Welcome!

Thank you for purchasing our *AZ-Delivery D1 Mini ESP8266 module*. On the following pages, you will be introduced to how to use and set-up this handy device.

Have fun!

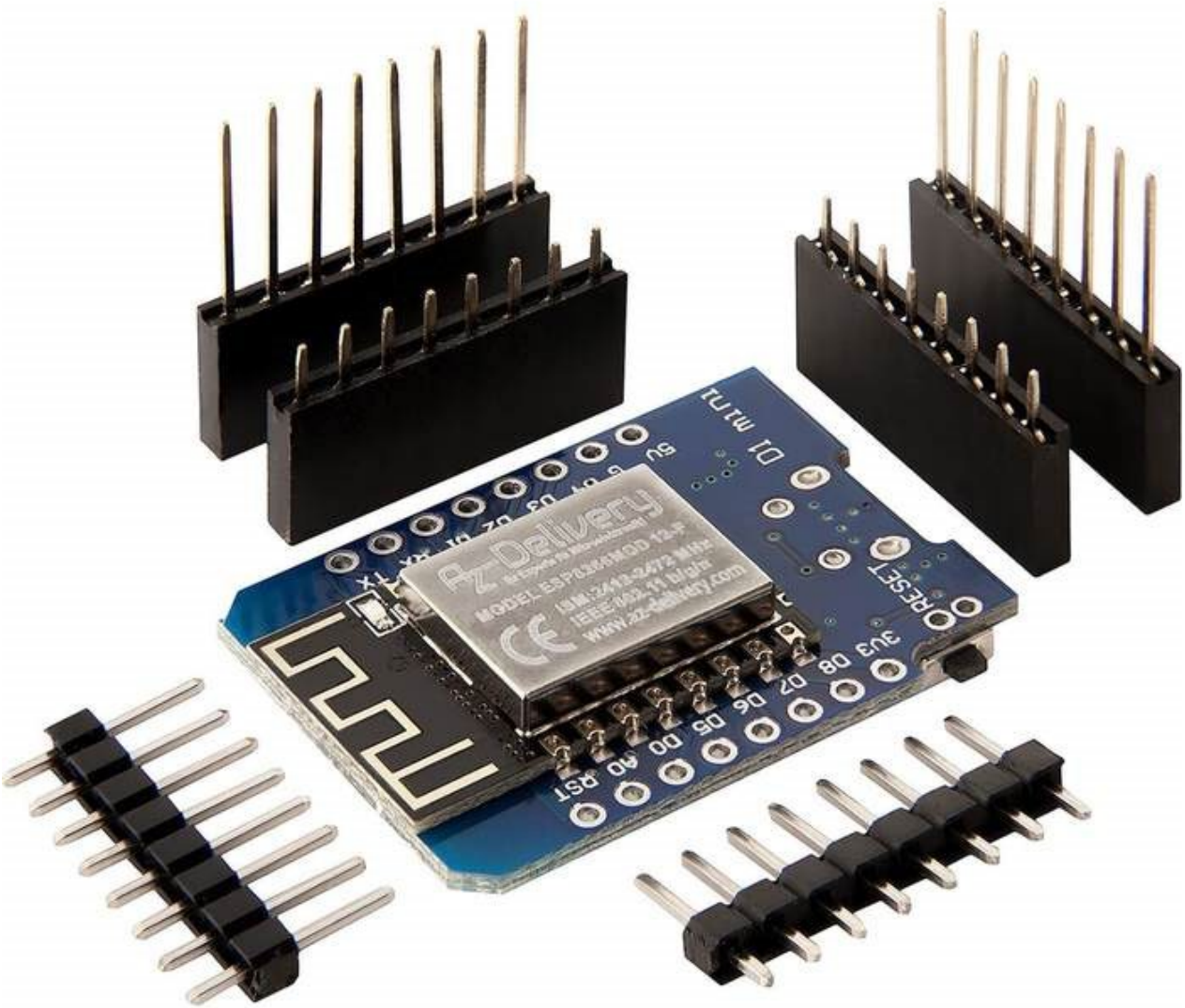


Table of Contents

Introduction

The ESP8266 module is a System on a Chip (*SoC*). It consists of a Tensilica L106 32-bit microcontroller and a Wi-Fi transceiver. It has *11* General Purpose Input/Output pins, or for short GPIO pins, and one analog input. This means that it can be programmed like any other microcontroller. The best thing about the ESP8266 is that it has Wi-Fi communication, which means that it can be connected to Wi-Fi, or the Internet, host a web server with real web pages, connect to a smartphone, etc. It supports network protocols like Wi-Fi, TCP, UDP, HTTP, DNS, etc.

AZ-Delivery D1 Mini module is a development board based on the ESP8266 chip. It has 11 digital input/output pins and one analog input pin. All digital I/O pins have interrupt, pwm, I2C and 1-wire capabilities, in software. The range of analog input voltage is between *0V* and *3.3V* DC. The module uses microUSB port and the CH340G chip with a programmer circuit for programming the ESP8266. Also, the microUSB port provides a power supply for the module.

There are different ways to program the D1 Mini module. If you have used Arduino boards before, then this is really easy for you. Just keep in mind that it is not limited to this option, there are many other ways to program the D1 Mini module (official ESP SDK for C programming, Lua interpreter, MicroPython firmware, are few of many).

Specifications of the ESP8266

- 802.11 b/g/n
- Integrated low power 32-bit MCU
- Integrated 10-bit ADC
- Integrated TCP/IP protocol stack
- Integrated TR switch, balun, LNA, power amplifier and matching network
- Integrated PLL, regulators, and power management units
- Supports antenna diversity
- Wi-Fi 2.4 GHz, support WPA/WPA2
- Support STA/AP/STA+AP operation modes
- Support Smart Link Function for both Android and iOS devices
- SDIO 2.0, (H) SPI, UART, I2C, I2S, IRDA, PWM, GPIO
- STBC, 1x1 MIMO, 2x1 MIMO
- A-MPDU & A-MSDU aggregation and 0.4s guard interval
- Deep sleep power <10uA, power down leakage current < 5uA
- Wake up and transmit packets in < 2ms
- Standby power consumption of < 1.0mW (DTIM3)
- +20dBm output power in 802.11b mode
- Operating temperature range: -40 °C ~ 125 °C

The D1 Mini module

The D1 Mini module comes unsoldered with pair of eight pin male headers, a pair of eight pin female headers and a pair of eight pin female headers with extra long legs (which can be seen on the cover image).

Specifications

- Operating voltage: 3.3V DC
- Main chip: ESP8266
- Clock speed: 80MHz / 160MHz
- Flash: 4MB
- Digital I/O pins: 11
- Analog input pins: 1
- Analog input voltage range: from 0V to 3.3V DC
- Port: microUSB
- USB chip: CH340 chip
- On-board LED: connected to GPIO2 pin
- Dimensions: 25 x 35 x 6mm [0.98 x 1.4 x 0.24in]
- Max. current per single digital I/O pin: 12mA

Digital I/O pins

Just like any Arduino board, the D1 Mini module has digital input/output pins or GPIO - General Purpose Input/Output pins. As the name implies, it can be used as digital inputs to read a digital voltage, or as digital outputs to output either *0V* (sink current) or *3.3V* (source current).

The D1 Mini module has a microcontroller that operate in voltager range of 0V – 3.3V.

The maximum current that can be drawn from a single GPIO pin is 12mA!

NOTE: The pins of the D1 Mini module are not 5V tolerant, applying more than 3.6V on any pin could damage the chip!

GPIO1 and *GPIO3* are used as *TX* and *RX* of the hardware Serial port (*UART*), so in most cases, it can not be used as a normal I/O while sending/receiving serial data.

D1 Mini module has one built-in LED connected to the *GPIO2* pin.

PWM

Unlike most Atmel chips (Arduino), the D1 Mini module does not support hardware PWM. However, software PWM is supported on all digital pins. The default PWM range is *10* bits at *1kHz*, but this can be changed (up to *14* bits at *1kHz*).

Analog input

The D1 Mini module has a single analog input, with an input voltage range of 0V - 3.0V. If more than *3.3V* is applied, the chip could be damaged. The analog to digital converter (ADC) has a resolution of *10* bits.

Serial

The D1 Mini module has two hardware *UARTS* (Serial ports):
UART0 on pins 1 and 3 (*TX0* and *RX0* respectively), and *UART1* on pins 2 and 8 (*TX1* and *RX1* respectively). However, *GPIO8* is used to connect the flash chip. This means that *UART1* can only transmit data. In most cases only one *UART* port is more than enough.

UART0 also has hardware flow control on pins 15 and 13 (*RTS0* and *CTS0* respectively). These two pins can also be used as an alternative for *TX0* and *RX0* pins.

The I2C

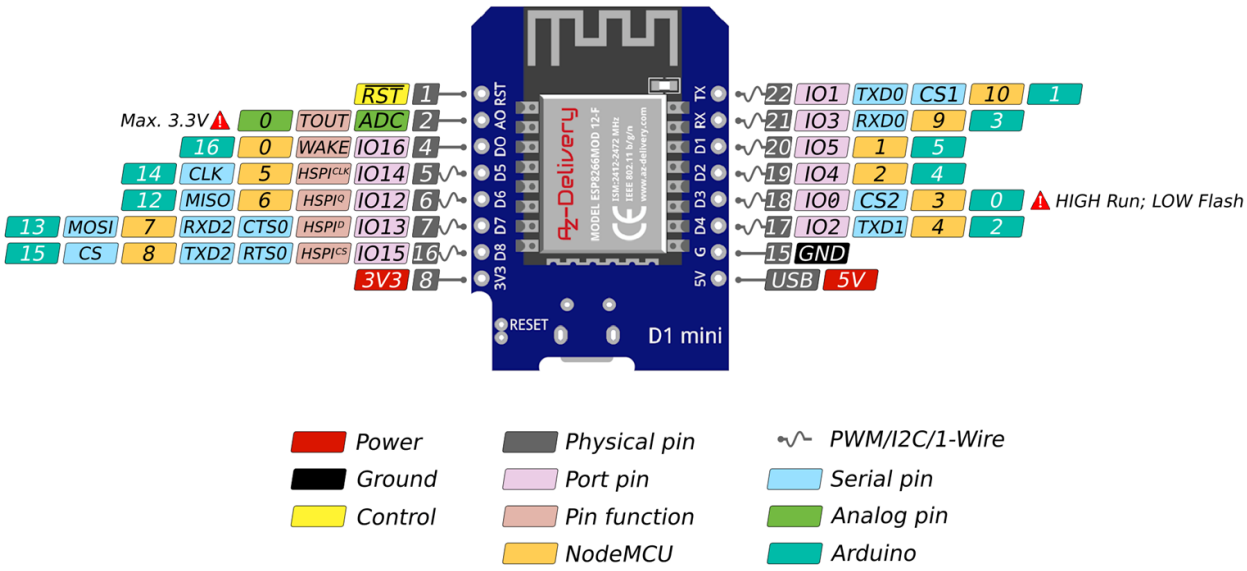
The D1 Mini module does not have a hardware I2C or TWI (Two Wire Interface), but it is implemented in software. This means that any two digital pins can be used as the I2C pins. By default, the I2C library uses *GPIO4* as *SDA* and *GPIO5* as *SCL* (the datasheet specifies *GPIO2* as *SDA* and *GPIO14* as *SCL*). The maximum speed of the I2C clock is approximately *450kHz*.

SPI

The D1 Mini module has one *SPI* connection available to the user, referred to as *HSPI*. It can be used in both Slave and Master mode (in software!).
It uses:
- *GPIO14* as a clock – *CLK*,
- *GPIO12* as *MISO*,
- *GPIO13* as *MOSI* and
- *GPIO15* as Slave Select - *SS*.

The pinout

The D1 Mini module has two rows of eight pins (sixteen pins in total). The pinout of the module is shown on the following image:



D1 Mini module - Software

Most of the microcontroller functionality of the ESP uses exactly the same syntax as a normal Arduino board, making it really easy to get started.

Digital I/O pins

Just like with a regular Arduino board, the pin function can be set using the following line of code:

pinMode(pin, mode)

where *pin* is the name of *GPIO* pin, and *mode* can be either *INPUT* (which is the default), or *OUTPUT*, or *INPUT_PULLUP* to enable the built-in pull-up resistors for pins *GPIO0 - 15*. To enable the pull-down resistor for *GPIO16* use *INPUT_PULLDOWN_16*.

To set an output pin *HIGH* (3.3V) or *LOW* (0V), use the following line of code:

digitalWrite(pin, value)

where *pin* is the name of *GPIO* pin, and *value* either *1* or *0* (or *HIGH* and *LOW*).

To read an input, use the following line of code: digitalRead(pin)

To enable PWM on a certain pin, use the following line of code:

analogWrite(pin, value)

where *pin* is the name of *GPIO* pin, and *value* a number between *0* and *1023*.

The range of the PWM output can be changed by using the following line of code: analogWriteRange(new_range)

The frequency of PWM can be changed by using the following line of code:

analogWriteFreq(new_frequency)

where *new_frequency* should be between *100Hz* and *1000Hz*.

Analog input pin

Just like on any Arduino board, function *analogRead(A0)* can be used to get the analog voltage on the analog input (0 = 0V, 1023 = 1.0V).

The D1 Mini module can also use the ADC to measure the supply voltage (VCC). To do this, include *ADC_MODE(ADC_VCC)* at the top of your sketch, and use *ESP.getVcc()* to actually get the voltage.

NOTE: If an analog input pin is used to read the supply voltage, anything else can not be connected to the analog input pin!

Serial communication

To use *UART0* (*TX* = *GPIO1*, *RX* = *GPIO3*), use the Serial object, just like on an Arduino board: Serial.begin(baud_reate)

To use the alternative pins (*TX* = *GPIO15*, RX = *GPIO13*), use the following line of code: Serial.swap()
after Serial.begin()

To use *UART1* (*TX* = *GPIO2*), use the *Serial1* object.

NOTE: All Arduino stream functions, like *read()*, *write()*, *print()*, *println()*, etc. are supported as well.

The I2C and SPI interfaces

For the I2C and SPI interface use the default Arduino library syntax.

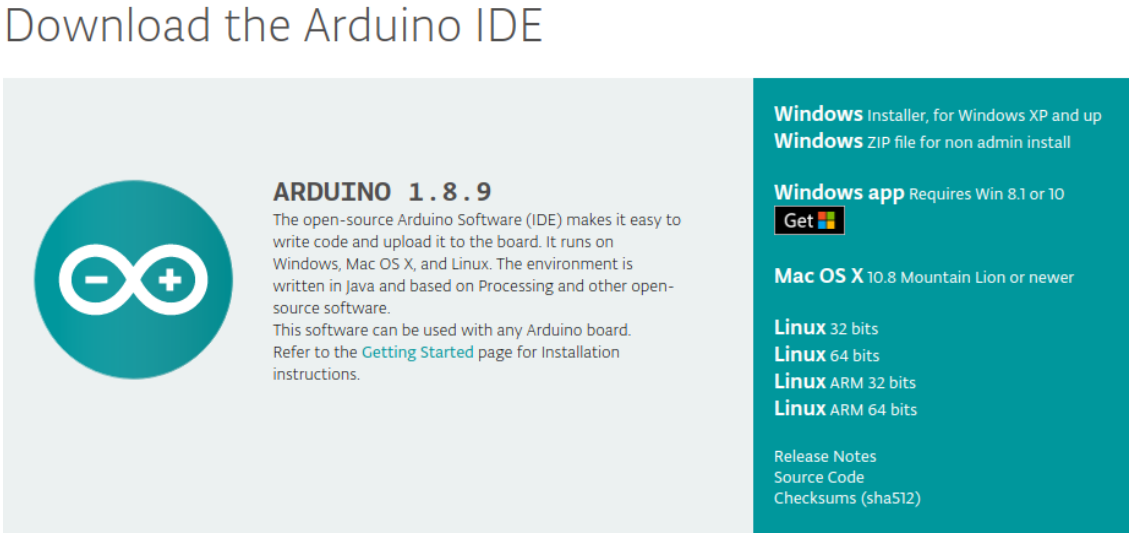
Sharing CPU time with the RF part

One thing to keep in mind while writing programs for the D1 Mini module (ESP8266) is that the sketch has to share resources (CPU time and memory) with the wifi and TCP-stacks (the software that runs in the background and handles all wifi and IP connections). If the code takes too long to execute, and do not let the TCP stacks do their thing, the program might crash, or the data could be lost. It is best to keep the execution time of the loop under a couple of hundreds of milliseconds. Every time the main loop is repeated, a sketch yields to the wifi and TCP to handle all wifi and TCP requests. If the loop takes longer than this, the CPU time have to explicitly be given to the wifi/TCP stacks, by using including *delay(0)* or *yield()*. If this is not done, network communication will not work as expected, and if it is longer than *3* seconds, the soft WDT (Watchdog Timer) resets the ESP. If the soft WDT is disabled, after a little over *8* seconds, the hardware WDT resets the chip. From the perspective of microcontroller however, *3* seconds is a very long time (*240* million clock cycles), so unless some extremely heavy number crunching is done, or sending extremely long strings over serial, the sketch will not be affected by this. Just keep in mind to add the *yield()* inside the *for* or *while* loops that could take longer than, say *100ms*.

How to set-up Arduino IDE

If the Arduino IDE is not installed, follow the *link* and download the installation file for the operating system of choice.

For



Windows users, double click on the downloaded *.exe* file and follow the instructions in the installation window.

For *Linux* users, download a file with the extension *.tar.xz*, which has to be extracted. When it is extracted, go to the extracted directory and open the terminal in that directory. Two *.sh* scripts have to be executed, the first called *arduino-linux-setup.sh* and the second called *install.sh*.

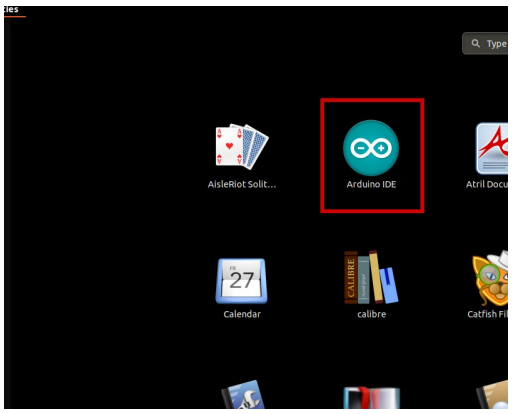
To run the first script in the terminal, open the terminal in the extracted directory and run the following command:

sh arduino-linux-setup.sh user_name

user_name - is the name of a superuser in the Linux operating system. A password for the superuser has to be entered when the command is started. Wait for a few minutes for the script to complete everything.

The second script called *install.sh* script has to be used after installation of the first script. Run the following command in the terminal (extracted directory): **sh install.sh**

After the installation of these scripts, go to the *All Apps*, where the *Arduino IDE* is installed.

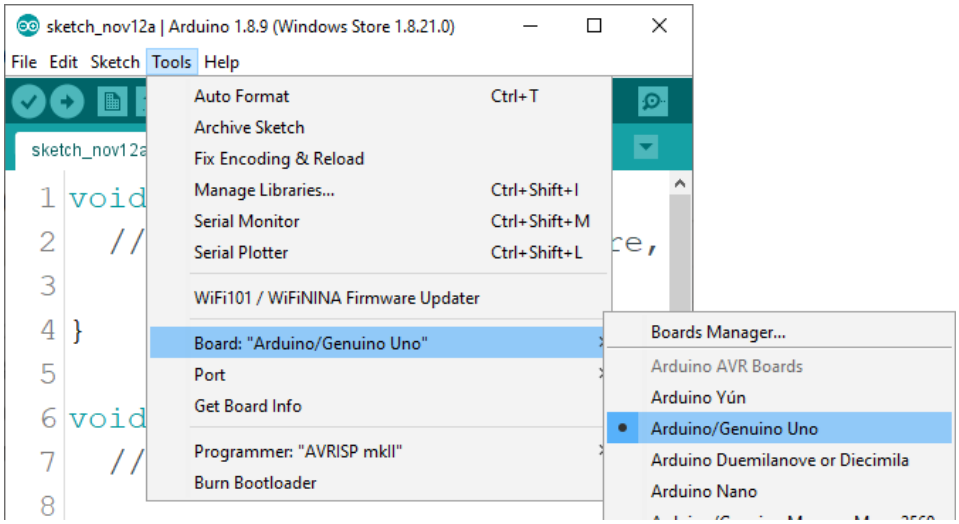


Almost all operating systems come with a text editor preinstalled (for example, *Windows* comes with *Notepad*, *Linux Ubuntu* comes with *Gedit*, *Linux Raspbian* comes with *Leafpad*, etc.). All of these text editors are perfectly fine for the purpose of the eBook.

Next thing is to check if your PC can detect an Arduino board. Open freshly installed Arduino IDE, and go to:

Tools > Board > {your board name here}

{your board name here} should be the *Arduino/Genuino Uno*, as it can be seen on the following image:

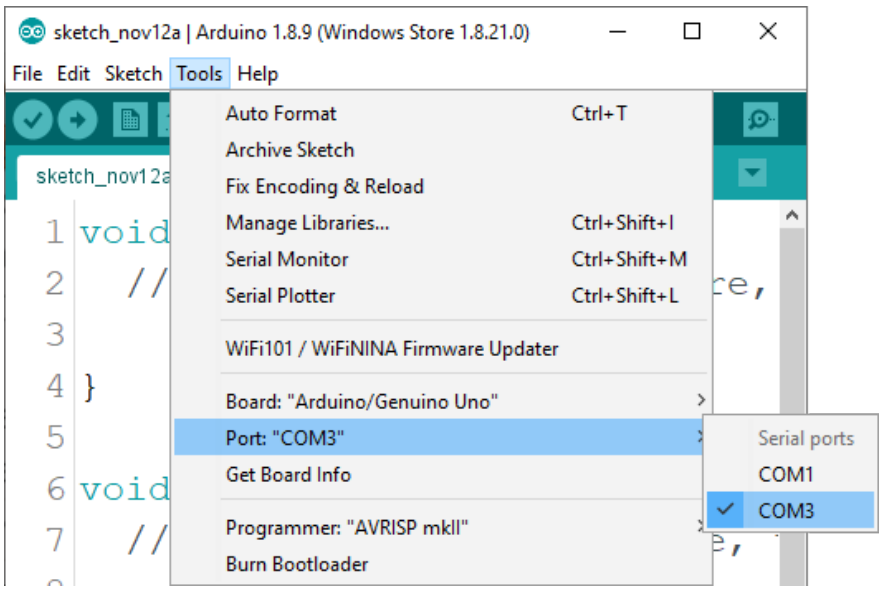


The port to which the Arduino board is connected has to be selected. Go to:

Tools > Port > {port name goes here}

and when the Arduino board is connected to the USB port, the port name can be seen in the drop-down menu on the previous image.

If the Arduino IDE is used on Windows, port names are as follows:



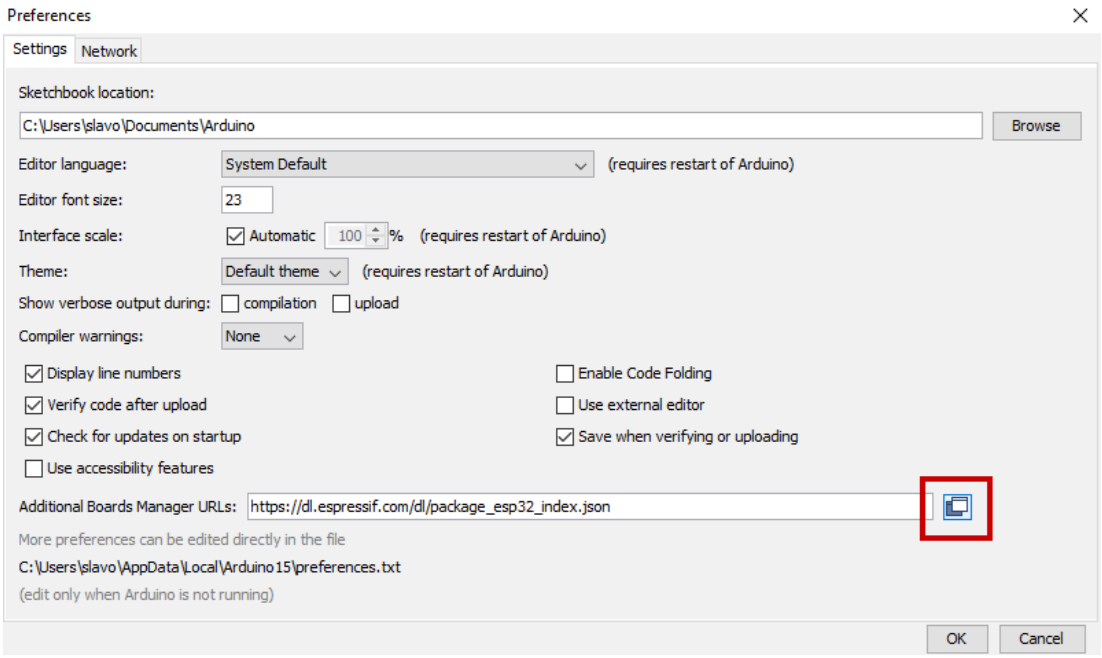
For *Linux* users, for example port name is */dev/ttyUSBx*, where *x* represents integer number between *0* and *9*.

D1 Mini with Arduino IDE

To set-up the Arduino IDE so that the D1 Mini module can be programmed via the Arduino IDE, follow the next steps.

The first thing to do is to install the USB driver, which can be downloaded here.

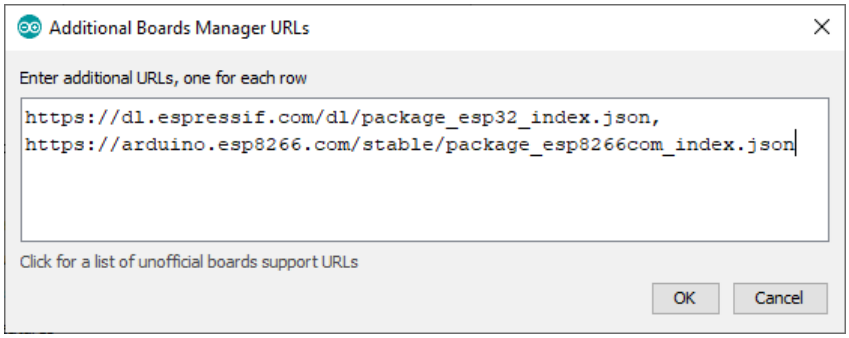
Next, install the ESP8266 core. To install it, open the Arduino IDE and go to:
File > Preferences
and find the *Additional URLs* field.



Then, copy the following URL:
https://arduino.esp8266.com/stable/package_esp8266com_index.json

and paste it in the *Additional URLs* field.

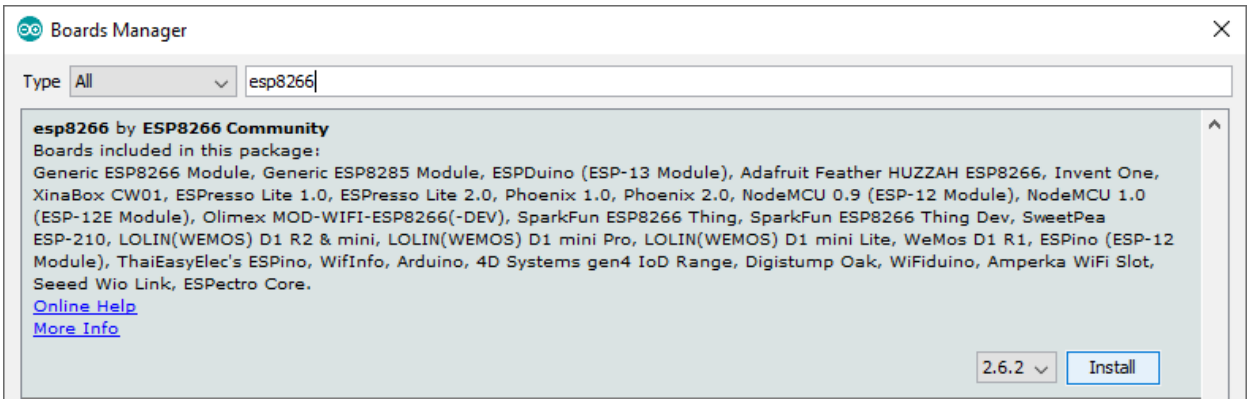
If there are one or more links inside *Additional URLs* field, just add one comma after the last link, paste a new link after a comma and click the *OK* button. Then close the Arduino IDE.



Open Arduino IDE again and go to:

Tools > Board > Boards Manager

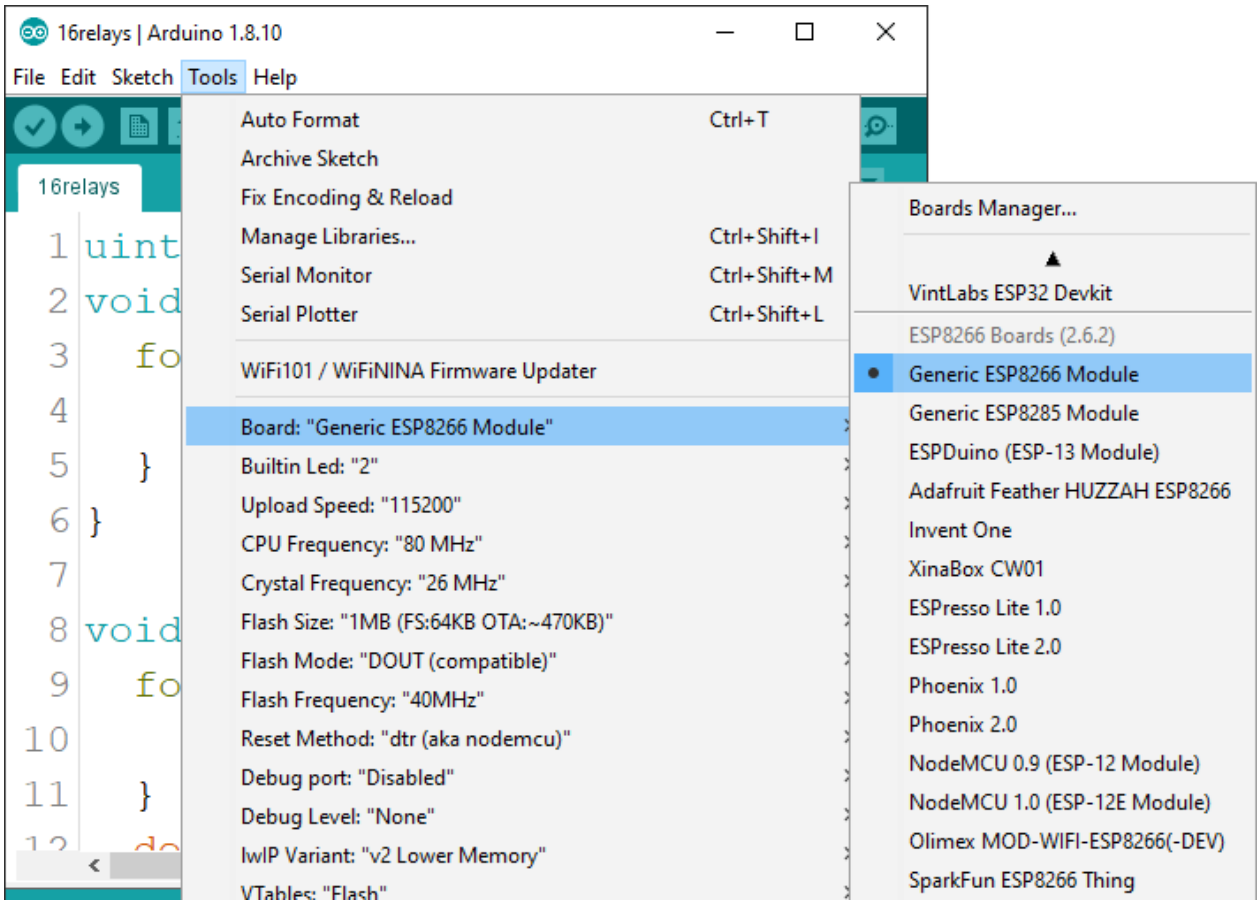
When new window opens, type *esp8266* in the search box and install the board called *esp8266* made by *ESP8266 Community*, as shown on the image below:



Now, the ESP8266 core is installed.

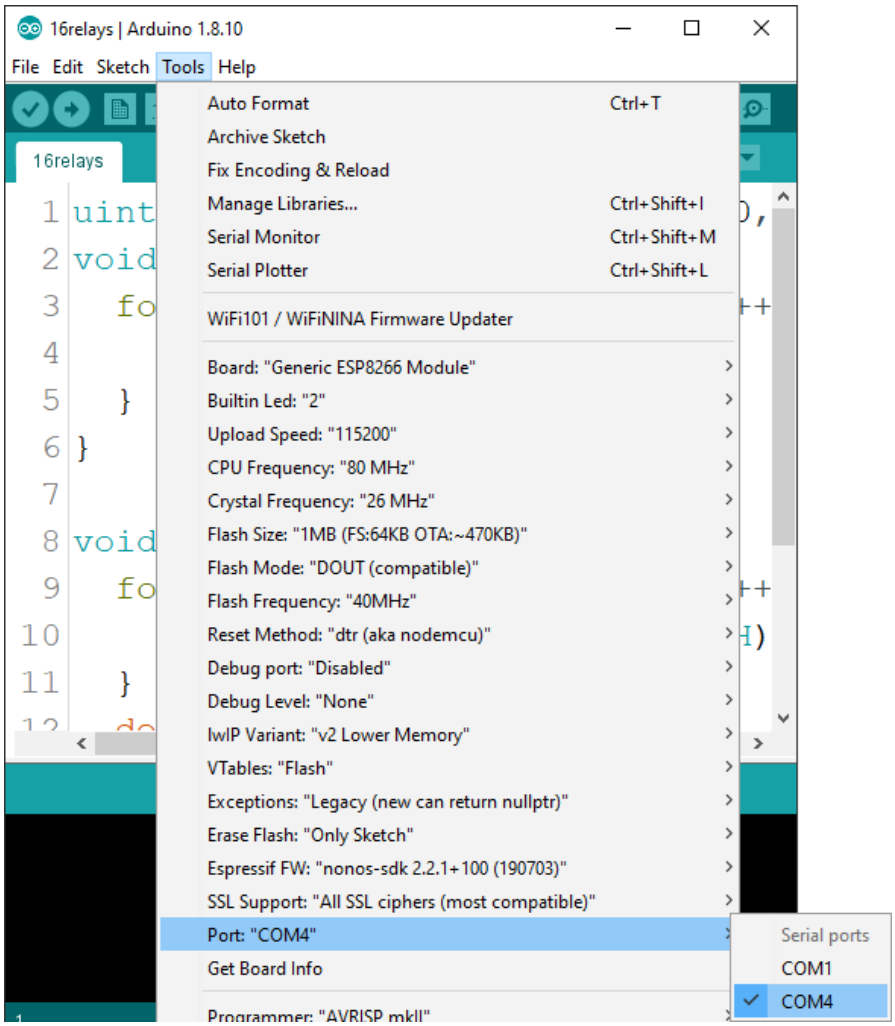
The next step is to select the right board in the Arduino IDE. Open Arduino IDE and go to: *Tools > Board > {board name}*

and select the first *Generic ESP8266 Module* as shown on the following image:



After that, select the port on which the D1 Mini module board is connected. Go to: *Tools > Port > {port name goes here}*

If the D1 Mini module board is connected on the USB port, there should be some port names. In this eBook the *Arduino IDE* is used on *Windows*, port names are as follows:



For Linux users, port name is */dev/ttyUSBx* for example, where "*x*" represents specific integer number between *0* and *9*.

Blink + PWM + Serial sketch examples

Blink sketch example

There is a blink sketch example that comes with the *ESP8266* board library. To open it, go to: *Files > Examples > ESP8266 > Blink*

```
void setup() {
  pinMode(LED_BUILTIN, OUTPUT);     // Initialize the LED_BUILTIN
}
void loop() {
  digitalWrite(LED_BUILTIN, LOW);   // Turn the LED on
  //  Note that LOW is the voltage level
  //  but actually the LED is on; this is because
  //  it is active LOW on the ESP
  delay(1000);                      // Wait for a second
  digitalWrite(LED_BUILTIN, HIGH);  // Turn the LED off
                                                  // by making the voltage HIGH
  delay(2000);                      // Wait for two seconds
}
```

For the D1 Mini module, *LED_BUILTIN* is equal to a number *2*, which means that on-board LED is connected to the *GPIO2* pin. To turn ON the LED the *GPIO2* pin has to be put in the *LOW* state, and to turn it OFF the *GPIO2* pin has to be put in the *HIGH* state.

Software PWM sketch example

```
int brightness = 1; //  do not set it to the zero
                    //  zero disables the PWM on a specific pin
uint8_t fadeAmount = 5;
void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
}
void loop() {
  analogWrite(LED_BUILTIN, 200);  //  high brightness
  delay(1000);
  analogWrite(LED_BUILTIN, 500);
  delay(1000);
  analogWrite(LED_BUILTIN, 800);
  delay(1000);
  analogWrite(LED_BUILTIN, 1000); //  low brightness
  delay(1000);

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

To use PWM on the D1 Mini module (ESP8266) the same function as on Arduino boards is used:

analogWrite(pin, value)

where *pin* is the name of *GPIO* pin, any free GPIO pin of the D1 Mini module. The *value* is the duty cycle, a value between *1* and *1023*.

Do not use number zero for *value*, because it turns OFF the PWM function on a specific pin!

Closer to the value zero the higher brightness level of a LED is and closer to the value *1023* the lover brightness level of a LED is.

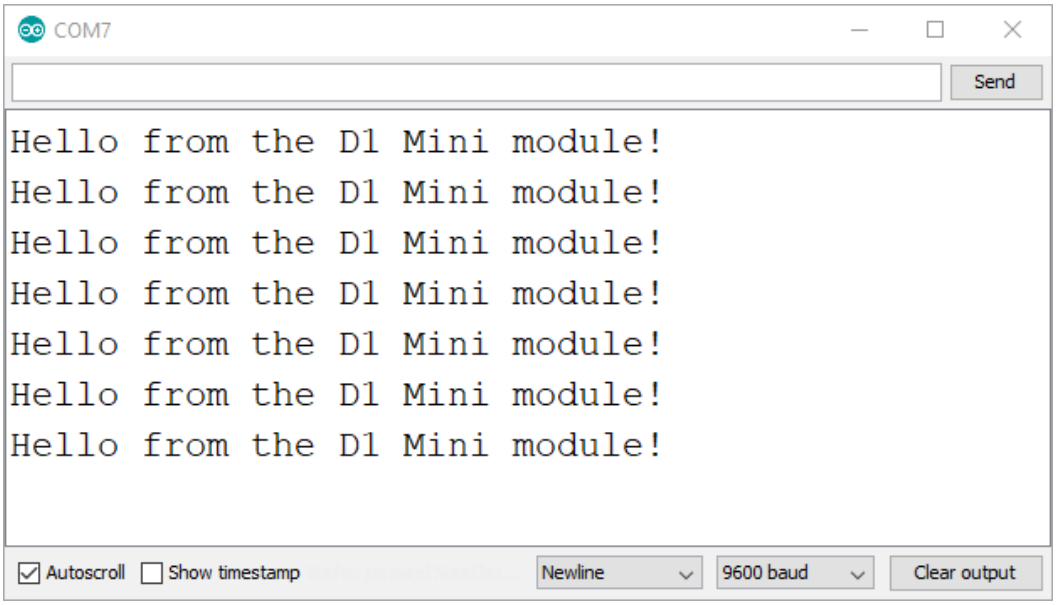
To make the LED to fade, the *while(1)* loop inside the *loop()* function has to be used, or else it will not work.

Serial communication sketch example

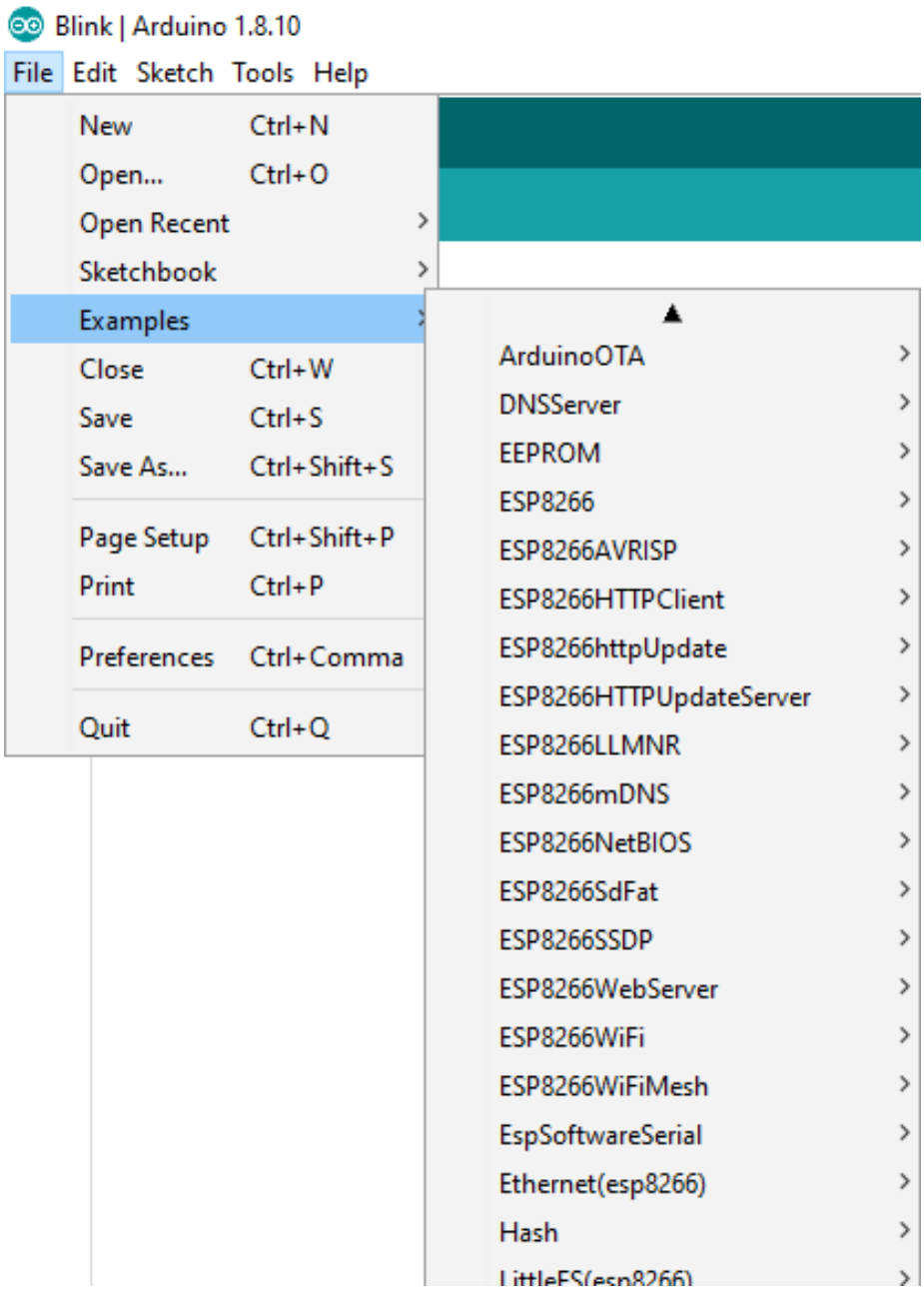
```
void setup() {
  Serial.begin(9600);
}

void loop() {
  Serial.println("Hello from the D1 Mini module!");
  delay(1000);
}
```

Upload the sketch to the D1 Mini module and open the Serial Monitor (*Tools > Serial Monitor*). The result should look like the on the following image:



With the *ESP8266* board library comes many other sketch examples. The wifi part of the D1 Mini module could be tested from there. It is not covered in this eBook.



Now it is the time to learn and make your own projects. You can do that with the help of many example scripts and other tutorials, which can be found on the internet.

If you are looking for the high quality products for Arduino and Raspberry Pi, AZ-Delivery Vertriebs GmbH is the right company to get them from. You will be provided with numerous application examples, full installation guides, eBooks, libraries and assistance from our technical experts.

https://az-delivery.de

Have Fun!

Impressum

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