# Esp8266 12e datasheet

The ESP8266 NodeMCU V1.0 ESP-12E WiFi module is the latest version of this popular module and can be used as a WiFi-enabled replacement for an Arduino in many applications. Package includes: ESP8266 NodeMCU V1.0 ESP-12E WiFi Module Key features of ESP8266 NodeMCU V1.0 ESP-12E Module: Microcontroller: ESP-8266 32-32-bit clock speed: 80 MHz USB converter: CP2102 USB connector: Micro USB operating voltage: 3.3V Flash memory: 4MB digital I/O: 11 Analog inputs: 1 Communication: Serial, SPI. I2C and 1-Wire via software libraries WiFi: Built-in 802.11 b/g/n Besides adding WiFi capacity, the main claim to fame for ESP8266 processor over AVR processor of standard Arduino is that it has a larger 4 MB Flash memory and runs at clock speeds of 80 MHz and can sometimes possibly be overclocked to 160 MHz and therefore has a very fast processing speed. Digital I/O supports all PWM and circuit breaker. In addition, they can be configured to have pull-up or pull-down resistors. Although there are 11 digital I/O pins, 2 are typically reserved for use as TX/RX lines if serial communication is used, leaving 9 digital I/O.000. On the negative side, it has only 1 analog input, which is probably the most significant limitation for some sensor type applications. It can always be overcome using an external Analog Mux module such as our 16-channel 74HC4067 or our ADS1115 4-channel 16-bit ADC module below if more analog I/O is requested. The module runs at 3.3V, so remember it when working with I/O. The digital I/O is indicated as being 5V tolerant, but the analogue input should be limited to < 3.3V. The blue on-board LED is connected to D0 (GPIO16) and can be accessed using LED_BUILTIN constantly. The module has a typical 'Reset' button as well as a 'Flash' button. The flash button is used for programming using the original NodeMCU firmware. If the module is being used with Arduino IDE, the Flash button does not need to be used to program the board and it will program just like any Arduino board would. The module comes loaded with NodeMCU software that accepts standard AT command sets. The module was originally designed primarily to be programmed using Lua, which is an interpreted language, but Lua has mostly been abandoned because it takes up a lot of memory and is slow because it is interpreted and more importantly it is generally buggy. It can also be programmed in C using Arduino IDE and is how the modules are used most often. An example program is shown down below. If a program is downloadable via IDE, it will overwrite NodeMCU software or whatever else was loaded before. If this is a problem for what you want to do, the NodeMCU software can always be reloaded. There are many instructions for installing and using ESP8266 based boards with Arduino IDE, but here is a Note that when ESP8266 board table added to the IDE, many more items are added to the Tools drop-down menu. Open the Options window and enter the following in the 'Additional Board Manager URLLs' field: . Under Boards Manager, install ESP8266 by the ESP8266 Community. Select NodeMCU 1.0 (ESP-12E Module). Set Transfer Rate to 115200. Select the port to which the board is associated. In my case, it happened COM19 In the Serial Monitor window, set the comm rate to 115200 and line ending to both NL & CR Here's what it looks like on my setup. Our evaluation results: These are Amica branded assemblies with good build quality To test whether the board is basically working using preloaded software, you can open a Serial Monitor Window and simply enter the 'AT' in the serial screen top window and hit ENTER. The board must return with 'OK'. This indicates that the board is alive and the setup is working. If you don't get OK, make sure you have the correct comm port selected, the serial screen window is set to a baud rate of 115200 and the line ends are set to both NL & CR. The NodeMCU has a few quirks to be aware of compared to a standard Arduino. First, the compilation time when using IDE tends to be longer than typical Arduino boards. This is especially true for the first time a program has been compiled or if building opportunities are changed that require a complete compilation. Subsequent compilations will run faster. The module does not like long delays in the code and it can cause the module to do a watchdog software reset. This is because the module has a network stack to handle WiFi and must be serviced regularly by the processor. An example of what not to do would be to use something like a tight DO/WHILE loop waiting for a button to be pressed. For example, in the program below, if you were to check the button using code like this, which blocks the program until the button is pushed, you will run into this problem as it does not allow any free time for the processor to go out and reset the watchdog timer on occasion so that it believes that it has unlocked and resets itself. Do { btn_Status = digitalRead (BUTTON_PIN) } while (btn_Status == HIGH) If the module keeps resetting every few seconds, look for this block type. This can be resolved by inserting the yield() function into the loop, as this feature lets the processor go out and take care of other outlets before returning to the loop. Using the delay function() does not create the same blocking problem because the delay() function internally calls the performance function() so often. The program below is based on one of the trial programs 'WiFiScan' that is available when ESP8266 boards are loaded into the IDE. The version here also adds a button to initiate the scan for all WiFi networks and turns on the on-board LED while a scan is in progress to show the use of some general I/O. In our example, we have a push button attached to pin D6, but another digital pin can be used. Be sure to ground the other side of the button. The internal pullup resistance is activated on this pin, so an external resistance is not required. /* * This sketch shows you how to scan for available WiFi networks. * A button input is used to start the scan and the on-board LED * is turned on to indicate when a scan is in progress * Connect one side of the push button to the ground and the other side to *pin D6 or any of the other digital input pins. */ #include ESP8266WiFi.h const int BUTTON_PIN = D6; Definer pin knappen er forbundet med //==========LED_BUILTIN LED_BUILTIN==========INPUT_PULLUP); Initialize button pin with built-in pullup. digitalWrite(LED_BUILTIN, HIGH); Make sure the LED is turned off Serial.begin(115200); Set the comm rate to 115200 // Set WiFi to station mode and disconnect from an AP if it was previously connected to WiFi mode(WIFI_STA); WiFi.disconnect(); delay(100) Serial.println(Installation complete); }//==========btn_Status BUTTON_PIN btn_Status btn_Status==========så gør noget Serial.println(scan start); digitalWrite(LED_BUILTIN, LOW); Turn LED ON // WiFi.scanNetworks will return the number of networks found int n = WiFi.scanNetworks(); Serial.println(scan done); if (n == 0) Serial.println(no networks were found) otherwise { Serial.print(n); Serial.println( network found); for (int i = 0; i < n; ++i) { // Print SSID and RSSI for each network found Serial.print(i + 1); Serial.print( ); Serial.print(WiFi.SSID(i)); Serial.print Serial.print(WiFi.RSSI(i)); Serial.print(|) Serial.println((WiFi.encryptionType(i) == ENC_TYPE_NONE)? : Unsafe: Encrypted); } } Serial.println(); digitalWrite(LED_BUILTIN, HIGH) Turn LED Off } } Before they are shipped, these modules are: Controlled power controlled using the command 'AT' Repackaged in high quality resealable ESD bag for safe storage. Notes: Technical Specifications Microcontroller ESP8266 Tensilica 32-bit Serial for USB Converter CP2102 Operating Voltage 3.3 V Input Voltage (Recommended) 7-12V Digital I/O Pins 11 PWM I/O Pins (shared with digital I/O) 10 Analog Input Pins 1 (10-bit) DC power per. I/O Pin 12mA (Max) Hardware Serial Ports 1 Flash Memory 4 MByte Instruction RAM 64 KBytes Data RAM 96 KBytes Clock 80MHz Network IEEE 802.0011 b/g/n WiF Built-in LED Attached to Digital Pin 13 USB Connector Style Micro-B Female Board Dimensions (PCB) 69 x 53mm (2.7 x 2.1) 2.1) Martyn Currey has an excellent series of articles related to the use of wireless communication, especially with Arduino. This is a link to a number of his articles related to ESP8266 ESP8266