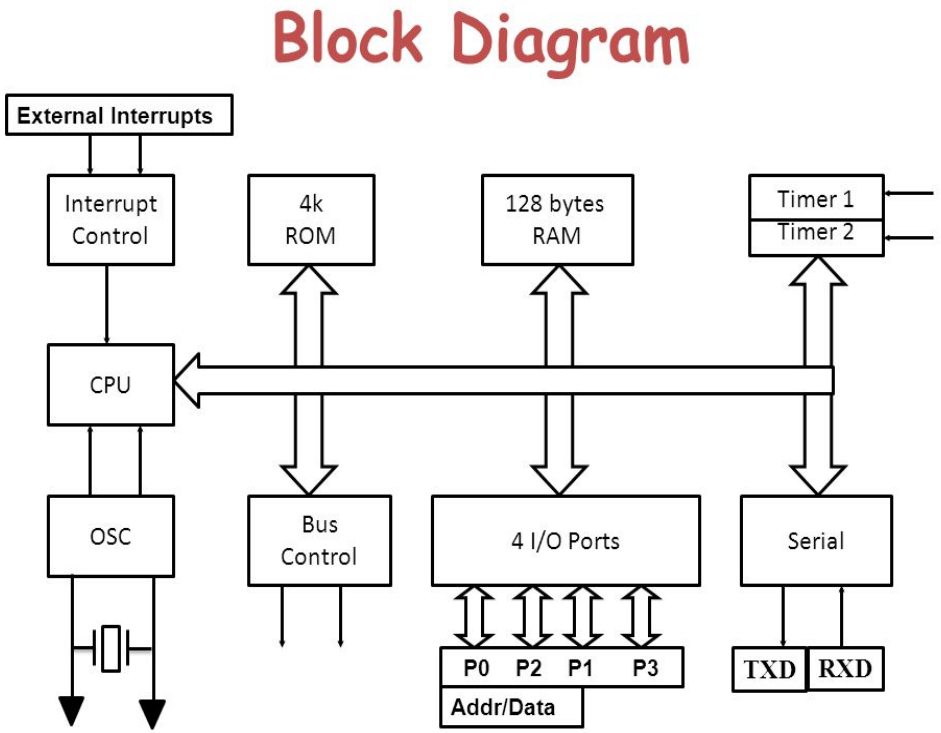
10

What is Microcontroller?

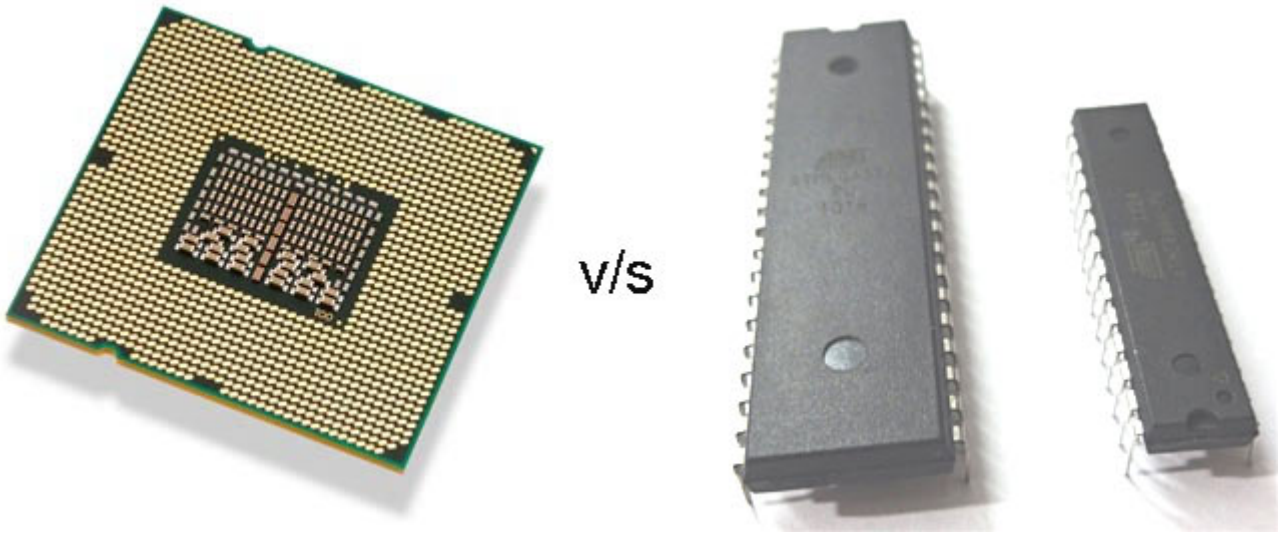


Microcontroller

A microcontroller (or MCU for microcontroller unit) is a small computer on a single integrated circuit



What is the difference between microprocessor and microcontroller?



13

Microcontroller

- are typically 8-bit, but may be 4-, 16-, or 32-bit
- run at speeds less than 200 MHz
- use very little power
- may provide enough current to operate an LED
- are useful to interface with sensors and motors
- are readily replaced, being inexpensive ($0.10 to $10)
- are really constrained for RAM and persistent storage (flash space)
- are really nice for electronics hobbyists

Microprocessors

- are often at least 16-bit, and typically 32-bit or 64-bit, though 8-bit still has a big market share
- many will be able to do floating point math in hardware
- run at speeds measured in hundreds of MHz
- are designed to be the brains of a system (and need a whole system to support them)
- need special hardware to interface with sensors, motors, LEDs, etc.
- are expensive (think $50 - $250 for 32 or 64-bit)
- are designed for external RAM and persistent storage (hard drives)
- are not as easily worked with by a hobbyist

AVR Microcontroller

AVR is a family of microcontrollers developed by Atmel beginning in 1996. These are modified Harvard architecture 8-bit RISC single-chip microcontrollers. AVR was one of the first microcontroller families to use on-chip flash memory for program storage, as opposed to one-time programmable ROM, EPROM, or EEPROM used by other microcontrollers at the time.

The '328 microcontroller has:

- 28 Pins
- Powered by 3 or 5 Volts
- Requires about 0.1 Watts of power
- Runs at 16 MHz
- 32 KB of flash storage
- 2 KB of RAM
- Costs about $5 per

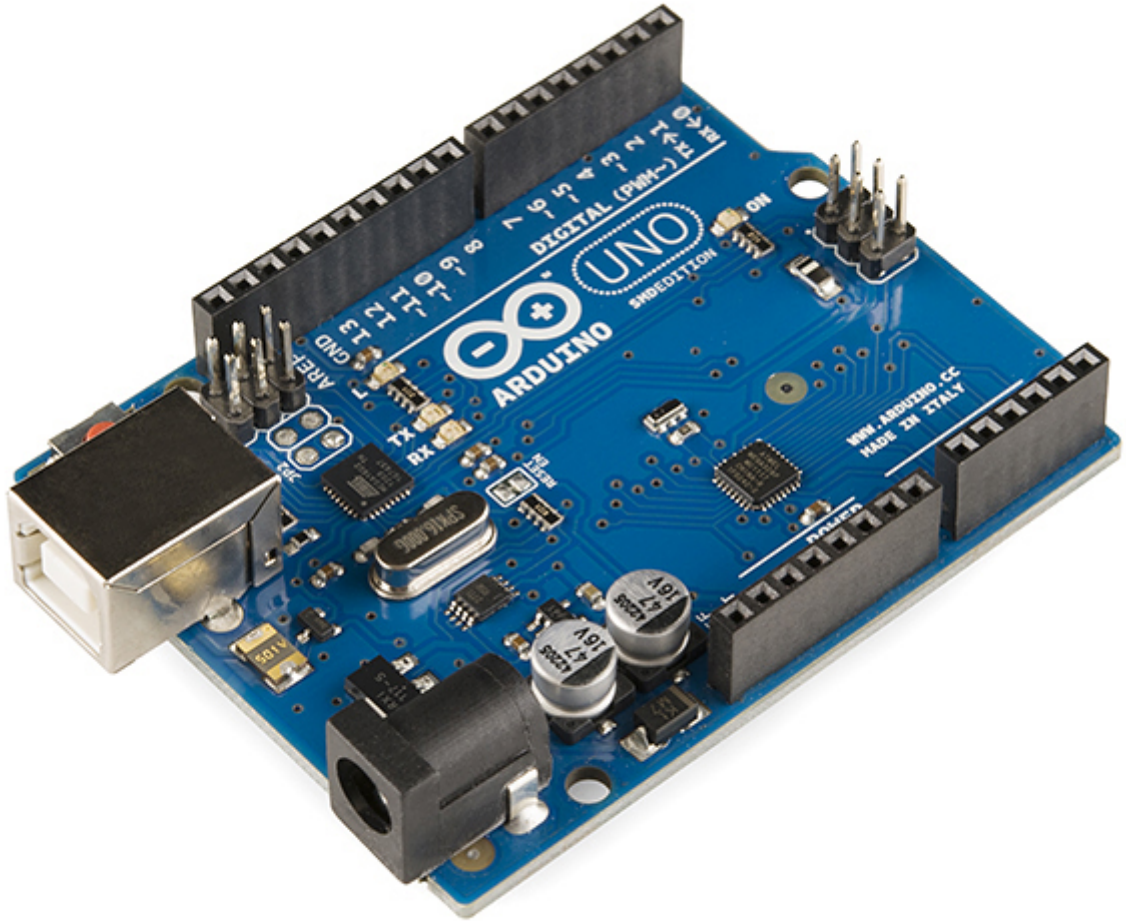
What is an Arduino?

17

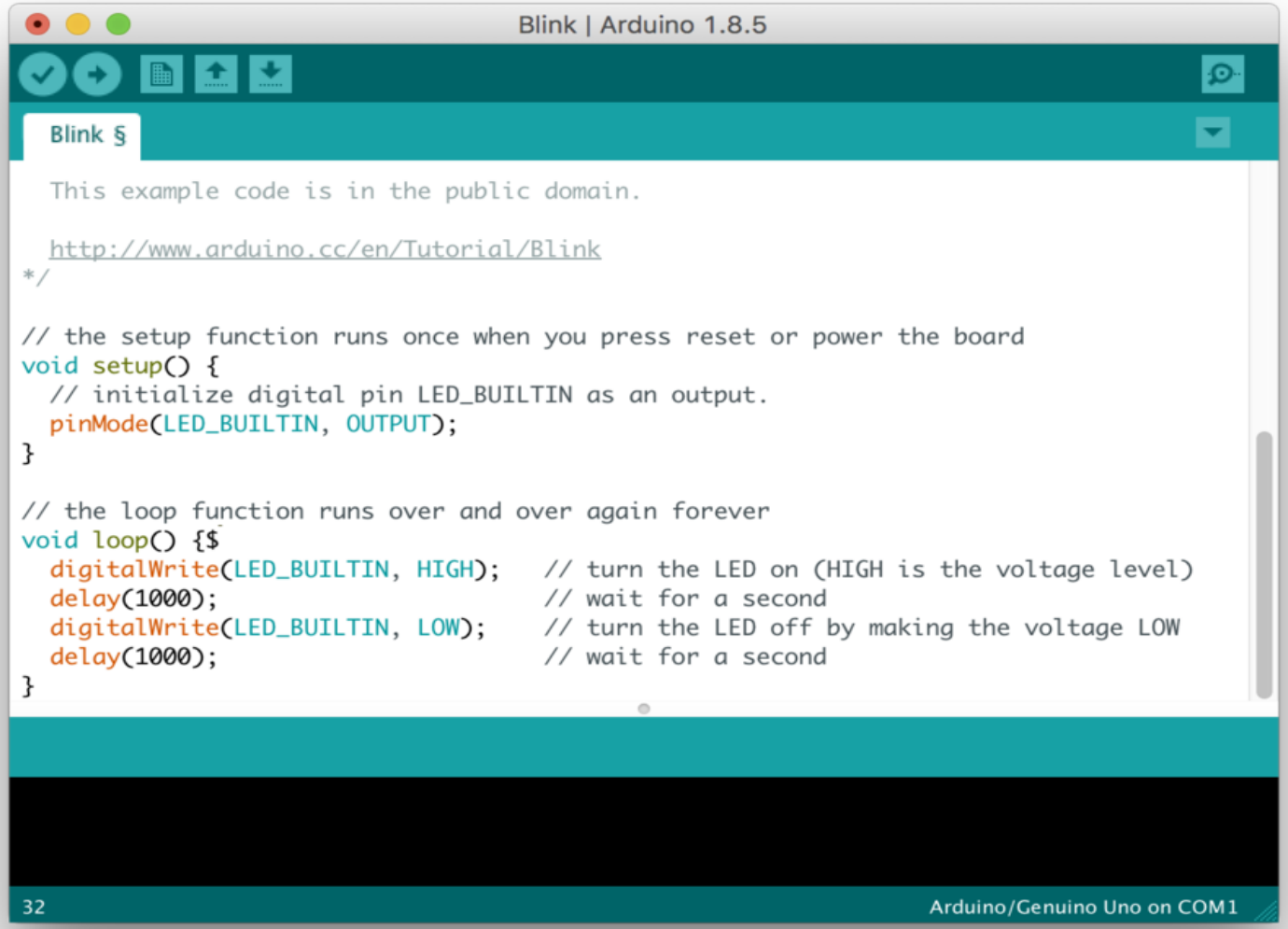
Arduino

Arduino is an open-source prototyping platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so you use the Arduino programming language (based on Wiring), and the Arduino Software (IDE), based on Processing.

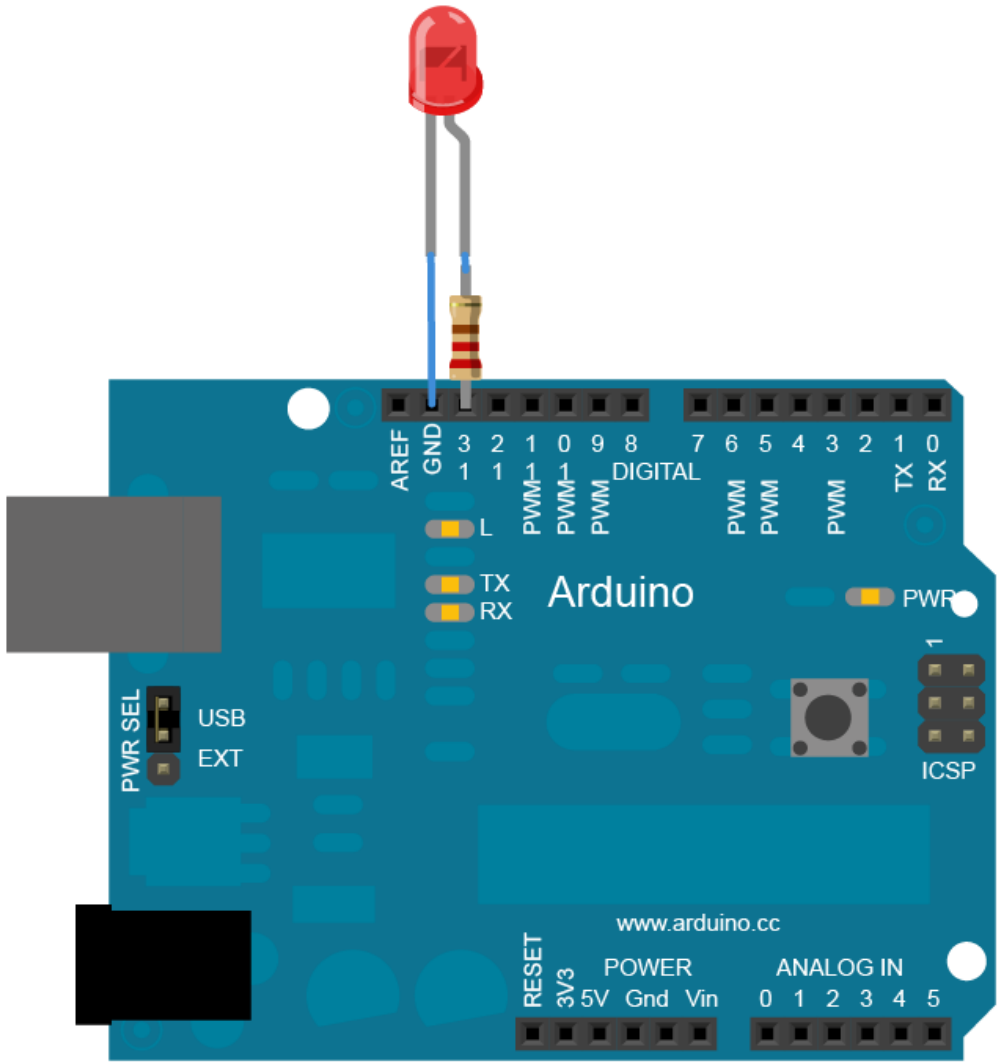
Arduino Board



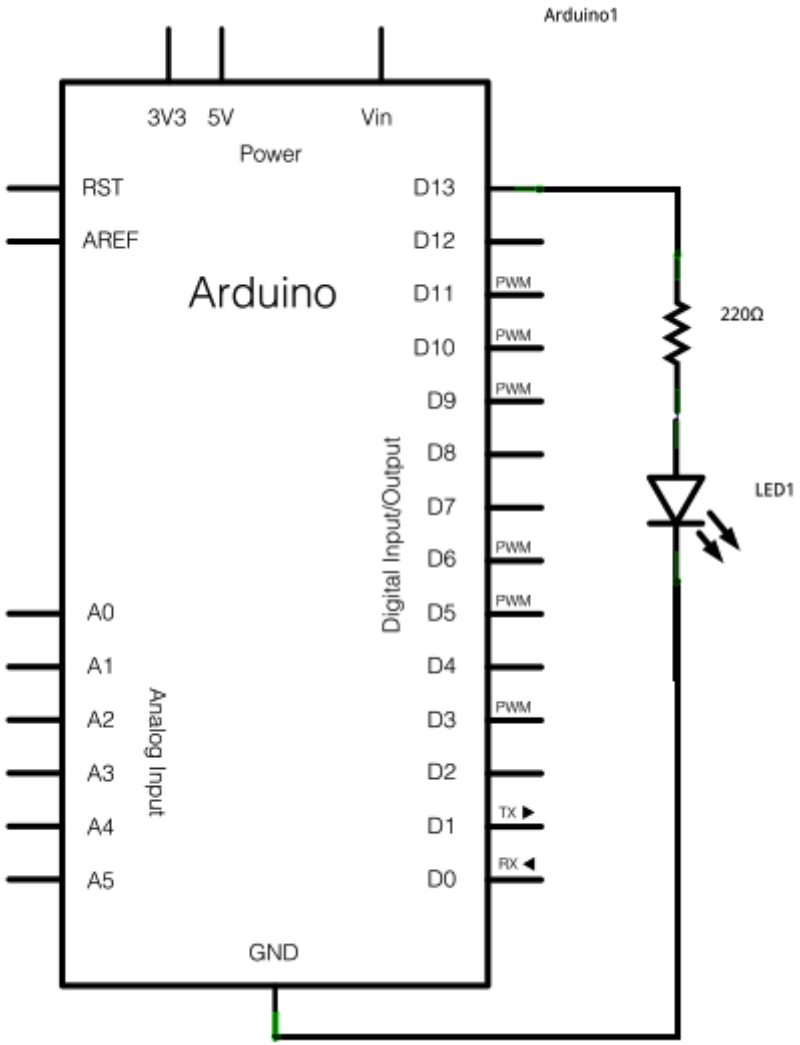
Ardiono IDE


Blink | Arduino 1.8.5
Blink §
This example code is in the public domain.
http://www.arduino.cc/en/Tutorial/Blink
*/
// the setup function runs once when you press reset or power the board
void setup() {
// initialize digital pin LED_BUILTIN as an output.
pinMode(LED_BUILTIN, OUTPUT);
}
// the loop function runs over and over again forever
void loop() {$
digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)
delay(1000); // wait for a second
digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making the voltage LOW
delay(1000); // wait for a second
}
32
Arduino/Genuino Uno on COM1

Blink


AREF
GND
3 2 1 0 9 8 7 6 5 4 3 2 1 0
1 1 1 1 DIGITAL
PWM PWM PWM PWM PWM
TX RX
L
TX
RX
Arduino
PWR
PWR SEL
USB
EXT
ICSP
www.arduino.cc
RESET
3V3 5V Gnd Vin
POWER
ANALOG IN
0 1 2 3 4 5

Schematic


Arduino1
3V3
5V
Vin
Power
RST
AREF
Arduino
D13
D12
D11
D10
D9
D8
D7
D6
D5
D4
D3
D2
D1
D0
PWM
PWM
PWM
PWM
PWM
PWM
TX
RX
Digital Input/Output
A0
A1
A2
A3
A4
A5
Analog Input
GND
220Ω
LED1

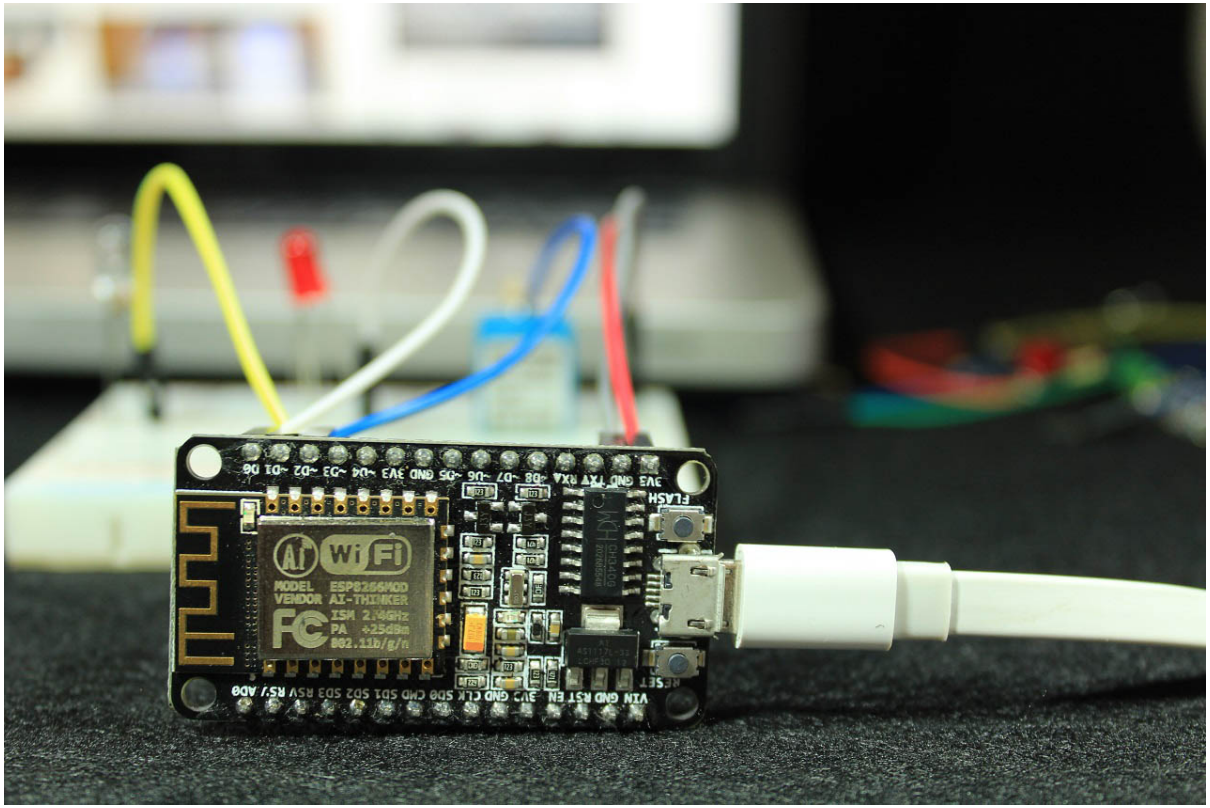
Code

```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                       // wait for a second
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);                       // wait for a second
}
```

23

ESP8266

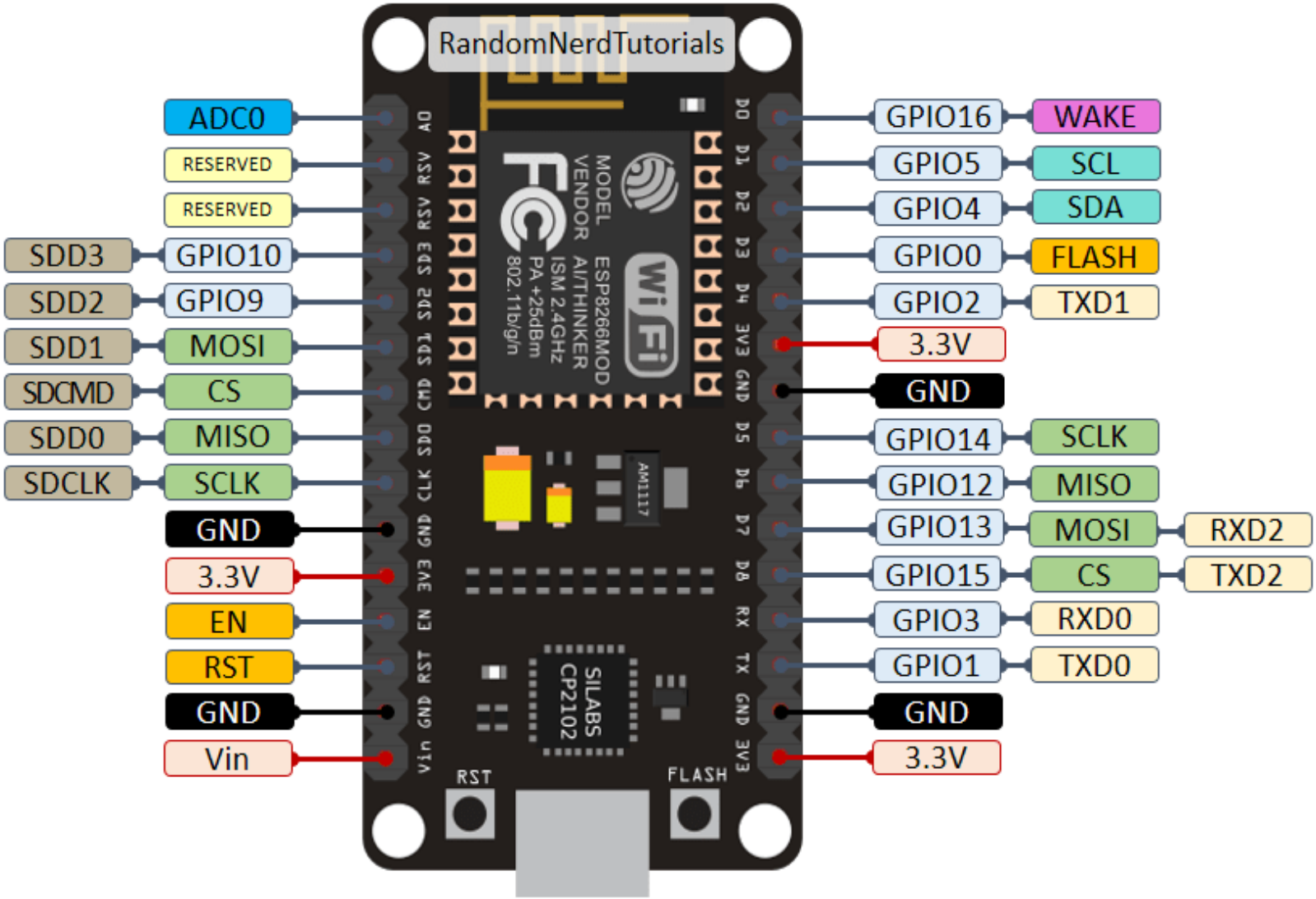
The ESP8266 is a $4 (up to $10) Wi-Fi module. It allows you to control inputs and outputs as you would do with an Arduino, but it comes with Wi-Fi.
So, it is great for home automation/internet of things applications.



ESP8266 Features

- Processor: L106 32-bit RISC microprocessor core based on the Tensilica Xtensa Diamond Standard 106Micro running at 80 MHz
- Memory: 32 KiB instruction RAM ,32 KiB instruction cache RAM ,80 KiB user-data RAM ,16 KiB ETS system-data RAM
- External QSPI flash: up to 16 MiB is supported (512 KiB to 4 MiB typically included)
- IEEE 802.11 b/g/n Wi-Fi
- Integrated TR switch, balun, LNA, power amplifier and matching network
- WEP or WPA/WPA2 authentication, or open networks
- 16 GPIO pins
- SPI ,I^2C ,I^2S
- UART on dedicated pins, plus a transmit-only UART can be enabled on GPIO2
- 10-bit ADC (successive approximation ADC)

ESP-12E NodeMCU Kit Pinout



Programming the ESP8266 Using Arduino IDE

There are several ways to program the ESP8266. We often use Arduino IDE or MicroPython.

- Install the current upstream Arduino IDE at the 1.8.9 level or later. The current version is on the Arduino website.
- Start Arduino and open the Preferences window.
- Enter https://arduino.esp8266.com/stable/package_esp8266com_index.json into the Additional Board Manager URLs field. You can add multiple URLs, separating them with commas.
- Open Boards Manager from Tools > Board menu and install esp8266 platform (and don't forget to select your ESP8266 board from Tools > Board menu after installation).

ESP8266 Peripherals

The ESP8266 peripherals include:

- 17 GPIOs
- SPI
- I2C (implemented on software)
- I2S interfaces with DMA
- UART
- 10-bit ADC

Best Pins to Use – ESP8266

One important thing to notice about ESP8266 is that the GPIO number doesn't match the label on the board silkscreen. For example, D0 corresponds to GPIO16 and D1 corresponds to GPIO5.

The following table shows the correspondence between the labels on the silkscreen and the GPIO number as well as what pins are the best to use in your projects, and which ones you need to be cautious.

The pins highlighted in green are OK to use. The ones highlighted in yellow are OK to use, but you need to pay attention because they may have unexpected behavior mainly at boot. The pins highlighted in red are not recommended to use as inputs or outputs.

Label	GPIO	Input	Output	Notes
D0	GPIO16	no interrupt	no PWM or I2C support	HIGH at boot used to wake up from deep sleep
D1	GPIO5	OK	OK	often used as SCL (I2C)
D2	GPIO4	OK	OK	often used as SDA (I2C)
D3	GPIO0	pulled up	OK	connected to FLASH button, boot fails if pulled LOW
D4	GPIO2	pulled up	OK	HIGH at boot connected to on-board LED, boot fails if pulled LOW
D5	GPIO14	OK	OK	SPI (SCLK)
D6	GPIO12	OK	OK	SPI (MISO)
D7	GPIO13	OK	OK	SPI (MOSI)
D8	GPIO15	pulled to GND	OK	SPI (CS) Boot fails if pulled HIGH
RX	GPIO3	OK	RX pin	HIGH at boot
TX	GPIO1	TX pin	OK	HIGH at boot debug output at boot, boot fails if pulled LOW
A0	ADC0	Analog Input	X	

Analog Input

The ESP8266 only supports analog reading in one GPIO. That GPIO is called ADC0 and it is usually marked on the silkscreen as A0.

The maximum input voltage of the ADC0 pin is 0 to 1V if you're using the ESP8266 bare chip. If you're using a development board like the ESP8266 12-E NodeMCU kit, the voltage input range is 0 to 3.3V because these boards contain an internal voltage divider.

On-board LED

Most of the ESP8266 development boards have a built-in LED. This LED is usually connected to GPIO2.

The LED is connected to a pull-down resistor, so when you send a HIGH signal the LED turns off.

I2C

The ESP8266 doens't have hardware I2C pins, but it can be implemented in software. So you can use any GPIOs as I2C. Usually, the following GPIOs are used as I2C pins:

- GPIO5: SCL
- GPIO4: SDA

SPI

The pins used as SPI in the ESP8266 are:

- GPIO12: MISO
- GPIO13: MOSI
- GPIO14: SCLK
- GPIO15: CS

34

PWM Pins

ESP8266 allows software PWM in all I/O pins: GPIO0 to GPIO16. PWM signals on ESP8266 have 10-bit resolution. Learn how to use ESP8266 PWM pins:

35

ESP8266 NodeMCU Control Digital Outputs

First you need set the GPIO you want to control as an OUTPUT. Use the pinMode() function as follows:

```
pinMode(GPIO, OUTPUT);
```

To control a digital output you just need to use the digitalWrite() function, that accepts as arguments, the GPIO (int number) you are referring to, and the state, either HIGH or LOW.

```
digitalWrite(GPIO, STATE);
```

ESP8266 NodeMCU Read Digital Inputs

First, set the GPIO you want to read as INPUT, using the pinMode() function as follows:

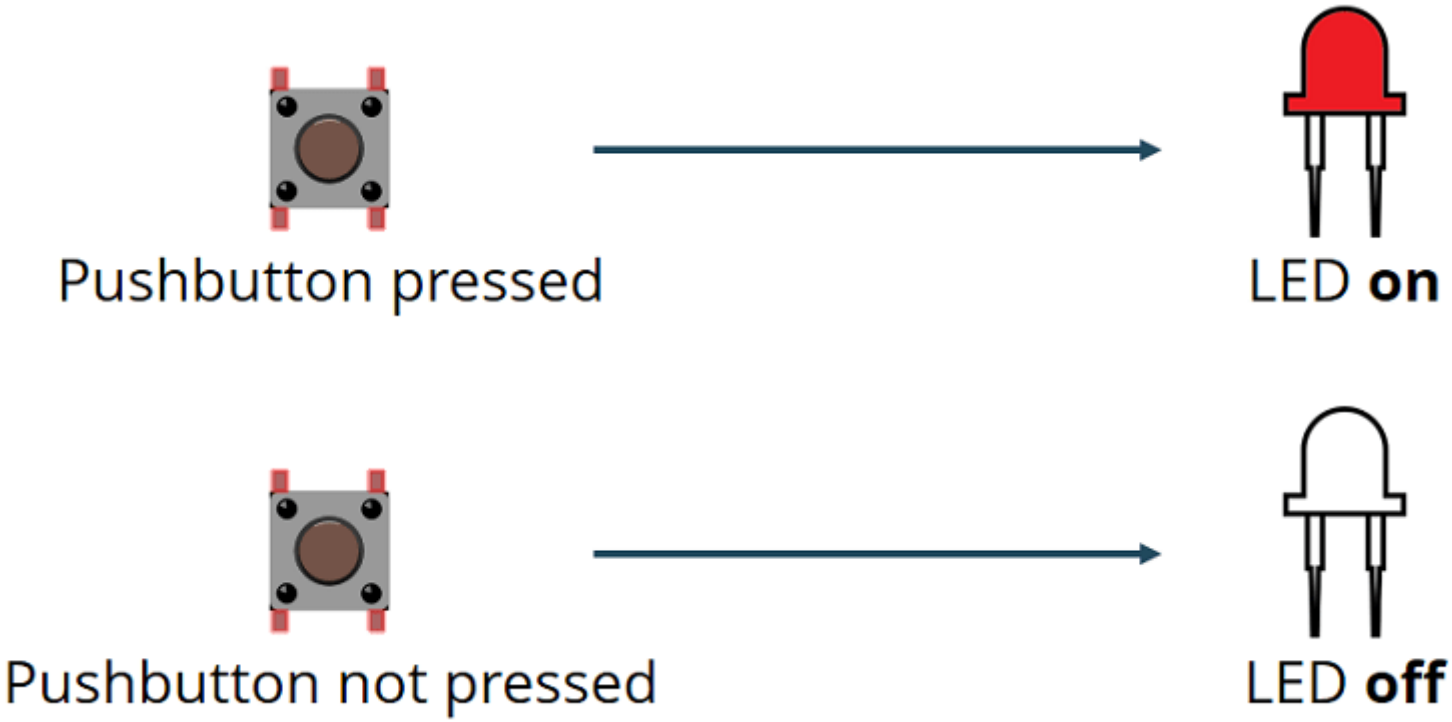
```
pinMode(GPIO, INPUT);
```

To read a digital input, like a button, you use the digitalRead() function, that accepts as argument, the GPIO (int number) you are referring to.

```
digitalRead(GPIO);
```

Project Example

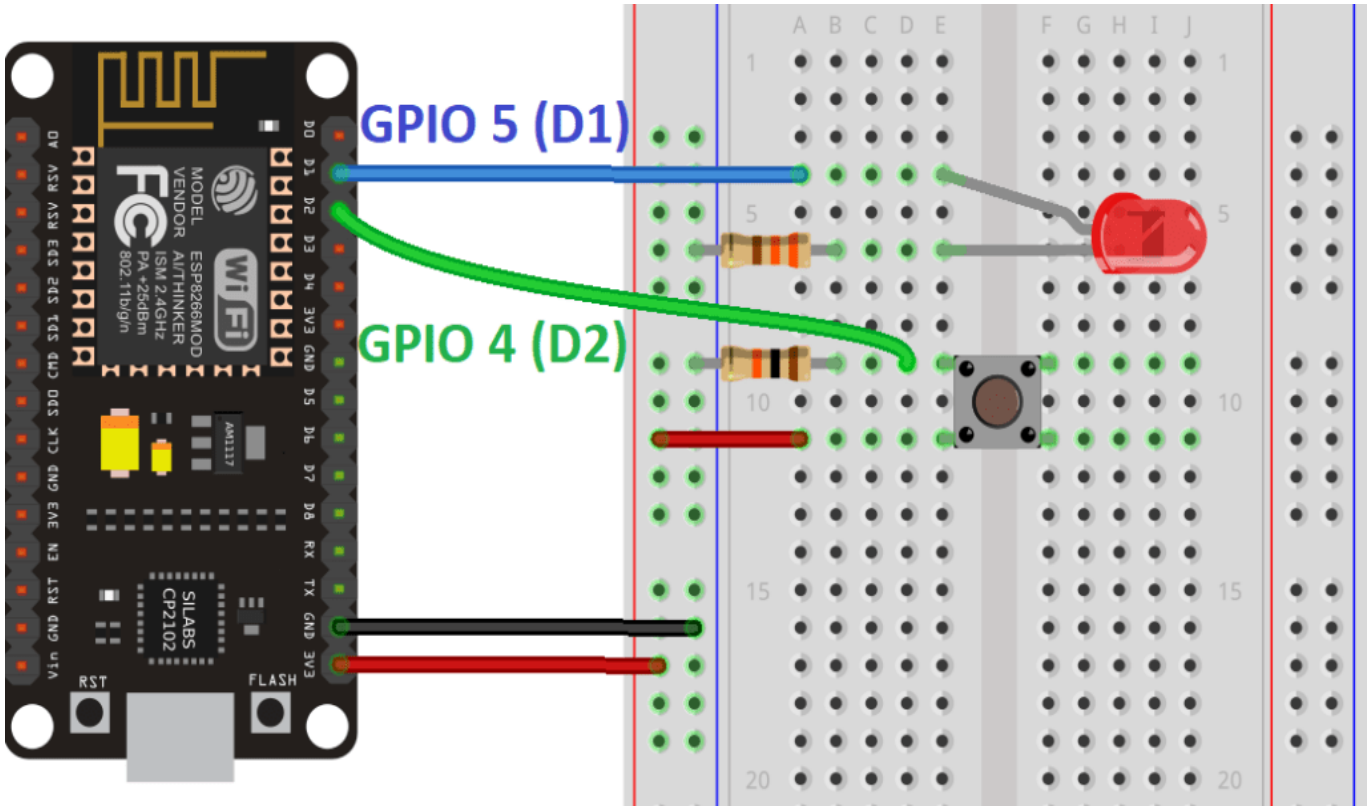
To show you how to use digital inputs and digital outputs, we'll build a simple project example with a pushbutton and an LED. We'll read the state of the pushbutton and light up the LED accordingly as illustrated in the following figure.



38

Schematic Diagram

Before proceeding, you need to assemble a circuit with an LED and a pushbutton. We'll connect the LED to GPIO 5 (D1) and the pushbutton to GPIO 4 (D2).

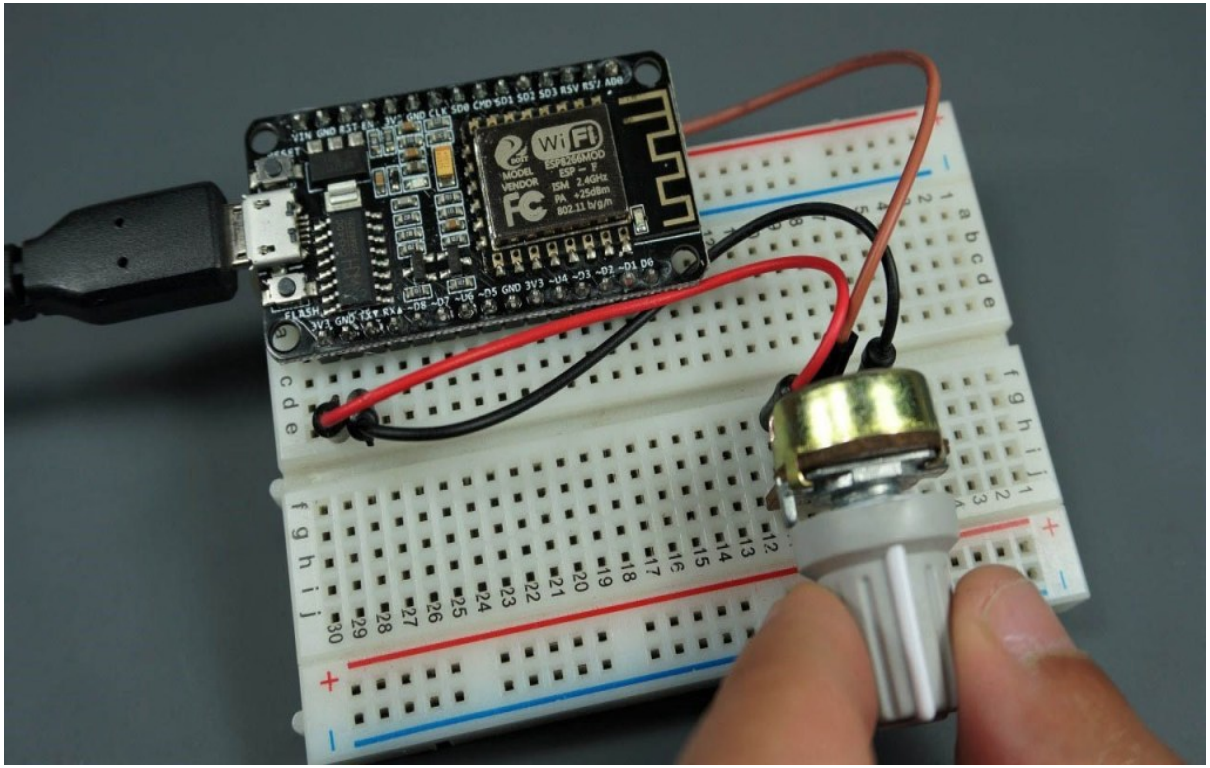


Code

```
const int buttonPin = 4;     // the number of the pushbutton pin
const int ledPin =  5;       // the number of the LED pin

int buttonState = 0;

void setup() {
  pinMode(buttonPin, INPUT);
  pinMode(ledPin, OUTPUT);
}

void loop() {
  // read the state of the pushbutton value
  buttonState = digitalRead(buttonPin);
  // check if the pushbutton is pressed.
  // if it is, the buttonState is HIGH
  if (buttonState == HIGH) {
    // turn LED on
    digitalWrite(ledPin, HIGH);
  } else {
    // turn LED off
    digitalWrite(ledPin, LOW);
  }
}
```

ESP8266 ADC – Read Analog Values

Both ESP8266-12E and ESP8266-07 have one ADC pin that is easily accessible. This means that those ESP8266 boards can read analog signals. In this tutorial we'll show you how to use analog reading with the ESP8266 using Arduino IDE, MicroPython or Lua firmware.



ESP8266 ADC Specifications

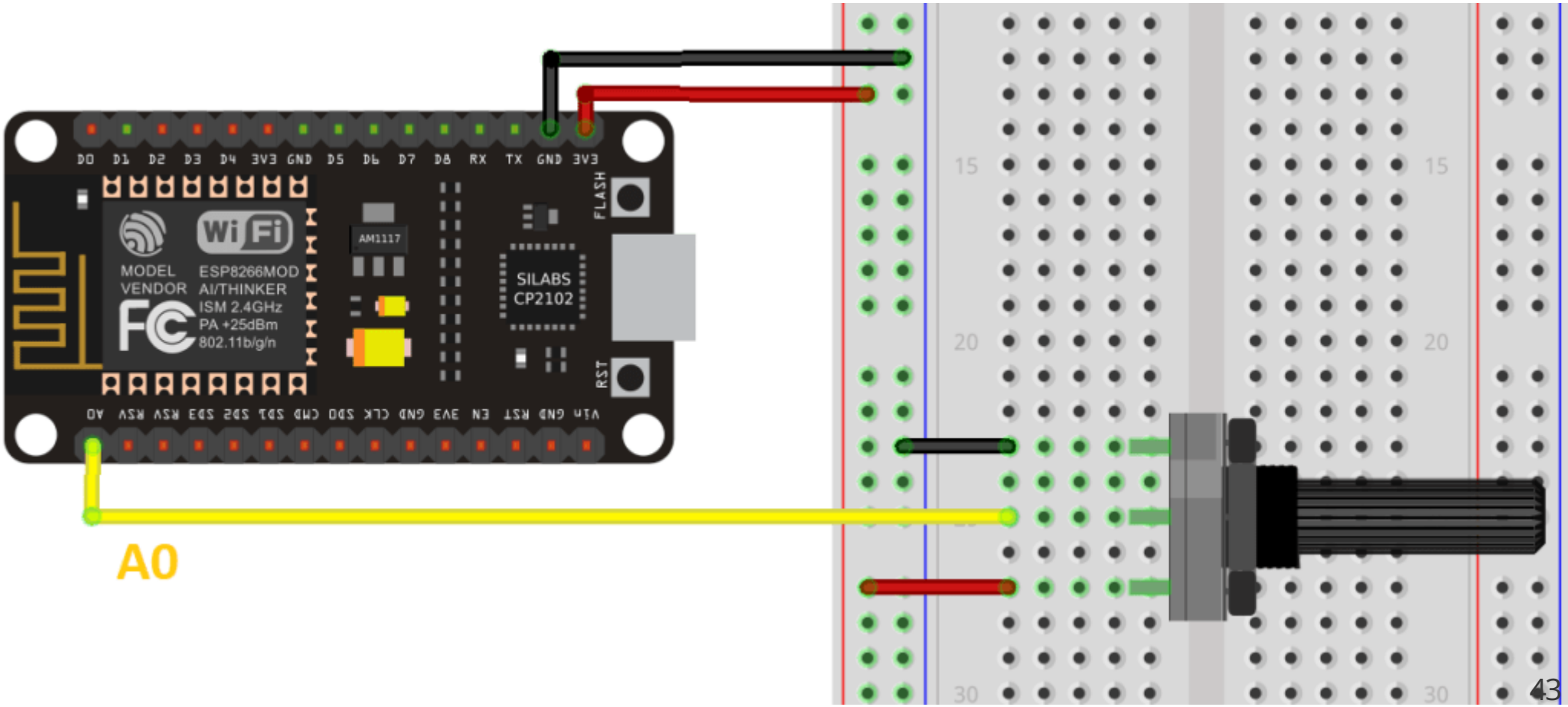
- ESP8266 ADC Resolution

The ADC pin has a 10-bit resolution, which means you'll get values between 0 and 1023.

- ESP8266 Input Voltage Range

The ESP8266 ADC pin input voltage range is 0 to 1V if you're using the bare chip. However, most ESP8266 development boards come with an internal voltage divider, so the input range is 0 to 3.3V. So, in sumary:

Schematic Diagram



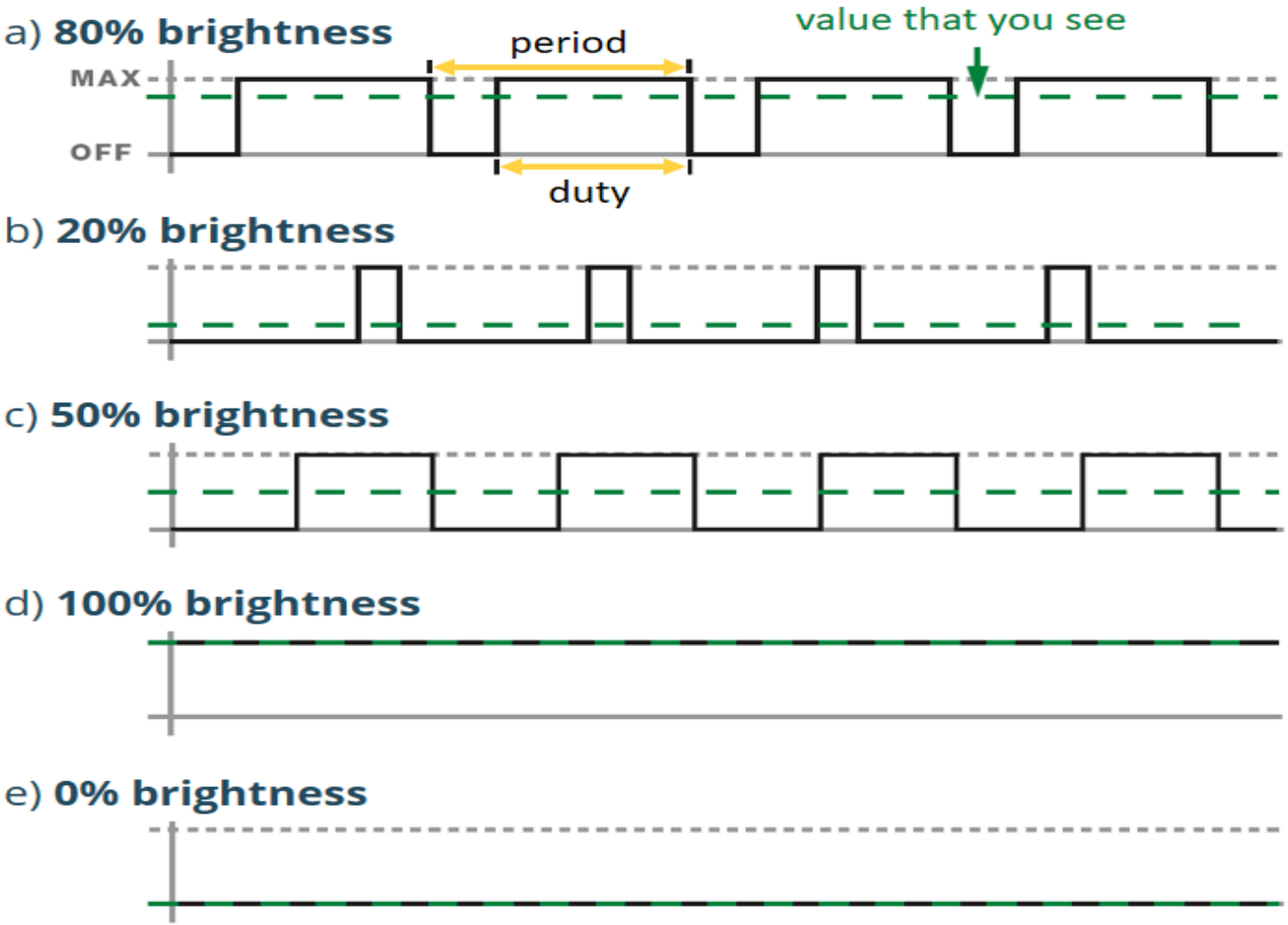
```
const int analogInPin = A0;  // ESP8266 Analog Pin ADC0 = A0

int sensorValue = 0;  // value read from the pot
int outputValue = 0;  // value to output to a PWM pin

void setup() {
  // initialize serial communication at 115200
  Serial.begin(115200);
}

void loop() {
  // read the analog in value
  sensorValue = analogRead(analogInPin);

  // map it to the range of the PWM out
  outputValue = map(sensorValue, 0, 1023, 0, 255);

  // print the readings in the Serial Monitor
  Serial.print("sensor = ");
  Serial.print(sensorValue);
  Serial.print("\t output = ");
  Serial.println(outputValue);

  delay(1000);
}
```

ESP8266 NodeMCU PWM

The ESP8266 GPIOs can be set either to output 0V or 3.3V, but they can't output any voltages in between. However, you can output "fake" mid-level voltages using pulse-width modulation (PWM), which is how you'll produce varying levels of LED brightness for this project.

If you alternate an LED's voltage between HIGH and LOW very fast, your eyes can't keep up with the speed at which the LED switches on and off; you'll simply see some gradations in brightness.That's basically how PWM works — by producing an output that changes between HIGH and LOW at a very high frequency.

The duty cycle is the fraction of time period at which LED is set to HIGH. The following figure illustrates how PWM works.


a) 80% brightness
period
value that you see
MAX
OFF
duty
b) 20% brightness
c) 50% brightness
d) 100% brightness
e) 0% brightness

analogWrite()

To produce a PWM signal on a given pin you use the following function:

```
analogWrite(pin, value);
```

- pin: PWM may be used on pins 0 to 16
- value: should be in range from 0 to PWMRANGE, which is 1023 by default. When value is 0, PWM is disable on that pin. A value of 1023 corresponds to 100% duty cycle

You can change the PWM range by calling:

```
analogWriteRange(new_range);
```

By default, ESP8266 PWM frequency is 1kHz. You can change PWM frequency with:

```
analogWriteFreq(new_frequency);
```

47

```
const int ledPin = 2;

void setup() {

}

void loop() {
  // increase the LED brightness
  for(int dutyCycle = 0; dutyCycle < 1023; dutyCycle++){
    // changing the LED brightness with PWM
    analogWrite(ledPin, dutyCycle);
    delay(1);
  }

  // decrease the LED brightness
  for(int dutyCycle = 1023; dutyCycle > 0; dutyCycle--){
    // changing the LED brightness with PWM
    analogWrite(ledPin, dutyCycle);
    delay(1);
  }
}
```

48

Thank you

Alireza Abdeshah
eMicro.ir@Gmail.com (mailto:eMicro.ir@Gmail.com)
http://iotplex.ir/ (http://iotplex.ir/)
@eMicro (http://twitter.com/eMicro)