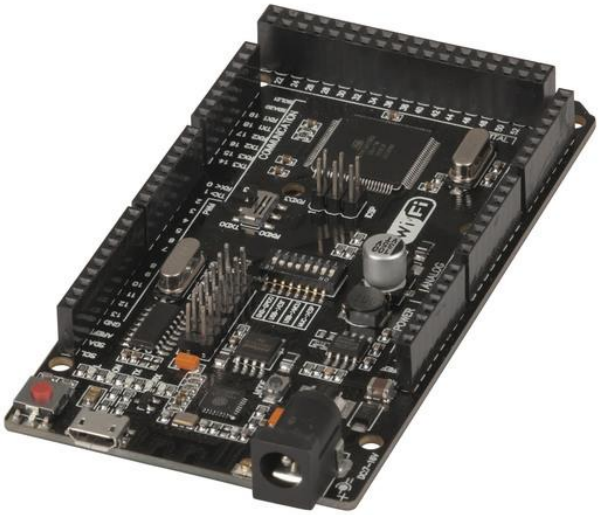
MEGA 2560 with Wi-Fi Dev Board

Part No: ML/SL - 0004

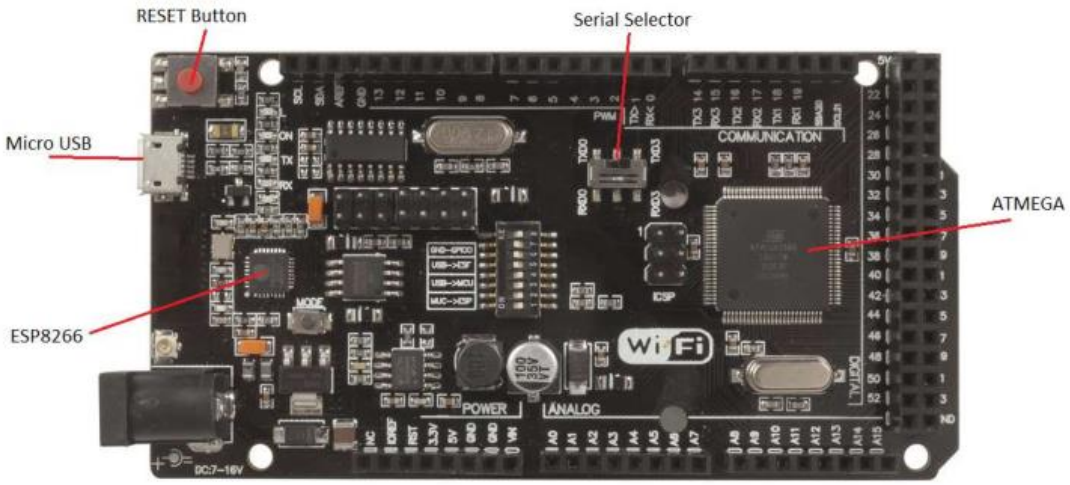


An Overview

An Arduino® Compatible + Wi-Fi Dual board that includes an ATMega2560 MCU with an ESP8266 chip to connect your projects to the cloud. The board includes a set of DIP switches to configure the connections, and uses a micro USB cable for connecting to the computer for programming.

This new boards have two chips traditional ATMega 328p chip as well as the ESP8266 Wi-Fi chips. This allows them to act as co-processors to each other and provide Wi-Fi functionality to your projects.

As these are two separate chips, the code must be uploaded to both. In the sources, they communicate with each other through Serial.read( ) and Serial.write( ) functions.
Set this by configuring the DIP switches as per the table below, so that the USB programmer can communicate between both chips individually. Once you have finished doing that, configuration of the DIP switches needs to be changed so that both these chips can communicate with each other, and send messages back and forth.



There are 8 DIP switches on the board, labelled 1 to 8 .

When you want to …	Dip Pins							
	1	2	3	4	5	6	7	8
… Program the Arduino MCU	OFF	OFF	ON	ON	OFF	OFF	OFF	OFF
… Program the ESP8266 Chip	OFF	OFF	OFF	OFF	ON	ON	ON	OFF
… Communicate between Computer serial and the ESP8266	OFF	OFF	OFF	OFF	ON	ON	OFF	OFF
… Connect the ESP8266 Serial to the ATMEGA Arduino Serial	ON	ON	OFF	OFF	OFF	OFF	OFF	OFF

Serial Selector

This board has an additional slide switch which will configure whether to use Serial 0 or Serial 3 for communication to the ESP8266. This is handy when you want to configure back and forth communication between the PC, Arduino, and ESP all at once.

Please note, when programming the ESP8266, you must manually press reset on the board to restart the ESP8266 and enable it to receive the programming instructions. Do this when it begins to say “uploading” on the Arduino IDE. This manual assumes you have configured the Arduino IDE to program ESP based chips, as a prior step.

For Example:

- When programming the Arduino Code, you must have switches 3 and 4 ON, and the others are off.
- Then when you want to upload the ESP8266 Code, you can switch 5,6, and 7 ON, & all the others are switched off.
- You can check with the serial monitor for the ESP8266 code, by turning switch 7 OFF (as a failsafe against reprogramming)
- Finally, when both codes are uploaded, you switch all switches off and turn on switches 1 and 2.
- This will enable your Arduino code to communicate to the ESP8266, and visa-versa, through the use of Serial.write() and Serial.read() commands, at whichever baud-rate that you define in code.
- Test each bit of code against the serial monitor to ensure that the flow is correct, and use the provided example code as a basis for your programming.