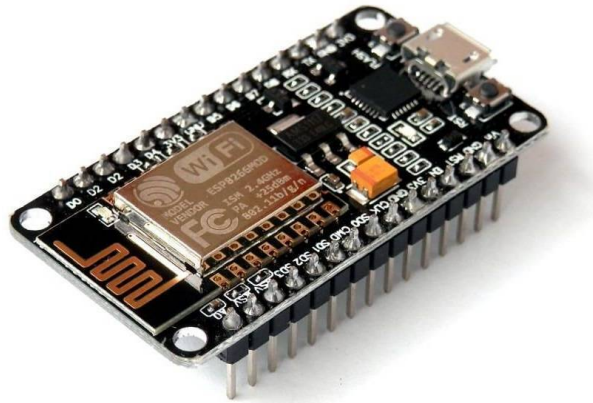


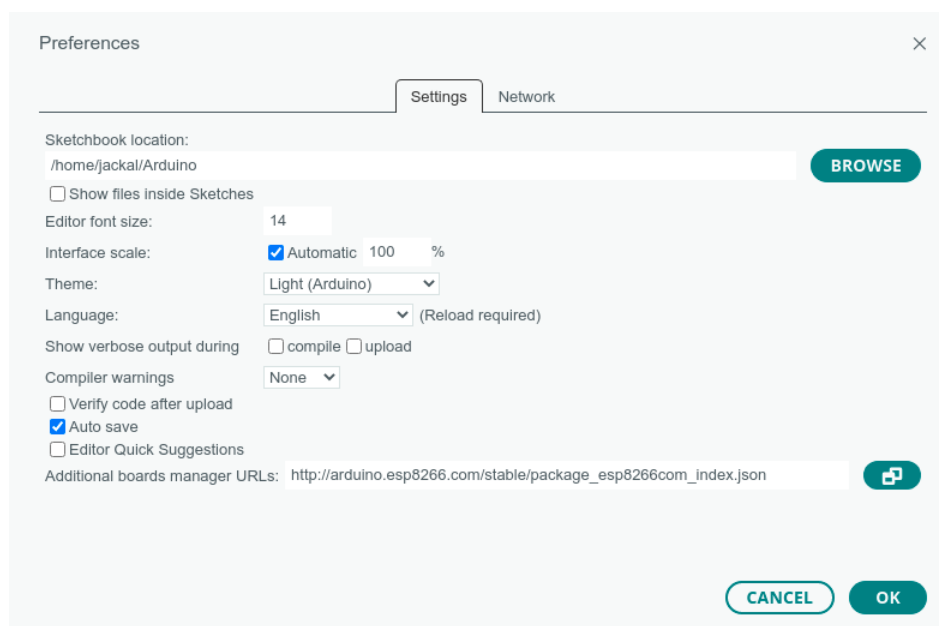
Introduction to ESP8266

ESP8266: The ESP8266 (shown in below figure by HallRoad.org) is a Wi-Fi enabled microcontroller that is commonly used in IoT (Internet of Things) projects. It is a low-cost, low-power system-on-a-chip (SoC) with a Tensilica L106 32-bit microcontroller unit (MCU) at its core. The ESP8266 has built-in Wi-Fi connectivity, which allows it to connect to the internet or a local network and communicate with other devices. We can program it using the same famous software; Arduino IDE.

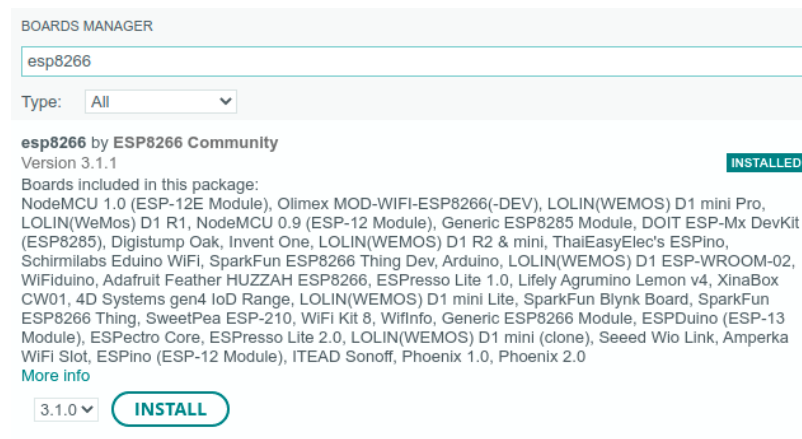


Arduino IDE: Arduino IDE is a software in which one can edit the code and upload on Arduino-based devices such as Arduino Uno, Arduino Mega, ESP8266 and so on.

Adding additional board manager URL: To install libraries for ESP8266 we need to add its respective URL to add this in our board manager of Aduino IDE. And for that we will have to open File >> Preferences and paste http://arduino.esp8266.com/stable/package_esp8266com_index.json as shown below.



Installing ESP8266 Libraries: Go to Tools >> Board >> Board Manager and search ESP8266 and install as shown below.



WiFi modes of ESP8266: ESP8266 can work in 3 modes; as WiFi Station, WiFi Access Point and both at the same time.

As WiFi Station: ESP8266 can connect to visible networks in its surrounding.

As WiFi Access Point: Other devices can connect to ESP8266.

WiFi Scan: ESP8266 can scan the networks within its range, counts the networks, prints their SSID with its respective signal strength in dbs. It will work as Wi-Fi station in this scenario.

Code:

```
#include <ESP8266WiFi.h>

void setup() {
  Serial.begin(115200);
  WiFi.mode(WIFI_STA);
}

void loop() {
  int n = WiFi.scanNetworks();
  Serial.println("Scanning for Wi-Fi networks...");
  if (n == 0) {
    Serial.println("No networks found.");
  } else {
    Serial.print(n);
    Serial.println(" networks found.");
    for (int i = 0; i < n; ++i) {
      Serial.print(i + 1);
      Serial.print(": ");
      Serial.print(WiFi.SSID(i));
      Serial.print(" ");
      Serial.print(WiFi.RSSI(i));
      Serial.println("");
      delay(10);
    }
  }
  delay(5000); // Wait 5 seconds before scanning again.
}
```

Code Explanation:

- Importing the library first (ESP8266WiFi).
- Setting the baud rate and the mode (Station in this case) in void setup using **begin(buad_rate)** function..
- Counting the WiFi networks with **scanNetworks()** fuction.
- Printing the number of scanned networks if found.
- Applying for loop to list the networks found with their SSID and signal strength using **SSID(WiFi_network_index)** and **RSSI(WiFi_network_index)** functions respectively.
- And it will keep repeating as it in loop function.
- Delay of 5 seconds after each scan.

Output: Here we can see the networks are being constantly listed below. Can be seen by opening serial monitor and make sure the baud rate is set to 115200 (in this case), otherwise it should match the baud rate defined in code.

```
Scanning for Wi-Fi networks...
4 networks found.
1: CRAIB-LAB (-64)
2: SmartLife-FE46 (-77)
3: DIRECT-c5-HP M132 LaserJet (-80)
4: CRAIB-LAB (-63)
Scanning for Wi-Fi networks...
4 networks found.
1: CRAIB-LAB (-66)
2: SmartLife-FE46 (-76)
3: DIRECT-c5-HP M132 LaserJet (-75)
```

ESP8266 connecting to a network: We will connect our ESP8266 to a network nearby our WiFi network and also here it will work as WiFi station as in below given code.

```
#include <ESP8266WiFi.h> // Include the Wi-Fi library
```

```
String ssid = "CRAIB-LAB"; // The SSID (name) of the Wi-Fi network you want to connect to
```

```
String password = "*****"; // Your password of the Wi-Fi network
```

```
void setup() {
```

```
  Serial.begin(115200); // Start the Serial communication to send messages to the computer
```

```
  delay(10);
```

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

```
  WiFi.begin(ssid, password); // Connect to the network
```

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

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

```
  int i = 0;
```

```
  while (WiFi.status() != WL_CONNECTED) { // Wait for the Wi-Fi to connect
```

```
    delay(1000);
```

```
    Serial.print(++i); Serial.print(' ');
```

```
  }
```

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

```
  Serial.print("Connection established with ");
```

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

```
  Serial.print("IP address:\t");
```

```
  Serial.println(WiFi.localIP()); // Send the IP address of the ESP8266 to the computer
```

```
}
```

```
void loop() { }
```

Code Explanation:

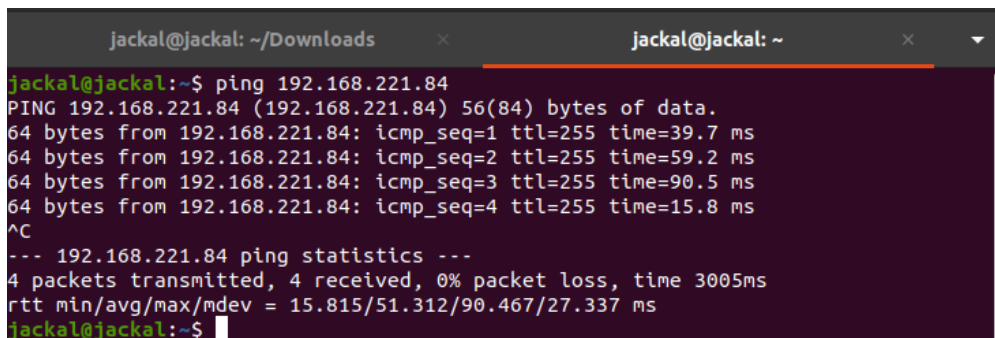
- Importing the library first (ESP8266WiFi).
- Defining credentials for WiFi network to be connected as SSID and Password in string form.
- Setting the baud rate as 115200 to communicate to computer.
- Connecting with WiFi using **begin(ssid,password)** function by providing respective SSID and password.
- Waiting for the WiFi to connect and update time in seconds using while loop as it is true until ESP8266 gets connected to network.
- Once Connection established (means now outside while loop) program prints the message of connection established and sends its IP to the computer using **localIP()** function.

Output: Below you can see ESP8266 is connected to CRAIB-LAB network.

```
Connecting to CRAIB-LAB ...
1 2 3 4 5 6

Connection established with CRAIB-LAB
IP address:      192.168.221.84
```

Validating Connection: We can validate its connectivity by pinging this IP by executing **ping <IP>** command in your terminal. If its connected properly, bytes or packets (messages) should be transferred to it successfully.



```
jackal@jackal: ~/Downloads
jackal@jackal: ~
jackal@jackal:~$ ping 192.168.221.84
PING 192.168.221.84 (192.168.221.84) 56(84) bytes of data.
64 bytes from 192.168.221.84: icmp_seq=1 ttl=255 time=39.7 ms
64 bytes from 192.168.221.84: icmp_seq=2 ttl=255 time=59.2 ms
64 bytes from 192.168.221.84: icmp_seq=3 ttl=255 time=90.5 ms
64 bytes from 192.168.221.84: icmp_seq=4 ttl=255 time=15.8 ms
^C
--- 192.168.221.84 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3005ms
rtt min/avg/max/mdev = 15.815/51.312/90.467/27.337 ms
jackal@jackal:~$
```

Output: Above one can see that all 4 bytes are sent successfully with 0% loss.

ESP8266 as an access point: As discussed ESP8266 can behave as an access point. Now we will connect our device(s) (smartphone, ESP8266, computers or other micro-controllers) as shown in below code.

```
#include <ESP8266WiFi.h> // Include the Wi-Fi library
String ssid = "ESP8266 Access Point"; // The name of the Wi-Fi network that will be created
String password = "12345678"; // The password required to connect to it, leave blank for an open network
void setup() {
  Serial.begin(115200);
  delay(10);
  Serial.println("\n");
}
```

```

void loop() {
  WiFi.softAP(ssid, password); // Start the access point
  Serial.print("Access Point \");
  Serial.print(ssid);
  Serial.println("\n started");
  Serial.print("IP address:\t");
  Serial.println(WiFi.softAPIP()); }

```

Code Explanation:

- Importing the library first (ESP8266WiFi).
- Defining credentials for WiFi network of ESP8266 for external devices to connect.
- Setting baud rate in void function.
- Starting the access point using **softAP(ssid,pass)** function and printing its SSID connection message.
- And finally printing the access point IP (APIP) for the other devices to connect **softAPIP()** function, so that we can see it on serial monitor.

Output: Below we can see that our access point is started with its respective IP. Now we can connect to ESP8266 either via SSID or IP and with the password that we have set.

```

Access Point "ESP8266 Access Point" started
IP address:      192.168.4.1
Access Point "ESP8266 Access Point" started
IP address:      192.168.4.1
Access Point "ESP8266 Access Point" started
IP address:      192.168.4.1

```

Connecting device with ESP8266: As ESP8266 has created an access network means it is open to be connected with devices as shown below.

