



SNS COLLEGE OF TECHNOLOGY

Coimbatore-35

An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with
'A++' Grade

Approved by AICTE, New Delhi & Affiliated to Anna University,
Chennai



DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

19ECT213- IoT SYSTEM ARCHITECTURE

II ECE / IV SEMESTER

UNIT 2 – MICROCONTROLLER AND INTERFACING TECHNIQUES FOR IoT

DEVICES

TOPIC 2 –Introduction to NodeMCU



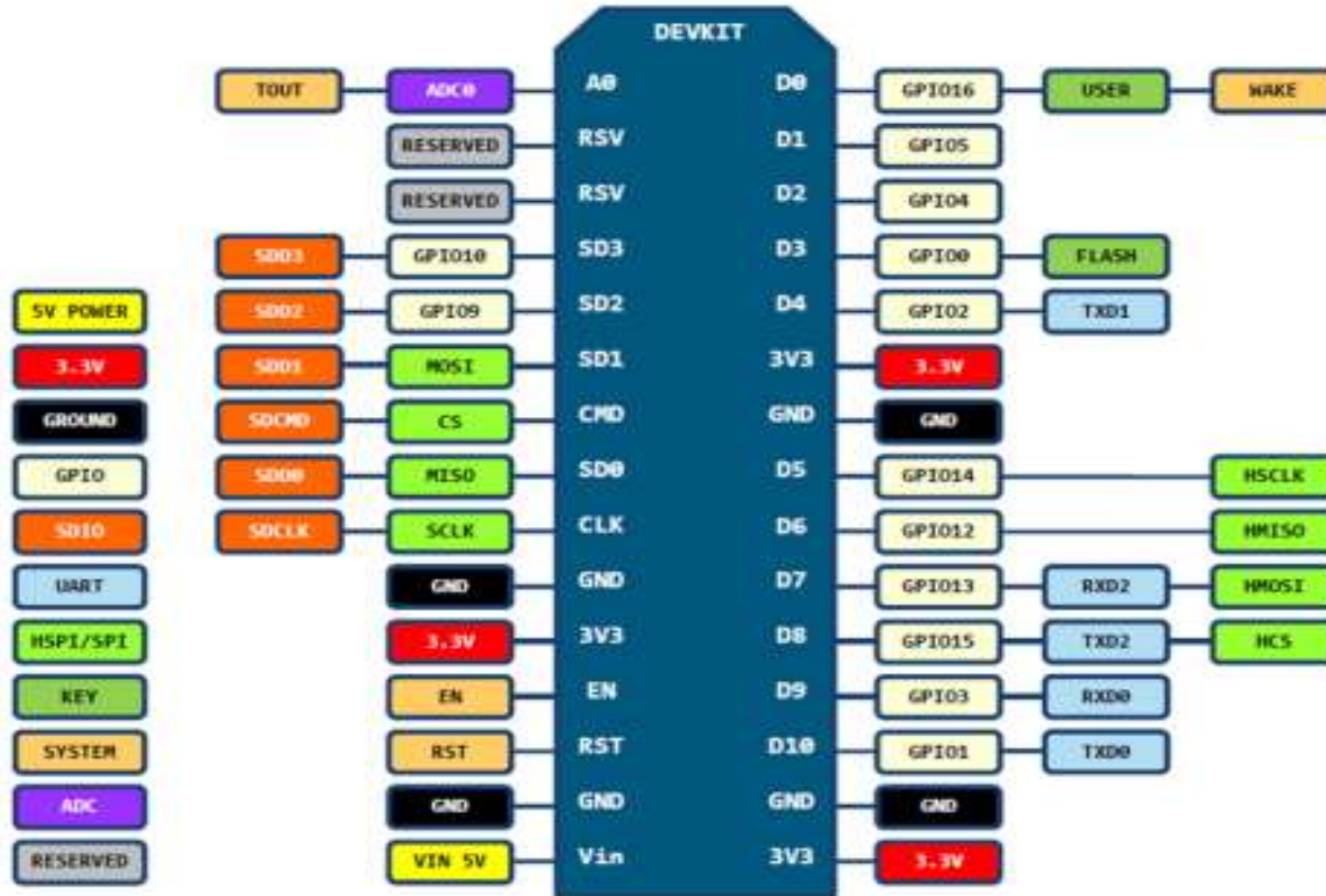
ESP8266 NodeMCU WiFi Development Board

Specification:

- Voltage: 3.3V.
- Wi-Fi Direct (P2P), soft-AP.
- Current consumption: 10uA~170mA.
- Flash memory attachable: 16MB max (512K normal).
- Integrated TCP/IP protocol stack.
- Processor: Tensilica L106 32-bit.
- Processor speed: 80~160MHz.
- RAM: 32K + 80K.
- GPIOs: 17 (multiplexed with other functions).
- Analog to Digital: 1 input with 1024 step resolution.
- +19.5dBm output power in 802.11b mode.
- 802.11 support: b/g/n.
- Maximum concurrent TCP connections: 5.



ESP8266 NodeMCU





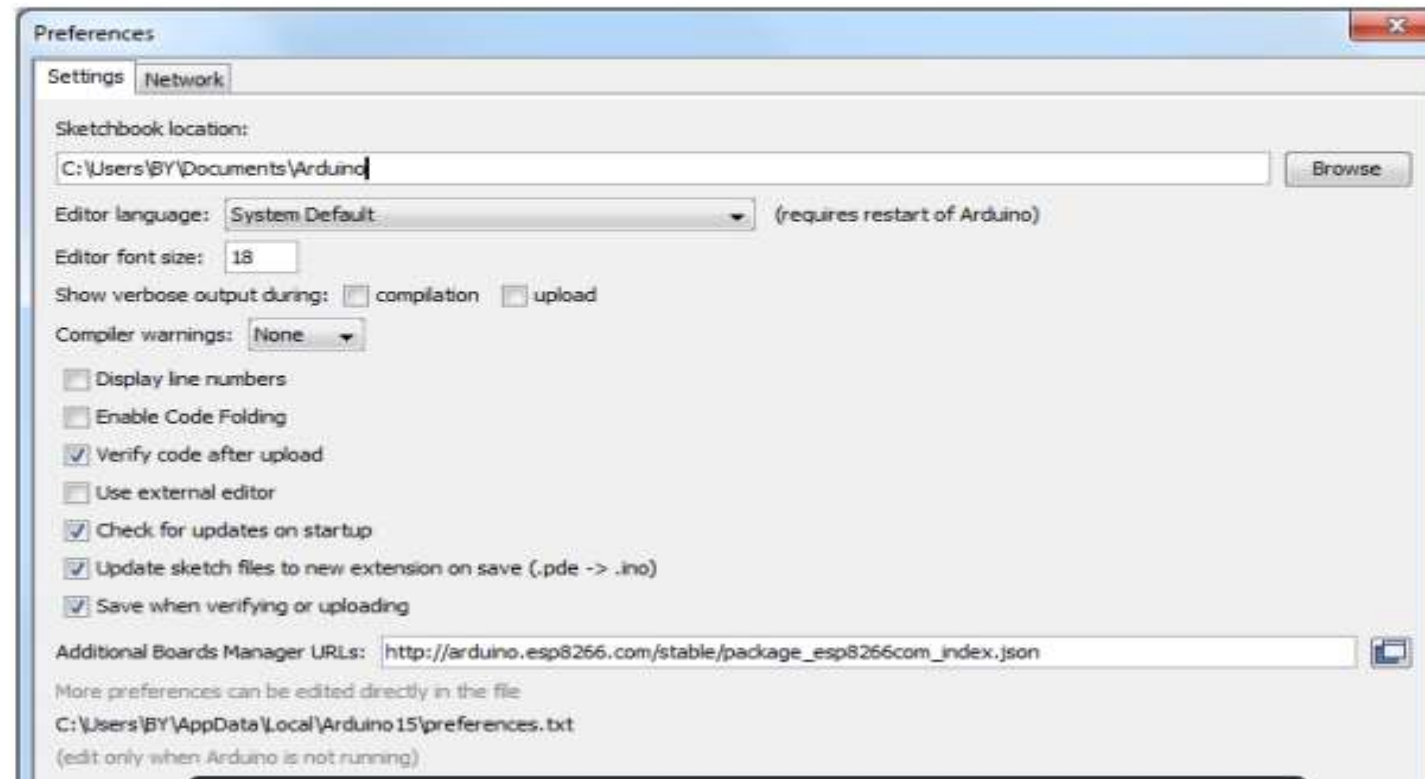
3.1 Install the Arduino IDE 1.6.4 or greater

[Download Arduino IDE from Arduino.cc \(1.6.4 or greater\) - don't use 1.6.2 or lower version! You can use your existing IDE if you have already installed it.](#)

[You can also try downloading the ready-to-go package from the ESP8266-Arduino project, if the proxy is giving you problems.](#)

3.2 Install the ESP8266 Board Package

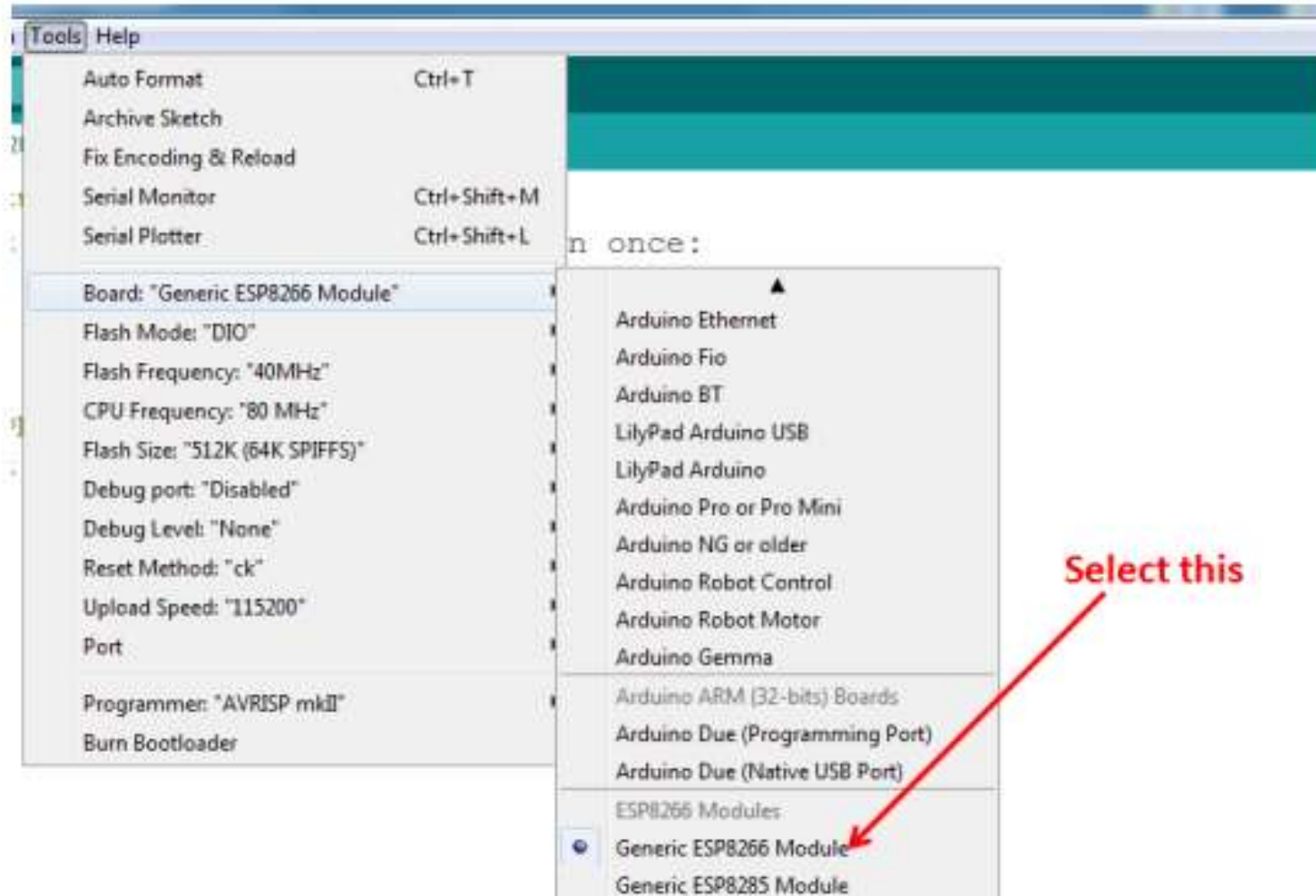
Enter **http://arduino.esp8266.com/stable/package_esp8266com_index.json** into *Additional Board Manage* field in the Arduino v1.6.4+ preferences.





3.3 Setup ESP8266 Support

When you've restarted Arduino IDE, select 'Generic ESP8266 Module' from the 'Tools' -> 'Board:' dropdown menu





The image shows two side-by-side screenshots. The left screenshot is of the Windows Device Manager, with the 'Ports (COM & LPT)' category expanded to show 'USB-SERIAL CH340 (COM11)' selected. The right screenshot is of the AVR Studio software interface, showing the 'Tools' menu with 'Serial Monitor' selected. A dropdown menu is open, showing 'Serial ports' with 'COM1' and 'COM11' listed, and 'COM11' is checked.

Find out which Com Port is assign for CH340

Select the correct Com Port as indicated on 'Device Manager'



Connecting via WiFi



3.4 Blink Test

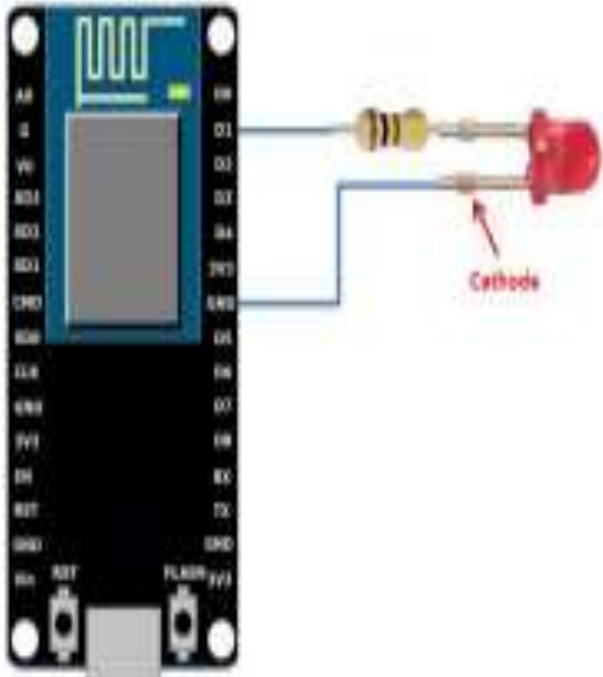
We'll begin with the simple blink test.

Enter this into the sketch window (and save since you'll have to). Connect a LED as shown in Figure3-1.

```
void setup() {  
  pinMode(5, OUTPUT); // GPIO05, Digital Pin D1  
}  
  
void loop() {  
  digitalWrite(5, HIGH);  
  delay(100);  
  digitalWrite(5, LOW);  
  delay(100);  
}
```

Now you'll need to put the board into bootload mode. You'll have to do this before each upload. There is no timeout for bootload mode, so you don't have to rush!

- Hold down the "Flash" button.
- While holding down "Flash", press the "RST" button.
- Release "RST", then release "Flash"



```
blinky | Arduino 1.6.7
File Edit Sketch Tools Help
blinky
void setup() {
  pinMode(5, OUTPUT); // GPIO05, Digital Pin D1
}

void loop() {
  digitalWrite(5, HIGH);
  delay(900);
  digitalWrite(5, LOW);
  delay(500);
}

Uploading...
WARNING: Spurious .github folder in 'Adafruit IO Arduino' library
WARNING: Spurious .tests folder in 'Adafruit IO Arduino' library
WARNING: Spurious .github folder in 'Adafruit MQTT Library' library
WARNING: Spurious .github folder in 'Adafruit IO Arduino' library
WARNING: Spurious .tests folder in 'Adafruit IO Arduino' library
WARNING: Spurious .github folder in 'Adafruit MQTT Library' library

Sketch uses 222,197 bytes (51%) of program storage space. Maximum is 434,160 bytes.
Global variables use 31,572 bytes (38%) of dynamic memory, leaving 50,348 bytes for local v
Uploading 226352 bytes from C:\Users\BY\AppData\Local\Temp\buildb7f3357d9ec338fa2a4043584dd
..... [ 36% ]
.....
Generic ESP8266 Module, 80 MHz, 40MHz, DIO, 115200, 512K(04K SPIFF
```


Connecting via WiFi

```
*/  
#include <ESP8266WiFi.h>
```

```
const char* ssid      = "handson";    // key in your own SSID  
const char* password = "abc1234";    // key in your own WiFi access point
```

```
password
```

```
const char* host = "www.handsontec.com";
```

```
void setup() {
```

```
  Serial.begin(115200);
```

```
  delay(100);
```

```
  // We start by connecting to a WiFi network
```

```
  Serial.println();
```

```
  Serial.println();
```

```
  Serial.print("Connecting to ");
```

```
  Serial.println(ssid);
```

```
  WiFi.begin(ssid, password);
```

```
  while (WiFi.status() != WL_CONNECTED) {
```

```
    delay(500);
```

```
    Serial.print(".");
```

```
  }
```

```
  Serial.println("");
```

```
  Serial.println("WiFi connected");
```

```
  Serial.println("IP address: ");
```

```
  Serial.println(WiFi.localIP());
```

```
}
```

```

int value = 0;

void loop() {
  delay(5000);
  ++value;

  Serial.print("connecting to ");
  Serial.println(host);

  // Use WiFiClient class to create TCP connections
  WiFiClient client;
  const int httpPort = 80;
  if (!client.connect(host, httpPort)) {
    Serial.println("connection failed");
    return;
  }

  // We now create a URI for the request
  String url = "/projects/index.html";
  Serial.print("Requesting URL: ");
  Serial.println(url);

  // This will send the request to the server
  client.print(String("GET ") + url + " HTTP/1.1\r\n" +
              "Host: " + host + "\r\n" +
              "Connection: close\r\n\r\n");

  delay(500);

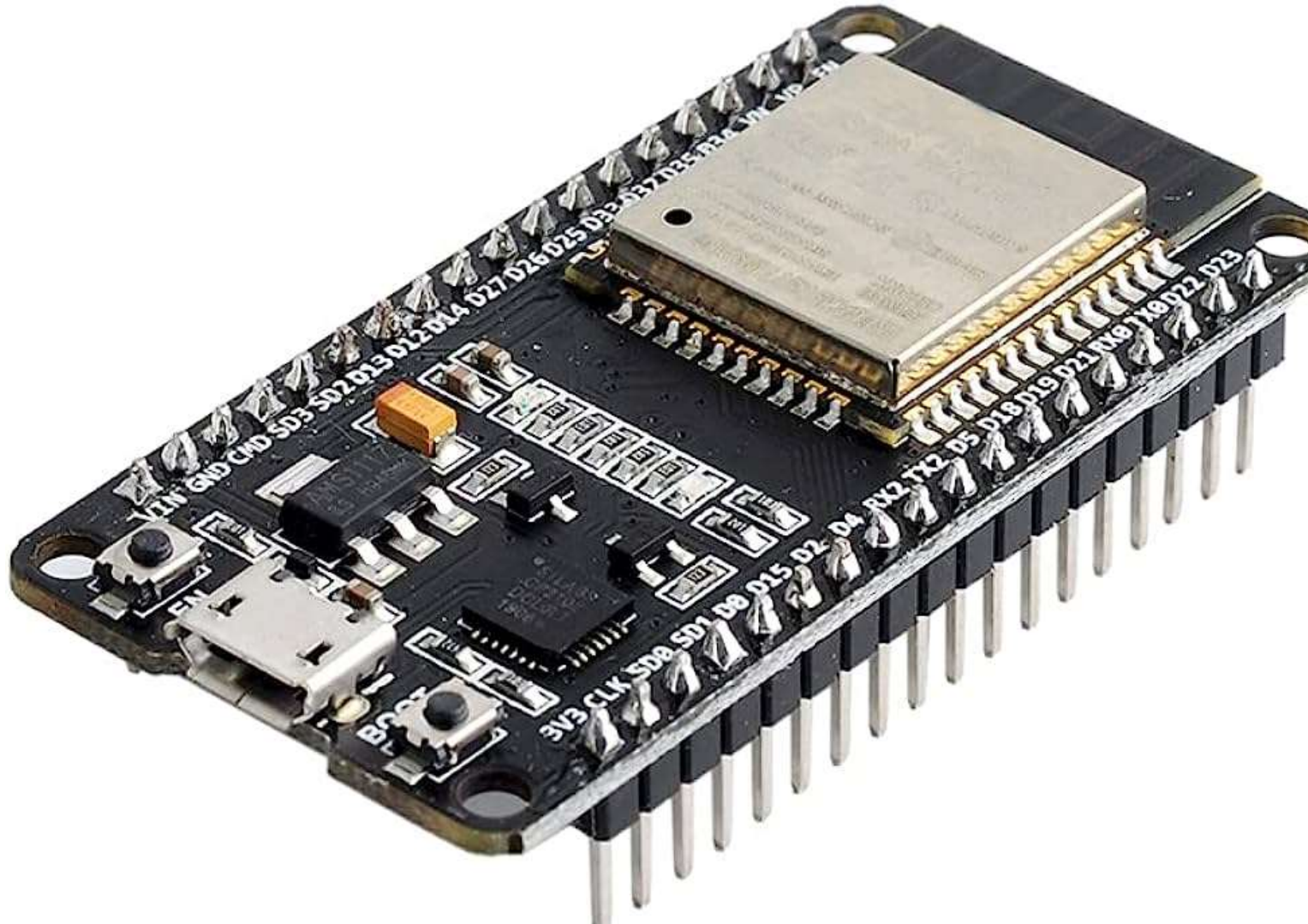
  // Read all the lines of the reply from server and print them to Serial
  while(client.available()){
    String line = client.readStringUntil('\r');
    Serial.print(line);
  }

  Serial.println();
  Serial.println("closing connection");
}

```



WHAT IS ESP32





WHAT IS ESP32



- ESP32 is a series of low cost, low power system on a chip microcontrollers with integrated Wi-Fi & dual-mode Bluetooth.
- CPU: Xtensa Dual-Core 32-bit LX6 microprocessor, operating at 160 or 240 MHz and performing at up to 600 DMIPS
- - Memory: 520 KiB SRAM
- - Wireless connectivity:
 - + Wi-Fi: 802.11 b/g/n/e/i
 - + Bluetooth: v4.2 BR/EDR and BLE



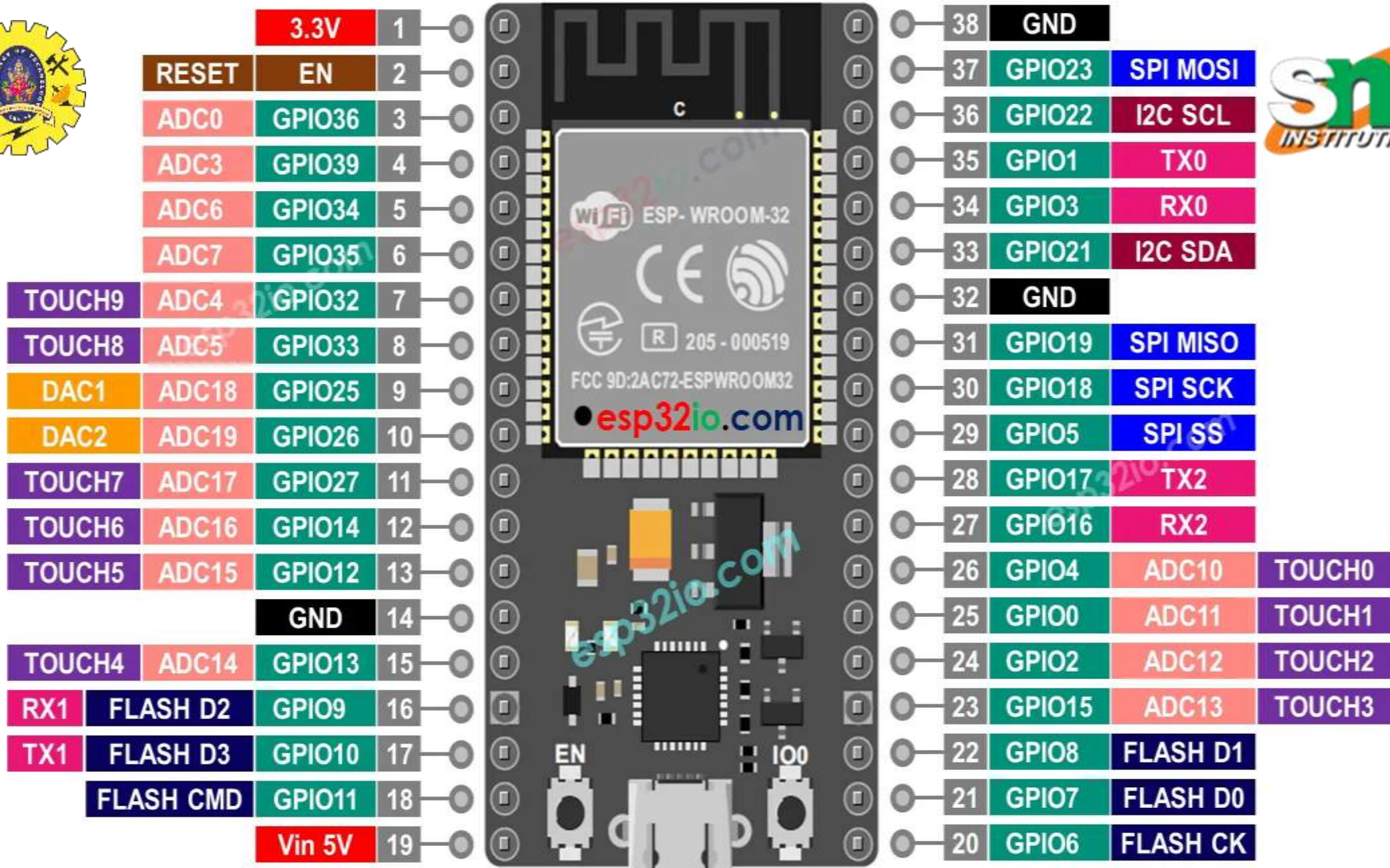
WHAT IS ESP32?



- Integrated Crystal- 40 MHz
- Module Interfaces- UART, SPI, I2C, PWM, ADC, DAC, GPIO, pulse counter, capacitive touch sensor
- Integrated SPI flash- 4 MB
- ROM- 448 KB (for booting and core functions)
- SRAM- 520 KB
- Integrated Connectivity Protocols- WiFi, Bluetooth, BLE
- On-chip sensor- Hall sensor
- Operating temperature range- -40 - 85 degrees Celsius
- Operating Voltage- 3.3V
- Operating Current- 80 mA (average)



ESP32





The ESP32 peripherals include:



- 18 Analog-to-Digital Converter (ADC) channels
- 3 SPI interfaces
- 3 UART interfaces
- 2 I2C interfaces
- 16 PWM output channels
- 2 Digital-to-Analog Converters (DAC)
- 2 I2S interfaces
- 10 Capacitive sensing GPIOs



ESP32



Input only pin

GPIOs 34 to 39 are GPIOs – input only pins. These pins don't have internal pull-up or pull-down resistors. They can't be used as outputs, so use these pins only as inputs:

GPIO 34

GPIO 35

GPIO 36

GPIO 39

SPI flash integrated on the ESP-WROOM-32

GPIO 6 to GPIO 11 are exposed in some ESP32 development boards. However, these pins are connected to the integrated SPI flash on the ESP-WROOM-32 chip and are not recommended for other uses. So, don't use these pins in your projects:

GPIO 6 (SCK/CLK)

GPIO 7 (SDO/SD0)

GPIO 8 (SDI/SD1)

GPIO 9 (SHD/SD2)

GPIO 10 (SWP/SD3)

GPIO 11 (CSC/CMD)



ESP32



The ESP32 has 10 internal capacitive touch sensors. These can sense variations in anything that holds an electrical charge, like the human skin. So they can detect variations induced when touching the GPIOs with a finger. These pins can be easily integrated into capacitive pads and replace mechanical buttons. The capacitive touch pins can also be used to wake up the ESP32 from deep sleep.

Those internal touch sensors are connected to these GPIOs:

- T0 (GPIO 4)
- T1 (GPIO 0)
- T2 (GPIO 2)
- T3 (GPIO 15)
- T4 (GPIO 13)
- T5 (GPIO 12)
- T6 (GPIO 14)
- T7 (GPIO 27)
- T8 (GPIO 33)
- T9 (GPIO 32)

5/8/2024



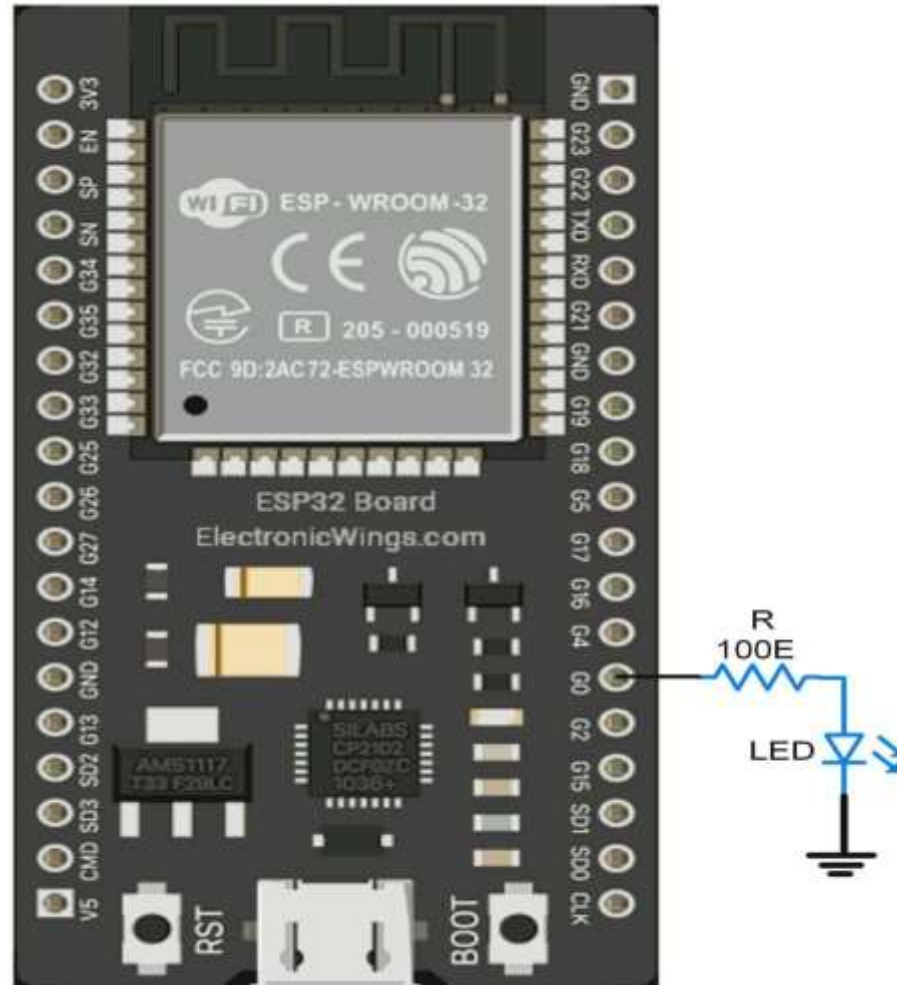
ESP32



- **Analog to Digital Converter (ADC)**
- ADC1_CH0 (GPIO 36)
- ADC1_CH1 (GPIO 37)
- ADC1_CH2 (GPIO 38)
- ADC1_CH3 (GPIO 39)
- ADC1_CH4 (GPIO 32)
- ADC1_CH5 (GPIO 33)
- ADC1_CH6 (GPIO 34)
- ADC1_CH7 (GPIO 35)
- ADC2_CH0 (GPIO 4)
- ADC2_CH1 (GPIO 0)
- ADC2_CH2 (GPIO 2)
- ADC2_CH3 (GPIO 15)
- ADC2_CH4 (GPIO 13)
- ADC2_CH5 (GPIO 12)
- ADC2_CH6 (GPIO 14)
- ADC2_CH7 (GPIO 27)
- ADC2_CH8 (GPIO 25)
- ADC2_CH9 (GPIO 26)



Blink LED Using ESP32





Blink LED Using ESP32



```
void setup()
{
  pinMode(0, OUTPUT);    // sets the digital pin 0 as output
}
```

```
void loop()
{
  digitalWrite(0, HIGH); // sets the digital pin 0 on
  delay(1000);           // waits for a second
  digitalWrite(0, LOW);  // sets the digital pin 0 off
  delay(1000);           // waits for a second
}
```



Blink LED Using ESP32



Output

