ESP8266 Cheatsheet

A comprehensive guide to the ESP8266 microcontroller, covering essential hardware details, programming with Arduino IDE, networking, and common troubleshooting tips. Ideal for quick reference and project development.

Hardware Essentials

Pinout Overview

The ESP8266 has multiple variants, but the ESP-12E is the most common.

Key Pins:

- **VCC:** 3.3V Power
- **GND:** Ground
- **GPIO0 & GPIO2:** General Purpose Input/Output pins (important for flashing)
- **CH_PD (EN):** Chip Enable (High to enable)
- **RST:** Reset (Low to reset)
- **TXD & RXD:** UART Communication

Important Note: The ESP8266 is *not* 5V tolerant. Applying 5V to any pin can damage the chip. Always use a 3.3V power supply and level shifters if interfacing with 5V logic.

Powering the ESP8266

Voltage	ESP8266 requires a stable 3.3V power supply. A dedicated 3.3V regulator is recommended.
Current	ESP8266 can draw up to 300mA during WiFi transmission. Ensure your power supply can provide sufficient current.
Decoupling Capacitor	Use a 100nF capacitor close to the VCC pin to reduce noise and improve stability.

Boot Modes

Flash Mode	GPIO0: LOW, GPIO2: HIGH - Used for flashing new firmware.
Normal Boot Mode	GPIO0: HIGH, GPIO2: HIGH - Executes the program in flash memory.
Serial Communication	Use a USB-to-Serial adapter (e.g., FTDI) to communicate with the ESP8266 via UART.

Arduino IDE Setup

Board Manager Installation

1. Open Arduino IDE.
2. Go to `File > Preferences`.
3. Add `http://arduino.esp8266.com/stable/package_esp8266com_index.json` to 'Additional Board Manager URLs'.
4. Go to `Tools > Board > Boards Manager...`.
5. Search for `esp8266` and install the `esp8266 by ESP8266 Community` package.

Board Selection

After installation, select your ESP8266 board from `Tools > Board`. Common options include:

- `Generic ESP8266 Module`
- `NodeMCU 1.0 (ESP-12E Module)`
- `WeMos D1 R2`

Basic Code Structure

```
void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
  Serial.println("Hello, ESP8266!");
}

void loop() {
  // put your main code here, to run
repeatedly:
  delay(1000);
  Serial.println("Looping...");
}
```

Common Libraries

`ESP8266WiFi.h`	For connecting to WiFi networks.
`WiFiClient.h`	For creating TCP client connections.
`ESP8266WebServer.h`	For creating a web server on the ESP8266.

Networking with ESP8266

Connecting to WiFi

```
#include <ESP8266WiFi.h>

const char* ssid = "your_SSID";
const char* password = "your_PASSWORD";

void setup() {
  Serial.begin(115200);
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("\nWiFi connected");
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());
}

void loop() {
  // Your code here
}
```

Creating a Web Server

```
#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>

const char* ssid = "your_SSID";
const char* password = "your_PASSWORD";

ESP8266WebServer server(80);

void handleRoot() {
  server.send(200, "text/plain", "Hello from
ESP8266!");
}

void setup() {
  Serial.begin(115200);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("WiFi connected");

  server.on("/", handleRoot);
  server.begin();
  Serial.println("Web server started");
}

void loop() {
  server.handleClient();
}
```

Access Point (AP) Mode

```
#include <ESP8266WiFi.h>

void setup() {
  Serial.begin(115200);
  WiFi.softAP("ESP8266-AP", "password");

  IPAddress IP = WiFi.softAPIP();
  Serial.print("AP IP address: ");
  Serial.println(IP);
}

void loop() {
  // Your code here
}
```

Troubleshooting and Tips

Common Issues

- **ESP not booting:** Check power supply (3.3V, sufficient current), wiring, and boot mode configuration (GPIO0, GPIO2).
- **WiFi connection issues:** Verify SSID and password, check WiFi signal strength, and ensure DHCP is enabled on your router.
- **Flashing errors:** Use the correct baud rate (usually 115200), ensure the ESP is in flash mode, and try a different USB cable/port.

Debugging Techniques

Serial Monitor	Use `Serial.print()` and `Serial.println()` to output debugging information to the Arduino IDE's Serial Monitor.
Exception Decoder	In case of crashes, use the Exception Decoder tool in the Arduino IDE (Tools > ESP Exception Decoder) to get more information about the error.
WiFi Scan	Use `WiFi.scanNetworks()` to scan for available WiFi networks and check signal strength.

Power Saving

`WiFi.mode(WIFI_OFF)`	Turn off WiFi to reduce power consumption.
`WiFi.disconnect()`	Disconnect from the WiFi network.
`delay(ms)`	Use `delay()` to put the ESP8266 into a light sleep mode. For deeper sleep use `ESP.deepSleep(microseconds)`.