

# ESP32 ARDUINO

[Pages Wiki original](#)

L'**ESP32** est un SOC développé par la société Espressif dédié à l'internet des objets (**IoT**) et plus particulièrement les communications sans fil Wifi et Bluetooth pour un coût réduit. Il a été décliné en version module l'**ESP-WROOM-32** qui a lui même été intégré par différents fabricants (Essentiellement Chinois) sur des cartes de développement.

Il présente plusieurs intérêts :

- Prix faible ~5€ pour le module ~8€ pour une petite carte de developpement
- Intègre du Wifi 802.11 b/g/n/e/i (WPA)/WPA2/WPA2-Enterprise/Wi-Fi Protected Setup (WPS)
- Intègre Bluetooth **4.2** - BLE Bluetooth low Energy
- Compatible avec l'environnement de développement **ARDUINO**
- Intègre un microcontrôleur 32 bits performants et de nombreux périphériques (ADC **12bit**, DAC, 3xUART, PWM, I2C, SPI, etc ...)
- S'alimente directement en USB
- etc.



## Caractéristiques

### CPU

Xtensa® single-/dual-core 32-bit LX6 microprocessor(s), up to 600 DMIPS (200 DMIPS for single-core microprocessor)

- 448 kB ROM
- 520 kB SRAM
- 16 kB SRAM in RTC
- QSPI flash/SRAM, up to 4 x 16 MB
- Power supply: 2.3V to 3.6V

### Hardware

- Périphériques : SD card, UART, SPI, SDIO, I2C, LED PWM, Motor

PWM, I2S, IR

- GPIO : capacitive touch sensor, ADC, DAC, LNA preamplifier
- Capteurs intégrés : Hall sensor, temperature sensor
- Alimentation : 2.7 ~ 3.6V
- Courant max : 500 mA
- Température de fonctionnement : -40°C ~ +85°C

## WIFI

- Wi-Fi Protocols : 802.11 b/g/n/e/i (802.11n up to 150 Mbps)
- Wi-Fi mode : Station/SoftAP/SoftAP+Station/P2P
- Wi-Fi Security : WPA/WPA2/WPA2-Enterprise/WPS
- Network protocols : IPv4, IPv6, SSL, TCP/UDP/HTTP/FTP/MQTT

## OS

- FreeRTOS

## Interruptions

[Interruptions ESP32](#)

## Brochage

## PINOUT

[esp32-wroom-32-pinout.png](#)

## ESP-WROOM-32

Spécifications du module ESP32 (ESP-WROOM-32)

L'ESP32 est une évolution importante de l'ESP8266. En plus du WiFi, elle apporte le support du Bluetooth basse énergie (4.0 LE). Elle corrige également le manque d'entrées/sorties. On dispose maintenant de 32 E/S dont 26 digitales et 18 analogiques (toujours avec une tension admissible maximale de 3.3V). On pourra également utiliser des écrans tactiles pour créer des interfaces homme/machine à base d'ESP32 (broches Touch 0 à 9). Pour le moment, la plupart des modules sont proposés avec 4MB de mémoire flash mais la puce peut supporter jusqu'à 16MB.

Module ESP32 : ESP-WROOM-32 d'Espressif. Microprocesseur dual core cadencé à 240MHz équipé de 4MB de mémoire flash SPI. Support jusqu'à 16MB de mémoire flash

Connectivité :

WiFi 802.11 b/g/n. Sécurité WEP, WPA/WPA2 PSK/Enterprise. Puce cryptographique intégrée prenant en charge les algorithmes AES/SHA2/Elliptical Curve Cryptography/RSA-4096 Puissance maximale pour le transfert de données : 19.5 dBm@11b, 16.5 dBm@11g, 15.5 dBm@11n Sensibilité max. de réception : -97 dBm Bluetooth 4.0 LE 32 Entrées/Sorties 26x E/S digitales (3.3V). Toutes les sorties peuvent être PWM 18x entrées analogiques 3x UART 3x SPI 2x I2S 2x DAC 2x I2C Consommation en mode sommeil (Deep Sleep mode) : 5 µA Capteurs intégrés Effet Hall 10x entrées pour interface tactile capacitive

## Installation de l'environnement de développement

### Procédure d'installation de l'environnement ARDUINO

- Installer l'environnement ARDUINO : <https://www.arduino.cc/en/Main/Software>
- Télécharger l'add-on pour l'ESP32 : <https://github.com/espressif/arduino-esp32>
- Dézipper l'add-on dans le répertoire : Mes Documents\Arduino\hardware\espressif\esp32



\* Exécuter get.exe qui se trouve dans le répertoire Mes Documents\Arduino\hardware\espressif\esp32\tools \* L'exécutable get.exe va télécharger et intégrer l'add-on dans l'environnement ARDUINO



L'exécutable get.exe va télécharger 3 fichiers ZIP qui seront téléchargés dans le répertoire dist (esptool-4dab24e-windows.zip, mkspiffs-0.2.1-windows.zip et xtensa-esp32-elf-win32-1.22.0-75-gbaf03c2-5.2.0.zip)

\* Démarrer l'environnement ARDUINO et vérifier que l'ESP32 apparaît bien dans les types de carte :



\* Brancher votre carte ESP32, vérifier le port COM qui lui a été attribué :



\* Créer un nouveau sketch, sélectionner la carte: ESP32 Dev Module et le port qui lui a été associé



\* Vérifier la compilation et le transfert

## 1er sketch de test - clignotement LED



Ma carte intègre une LED sur la sortie IO2

```
void setup() {  
  // put your setup code here, to run once:  
  pinMode(2, OUTPUT);          // set pin to output  
}  
  
void loop() {  
  // put your main code here, to run repeatedly:  
  
  digitalWrite(2, HIGH);      // sets the LED on  
  delay(200);                  // waits for a second  
  digitalWrite(2, LOW);       // sets the LED off  
}
```

```
delay(800);  
}
```

## Sketch exemples



De nombreux exemples sont fournis dans le répertoire :  
Documents\Arduino\hardware\espressif\esp32\libraries

## Sketch transmission wifi

Le sketch ci dessous va transmettre une trame toutes les secondes vers un serveur UDP. Remplacer par **your-ssid** par le nom de votre réseau WIFI, **your-password** par le mot de passe de votre réseau WIFI et 192.168.1.10 par l'adresse IP de votre PC.

### [exemple001.ino](#)

```
/*  
 * This sketch sends random data over UDP on a ESP32 device  
 */  
#include <WiFi.h>  
#include <WiFiUdp.h>  
  
// WiFi network name and password:  
const char * networkName = "**your-ssid**";  
const char * networkPswd = "**your-password**";  
  
//IP address to send UDP data to:  
// either use the ip address of the server or  
// a network broadcast address  
const char * udpAddress = "**192.168.1.10**";  
const int udpPort = 2205;  
  
//Are we currently connected?  
boolean connected = false;  
  
//The udp library class  
WiFiUDP udp;  
  
void setup(){  
  // Initilize hardware serial:  
  Serial.begin(115200);  
  
  //Connect to the WiFi network  
  connectToWiFi(networkName, networkPswd);  
}
```

```
}

void loop(){
  //only send data when connected
  if(connected){
    //Send a packet
    udp.beginPacket(udpAddress,udpPort);
    udp.printf("Seconds since boot: %u", millis()/1000);
    udp.endPacket();
  }
  //Wait for 1 second
  delay(1000);
}

void connectToWiFi(const char * ssid, const char * pwd){
  Serial.println("Connecting to WiFi network: " + String(ssid));

  // delete old config
  WiFi.disconnect(true);
  //register event handler
  WiFi.onEvent(WiFiEvent);

  //Initiate connection
  WiFi.begin(ssid, pwd);

  Serial.println("Waiting for WIFI connection...");
}

//wifi event handler
void WiFiEvent(WiFiEvent_t event){
  switch(event) {
    case SYSTEM_EVENT_STA_GOT_IP:
      //When connected set
      Serial.print("WiFi connected! IP address: ");
      Serial.println(WiFi.localIP());
      //initializes the UDP state
      //This initializes the transfer buffer
      udp.begin(WiFi.localIP(),udpPort);
      connected = true;
      break;
    case SYSTEM_EVENT_STA_DISCONNECTED:
      Serial.println("WiFi lost connection");
      connected = false;
      break;
  }
}
```

L'ESP32 va transmettre toutes les secondes une trame vers le port 3333 de votre PC. Pour visualiser les trames envoyées, il suffit d'utiliser un serveur UDP comme [HERCULE](https://www.magenealogie.chanterie37.fr/www/fablab37110/)

Démarrer HERCULE et dans l'onglet UDP mettre 3333 dans le port à utiliser puis cliquer sur "Listen"



## Liens utiles

- [Site Expressif sur le ESP32](#)
- [Datasheet ESP32](#)
- [Datsheet carte ESP WROOM 32](#)
- [Tutos sur l'ESP32 EN](#)
- [Projets DIY - Excellent site avec pas mal d'infos en français](#)

[Schema de la carte de developpement ESP32 DEVKIT V1](#)

From:

<https://www.magenealogie.chanterie37.fr/www/fablab37110/> - Castel'Lab le Fablab MJC de Château-Renault

Permanent link:

<https://www.magenealogie.chanterie37.fr/www/fablab37110/doku.php?id=start:arduino:esp32:wikiexterne>

Last update: **2024/06/14 16:09**

