Generic ESP8266 ESP-12E NodeMcu Development Board with Built-in USB Port in the Arduino IDE

by Kevin Petrasek

The Generic ESP8266 ESP-12E NodeMcu Development Board with built-in USB port is widely available, and quite cheap. It offers wireless connectivity, USB connectivity, and a reasonably powerful MCU, all for a few dollars and a pretty small footprint. The type of board I talk about here looks like this:

Here are some instructions for how to use the Generic ESP8266 ESP-12E NodeMcu Development Board with the Arduino IDE through the built-in USB port:

1) Download the latest version of the Arduino IDE.
2) Install the IDE.
3) Run the Arduino IDE and Go to File->Preferences.
4) In the Settings Tab of the Preferences Dialog, Copy the URL below into the "Additional Boards Manager URLs:" textbox. If there is already one or more URL's in this box, add this new one to the end of the string, separated from the others by a comma. This sets up the Arduino IDE to get the ESP board manager extensions for various ESP8266 boards. The URL is: http://arduino.esp8266.com/stable/package_esp8266com_index.json
5) If you don't have the "http://" at the beginning of the address, the Arduino IDE gives you a protocol error when you try to load the toolchain in the Boards Manager.
6) Go to Tools > Board > Board Manager and Type "esp8266" in the filter box. You should see the "ESP8266 Community esp8266" package in the resulting list (it will probably be the only one).
7) Click on the "ESP8266 Community esp8266" package, select a version (the latest) in the version list box, and click the Install button.
8) Plug a USB cable into the board, and into your computer. It should come up as a serial COM port. Check Windows Device Manager (right-click on the "My Computer" or "This PC" icon and choose "Manage" on the menu. Then choose "Device Manager" in the left pane and expand "Ports (COM & LPT)" in the right pane. On the ports list, you should see a port for your board. If you don't know which one it is, you can unplug the board and plug it back in, and you'll see the port for it disappear and reappear. Mine came up as "Silicon Labs CP210x USB to UART Bridge (COM14), but yours will depend on your ESP8266 board and your computer. You don't need to worry about port settings here -- the Arduino IDE will set these later.
9) In the Arduino Tools Menu, Set up your parameters as follows:

   Tools -> Board -> NodeMCU 1.0 (ESP-12E Module)

Tools -> Flash Size -> 4M (3M SPIFFS)
Tools -> CPU Frequency -> 80 Mhz
Tools -> Upload Speed -> 115200
Tools-->Port--> (whatever it was in Device Manager)

10) Here, it gets a little tricky, because that 115200 Upload Speed you just set may not be right for your board. As far as I can tell, different brands of this board style will have different baud rates. The reference I used to learn how to do this said to use 921600 as the Upload Speed. That seemed to work fine for upload, but my sketch wouldn't run. As it turns out, the Arduino IDE can't tell if the Upload Baud Rate is right, so it throws NO error if you have the wrong baud rate set (Umm, I KNOW, right?). So you THINK you're uploading, but all the board hears is garbage, so you haven't really flashed your program after all. I found the correct baud rate for my board by trial and error. I downloaded the blink sketch with each baud rate in turn until it started blinking.
11) Here's another fun baud rate idiosyncrasy: If you bring up the Serial Monitor from the IDE Tools Menu, and then press the reset button on your board, you'll see that the board is trying to send you a message. I had to trial-and-error this baud rate too, and it turned out to be different than the Upload Speed. In my case, this baud rate is 74880. The message it sends when you reset looks like this:

```
 ets Jan  8 2013,rst cause:2, boot mode:(3,7)

load 0x4010f000, len 1384, room 16
tail 8
chksum 0x2d
csum 0x2d
vbb28d4a3
~ld
```

12) Alternatively, you might be able to gain a greater amount of control over the settings of this board's boot loader by flashing the board. MakerFocus (from whom I bought my boards) had some obscure instructions to "Download and run the 32 bit flasher exe at Github". I didn't do this, so I'm not much help on this part. MakerFocus didn't provide much guidance on what this does for you or really how to do it, but here is the Github program they point to:
    https://github.com/nodemcu/nodemcu-flasher
    Read the readme file to figure out how to use it.

    You can Download and run the compiled 32-bit flasher program as an executable file at:
    https://github.com/nodemcu/nodemcu-flasher/tree/master/Win32/Release
    -Or-
    The compiled 64-bit flasher program as an executable file at:
    https://github.com/nodemcu/nodemcu-flasher/tree/master/Win64/Release

13) To Upload and run a sketch in the Arduino IDE, create a blank sketch, and copy the code below into it and save it. Hit the Upload button, and wait a couple of minutes for it to compile. It should automatically start uploading, and will run the sketch automatically when it's done, if you've got the baud rate right.

```
//************************************************************************************************
//  ESP8266 Blink by Kevin Petrasek
//  Generic NodeMCU-style ESP12e Board
//  with the ESP8266MOD by AI-THINKER
//  Blink the built-in blue LED on GPIO-16 (D0) pin.
//
//  The built-in LED on this style of generic boards is connected
//  to GPIO16(as marked on the board bottom silkscreen markings),
//  Which is marked D0 on the board's top silkscreen markings.
//************************************************************************************************

//Here is a map of the "D" markings on the board vs. actual ESP8266 chip GPIO numbers
//static const uint8_t D0   = 16;
//static const uint8_t D1   = 5;
//static const uint8_t D2   = 4;
//static const uint8_t D3   = 0;
//static const uint8_t D4   = 2;
//static const uint8_t D5   = 14;
//static const uint8_t D6   = 12;
//static const uint8_t D7   = 13;
//static const uint8_t D8   = 15;
//static const uint8_t D9   = 3;
//static const uint8_t D10  = 1;

static const uint8_t LED_PIN = 16; //The LED is on GPIO-16 pin (D0)

void setup() {
  pinMode(LED_PIN, OUTPUT);     // Initialize the D0 pin as an output
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_PIN, LOW);  // Turn the LED on by bringing the ouput pin to LOW voltage level
                               // the LED turns on because the other side of the LED is tied to 3.3V
                               // and sending this pin LOW sinks this side to ground.
  delay(1000);                 // Wait for a second
  digitalWrite(LED_PIN, HIGH); // Turn the LED off by bringing both leads to 3.3v
  delay(2000);                 // Wait for two seconds, then loop
}
```

For further learning, Here is a link to the Arduino IDE Core for the ESP8266 boards:
https://github.com/esp8266/arduino