

AZ-Delivery

Bienvenue !

Merci d'avoir acheté notre AZ-Delivery ESP-32 Dev Kit C V2. Dans les pages suivantes, nous allons vous présenter comment utiliser et configurer cet appareil pratique.

Amusez-vous bien !



Sommaire

Introduction	3
Caractéristiques	4
ESP32 Dev Kit C V2	5
Pinout	6
Description des broches	7
Broches du capteur tactile capacitif	8
Broches du convertisseur analogique-numérique	9
Broches du convertisseur numérique-analogique	9
Broches GPIO de l'horloge en temps réel	10
Broches PWM (Pulse Width Modulation)	11
Broches de l'interface I2C	11
Broches de l'interface SPI	12
Broches de fixation	12
Broches HIGH au démarrage	13
Activation (EN)	13
Communication USB vers série	14
Communication WiFi	15
Communication Bluetooth	16
Autres caractéristiques	18
Comment configurer l'IDE Arduino	19
Configuration supplémentaire	23
ESP32 Dev Kit C V2 : exemple de câblage	27
Exemples de sketches	28

Introduction

L'*ESP32 Dev Kit C V2* est une carte de développement créée autour de la puce *ESP32 WROOM 32*, contenant un régulateur de tension et un circuit de programmation *USB* pour la puce *ESP32*, et quelques autres fonctionnalités.

Pour le développement d'applications, vous avez le choix entre *Arduino IDE* et *ESP-IDF* (plate-forme native). La plupart des utilisateurs choisissent l'*Arduino IDE* en raison de sa simplicité et de sa compatibilité. La communauté des utilisateurs d'*Arduino* est très active et soutient des plateformes telles que l'*ESP32*.

ESP32 Dev Kit C V2 est livré avec un firmware préinstallé qui permet de travailler avec le langage interprété, en envoyant des commandes via le port série (puce *CP2102*). Les cartes *ESP32* sont l'une des plateformes les plus utilisées pour les projets d'Internet des objets (*IoT*).

La carte *ESP32 Dev Kit C V2* est spécialement conçue pour fonctionner sur breadboard. Elle possède un régulateur de tension qui lui permet de s'alimenter directement à partir du port *USB*. Les broches d'entrée/sortie fonctionnent à 3,3V. La puce *CP2102* est responsable de la communication *USB* vers série.



Caractéristiques

Tension d'alimentation (USB)	5V DC
Tension d'entrée/sortie	3.3V DC
Courant de fonctionnement requis	min. 500mA
SoC	ESP32-WROOM 32
CPU	Xtensa® single-dual-core 32-bit LX6
Gamme de fréquences d'horloge	80MHz / 240MHz
RAM	512kB
Mémoire flash externe	4MB
I/O pins	34
ADC canaux	18
ADC Résolution	12-bit
DAC canaux	2
DAC Résolution	8-bit
Interfaces de communication	SPI, I2C, I2S, CAN, UART
Protocoles Wi-Fi	802.11 b/g/n (802.11n jusqu'à 150 Mbps)
Fréquence Wi-Fi	2.4 GHz - 2.5 GHz
Bluetooth	V4.2 - BLE et Bluetooth classique
Antenne Wireless	PCB
Dimensions	56x28x13mm(2.2x1.1x0.5in)

ESP32 Dev Kit C V2

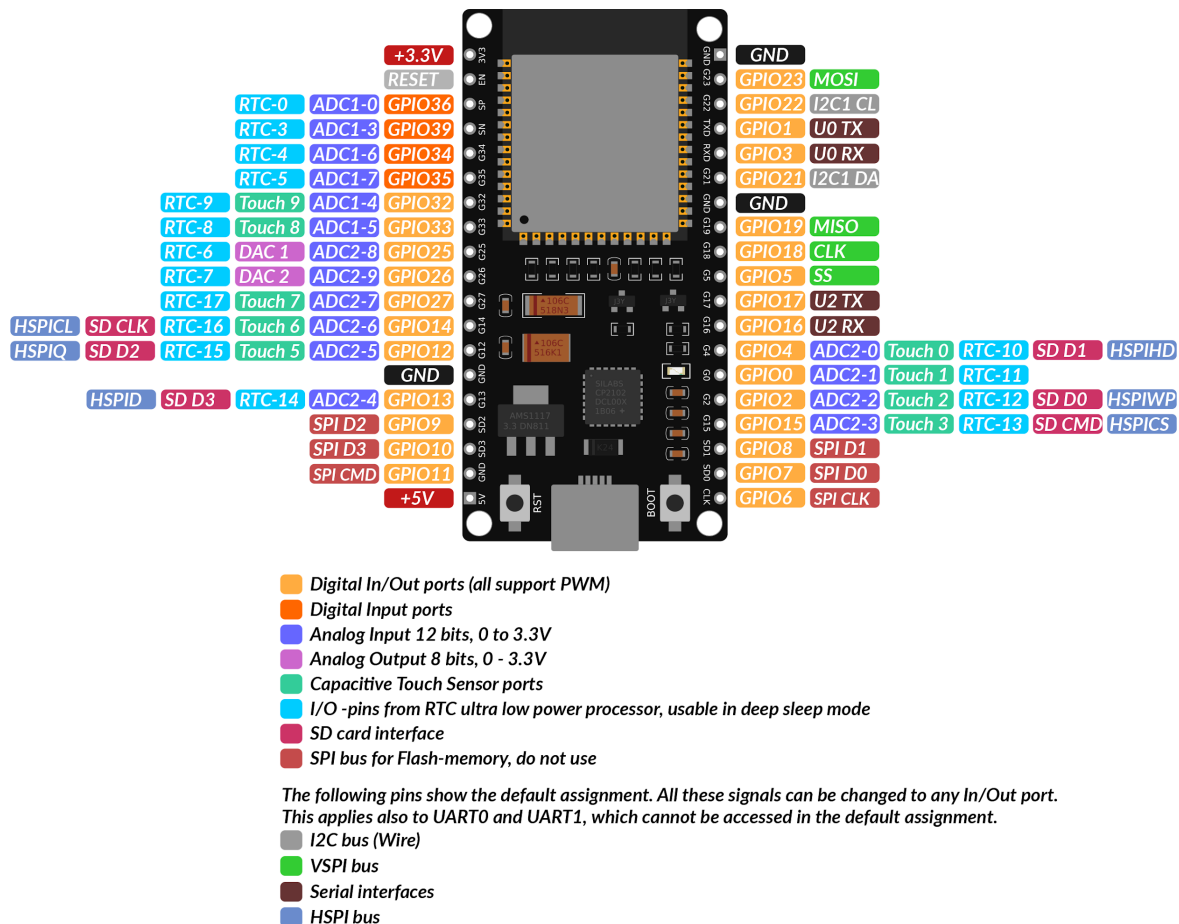
La série de puces *Wi-Fi ESP32 WROOM 32* est produite par *Espressif Systems*. L'*ESP32 WROOM-32* est un module *Wi-Fi* abordable adapté aux projets de bricolage dans le domaine de l'*Internet des objets (IoT)*. Ce module est livré avec de nombreux *GPIO* et supporte une variété de protocoles comme *SPI*, *I2C*, *I2S*, *UART*, et plus encore. La meilleure partie est qu'il est livré avec un réseau sans fil inclus, ce qui le rend différent des autres microcontrôleurs comme l'*Arduino*. Cela signifie qu'il peut facilement contrôler et surveiller des appareils à distance via *Wi-Fi* et *Bluetooth®* à un prix accessible.

ESP32 WROOM 32 est un système sur puce (SoC) intégrant dans un petit boîtier un microcontrôleur *Tensilica 32 bits*, des interfaces périphériques numériques standard, des commutateurs d'antenne, un balun *RF*, un amplificateur de puissance, un amplificateur de réception à faible bruit, des filtres et des modules de gestion de l'alimentation. Il offre une communication sans fil *Wi-Fi 2,4GHz (802.11 b/g/n)*, supportant des vitesses jusqu'à *150MB/s*, *BLE* et *Bluetooth®* classique, 34 broches d'E/S, des interfaces *I2C* et *I2S*, *ADC* (conversion analogique-numérique), *DAC* (conversion numérique-analogique), une interface *SPI*, *UART* sur broches dédiées et *PWM (Pulse Width Modulation)*.

Le cœur du processeur, appelé *LX6* par *Espressif*, est basé sur le contrôleur de processeur *Xtensa® dual-core 32-bit LX6* et fonctionne à une fréquence comprise entre *80* et *240MHz*. Il dispose d'une *ROM* de démarrage de *448 Ko*, d'une *SRAM* intégrée de *520 Ko* et d'une mémoire flash externe de *4 Mo* accessible par l'interface *SPI*.

Le Pinout

L'ESP32 Dev Kit C V2 possède 38 broches. Le brochage est présenté sur l'image suivante :



Pour une description détaillée du brochage et des capacités E/S, veuillez vous référer à la fiche technique qui se trouve sur ce [link](#).

NOTE : Le courant maximal absolu consommé par une GPIO est de 40 mA, conformément à la section "Recommended Operating Conditions " de la fiche technique de l'ESP32.

Description des pins

Tout comme une carte Arduino normale, l'*ESP32 Dev Kit C V2* possède des broches d'entrée/sortie numériques (broches *GPIO* - General Purpose Input/Output pins). Ces entrées/sorties numériques fonctionnent à 3,3 V.

La tension de 5V ne doit pas être connectée aux broches de la puce ESP32 !

Les broches ne sont pas tolérantes à 5V, appliquer plus de 3,3V sur une broche détruira la puce.

Les broches *GPIO 34* à *39* sont des *GPI* - broches d'entrée uniquement. Ces broches n'ont pas de résistances internes de type *pull-up* ou *pull-down*. Elles ne peuvent pas être utilisées comme sorties, donc utilisez ces broches uniquement comme entrées : *GPIO 34*, *GPIO 35*, *GPIO 36*, *GPIO 39*

Il y a une *flash SPI* intégrée sur la puce *ESP-WROOM-32*. Les broches *GPIO6* à *GPIO 11* sont exposées dans certaines cartes de développement *ESP32*. Ces broches sont connectées à la *flash SPI* intégrée sur la puce et ne sont pas recommandées pour d'autres utilisations.

GPIO 6 (SCK/CLK), *GPIO 7* (SDO/SD0), *GPIO 8* (SDI/SD1), *GPIO 9* (SHD/SD2), *GPIO 10* (SWP/SD3), *GPIO 11* (CSC/CMD).

Broches du capteur tactile capacitif

L'*ESP32* possède 10 capteurs tactiles capacitifs internes. Les broches tactiles capacitives peuvent également être utilisées pour réveiller l'*ESP32* de son sommeil profond. Ces capteurs tactiles internes sont connectés aux *GPIO* suivants : 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).

Broches du convertisseur analogique-numérique

L'*ESP32* possède 18 canaux d'entrée *ADC* (convertisseur analogique-numérique) de 12 bits (alors que l'*ESP8266* ne possède qu'un *ADC* de 10 bits). Ce sont les *GPIOs* qui peuvent être utilisés comme *ADC* et les canaux respectifs :

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).

Broches du convertisseur numérique-analogique

L'*ESP32* dispose de 2 canaux *DAC* (convertisseur numérique-analogique) de 8 bits pour convertir les signaux numériques en signaux analogiques de sortie. Ce sont les canaux *DAC* :

DAC1 (GPIO25), *DAC2* (GPIO26).



Broches GPIO Real Time Clock

L'*ESP32* prend en charge les *GPIO* de la *RTC* (horloge en temps réel).

Les *GPIO* orientés vers le sous-système à faible consommation de la *RTC* peuvent être utilisés lorsque l'*ESP32* est en veille profonde. Ces

GPIO RTC peuvent être utilisés pour réveiller l'*ESP32* de son sommeil profond lorsque le coprocesseur *Ultra Low Power (ULP)* est actif. Les

GPIO suivants peuvent être utilisés comme source de réveil externe :

RTC_GPIO0 (GPIO36), RTC_GPIO3 (GPIO39), RTC_GPIO4 (GPIO34),

RTC_GPIO5 (GPIO35), RTC_GPIO6 (GPIO25), RTC_GPIO7 (GPIO26),

RTC_GPIO8 (GPIO33), RTC_GPIO9 (GPIO32), RTC_GPIO10 (GPIO4),

RTC_GPIO11 (GPIO0), RTC_GPIO12 (GPIO2), RTC_GPIO13

(GPIO15), RTC_GPIO14 (GPIO13), RTC_GPIO15 (GPIO12),

RTC_GPIO16 (GPIO14), RTC_GPIO17 (GPIO27).

Pin PWM (Pulse Width Modulation)

Le contrôleur *PWM* (modulation de largeur d'impulsion) pour *LED ESP32* possède 16 canaux indépendants qui peuvent être configurés pour générer des signaux *PWM* avec différentes propriétés. Toutes les broches qui peuvent agir comme des sorties peuvent être utilisées comme broches *PWM* (les *GPIOs* 34 à 39 ne peuvent pas générer de *PWM*). Pour configurer un signal *PWM*, vous devez définir ces paramètres dans le code : La fréquence du signal, le rapport cyclique, le canal *PWM*, le *GPIO* où vous voulez sortir le signal.

Les broches de l'interface I2C

L'*ESP32* possède deux canaux *I2C* et n'importe quelle broche peut être définie comme *SDA* ou *SCL*. Lorsque vous utilisez l'*ESP32* avec l'*IDE Arduino*, les broches *I2C* par défaut sont :

GPIO 21 (SDA), GPIO 22 (SCL).

Pins de l'interface SPI

Par défaut, le mappage des broches *SPI* est le suivant :

SPI	MOSI	MISO	CLK	CS
VSPI	GPIO 23	GPIO 19	GPIO 18	GPIO 5
HSPI	GPIO 13	GPIO 12	GPIO 14	GPIO 15

Pins de fixation

Les broches suivantes sont utilisées pour mettre l'*ESP32* en mode bootloader ou flashage :

GPIO 0, *GPIO 2*, *GPIO 4*, *GPIO 5* (doit être HIGH pendant le démarrage), *GPIO 12* (doit être LOW pendant le démarrage), *GPIO 15* (doit être HIGH pendant le démarrage).

La plupart des cartes de développement mettent les broches dans le bon état pour le flashage ou le mode de démarrage. Si certains périphériques sont connectés aux broches de fixation et que l'*IDE* ne parvient pas à télécharger le code ou à flasher l'*ESP32*, c'est peut-être parce que ces périphériques empêchent l'*ESP32* d'entrer dans le bon mode. Après la réinitialisation, le flashage ou le démarrage, ces broches fonctionnent comme prévu. Il existe un guide de documentation sur la sélection du mode de démarrage sur le lien suivant. Des explications plus approfondies ne sont pas dans le cadre de cet eBook, veuillez donc vous référer à la fiche technique.

Pins HIGH au Boot

Certains GPIOs changent leur état en HIGH ou émettent des signaux PWM au démarrage ou à la réinitialisation. Cela signifie que si des sorties sont connectées à ces GPIO, on peut obtenir des résultats inattendus lorsque l'ESP32 se réinitialise ou démarre.

GPIO 1, GPIO 3, GPIO 5, GPIO 6 à GPIO 11 (connecté à la mémoire *flash SPI* intégrée de l'ESP32 - utilisation non recommandée), *GPIO 14, GPIO 15*.

Enable (EN)

Enable (EN) est la broche d'activation du régulateur 3,3V. Elle a un état tiré vers le haut et doit être connectée à la surface pour désactiver le régulateur 3,3V. Cela signifie que cette broche peut être connectée à un bouton poussoir pour redémarrer votre ESP32, par exemple.



Communication USB vers série

L'*ESP32 Dev Kit C V2* possède un port de connexion *microUSB*. Il est constitué d'une puce *CP21202* fabriquée par *Silicon Laboratories* qui permet une communication série *USB* vers *UART*. La puce possède la fonction de port *COM virtuel (VCP)* qui apparaît comme un port *COM* dans les applications *PC*. L'interface *UART CP2102* met en œuvre tous les signaux *RS-232*, y compris les signaux de contrôle et de *handshaking*, de sorte qu'il n'est pas nécessaire de modifier le *firmware* du système existant. Pour pouvoir utiliser l'*ESP32*, il faut installer le pilote.

Communication WiFi

L'*ESP32 Dev Kit C V2* dispose d'une interface de communication *Wi-Fi* intégrée et peut fonctionner en trois modes différents : *Station Wi-Fi*, point d'accès *Wi-Fi*, et les deux en même temps. Il prend en charge les fonctionnalités suivantes :

- Débits de données 802.11b et 802.11g
- 802.11n MCS0-7 en bande passante de 20MHz et 40MHz
- 802.11n MCS32
- Intervalle de garde de 0,4μS pour 802.11n
- Débit de données jusqu'à 150 Mbps
- Réception STBC 2x1
- Jusqu'à 20 dBm de puissance d'émission
- Puissance d'émission réglable
- Diversité et sélection des antennes (matériel géré par logiciel)

Communication Bluetooth

L'*ESP32 Dev Kit C V2* possède une radio *Bluetooth* intégrée et prend en charge les fonctions suivantes :

- Puissances de sortie d'émission de Class-1, Class-2 et Class-3 et gamme de contrôle dynamique de plus de 30 dB
- Modulation $\pi/4$ DQPSK et 8 DPSK
- Haute performance dans la sensibilité du récepteur NZIF avec une gamme dynamique de plus de 98 dB
- Fonctionnement en Class-1 sans PA externe
- La SRAM interne permet un transfert de données à pleine vitesse, un mélange de voix et de données, et un fonctionnement en mode piconet complet.
- Logique pour la correction d'erreur directe, le contrôle d'erreur d'en-tête, la corrélation de code d'accès, le CRC, la démodulation, la génération de flux de bits de cryptage, le traitement des blancs et la mise en forme des impulsions d'émission.
- ACL, SCO, eSCO et AFH
- CODEC audio numérique A-law, μ -law et CVSD en interface PCM
- SBC audio CODEC
- Gestion de l'alimentation pour les applications à faible consommation
- SMP avec AES 128 bits

En outre, la radio *Bluetooth* prend en charge les protocoles d'interface de communication suivants :

- Interface UART HCI, jusqu'à 4 Mbps
- Interface HCI SDIO / SPI
- Interface I2C
- Interface audio PCM / I2S.

Autres caractéristiques

La puce *ESP32-WROOM 32D* est dotée d'un capteur à *effet Hall* intégré qui détecte les variations du champ magnétique dans son environnement.


Le capteur de *Hall* est basé sur une résistance à *N-porteuse*. Lorsque la puce se trouve dans le champ magnétique, le capteur de *Hall* développe une petite tension sur la résistance, qui peut être mesurée directement par le convertisseur analogique-numérique (*CAN*), ou amplifiée par le préamplificateur analogique à très faible bruit, puis mesurée par le *CAN*.

Le capteur de température génère une tension qui varie avec la température. Cette tension est convertie en interne en un code numérique via un convertisseur analogique-numérique. Le capteur de température a une gamme de -40°C à 125°C . Étant donné que le décalage du capteur de température varie d'une puce à l'autre en raison de la variation du processus, ainsi que de la chaleur générée par le circuit *Wi-Fi* lui-même (qui affecte les mesures), le capteur de température interne ne convient qu'aux applications qui détectent les changements de température au lieu des températures absolues et à des fins d'étalonnage également. Toutefois, si l'utilisateur étalonne le capteur de température et utilise l'appareil dans une application à alimentation minimale, les résultats pourraient être suffisamment précis.

Comment configurer l'IDE Arduino

Si l'*Arduino IDE* n'est pas installé, suivez le lien et téléchargez le fichier d'installation pour le système d'exploitation de votre choix. La version d'*Arduino IDE* utilisée pour cet eBook est 1.8.13.

Download the Arduino IDE




ARDUINO 1.8.13

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software.

This software can be used with any Arduino board. Refer to the [Getting Started](#) page for Installation instructions.

Windows Installer, for Windows 7 and up
Windows ZIP file for non admin install

Windows app Requires Win 8.1 or 10  Get

Mac OS X 10.10 or newer

Linux 32 bits
Linux 64 bits
Linux ARM 32 bits
Linux ARM 64 bits

[Release Notes](#)
[Source Code](#)
[Checksums \(sha512\)](#)

Pour les utilisateurs de *Windows*, double-cliquez sur le fichier .exe téléchargé et suivez les instructions de la fenêtre d'installation.

Az-Delivery

Pour les utilisateurs de *Linux*, téléchargez un fichier portant l'extension *.tar.xz*, qui doit être extrait. Lorsqu'il est extrait, allez dans le répertoire extrait et ouvrez le terminal dans ce répertoire. Deux scripts *.sh* doivent être exécutés, le premier appelé *arduino-linux-setup.sh* et le deuxième appelé *install.sh*.

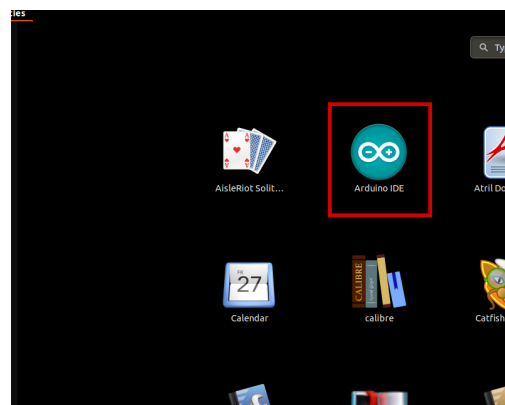
Pour exécuter le premier script dans le terminal, ouvrez le terminal dans le répertoire extrait et exécutez la commande suivante :

sh arduino-linux-setup.sh user_name

user_name - est le nom d'un superutilisateur dans le système d'exploitation Linux. Un mot de passe pour le superutilisateur doit être saisi au moment du lancement de la commande. Attendez quelques minutes pour que le script complète tout.

Le second script, appelé *install.sh*, doit être utilisé après l'installation du premier script. Exécutez la commande suivante dans le terminal (répertoire extrait) : **sh install.sh**

Après l'installation de ces scripts, allez dans le dossier *All Apps*, où est installé l'*IDE Arduino*.

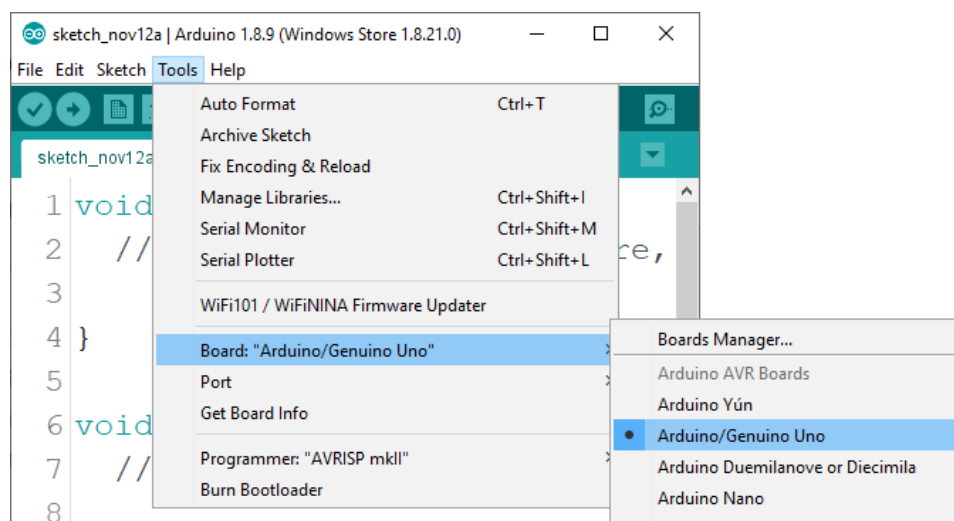


Presque tous les systèmes d'exploitation sont livrés avec un éditeur de texte préinstallé (par exemple, *Windows* est livré avec *Notepad*, *Linux Ubuntu* avec *Gedit*, *Linux Raspbian* avec *Leafpad*, etc.) Tous ces éditeurs de texte conviennent parfaitement à l'objectif de l'eBook.

La prochaine étape est de vérifier si votre PC peut détecter une carte Arduino. Ouvrez l'*IDE Arduino* fraîchement installé, et allez dans :

Tools > Board > {votre nom de conseil ici}

{votre nom de conseil ici} devrait être l'Arduino/Genuino Uno, comme on peut le voir sur l'image suivante :



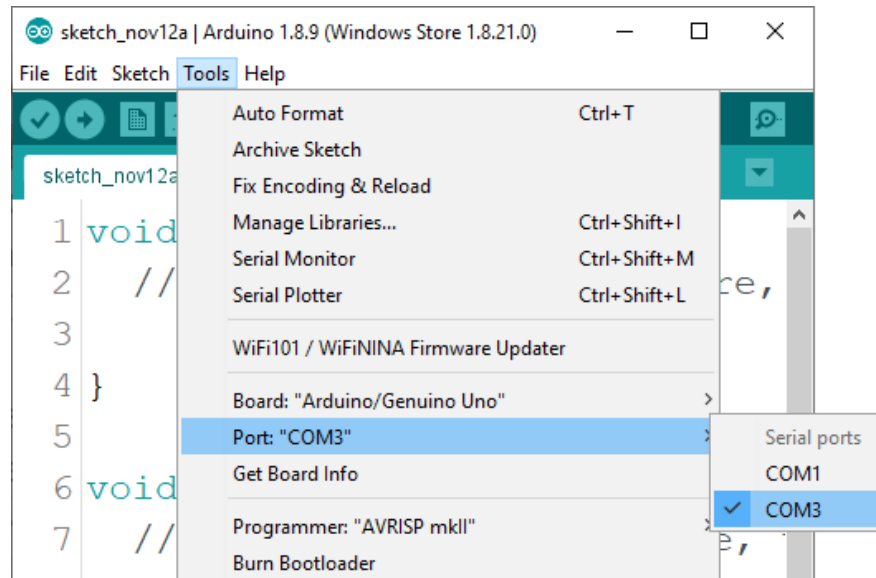
Le port auquel la carte Arduino est connectée doit être sélectionné. Aller à :

Tools > Port > {le nom du port va ici}

et lorsque la carte Arduino est connectée au port USB, le nom du port peut être vu dans le menu déroulant de l'image précédente.

Az-Delivery

Si l'*Arduino IDE* est utilisé sous *Windows*, les noms des ports sont les suivants :



Pour les utilisateurs de Linux, par exemple, le nom du port est */dev/ttyUSBx*, où *x* représente un nombre entier entre 0 et 9.



Configuration supplémentaire

Pour utiliser l'*ESP32 Dev Kit C V2* avec *Arduino IDE*, suivez quelques étapes simples. Avant de configurer l'*IDE Arduino*, il faut installer le pilote pour la communication *USB-série*. Si le pilote n'est pas installé automatiquement, il existe une page d'assistance qui contient les pilotes pour *Windows/Mac* ou *Linux* et peut être choisi en fonction de celui qui est utilisé. Les pilotes peuvent être téléchargés à partir de ce [link](#).

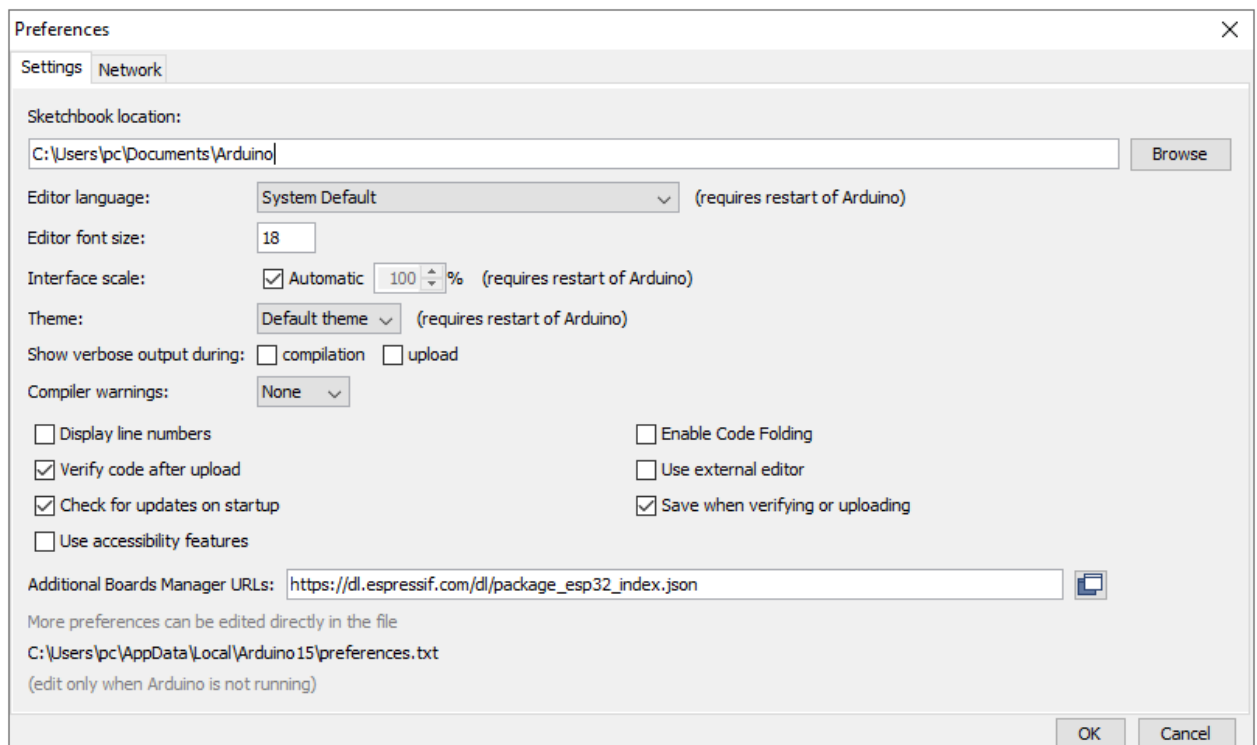
Az-Delivery

Ensuite, pour installer le support pour la plateforme *ESP32*, ouvrez l'*IDE Arduino* et allez dans :

File > Preferences, et trouvez le champ *Additional URLs*.

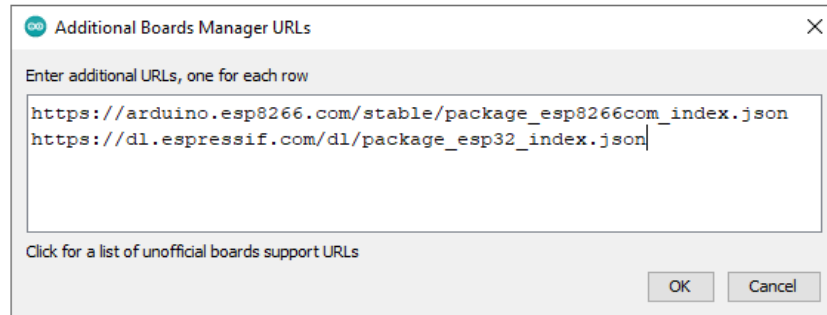
Copiez ensuite l'URL suivante :

https://dl.espressif.com/dl/package_esp32_index.json



Az-Delivery

Collez ce lien dans le champ *Additional URLs*. Si un ou plusieurs liens se trouvent dans ce champ, ajoutez simplement une virgule après le dernier lien, collez le nouveau lien après la virgule et cliquez sur le bouton *OK*.



Ouvrez à nouveau *Arduino IDE* et allez à :

Tools > Board > Boards Manager

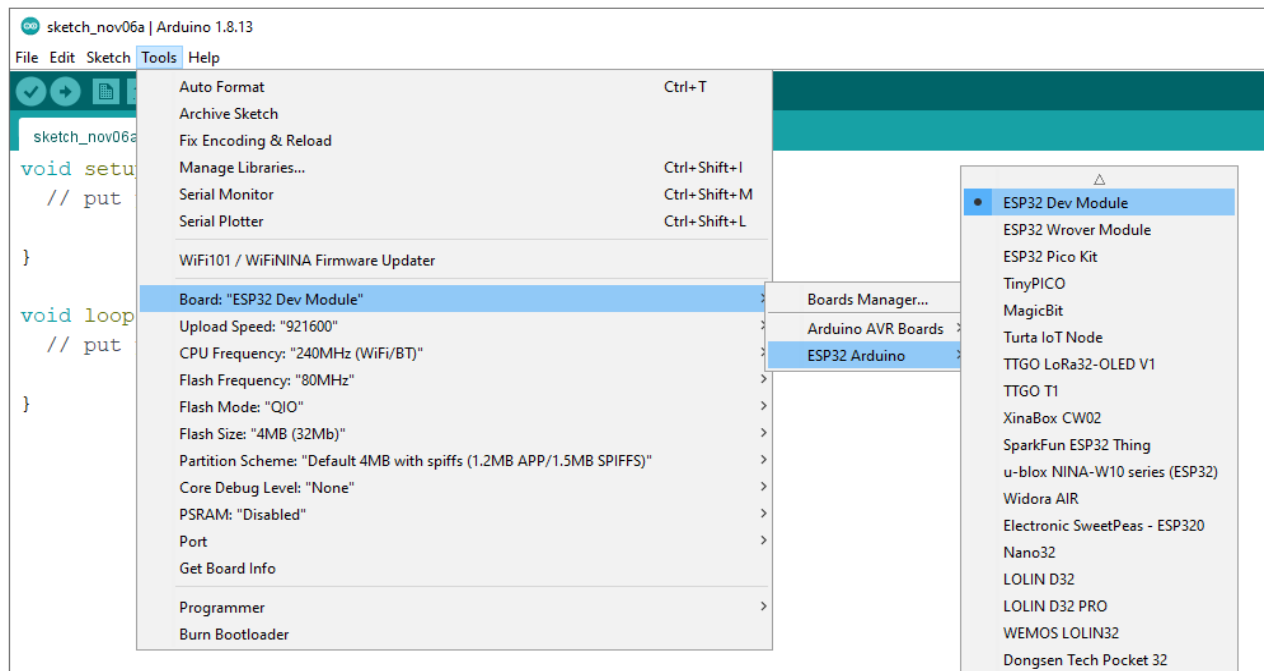
Lorsque la nouvelle fenêtre s'ouvre, tapez *esp32* dans la boîte de recherche et installez la carte appelée *esp32* fabriquée par *Espressif Systems*, comme le montre l'image suivante :



Az-Delivery

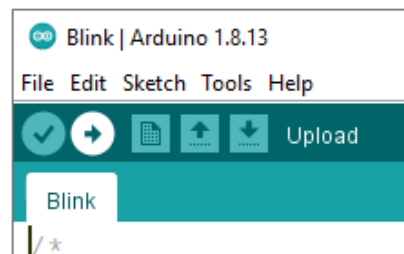
Pour sélectionner la carte ESP32, allez sur :

Tools > Board > ESP32 Arduino > ESP32 Dev Module



Pour télécharger le code du sketch sur la carte ESP32, sélectionnez d'abord le port sur lequel vous avez connecté la carte. Aller à :

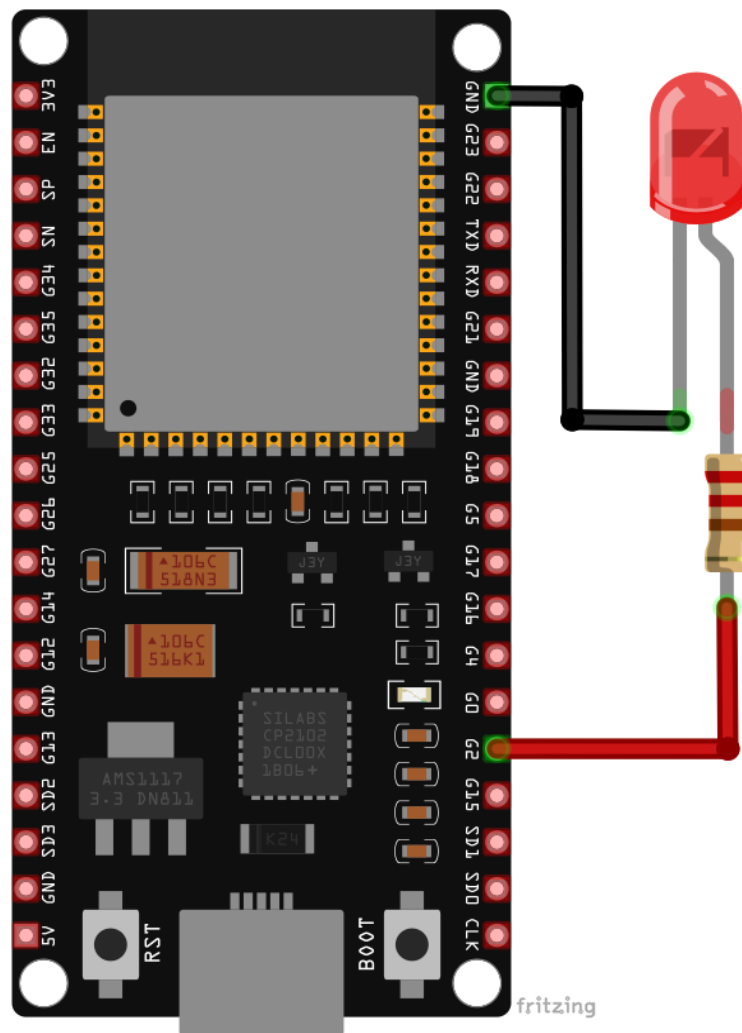
Tools > Port > {nom du port}0



Si le téléchargement ne fonctionne pas à la première tentative, il peut être utile d'appuyer sur le bouton *"Boot"* du module pendant le téléchargement. Veuillez utiliser un câble certifié *USB 2.0* pour la programmation.

ESP32 Dev Kit C V2 exemple de câblage

Connectez l'ESP32 Dev Kit C V2 avec une LED et une résistance comme indiqué sur le schéma de connexion suivant :



Pin ESP32 Dev Kit C V2	Pin LED	Couleur du câble
GPIO2 (pin2)	Anode (+) à travers la résistance	Câble Rouge
GND	Cathode (-)	Câble Noir

Exemples de Sketch

LED clignotante

```
int ledPin = 2;

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

void loop() {
  digitalWrite(ledPin, HIGH);
  delay(1000);
  digitalWrite(ledPin, LOW);
  delay(1000);
}
```

PWM - Modulation de largeur d'impulsion

```
#define LEDC_CHANNEL_0 0
#define LEDC_TIMER_13_BIT 13
#define LEDC_BASE_FREQ 5000
#define LED_PIN 2

int brightness = 0;
int fadeAmount = 5;

void ledcAnalogWrite(uint8_t channel, uint32_t value, uint32_t valueMax = 255) {
    uint32_t duty = (8191 / valueMax) * min(value, valueMax);
    ledcWrite(channel, duty);
}

void setup() {
    ledcSetup(LEDC_CHANNEL_0, LEDC_BASE_FREQ, LEDC_TIMER_13_BIT);
    ledcAttachPin(LED_PIN, LEDC_CHANNEL_0);
}

void loop() {
    ledcAnalogWrite(LEDC_CHANNEL_0, brightness);
    brightness = brightness + fadeAmount;
    if (brightness <= 0 || brightness >= 255) {
        fadeAmount = -fadeAmount;
    }
    delay(30);
}
```



Il est maintenant temps d'apprendre et de réaliser vos propres projets. Vous pouvez le faire à l'aide de nombreux exemples de scripts et autres tutoriels, que vous trouverez sur Internet.

Si vous recherchez des produits de haute qualité pour Arduino et Raspberry Pi, AZ-Delivery Vertriebs GmbH est l'entreprise idéale pour vous les procurer. Vous recevrez de nombreux exemples d'application, des guides d'installation complets, des livres électroniques, des bibliothèques et l'assistance de nos experts techniques.

<https://az-delivery.de>

Amusez-vous !

Mentions légales

<https://az-delivery.de/pages/about-us>