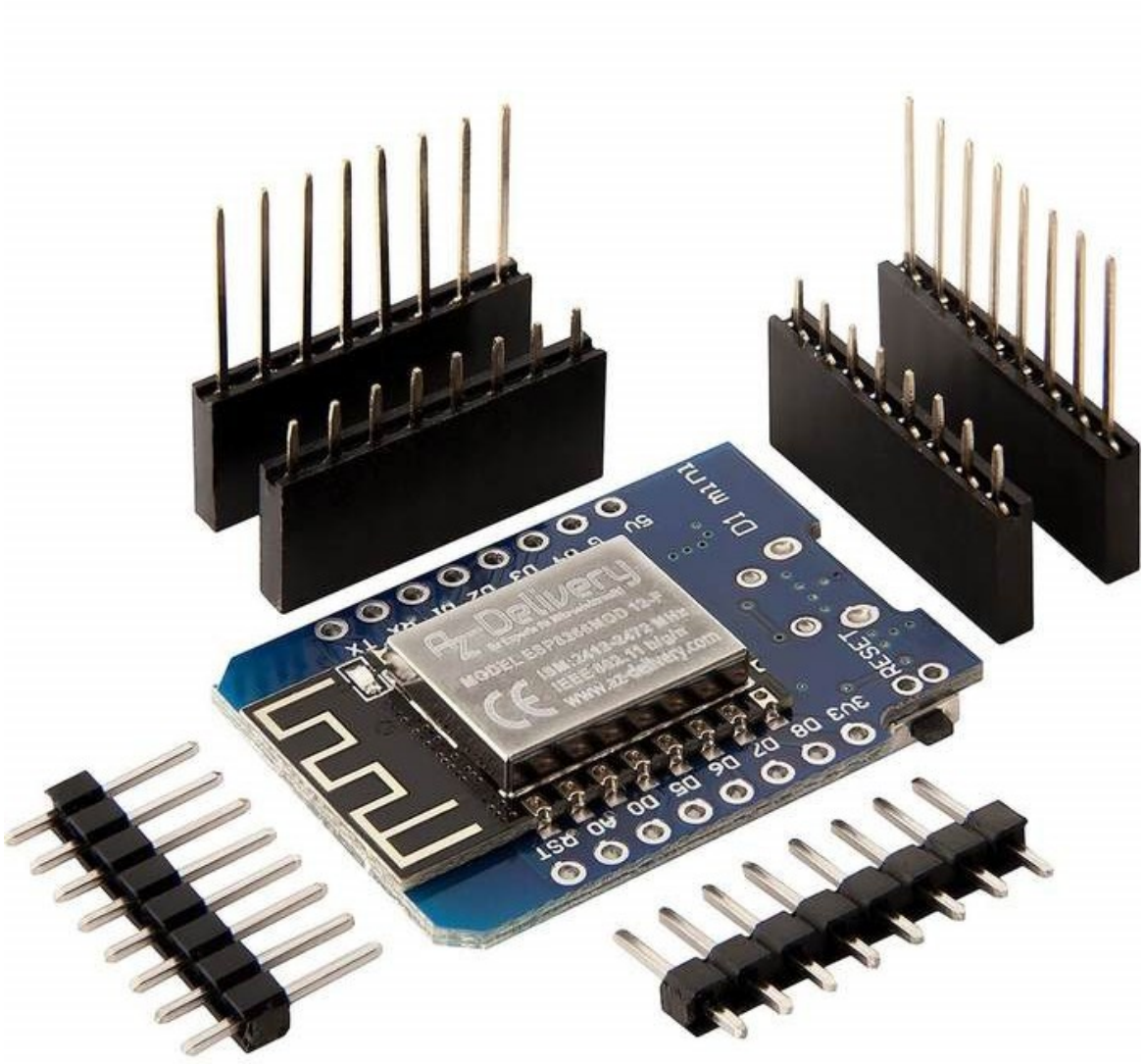
Bienvenido.

Gracias por adquirir nuestro *módulo AZ-Delivery D1 Mini ESP8266*. En las siguientes páginas, se le presentará cómo utilizar y configurar este práctico dispositivo.

¡Que te diviertas!




AZ-Delivery
Ihr Experte für Mikroelektronik!


AZ-Delivery
Ihr Experte für Mikroelektronik!

Índice

Introducción

El módulo ESP8266 es un sistema en un chip (*SoC*). Se compone de un microcontrolador de 32 bits Tensilica L106 y un transceptor Wi-Fi. Tiene *11* pines de entrada/salida de propósito general, o para abreviar pines GPIO, y una entrada analógica. Esto significa que se puede programar como cualquier otro microcontrolador. Lo mejor del ESP8266 es que tiene comunicación Wi-Fi, lo que significa que se puede conectar a Wi-Fi, o a Internet, alojar un servidor web con páginas web reales, conectarse a un smartphone, etc. Soporta protocolos de red como Wi-Fi, TCP, UDP, HTTP, DNS, etc.

El módulo AZ-Delivery D1 Mini es una placa de desarrollo basada en el chip ESP8266. Tiene 11 pines de entrada / salida digital y un pin de entrada analógica. Todos los pines de E/S digitales tienen capacidades de interrupción, pwm, I2C y 1-wire, en software. El rango de tensión de entrada analógica está entre *0V* y *3,3V* DC. El módulo utiliza el puerto microUSB y el chip CH340G con un circuito programador para programar el ESP8266. Además, el puerto microUSB proporciona una fuente de alimentación para el módulo.

Hay diferentes maneras de programar el módulo D1 Mini. Si usted ha utilizado placas Arduino antes, entonces esto es muy fácil para usted. Sólo ten en cuenta que no se limita a esta opción, hay muchas otras maneras de programar el módulo D1 Mini (ESP SDK oficial para la programación C, intérprete Lua, firmware MicroPython, son algunos de muchos).

Especificaciones de ESP8266

" 802.11 b/g/n

" MCU de 32 bits de bajo consumo integrada

" ADC de 10 bits integrado

" Pila de protocolos TCP/IP integrada

" Conmutador TR integrado, balun, LNA, amplificador de potencia y red de adaptación

" PLL integrado, reguladores y unidades de gestión de potencia

" Admite diversidad de antenas

" Wi-Fi 2,4 GHz, compatible con WPA/WPA2

" Admite los modos de funcionamiento STA/AP/STA+AP

" Admite la función Smart Link tanto para dispositivos Android como iOS

" SDIO 2.0, (H) SPI, UART, I2C, I2S, IRDA, PWM, GPIO

" STBC, 1x1 MIMO, 2x1 MIMO

" Agregación de A-MPDU & A-MSDU e intervalo de guarda de 0,4s

" Deep sleep power <10uA, corriente de fuga de apagado < 5uA

" Despierta y transmite paquetes en < 2ms

" Consumo en modo de espera < 1,0 mW (DTIM3)

" Potencia de salida de +20dBm en modo 802.11b

" Temperatura de funcionamiento: -40 °C ~ 125 °C

El módulo D1 Mini

El módulo D1 Mini viene desoldado con un par de cabezales macho de ocho pines, un par de cabezales hembra de ocho pines y un par de cabezales hembra de ocho pines con patas extra largas (que se pueden ver en la imagen de portada).

Especificaciones

" Tensión de	funcionamiento: 3,3 V CC
"Chip	principal: ESP8266
" Velocidad de reloj:	80MHz / 160MHz
" Flash:	4MB
"Pines digitales de E/S":	11
" Pines de entrada analógica:	1
" Rango de tensión de entrada	analógica: de 0V a 3,3V CC
" Puerto	microUSB
" Chip	USB:Chip CH340
" LED	a bordo: conectado al pin GPIO2
" Dimensiones:	25 x 35 x 6 mm [0,98 x 1,4 x 0,24 pulgadas].
" Corriente	máx. por pin de E/S digital único: 12mA

Pines digitales de E/S

Al igual que cualquier placa Arduino, el módulo D1 Mini tiene pines de entrada/salida digital o GPIO - General Purpose Input/Output pins. Como su nombre indica, se pueden utilizar como entradas digitales para leer un voltaje digital, o como salidas digitales para dar salida a *0V* (corriente de sumidero) o *3,3V* (corriente de origen).

El módulo D1 Mini tiene un microcontrolador que funciona en un rango de tensión de 0 V a 3,3 V.

La corriente máxima que se puede extraer de un solo pin GPIO es de 12 mA.

NOTA: Los pines del módulo D1 Mini no son tolerantes a 5V, ¡aplicar más de 3,6V en cualquier pin podría dañar el chip!

GPIO1 y *GPIO3* se utilizan como *TX* y *RX* del puerto serie de hardware (*UART*), por lo que en la mayoría de los casos, no se puede utilizar como una E / S normal, mientras que el envío / recepción de datos en serie.

El módulo D1 Mini tiene un LED integrado conectado al pin *GPIO2*.

PWM

A diferencia de la mayoría de los chips Atmel (Arduino), el módulo D1 Mini no soporta PWM por hardware. Sin embargo, el software PWM es compatible con todos los pines digitales. El rango PWM por defecto es de *10* bits a *1kHz*, pero esto se puede cambiar (hasta *14* bits a *1kHz*).

Entrada analógica

El módulo D1 Mini tiene una única entrada analógica, con un rango de tensión de entrada de 0V - 3,0V. Si se aplican más de *3,3 V,* el chip podría resultar dañado. El convertidor analógico a digital (ADC) tiene una resolución de *10* bits.

Serie

El módulo D1 Mini dispone de dos *UARTS* hardware (puertos serie):
UART0 en los pines 1 y 3 (*TX0* y *RX0 respectivamente*), y *UART1* en los pines 2 y 8 (*TX1* y *RX1* respectivamente). Sin embargo, *GPIO8* se utiliza para conectar el chip flash. Esto significa que *UART1* sólo puede transmitir datos. En la mayoría de los casos un solo puerto *UART* es más que suficiente.

UART0 también tiene control de flujo por hardware en los pines 15 y 13 (*RTS0* y *CTS0* respectivamente). Estos dos pines también se pueden utilizar como alternativa para los pines *TX0* y *RX0*.

El I2C

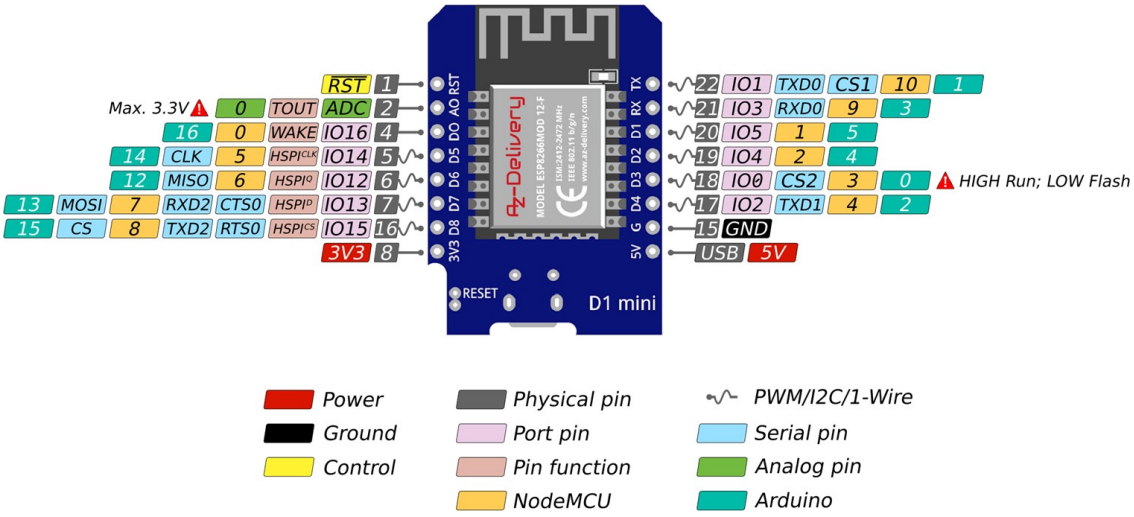
El módulo D1 Mini no tiene un hardware I2C o TWI (Two Wire Interface), sino que está implementado en software. Esto significa que se pueden utilizar dos pines digitales cualesquiera como pines I2C. Por defecto, la librería I2C utiliza *GPIO4* como *SDA* y *GPIO5* como *SCL* (la hoja de datos especifica *GPIO2* como *SDA* y *GPIO14* como *SCL*). La velocidad máxima del reloj I2C es de aproximadamente *450kHz*.

SPI

El módulo D1 Mini tiene una conexión *SPI* disponible para el usuario, denominada *HSPI*. Puede utilizarse tanto en modo esclavo como maestro (¡por software!).
Utiliza:
- *GPIO14* como reloj - *CLK*,
- *GPIO12* como *MISO*,
- *GPIO13* como *MOSI* y
- *GPIO15* como Selección Esclavo - *SS*.

El pinout

El módulo D1 Mini tiene dos filas de ocho pines (dieciséis pines en total). El pinout del módulo se muestra en la siguiente imagen:



La mayor parte de la funcionalidad del microcontrolador del ESP utiliza exactamente la misma sintaxis que una placa Arduino normal, por lo que es muy fácil empezar a utilizarlo.

Pines digitales de E/S

Al igual que con una placa Arduino normal, la función del pin se puede establecer utilizando la siguiente línea de código:

```
pinMode(pin, mode)
```

donde *pin* es el nombre del pin *GPIO,* y *mode* puede ser *INPUT* (que es el predeterminado), o *OUTPUT*, o *INPUT_PULLUP* para habilitar las resistencias pull-up incorporadas para los pines *GPIO0 - 15*. Para habilitar la resistencia pull-down para *GPIO16* utilice *INPUT_PULLDOWN_16*.

Para poner un pin de salida *en ALTO* (3,3V) o *BAJO* (0V), utilice la siguiente línea de código:

```
digitalWrite(pin, valor)
```

donde *pin* es el nombre del pin *GPIO*, y *valor 1* o *0* (o *HIGH* y *BAJA*).

Para leer una entrada, utilice la siguiente línea de código:

```
digitalRead(pin)
```

Para activar PWM en un pin determinado, utiliza la siguiente línea de código:

```
analogWrite(pin, valor)
```

donde *pin* es el nombre del pin *GPIO,* y *value* un número entre *0* y *1023*.

El rango de la salida PWM se puede cambiar utilizando la siguiente línea de código: `analogWriteRange(new_range)`

La frecuencia del PWM se puede cambiar utilizando la siguiente línea de código:

```
analogWriteFreq(nueva_frecuencia)
```

donde *nueva_frecuencia* debe estar entre *100Hz* y *1000Hz*.

Entrada analógica pin

Al igual que en cualquier placa Arduino, la función *analogRead(A0)* se puede utilizar para obtener la tensión analógica en la entrada analógica (0 = 0V, 1023 = 1.0V).

El módulo D1 Mini también puede utilizar el ADC para medir la tensión de alimentación (VCC). Para ello, incluye *ADC_MODE(ADC_VCC)* en la parte superior de tu sketch, y utiliza *ESP.getVcc()* para obtener realmente el voltaje.

NOTA: Si se utiliza una patilla de entrada analógica para leer la tensión de alimentación, no se puede conectar nada más a la patilla de entrada analógica.

Comunicación serie

Para usar *UART0* (*TX* = *GPIO1*, *RX* = *GPIO3*), usa el objeto Serial, igual que en una placa Arduino: `Serial.begin(baud_reate)`

Para utilizar los pines alternativos (*TX* = *GPIO15*, RX = *GPIO13*), utilice la siguiente línea de código: `Serial.swap()`
después de `Serial.begin()`

Para utilizar *UART1* (*TX* = *GPIO2*), utilice el objeto *`Serial1`*.

NOTA: Todas las funciones de flujo de Arduino, como *`read()`*, *`write()`*, *`print()`*, *`println()`*, etc. también están soportadas.

Las interfaces I2C y SPI

Para la interfaz I2C y SPI utiliza la sintaxis por defecto de la librería Arduino.

Compartiendo tiempo de CPU con la parte RF

Una cosa a tener en cuenta al escribir programas para el módulo D1 Mini (ESP8266) es que el boceto tiene que compartir recursos (tiempo de CPU y memoria) con el wifi y TCP-stacks (el software que se ejecuta en segundo plano y maneja todas las conexiones wifi e IP). Si el código tarda demasiado en ejecutarse, y no deja que las pilas TCP hagan lo suyo, el programa podría bloquearse, o los datos podrían perderse. Es mejor mantener el tiempo de ejecución del bucle por debajo de un par de cientos de milisegundos. Cada vez que se repite el bucle principal, un sketch cede al wifi y al TCP para manejar todas las peticiones wifi y TCP. Si el bucle tarda más que esto, el tiempo de CPU tiene que ser dado explícitamente a las pilas wifi/TCP, usando incluyendo `delay(0)` o `yield()`. Si esto no se hace, la comunicación de red no funcionará como se espera, y si dura más de *3* segundos, el soft WDT (Watchdog Timer) reinicia el ESP. Si el WDT suave está desactivado, después de un poco más de *8 segundos*, el WDT hardware reinicia el chip. Sin embargo, desde la perspectiva del microcontrolador, *3* segundos es un tiempo muy largo (*240* millones de ciclos de reloj), así que a menos que se haga algún cálculo numérico extremadamente pesado, o se envíen cadenas extremadamente largas por serie, el sketch no se verá afectado por esto. Sólo tenga en cuenta añadir el `yield()` dentro de los bucles `for` o `while` que podrían tardar más de, digamos *100ms*.

Cómo configurar Arduino IDE

Si el IDE de Arduino no está instalado, sigue el *enlace* y descarga el archivo de instalación para el sistema operativo que elijas.

Para

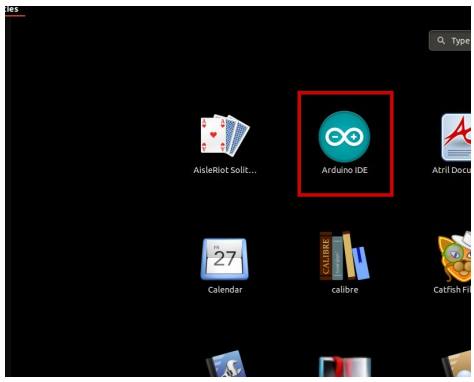
Usuarios *de Windows*, hagan doble clic en el archivo *.exe* descargado y sigan las instrucciones de la ventana de instalación.

Para los usuarios *de Linux,* descarga un archivo con la extensión *.tar.xz*, que hay que extraer. Cuando esté extraído, ve al directorio extraído y abre el terminal en ese directorio. Hay que ejecutar dos scripts *.sh*, el primero llamado *arduino-linux-setup.sh* y el segundo llamado *install.sh*.

Para ejecutar el primer script en el terminal, abra el terminal en el directorio extraído y ejecute el siguiente comando:
sh arduino-linux-setup.sh nombre_usuario
nombre_usuario - es el nombre de un superusuario en el sistema operativo Linux. Se debe introducir una contraseña para el superusuario cuando se inicie el comando. Espera unos minutos a que el script lo complete todo.

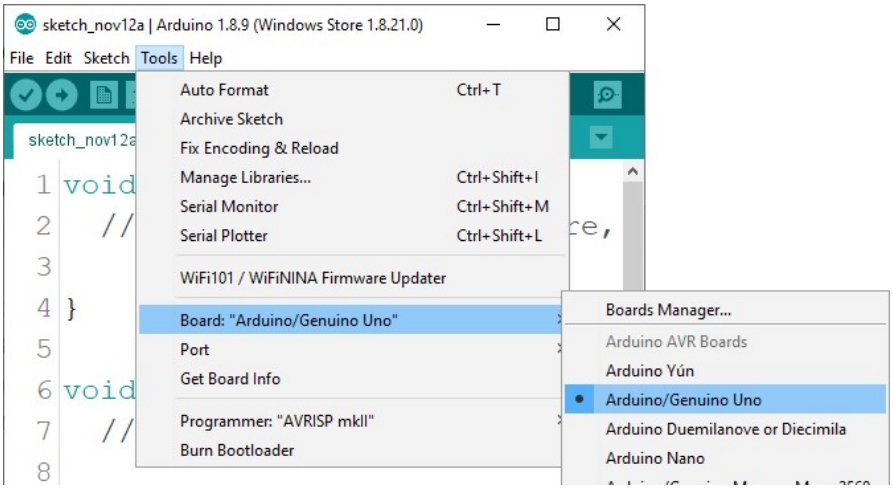
El segundo script llamado *install.sh* script tiene que ser utilizado después de la instalación del primer script. Ejecute el siguiente comando en el terminal (directorio extraído): **sh install.sh**

Después de la instalación de estos scripts, vaya a *Todas las aplicaciones*, donde está instalado el *IDE Arduino*.



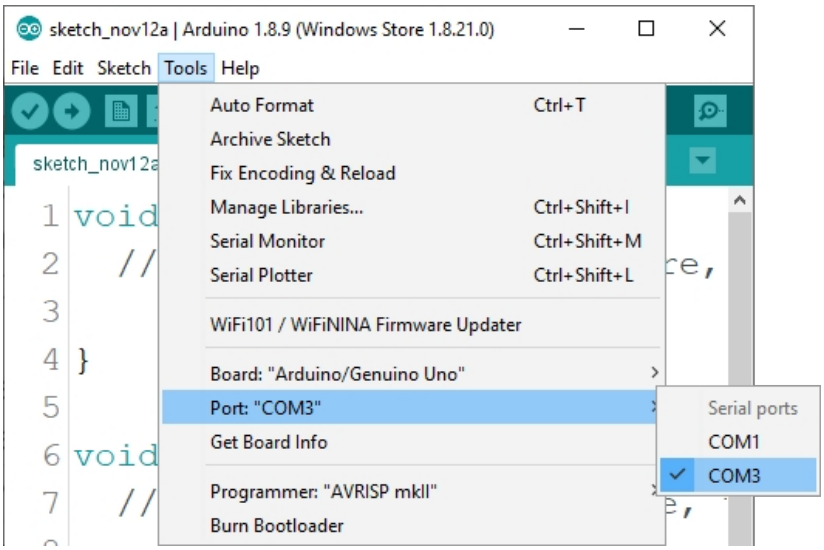
Casi todos los sistemas operativos vienen con un editor de texto preinstalado (por ejemplo, *Windows* viene con *Notepad*, *Linux Ubuntu* viene con *Gedit*, *Linux Raspbian* viene con *Leafpad*, etc.). Todos estos editores de texto son perfectamente adecuados para el propósito del libro electrónico.

Lo siguiente es comprobar si tu PC puede detectar una placa Arduino. Abra recién instalado Arduino IDE, y vaya a:
Herramientas > Tablero > {nombre de su tablero aquí}
{el nombre de tu placa aquí} debería ser el *Arduino/Genuino Uno*, como se puede ver en la siguiente imagen:



Hay que seleccionar el puerto al que está conectada la placa Arduino. Ir a:
Herramientas > Puerto > {el nombre del puerto va aquí}
y cuando la placa Arduino está conectada al puerto USB, se puede ver el nombre del puerto en el menú desplegable de la imagen anterior.

Si se utiliza el IDE Arduino en Windows, los nombres de los puertos son los siguientes:



Para los usuarios *de Linux,* por ejemplo, el nombre del puerto es */dev/ttyUSBx*, donde *x*

representa un número entero entre *0* y *9*.

D1 Mini con Arduino IDE

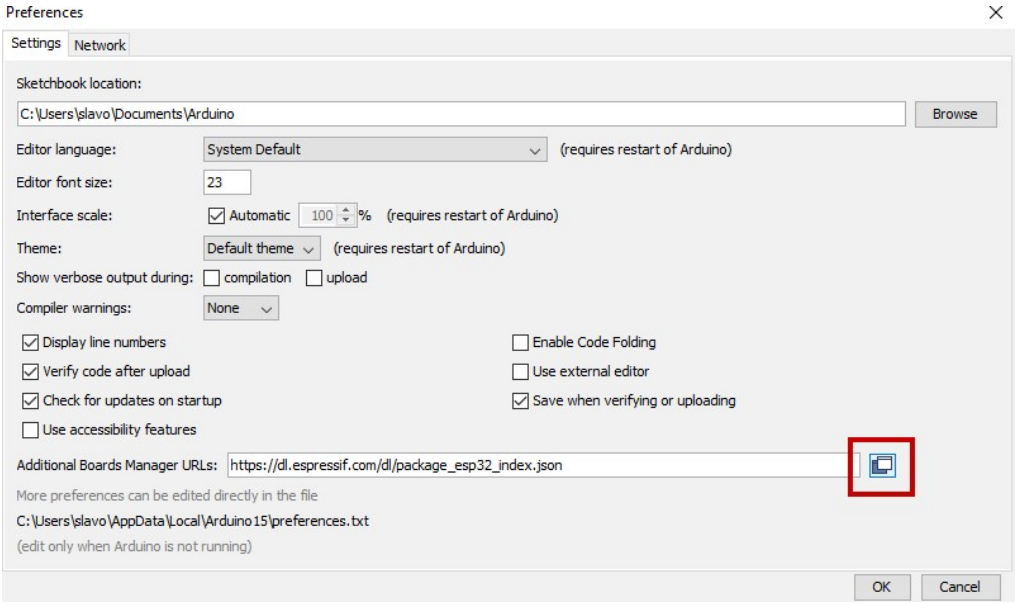
Para configurar el IDE Arduino de forma que el módulo D1 Mini pueda programarse a través del IDE Arduino, sigue los siguientes pasos.

Lo primero que hay que hacer es instalar el controlador USB, que puede descargarse aquí.

A continuación, instale el núcleo ESP8266. Para instalarlo, abra el IDE de Arduino y vaya a:

Archivo > Preferencias

y busque el campo *URLs adicionales*.



A continuación, copie la siguiente URL:

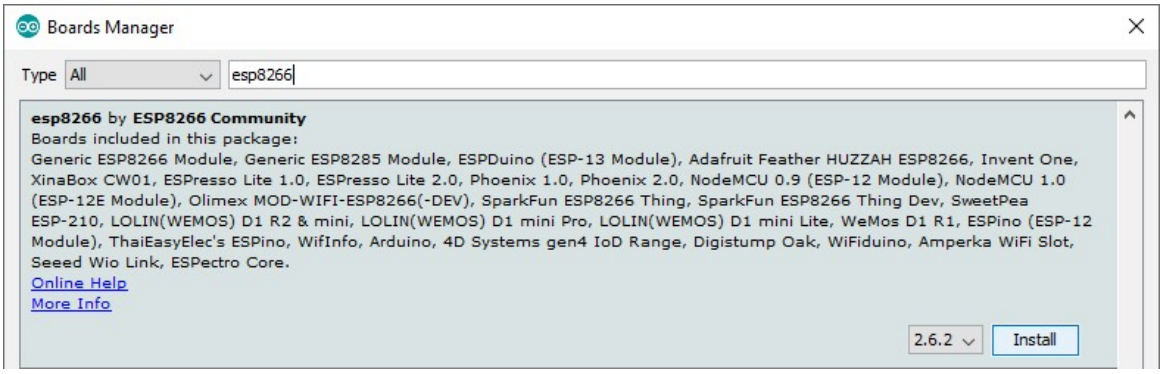
https://arduino.esp8266.com/stable/package_esp8266com_index.json

y péguelo en el campo *URLs adicionales*.

Si hay uno o más enlaces dentro del campo *URLs adicionales*, basta con añadir una coma después del último enlace, pegar un nuevo enlace después de una coma y hacer clic en el botón *Aceptar*. A continuación, cierre el IDE Arduino.

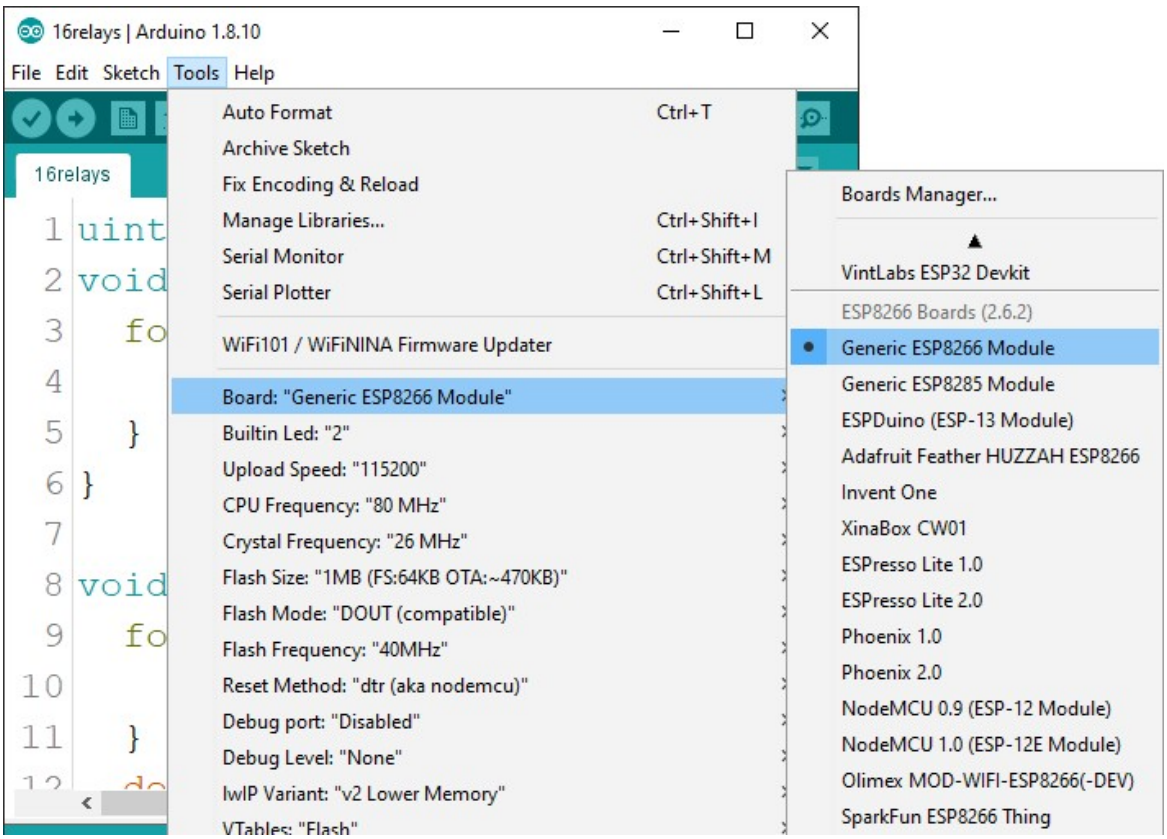
Abra Arduino IDE de nuevo y vaya a:
Herramientas > Tablero > Administrador de tableros

Cuando se abra la nueva ventana, escribe *esp8266* en la caja de búsqueda e instala la placa llamada *esp8266* hecha por *ESP8266 Community*, como se muestra en la imagen de abajo:



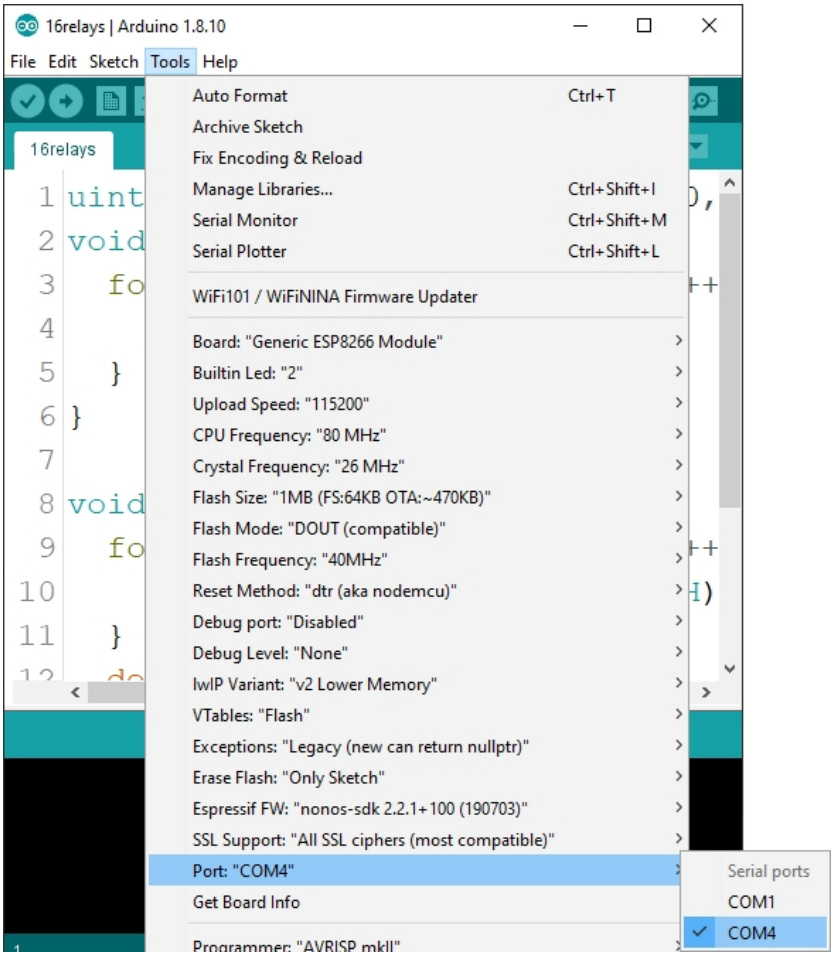
Ahora, el núcleo ESP8266 está instalado.

El siguiente paso es seleccionar la placa correcta en el IDE de Arduino. Abre Arduino IDE y ve a *Herramientas > Placa > {nombre de la placa}* y seleccione el primer *Módulo Genérico ESP8266* como se muestra en la siguiente imagen:



A continuación, seleccione el puerto al que está conectada la placa del módulo D1 Mini. Vaya a: *`Herramientas > Puerto > {el nombre del puerto va aquí}`*

Si la placa módulo D1 Mini está conectado en el puerto USB, debe haber algunos nombres de puerto. En este eBook el *IDE Arduino* se utiliza en *Windows*, los nombres de los puertos son los siguientes:



Para los usuarios de Linux, el nombre del puerto es *`/dev/ttyUSBx`*, por ejemplo, donde "*x*" representa un número entero específico entre *0* y *9*.

Ejemplos de sketch de parpadeo +

PWM + Serie Ejemplo de sketch de parpadeo

Hay un ejemplo de blink sketch que viene con la librería de la placa *ESP8266*. Para abrirlo, vaya a: *Archivos > Ejemplos > ESP8266 > Blink*

```
void setup() {
  pinMode(LED_BUILTIN, OUTPUT);     // Inicializar el LED_BUILTIN
}
void loop() {
  digitalWrite(LED_BUILTIN, LOW);   // Enciende el LED
  // Tenga en cuenta que LOW es el nivel de tensión
  // pero en realidad el LED está encendido; esto se debe a que
  // está activo BAJO en el ESP
  delay(1000);                      // Espera un segundo
  digitalWrite(LED_BUILTIN, HIGH); // Apaga el LED.
                                    // haciendo que la tensión sea
                                    ALTA
  delay(2000);                      // Espera dos segundos
}
```

Para el módulo D1 Mini, *LED_BUILTIN* es igual a un número *2*, lo que significa que el LED de a bordo está conectado al pin *GPIO2*. Para encender el LED hay que poner el pin *GPIO2 en* estado *BAJO*, y para apagarlo hay que poner el pin *GPIO2 en* estado *ALTO*.

Software PWM sketch ejemplo

```
int brillo = 1; // no lo pongas a cero
                    // cero desactiva el PWM en un pin específico
uint8_t fadeAmount = 5;
void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
}
void loop() {
  analogWrite(LED_BUILTIN, 200); // brillo alto
  delay(1000);
  analogWrite(LED_BUILTIN, 500);
  delay(1000);
  analogWrite(LED_BUILTIN, 800);
  delay(1000);
  analogWrite(LED_BUILTIN, 1000); // brillo bajo
  delay(1000);

  // led desvanecido
  while (1) {
    analogWrite(LED_BUILTIN, brillo); brillo =
    brillo + fadeAmount;
    if (brillo < 0 || brillo >= 1023) { fadeAmount = -
      fadeAmount;
    }
    retraso(10);
  }
}
```

Para utilizar PWM en el módulo D1 Mini (ESP8266) se utiliza la misma función que en las placas Arduino:

`analogWrite(pin, valor)`

donde `pin` es el nombre del pin *GPIO*, cualquier pin GPIO libre del módulo D1 Mini. El `valor` es el ciclo de trabajo, un valor entre *1* y *1023*.

No utilice el número cero para el `valor`, ya que desactiva la función PWM en un pin específico.

Cuanto más se acerque al valor cero, mayor será el nivel de brillo de un LED, y cuanto más se acerque al valor *1023,* mayor será el nivel de brillo de un LED.

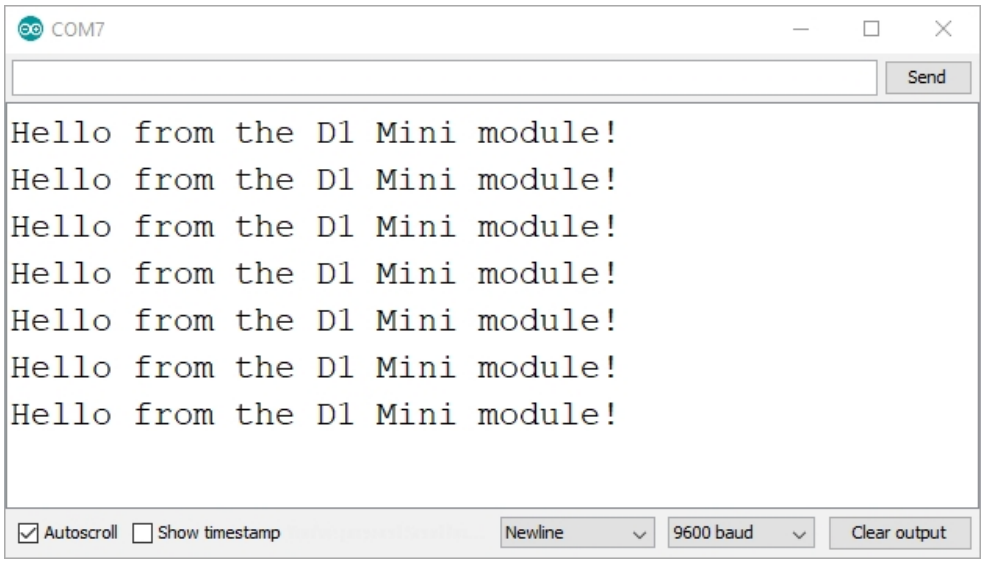
Para hacer que el LED se desvanezca, el bucle `while(1)` dentro de la función `loop()` tiene que ser utilizado, o de lo contrario no funcionará.

Sketch de comunicación serie ejemplo

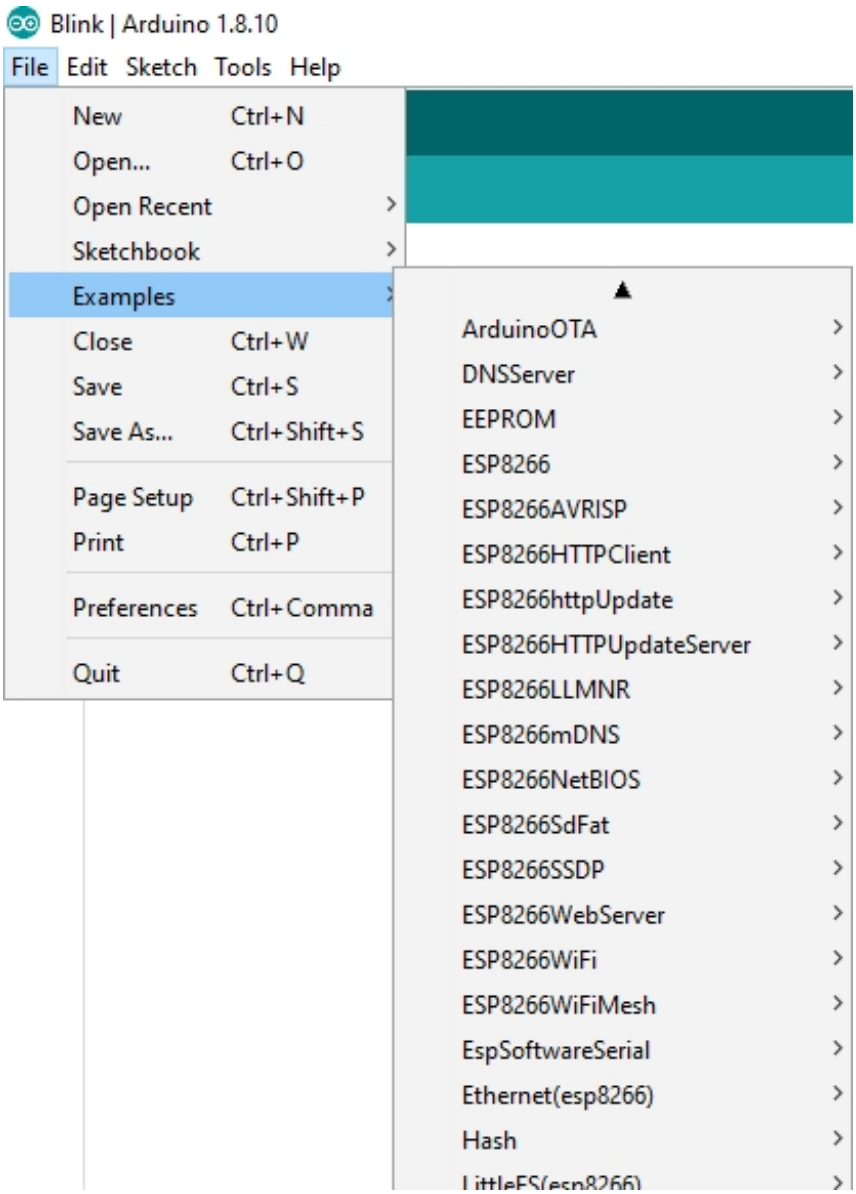
```
void setup() {
  Serial.begin(9600);
}

void loop() {
  Serial.println("¡Hola desde el módulo D1 Mini!");
  delay(1000);
}
```

Sube el sketch al módulo D1 Mini y abre el Serial Monitor (*Herramientas > Serial Monitor*). El resultado debería ser como el de la siguiente imagen:



Con la biblioteca de la placa *ESP8266* vienen muchos otros ejemplos de bocetos. La parte wifi del módulo D1 Mini podría probarse desde allí. No está cubierto en este eBook.



Ahora es el momento de aprender y hacer tus propios proyectos. Puedes hacerlo con la ayuda de muchos scripts de ejemplo y otros tutoriales, que puedes encontrar en internet.

Si está buscando productos de alta calidad para Arduino y Raspberry Pi, AZ-Delivery Vertriebs GmbH es la empresa adecuada. Dispondrá de numerosos ejemplos de aplicación, guías de instalación completas, libros electrónicos, bibliotecas y asistencia de nuestros expertos técnicos.

https://az-delivery.de

¡Diviértete!

Impresionante

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