

CURSO BLYNK CON ARDUINO

CONECTA TUS PROYECTOS Y CONTROLALOS DESDE TU MOVIL

Instructor: Fidas Rodríguez

Contenido del curso

- Que es Blynk?
 - Como funciona Blynk
 - Instalando la App Blynk
 - Instalando la librería para Arduino
 - Arduino MKR1000
 - Instalar las librerías para MKR1000
 - Como diseñar una interfaz con Blynk
 - Primer proyecto
 - Operaciones principales de Blynk
 - Sincronización
 - Limitaciones y recomendaciones
- Que es el chip ESP8266?
 - Modelos disponibles
 - Como usar el ESP8266
 - Configuración Standalone
 - Instalando la tarjeta en el IDE
 - Instalando la librería
 - Primer proyecto con el ESP8266

Que es Blynk?

www.Blynk.cc

- Blynk es una plataforma que usa APPs en iOS y Android para controlar Arduino, Raspberry Pi y otras placas de desarrollo usando internet.
- Es un panel de control digital donde creas una interfaz gráfica para tu proyecto simplemente arrastrando y soltando widgets.
- Blynk no está ligado a una placa o shield específico. En su lugar, es compatible con el hardware de tu elección. Ya sea que tu Arduino o Raspberry Pi esté conectado a Internet a través de Wi-Fi, Ethernet o el nuevo ESP8266.

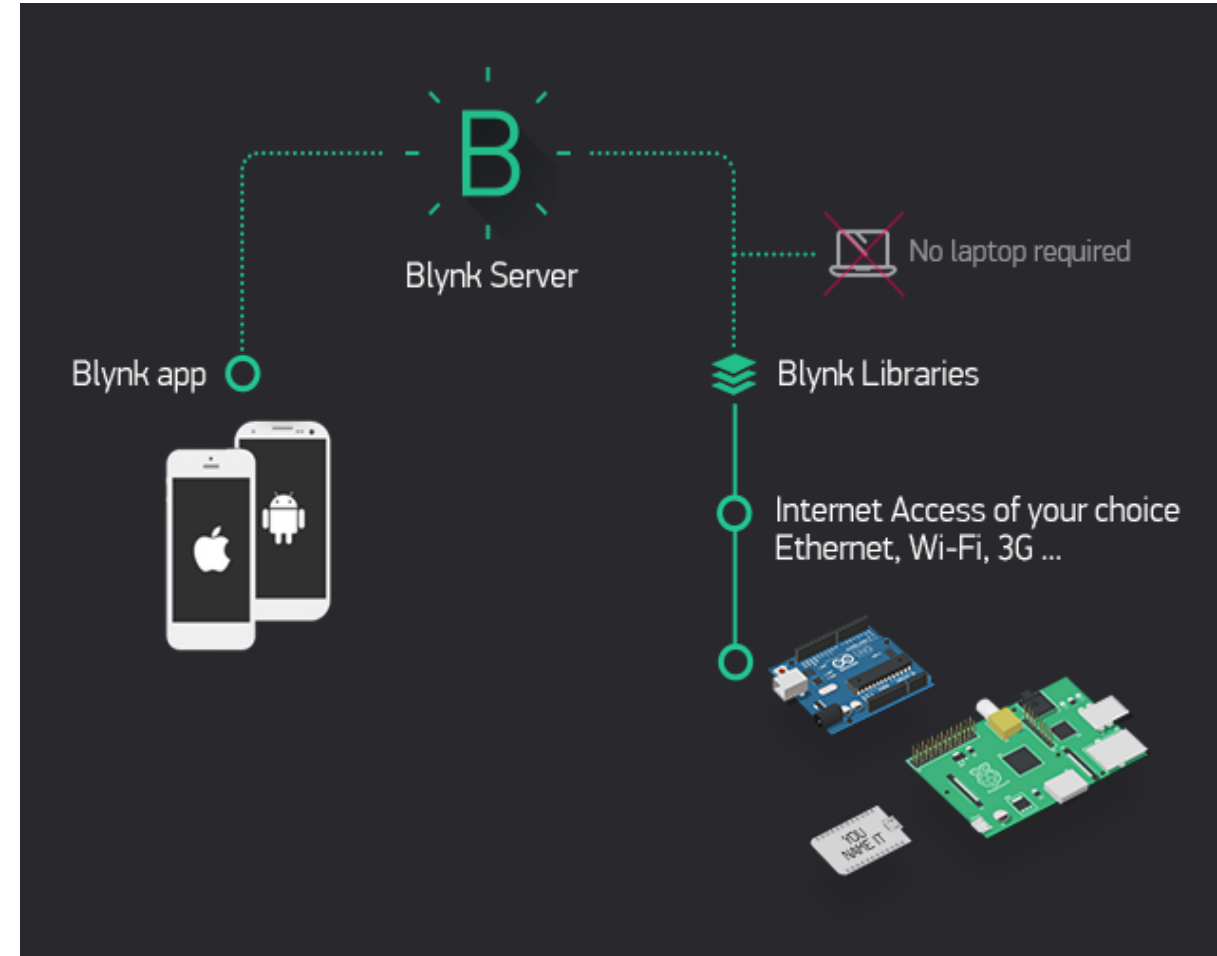


Como funciona Blynk

Blynk fue diseñado para el Internet de las Cosas. Puede controlar el hardware remotamente, puede exhibir datos del sensor, puede almacenar datos y visualizarlos.

Hay tres componentes principales en la plataforma:

- Blynk App - permite crear interfaces para tus proyectos usando varios widgets.
- Blynk Server - responsable de todas las comunicaciones entre el smartphone y el hardware. Se puede utilizar Blynk Cloud o ejecutar un servidor Blynk privado local. Es de código abierto, podría manejar fácilmente miles de dispositivos e incluso puede ser lanzado en una Raspberry Pi.
- Las bibliotecas de Blynk - para todas las plataformas de hardware populares - permiten la comunicación con el servidor y procesan todos los comandos entrantes y salientes.



Características

- API (*Application Programming Interface*) y UI (User Interface) similares para todos los dispositivos y hardware soportados
- Conexión a la nube usando: Ethernet, WiFi, Bluetooth y BLE, USB (Serial) ...
- Conjunto de widgets fáciles de usar
- Manipulación directa de pin sin escritura de código
- Fácil de integrar y agregar nueva funcionalidad usando Pines virtuales
- Monitorización de datos de historicos a través del widget Historial Gráfico
- Comunicación Device-to-Device con Bridge Widget
- Envío de correos electrónicos, tweets, etc ...

Que necesito para usar Blynk?

Sólo un par de cosas

1. Hardware. Un Arduino, Raspberry Pi, o un kit de desarrollo similar. Blynk trabaja a través de Internet. Esto significa que el hardware que elija debe ser capaz de conectarse a Internet. Algunas de las placas, como Arduino Uno necesitarán un shield Ethernet o Wi-Fi para comunicarse, otros ya están habilitados para Internet: como el ESP8266, Raspberri Pi con dongle WiFi, Photon de Partículas o SparkFun Blynk Board.
2. Un teléfono inteligente. La aplicación de Blynk es un constructor de interfaz bien diseñado. Funciona tanto en iOS como en Android.

Hardware soportado

Plataformas

Arduino (<https://github.com/blynkkk/blynk-library>)

- Arduino Uno, Duemilanove
- Arduino Nano, Mini, Pro Mini, Pro Micro, Due, Mega
- Arduino 101 (Intel Curie, with BLE)
- Arduino MKR1000
- Arduino Zero
- Arduino Yún (onboard WiFi and Ethernet, via Bridge)

Arduino-like

- Blynk Board
- ESP8266 (Generic, NodeMCU, Witty Cloud, Huzzah, WeMos D1, Seeed Wio Link, etc.)
- ESP32 Dev Board
- Intel Edison
- Intel Galileo
- Teensy 3.2/3.1
- Blue Pill (STM32F103C)
- BBC micro:bit
- LightBlue Bean , *soon*
- DFRobot Bluno
- RedBear Duo (WiFi, BLE)
- RedBearLab Blend Micro
- RedBearLab BLE Nano

- Seeed Tiny BLE
- Simblee BLE
- RFduino BLE
- The AirBoard
- Fishino Guppy, Uno, Mega
- TinyCircuits TinyDuino (CC3000)
- Microduino/mCookie Core, Core+, CoreUSB
- Wicked WildFire V2, V3, V4
- Digistump Oak
- chipKIT Uno32
- Alorium XLR8 (FPGA)
- LinkIt ONE (WiFi only)

Energia

- Texas Instruments
 - CC3200-LaunchXL
 - Tiva C Connected LaunchPad
 - Stellaris LM4F120 LaunchPad
 - MSP430F5529 + CC3100
- RedBearLab (CC3200, WiFi Mini)

Particle (formalmente Spark: <https://github.com/vshymanskyy/blynk-library-spark>)

- Core
- Photon
- Electron
- SparkFun RedBoard
- RedBear Duo (WiFi & BLE)

Hardware soportado

Plataformas

ARM mbed (<https://developer.mbed.org/users/vshymanskyi/code/Blynk/>)

- Seeed Tiny BLE
- RedBearLab BLE Nano
- BBC micro:bit
- STM32 Nucleo + Wiznet 5100 , *soon*

JavaScript (Node.js, Espruino, Browsers)
(<https://www.npmjs.com/package/blynk-library>)

- Regular PC with Linux / Windows / OS X
- Raspberry Pi (Banana Pi, Orange Pi, ...)
- BeagleBone Black
- Onion Omega
- Intel Galileo
- Intel Edison
- Intel Joule
- LeMaker Guitar
- LeMaker Banana Pro
- Samsung ARTIK 5
- PandaBoard, CubieBoard, pcDuino, Tessel 2
- VoCore (OpenWRT + [Espruino package](#))
- Espruino Pico

Python (MicroPython) (<https://github.com/wipy/wipy/tree/master/lib/blynk>)

WiPy

Lua (<https://github.com/blezek/blynk-esp>)

NodeMCU

Hardware soportado

Tipos de conexión a Arduino

- USB (Serial), conectado a tu computadora
- **Ethernet:**
 - Arduino Ethernet Shield (W5100)
 - Arduino Ethernet Shield 2 (W5500)
 - SeeedStudio Ethernet Shield V2.0 (W5200)
 - ENC28J60-based módulos
- **WiFi:**
 - ESP8266 como modem WiFi (con el firmware original)
 - Arduino WiFi 101 Shield
 - Arduino WiFi Shield
 - Adafruit CC3000 WiFi Breakout / Shield
 - RN-XV WiFly

Bluetooth Smart (BLE 4.0):

HM-10, HC-08
DFRobot BLE-Link módulo
Microduino/mCookie BLE
RedBearLab BLE Mini
nRF8001-based placas (Adafruit Bluefruit LE, etc.)

Bluetooth 2.0 Serial Port Profile (SPP)

HC-05, HC-06, ...

GSM/3G:

SIM800L , *pronto...*

Descargar e instalar Blynk

Instalar la APP

Visitemos la pagina www.Blynk.cc desde nuestro teléfono móvil y encontraremos el enlace para descargar e instalar la APP. O también la podemos buscar en google play <https://play.google.com/>.

Instalar las librerías Blynk para Arduino

Visitemos la pagina <http://docs.blynk.cc/> buscamos la sección Downloads y seleccionamos “Download the Blynk library”

Arduino MKR1000

www.Blynk.cc

Arduino MKR1000 ha sido diseñado para ofrecer una solución práctica y rentable para los que buscan agregar conectividad Wi-Fi a sus proyectos y cuentan con mínima experiencia previa en redes. El diseño incluye un circuito de carga de Li-Po que permite que el Arduino MKR1000 funcione con batería o 5V externo, cargando la batería Li-Po mientras esta conectado a la alimentación externa. El cambio de una fuente a otra se realiza automáticamente.



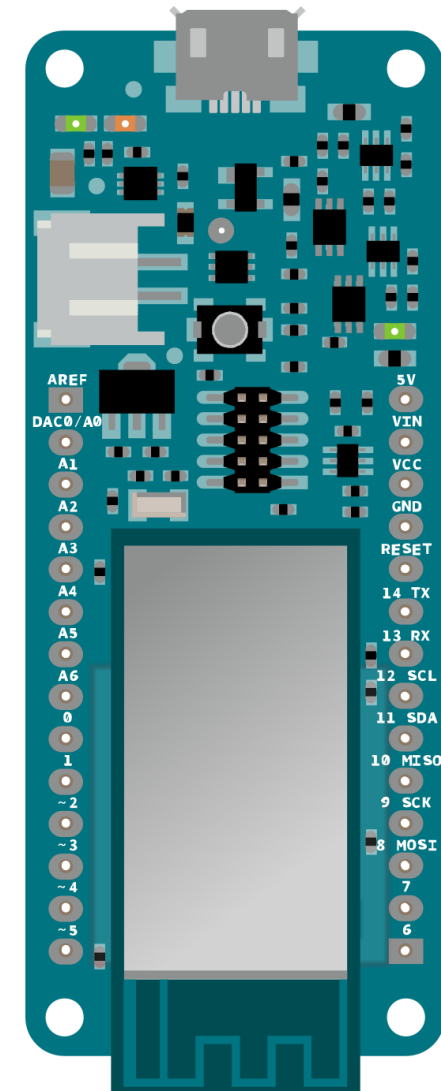
Características de la placa MKR1000

- A diferencia de la mayoría de las placas Arduino, la MKR1000 funciona a 3.3V. El voltaje máximo que los pines I / O pueden tolerar es 3.3V. Aplicar tensiones superiores a 3,3 V a cualquier pin de E / S podría dañar la placa. Mientras que la salida a los dispositivos digitales 5V es posible, la comunicación bidireccional con los dispositivos 5V necesita el cambio apropiado de nivel.
- MKR1000 utiliza un chip especializado que tiene una corriente de carga preestablecida de 350mAh. Esto significa que la capacidad MÍNIMA de la batería Li-Po será de 700 mAh. Una celda más grande tardará más tiempo en cargarse,. El chip esta programado con 4 horas de tiempo de carga, luego entra en el modo de reposo automático.

Microcontrolador	SAMD21 Cortex-M0+
Voltaje de operación	3.3 V
Voltaje de entrada (recomendado)	5 V
Pines Digitales I/O	8
Pines PWM Digitales I/O	4
Pines de entrada analogica	6
Pines de salida analogica	1
Memoria Flash	256 Kb
SRAM	32 Kb
Velocidad de reloj	48MHz
Longitud	68 mm
Alto	30 mm
Peso	12 gr.

Probemos la placa MKR1000

1. Abrir el IDE de Arduino
2. Conectar la placa MKR1000 al PC usando el cable serial
3. El IDE nos dará un mensaje de que es necesario descargar las librerías de la tarjeta, descargaremos el software y se instala automáticamente. Ya el IDE esta listo para trabajar con la placa.
4. Seleccionamos la placa y el puerto para conectarnos.
5. Buscamos el ejemplo “blink” en los programas de ejemplo, modificaremos el pin al cual esta conectado el Led, que es el pin 6, y no el 13 como como las otras placas.
6. Subimos el programa y verificamos el funcionamiento.

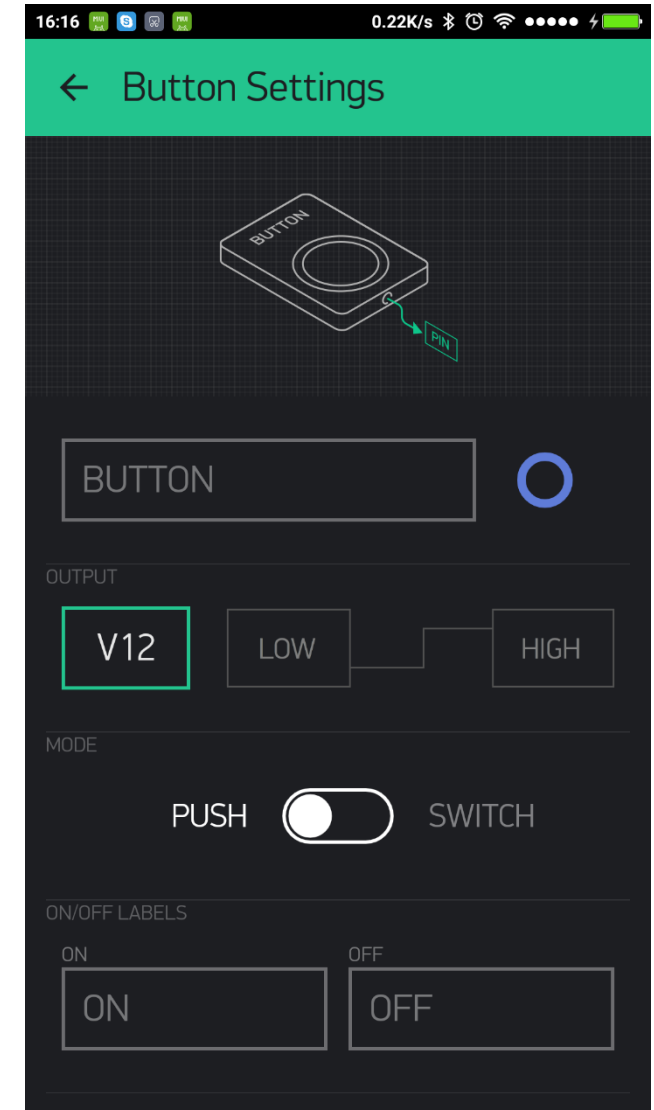


Ejercicio 1: Encendamos un Led con Blynk

1. Comenzaremos con la App de Blynk. Creemos una cuenta
2. Creamos un nuevo proyecto, debemos ponerle un nombre
3. Seleccionamos el hardware, en nuestro caso MKR1000
4. El código de autorización es necesario para conectar el hardware con la App, puedes copiarlo o enviártelo por email, también puedes tocarlo y se copiara en el portapapeles
5. Presionamos crear y el proyecto se habrá creado
6. Nos aparecerá la portada de nuestra interfaz vacía, presionamos en el signo “+”, y nos aparecerá la lista de Widgets
7. Seleccionaremos un “Button” que es un botón para encender nuestro Led, y lo posicionaremos en cualquier lugar que elijamos de la pantalla
8. Cada “widget” tiene su configuración, tocamos 2 veces sobre el, y podremos configurarlo

Proyecto 1: Encendamos un Led con Blynk

9. Podremos colocarle un nombre, elegir el color del led, seleccionar el pin, si el botón tiene retención, y las etiquetas para cuando esta encendido y apagado.
10. Una vez que terminemos de configurar el Led, y presionamos play, el proyecto se conectara a la placa Arduino. Pero primero hay que cargar el programa en la placa.
11. Los programas de ejemplo de la librería de Blynk son de gran ayuda para crear nuestros proyectos, seleccionaremos “Archivo/Ejemplos/Blynk/Boards_WiFi/Arduino_MKR1000”
12. En el programa de ejemplo colocaremos:
YourAuthToken es el código que nos enviaron por email
YourNetworkName es el nombre de la red WiFi
YourPassword es la contraseña de la red WiFi

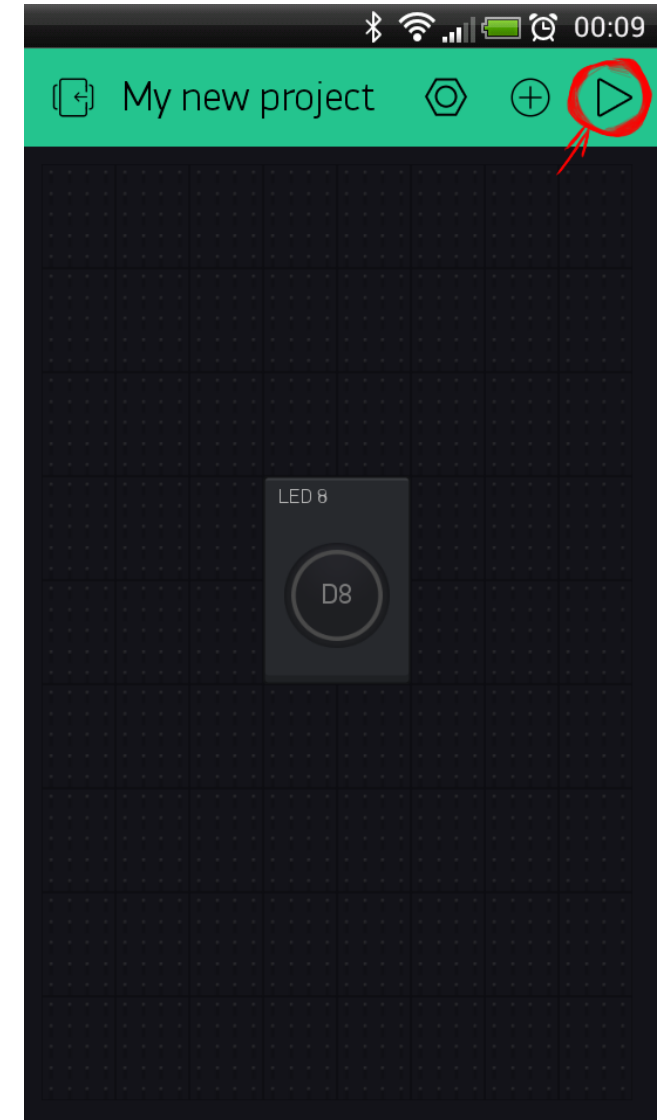


Proyecto 1: Encendamos un Led con Blynk

```
#define BLYNK_PRINT Serial // Comment this out to disable prints and save space
#include <SPI.h>
#include <WiFi101.h>
#include <BlynkSimpleMKR1000.h>
char auth[] = "YourAuthToken";
char ssid[] = "YourNetworkName";
char pass[] = "YourPassword";
void setup() {
  Serial.begin(9600);
  Blynk.begin(auth, ssid, pass);
}
void loop() {
  Blynk.run();
}
```


Proyecto 1: Encendamos un Led con Blynk

13. Subimos el programa a la placa Arduino
14. Vamos a la App y presionamos el triangulo “play” en nuestro proyecto y estaremos conectados a la placa
15. Al presionar el botón podremos controlar el pin 6 directamente



Operaciones principales de Blynk

- **Pines Virtuales**
- **Enviar datos de la App al hardware**
- **Tomar datos del hardware**
- **Sincronización**
- **Limitaciones y recomendaciones**

Pines Virtuales

Blynk puede controlar los pines digitales y analógicos del hardware directamente.

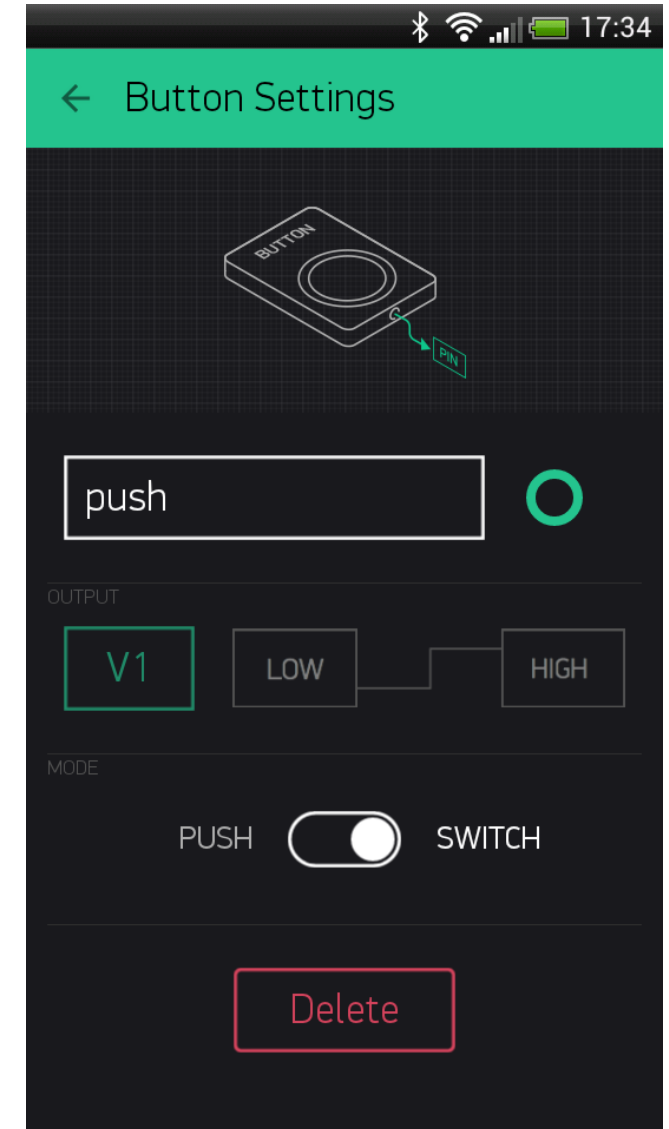
Los pines virtuales permiten enviar cualquier información del microcontrolador a la aplicación de Blynk y viceversa. Cualquier cosa que se conecte al hardware será capaz de hablar con Blynk. Con los pines virtuales puedes enviar algo desde la aplicación, procesarlo en el microcontrolador y luego devolverlo al smartphone. Puede activar funciones, leer dispositivos I2C, convertir valores, controlar servos y motores de corriente continua, etc.

Los pines virtuales se pueden utilizar para interactuar con bibliotecas externas (Servo, LCD y otros) e implementar funcionalidades personalizadas.

Enviar datos de la App al hardware

Es posible enviar cualquier información desde un Widgets en la aplicación al hardware. Todos los Widgets de Control pueden enviar datos a Pines virtuales en el hardware. Por ejemplo, el código siguiente muestra cómo obtener valores del widget de botones en la aplicación.

```
BLYNK_WRITE(V1) //Widget button escribe al pin V1
{
  int pinData = param.asInt();
}
```



Enviar datos de la App al hardware

Explicación del programa de ejemplo “GetData”.

Ejercicio 2: Basándose en el programa “GetData” de ejemplo de Blynk ubicado en “Archivo/Ejemplos/Blynk/GettingStarted” modificar el programa MKR1000 para que imprima en el monitor serial el valor que enviara un widget de Slider desde la App. Luego ubicar un Widget Slider L en la portada de la aplicación y configurarlo para que interaccione con el pin virtual 1 y envíe valores entre 0 y 100.

Ejercicio 3: Basándonos en el programa del ejercicio 2, modificarlo para que a través del slider nos envíe el tiempo en alto y bajo de un led (en D6) que parpadea, el rango de valores que enviara el slider será de 100 a 1000 ms.

Enviar datos de la App al hardware

Ejercicio 4: Basándonos en el programa del ejercicio 3, modificarlo para que a través de otro slider se controle adicionalmente el brillo del led mientras parpadea. Usaremos el pin virtual 2.

Tomar datos del hardware

Hay dos maneras de leer los datos del hardware y pasarlos a los widgets en la App usando pines virtuales.

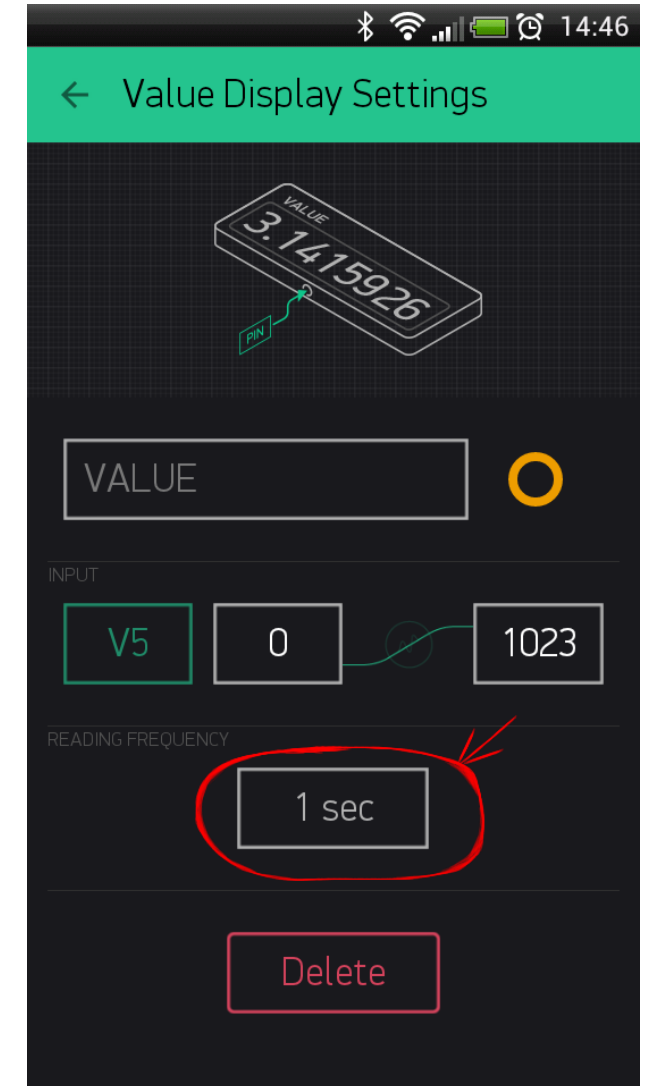
1. Realizando solicitudes mediante el Widget (requests by Widget)
2. Enviando datos desde el hardware (Pushing data)

Tomar datos del hardware

Realizando solicitudes mediante el Widget

Se realiza utilizando la frecuencia de lectura incorporada de Blynk mientras la App está activa estableciendo el parámetro frecuencia de lectura (Reading frequency) en el intervalo deseado.

```
BLYNK_READ(V5) // Widget lee el pin virtual V5
periodicamente
{
  Blynk.virtualWrite(5, millis() / 1000);
}
```

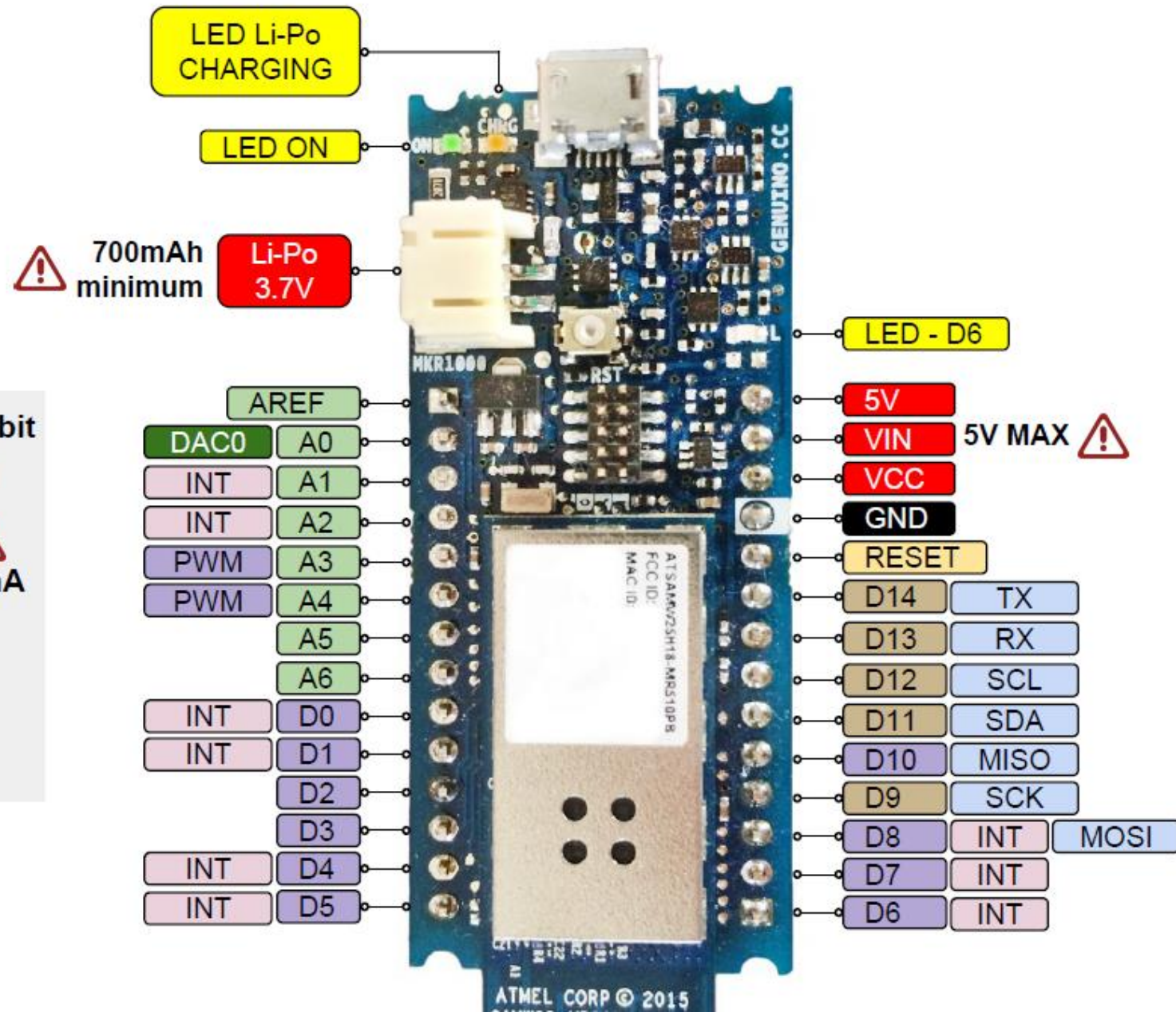


Tomar datos del hardware


Realizando solicitudes mediante el Widget

Explicación del programa de ejemplo “PushDataOnRequest”.










Ejercicio 5: Conectar un potenciómetro al pin analógico A1. Y basándose en el programa “PushDataOnRequest” de ejemplo de Blynk ubicado en “Archivo/Ejemplos/Blynk/GettingStarted” modificar el programa del ejercicio 4 para que un widget “Value Display” muestre en la portada de la App el voltaje presente en la entrada analógica. Usaremos el pin virtual 3.



Analog Input = **ADC 8/10/12 bit**
 Analog Output = **DAC 10 bit**

DC Current per I/O Pin = **7 mA** 

Flash Memory = **256 KB**
 SRAM = **32 KB**

	Power
	GND
	Analog Input
	Analog Output
	PWM Pin
	Digital Pin I/O
	Interrupt Pin
	Control Pin
	LED

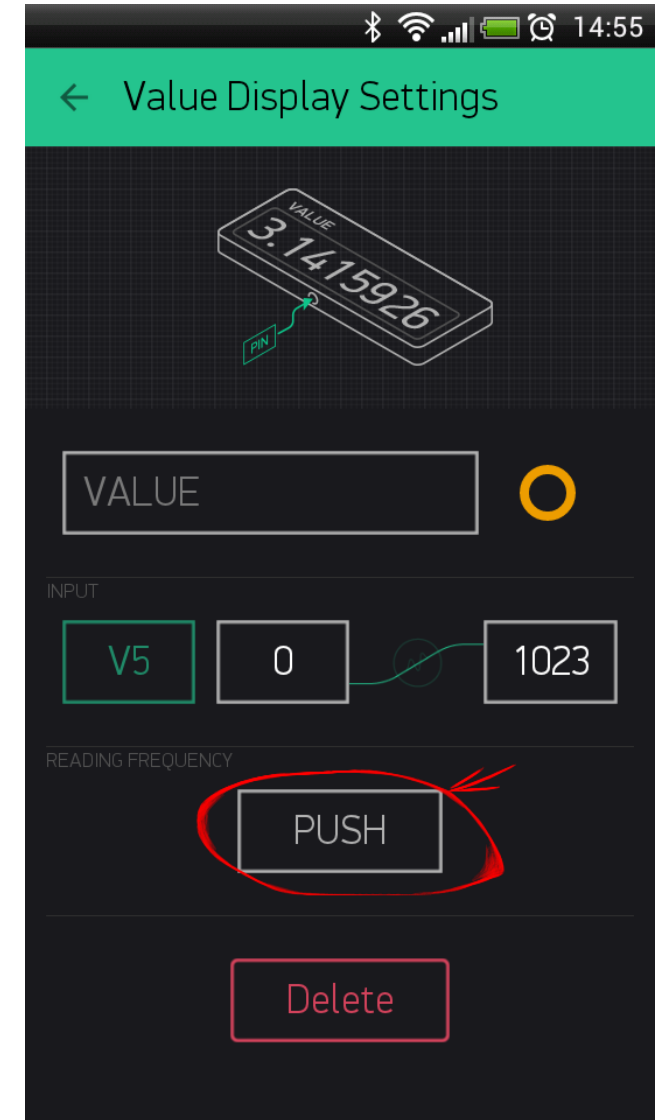
Tomar datos del hardware

Enviando datos desde el hardware

Si necesitamos enviar información de un sensor u otros datos del hardware a un Widget. Simplemente ajustemos la frecuencia en el modo PUSH.

Cualquier comando que el hardware envía a “Blynk Cloud” se almacena automáticamente en el servidor y es posible obtener esa información con un widget Gráfico.

Se recomienda enviar datos en intervalos (no mas de 100 por segundo) y para evitar errores. Podemos usar la biblioteca SimpleTimer que nos es útil para eventos periodicos. SimpleTimer está incluido en la biblioteca de Blynk.

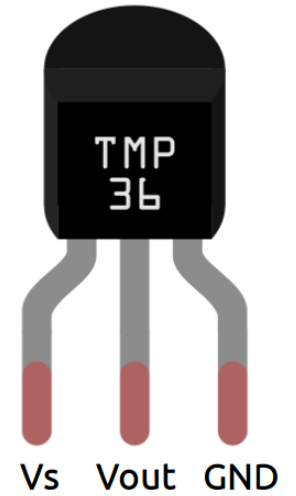
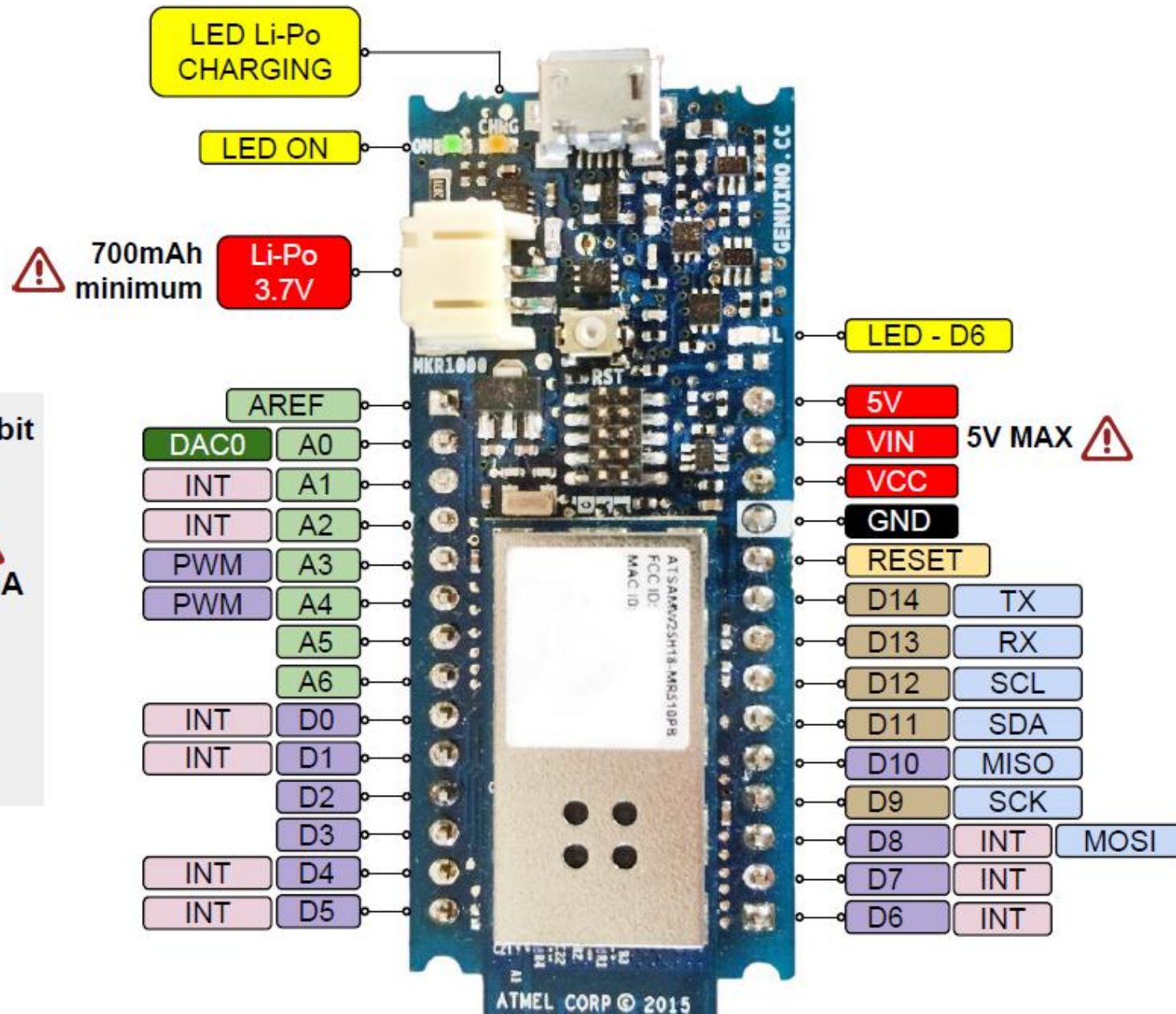


Tomar datos del hardware

Enviando datos desde el hardware


Explicación del programa de ejemplo “PushData”.

Ejercicio 6: Conectar un sensor de temperatura al pin analógico A2. Y basándose en el programa “PushData” de ejemplo de Blynk ubicado en “Archivo/Ejemplos/Blynk/GettingStarted” modificar el programa del ejercicio 5 para que un widget “Value Display Labeled” muestre en la portada de la App la temperatura medida por el sensor, con un decimal cada segundo. Usaremos el pin virtual 4.



- Power
- GND
- Analog Input
- Analog Output
- PWM Pin
- Digital Pin I/O
- Interrupt Pin
- Control Pin
- LED

Analog Input = **ADC 8/10/12 bit**
 Analog Output = **DAC 10 bit**

DC Current per I/O Pin = **7 mA** 

Flash Memory = **256 KB**
 SRAM = **32 KB**

Sincronización

Si el hardware pierde conexión a Internet o se restablece, es posible restaurar todos los valores de Widgets en la aplicación Blynk.

El comando `Blynk.syncAll()` restaura todos los valores de los Widgets basados en los últimos valores guardados en el servidor. Todos los estados de pines analógicos y digitales se restaurarán. Cada pin virtual realizará `BLYNK_WRITE`.

```
bool isFirstConnect = true;
BLYNK_CONNECTED() {
  if (isFirstConnect) {
    Blynk.syncAll();
    isFirstConnect = false;
  }
}
```

Sincronización

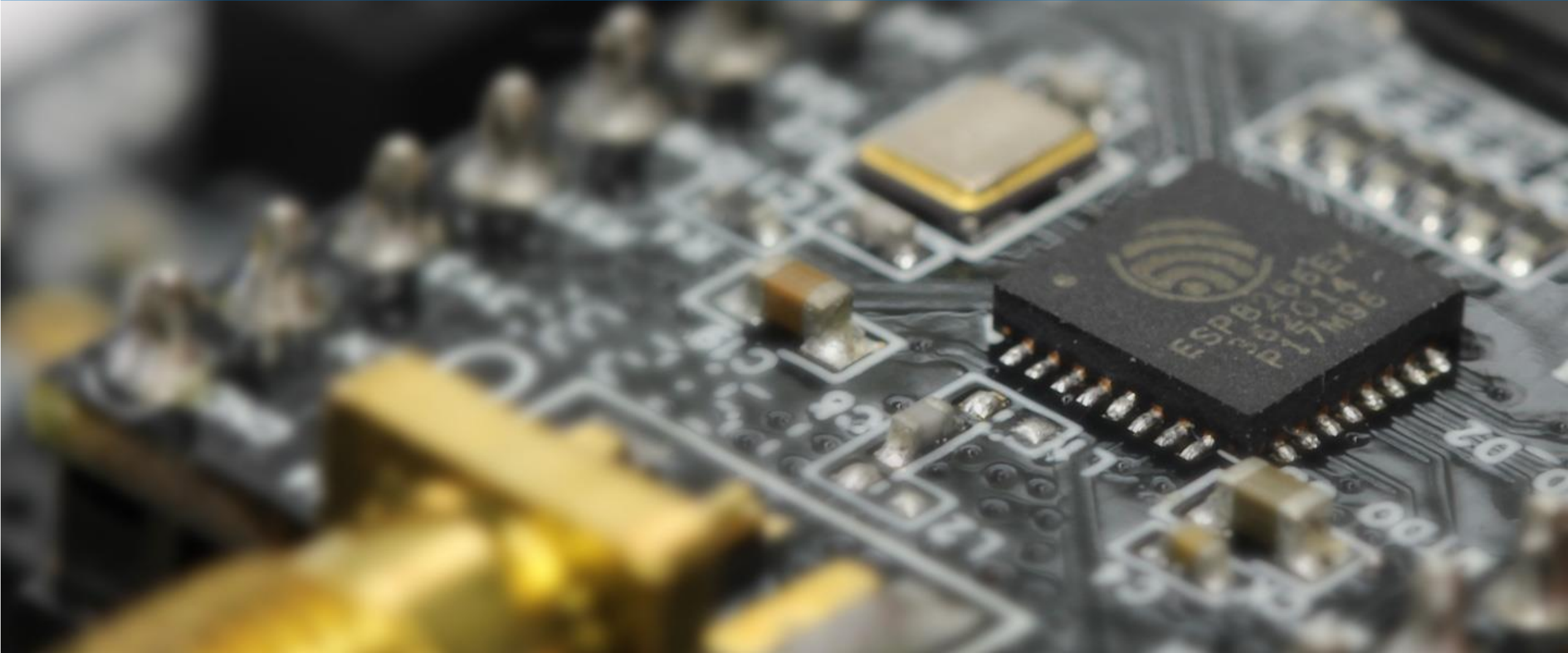
Ejercicio 7: Agregar el comando `Blynk.syncAll()` al programa del ejercicio 6, y comprobar que funciona correctamente.

Limitaciones y recomendaciones

- No poner `Blynk.virtualWrite` ni cualquier otro comando de Blynk.* dentro de `void loop ()` – esto causará muchos mensajes salientes al servidor de Blynk y la conexión será terminada.
- Se recomienda llamar a las funciones periódicamente. Se recomienda la biblioteca `SimpleTimer` es una biblioteca sencilla para eventos cronometrados.
- Evitar usar largos retrasos con `delay()` - puede causar interrupciones en la conexión.
- Si se envían más de 100 valores por segundo, puede causar un error de inundación y el hardware se desconectará automáticamente del servidor.
- Hay que tener cuidado al enviar una gran cantidad de comandos `Blynk.virtualWrite` ya que la mayoría del hardware no es muy potente (como ESP8266) por lo que puede no manejar muchas solicitudes.

ESP8266

Una alternativa a considerar para algunas aplicaciones



Que es el ESP8266?

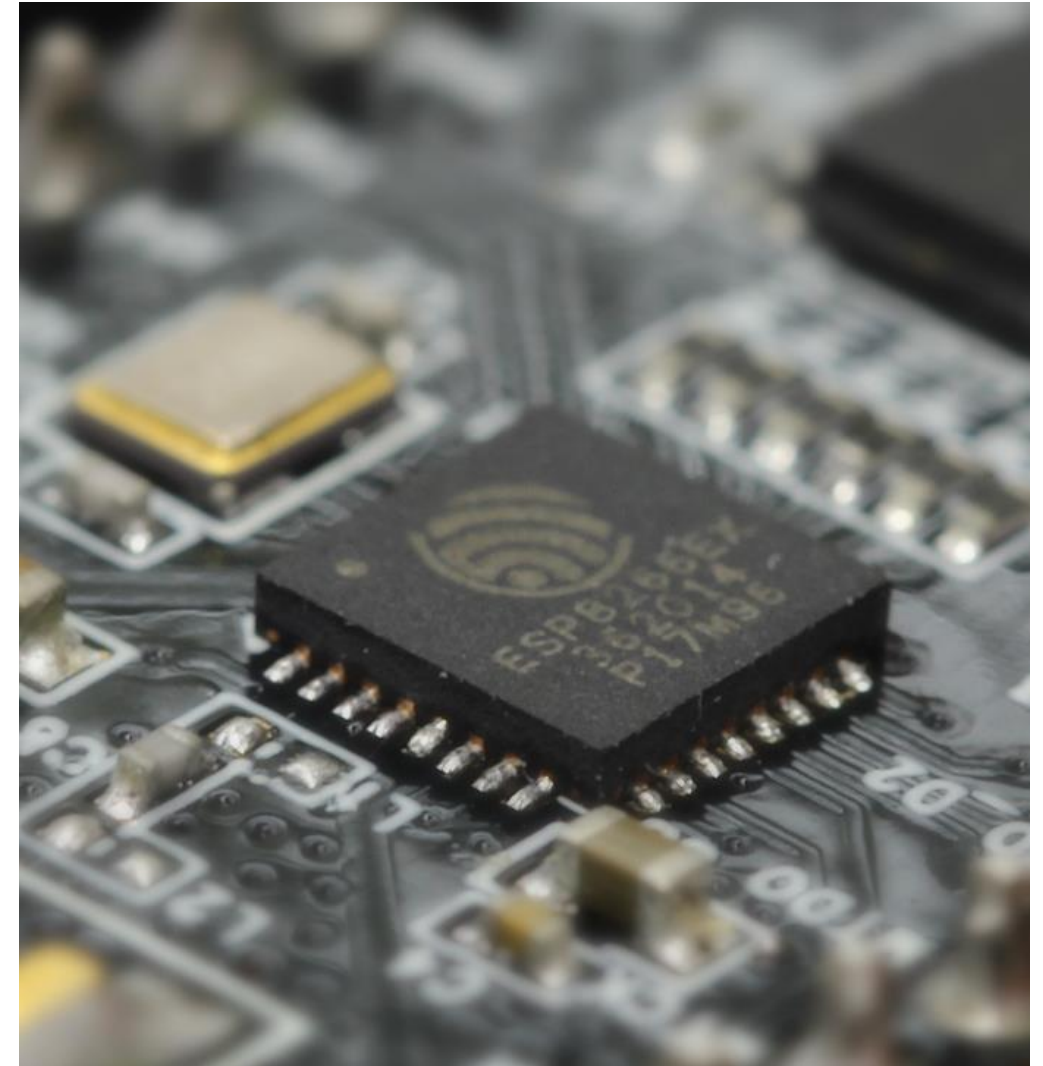
<https://espressif.com>

El ESP8266 es un chip Wi-Fi de bajo coste con pila TCP/IP completa y capacidad de MCU (Micro Controller Unit) producido por el fabricante chino **Espressif Systems**.

Este pequeño módulo permite a los microcontroladores conectarse a una red Wi-Fi y realizar conexiones TCP / IP sencillas utilizando comandos de tipo Hayes (AT).

El ESP8285 es un ESP8266 con 1 MB de flash incorporado, lo que permite dispositivos de un solo chip capaz de conectarse a Wi-Fi.

El sucesor de estos módulos es el ESP32.



Características

- 32-bit RISC CPU: Tensilica Xtensa LX106 corriendo a 80 MHz
- 64 KB de RAM para instrucciones, 96 KB de RAM para datos
- Externo QSPI flash - 512 KiB to 4 MiB* (hasta 16 MB es soportado)
- IEEE 802.11 b/g/n Wi-Fi
- 16 GPIO pines
- SPI, I²C,
- I²S interface con DMA (compartiendo pines con GPIO)
- UART en pines dedicados, adicionalmente un solo para transmisión UART puede ser habilitado en GPIO2
- 1 10-bit ADC
- 3.3V de voltaje de operación

SDK (Software Development Kit)

A finales de octubre de 2014, Espressif lanzó un kit de desarrollo de software (SDK) que permitió que el chip fuera programado, eliminando la necesidad de un microcontrolador separado. Espressif mantiene dos versiones del SDK - una basada en RTOS y la otra basada en callbacks.

Una alternativa al SDK oficial de Espressif es esp-open-sdk de código abierto que se basa en la cadena de herramientas GCC y es mantenida por Max Filippov.

Otra alternativa es "Unofficial Development Kit" de Mikhail Grigorev.

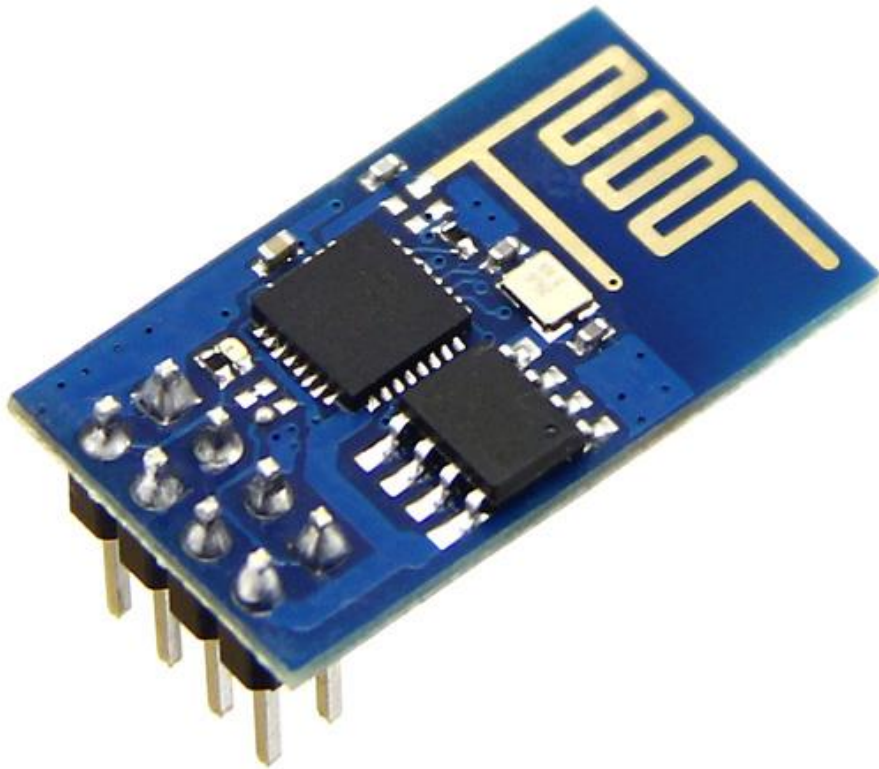
Otros SDK de código abierto incluyen:

- NodeMCU: un firmware basado en Lua.
- Arduino: un firmware basado en C ++. Este núcleo permite que la CPU ESP8266 y sus componentes Wi-Fi sean programados como cualquier otro dispositivo Arduino. El Arduino Core ESP8266 está disponible a través de GitHub.
- MicroPython: un puerto del MicroPython (una implementación de Python para dispositivos embebidos) a la plataforma ESP8266.
- ESP8266 BASIC: Un intérprete básico de código abierto específicamente diseñado para la Internet de las cosas. Entorno de desarrollo basado en el navegador de autoservicio.
- Mongoose Firmware: Un firmware de código abierto con servicio gratuito en la nube [11]

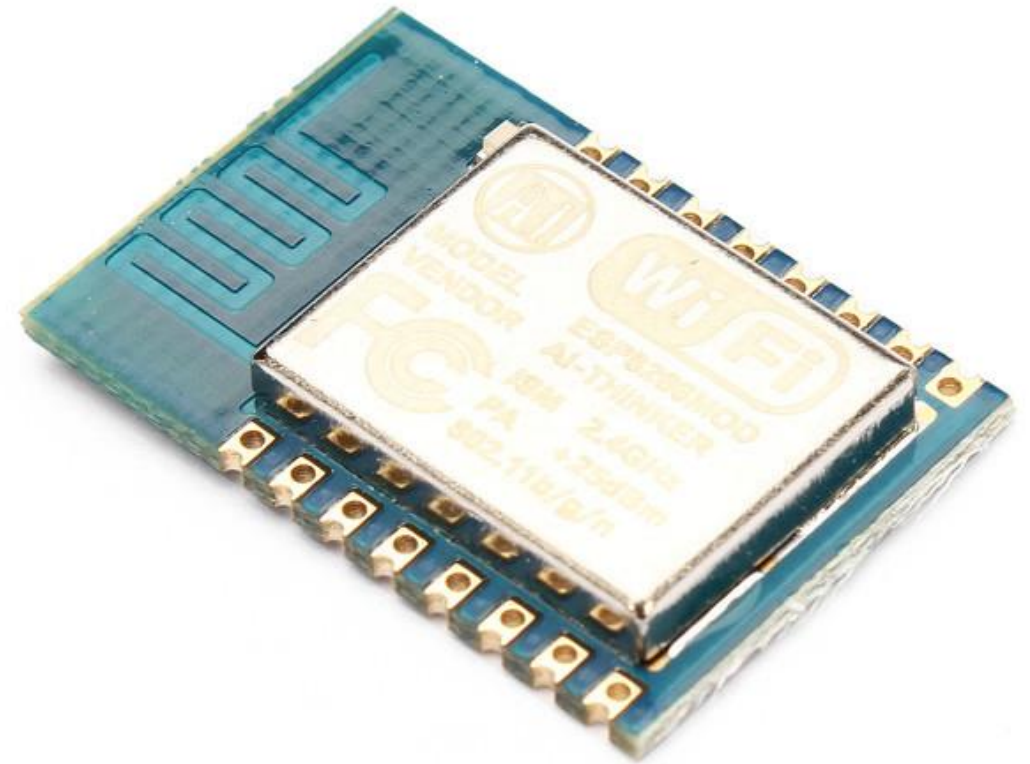
Módulos AI Thinker

Name ↕	Active pins ↕	Pitch ↕	Form factor ↕	LEDs ↕	Antenna ↕	Shielded? ↕	dimensions (mm) ↕	Notes ↕
ESP-01	6	0.1"	2×4 DIL	Yes	PCB trace	No	14.3 × 24.8	
ESP-02	6	0.1"	2×4 castellated	No	U-FL connector	No	14.2 × 14.2	
ESP-03	10	2 mm	2×7 castellated	No	Ceramic	No	17.3 × 12.1	
ESP-04	10	2 mm	2×4 castellated	No	None	No	14.7 × 12.1	
ESP-05	3	0.1"	1×5 SIL	No	U-FL connector	No	14.2 × 14.2	
ESP-06	11	misc	4×3 dice	No	None	Yes	14.2 × 14.7	Not FCC approved
ESP-07	14	2 mm	2×8 pinhole	Yes	Ceramic + U-FL connector	Yes	20.0 × 16.0	Not FCC approved
ESP-08	10	2 mm	2×7 castellated	No	None	Yes	17.0 × 16.0	Not FCC approved
ESP-09	10	misc	4×3 dice	No	None	No	10.0 × 10.0	
ESP-10	3	2 mm?	1×5 castellated	No	None	No	14.2 × 10.0	
ESP-11	6	0.05"	1×8 pinhole	No	Ceramic	No	17.3 × 12.1	
ESP-12	14	2 mm	2×8 castellated	Yes	PCB trace	Yes	24.0 × 16.0	FCC and CE approved ^[14]
ESP-12E	20	2 mm	2×8 castellated	Yes	PCB trace	Yes	24.0 × 16.0	4 MB Flash
ESP-12F	20	2 mm	2×8 castellated	Yes	PCB trace	Yes	24.0 × 16.0	FCC and CE approved. Improved antenna performance. 4 MB Flash
ESP-13	16	1.5 mm	2×9 castellated	No	PCB trace	Yes	W18.0 x L20.0	Marked as "FCC". Shielded module is placed sideways, as compared to the ESP-12 modules.
ESP-14	22	2 mm	2×8 castellated +6	No	PCB trace	Yes	24.3 x 16.2	

Módulos AI Thinker mas populares



ESP-01

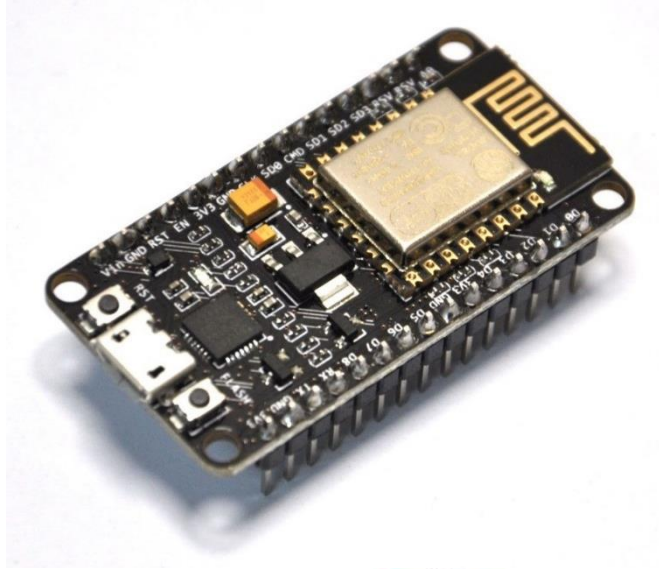


ESP-12

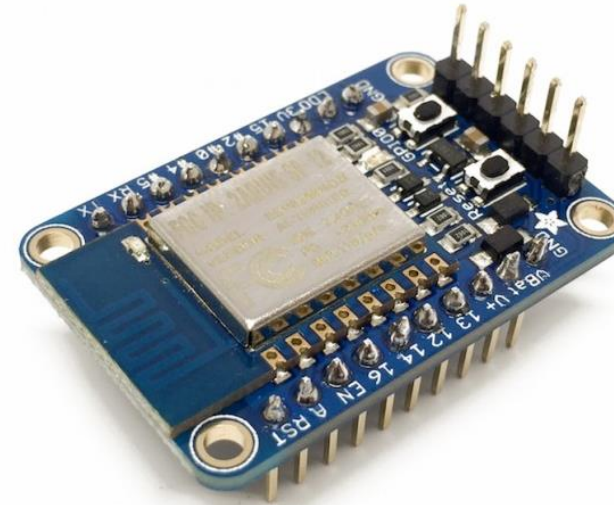
Módulos de otros fabricantes

Name	Active pins	Pitch	Form factor	LEDs	Antenna	Shielded?	dimensions (mm)	Notes
Bolt IoT	14	0.1"	2×14 DIL	Yes	PCB trace	Yes	30 × 40	Comes with an on Board SD card and technologies like Lib-Discovery and Fail Safe Mode. Has its own cloud for IoT.
Olimex MOD-WIFI-ESP8266 ^[15]	2	0.1"	UEXT module	Yes	PCB trace	No	?	Only RX/TX are connected to UEXT connector
Olimex MOD-WIFI-ESP8266-DEV ^[16]	20	0.1"	2×11 DIL + castellated	Yes	PCB trace	No	?	All available GPIO pins are connected, also has pads for soldering UEXT connector (with RX/TX and SDA/SCL signals)
NodeMCU DEVKIT	14	0.1"	2×15 DIL	Yes	PCB trace	Yes	?	Uses the ESP-12 module, includes USB serial interface
Adafruit Huzzah ESP8266 breakout ^[17]	14	0.1"	2×10 DIL	Yes	PCB trace	Yes	25 × 38	Uses the ESP-12 module
SparkFun ESP8266 Thing ^[18] WRL-13231	12	0.1"	2×10 DIL	Yes	PCB trace + U.FL socket	No	58 × 26	FTDI serial header, Micro-USB socket for power, includes Li-ion battery charger
KNEWRON Technologies smartWIFI ^[19]	12	0.1"	2×20 DIL	Yes 1 RGB	PCB trace	Yes	25.4 × 50.8	CP2102 USB bridge, includes battery charger, micro-USB socket for power and battery charging, 1 RGB LED and USER / Reflash button
WeMos D1	12	0.1"	Arduino Uno	Yes	PCB trace	Yes	53.4 × 68.6	Uses the ESP-12F module, Micro-USB socket
WeMos D1 R2 ^[20]	12	0.1"	Arduino Uno	Yes	PCB trace	Yes	53.4 × 68.6	Uses the ESP-12F module, Micro-USB socket
WeMos D1 Mini ^[21]	12	0.1"	2×8 DIL	Yes	PCB trace	Yes	25.6 × 34.2	Uses the ESP-12F module, Micro-USB socket
ESPert ESPresso Lite ^[22]	16	0.1"	2×8 DIL	Yes	PCB trace	Yes	26.5 × 57.6	Uses the WROOM-02 module. Produced in limited quantity as beta version.
ESPert ESPresso Lite V2.0 ^[23]	24	0.1"	2×10 DIL	Yes	PCB trace	Yes	28 × 61	Improved design and feature to ESPresso Lite.
In-Circuit ESP-ADC ^[24]	18	0.1"	2×9 DIL	No	U.FL socket	No	22.9 × 14.9	Uses the ESP8266EX
Watterott ESP-WROOM02-Breakout ^[25]	14	0.1"	2×10 DIL	Yes	PCB trace	Yes	40.64 × 27.94	Uses the Espressif ESP-WROOM-02 module.

Módulos de otros fabricantes mas populares



NodeMCU



Adafruit
Huzzah



Sparkfun
Thing



Wemos

Arduino core para chip ESP8266 WiFi

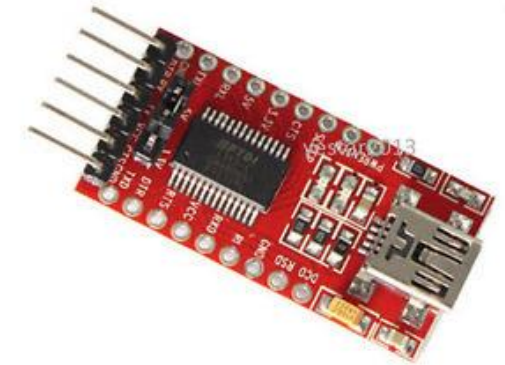
El *ESP8266 Arduino core* brinda soporte para el chip ESP8266 en el entorno Arduino. Permite escribir sketches utilizando funciones y bibliotecas familiares de Arduino, y ejecutarlos directamente en el ESP8266, no requiere microcontrolador externo.

El *ESP8266 Arduino core* viene con las bibliotecas para comunicarse a través de WiFi usando TCP y UDP, configurar servidores HTTP, mDNS, SSDP y DNS, hacer actualizaciones OTA, usar un sistema de archivos en memoria flash, trabajar con tarjetas SD, servos, periféricos SPI e I2C .

La pagina del proyecto en GitHub es: <https://github.com/esp8266/Arduino>, hay se encuentra toda su documentación.

Requerimientos de hardware

1. El modulo ESP8266 requiere un voltaje de alimentación de 3,3V con una capacidad de alrededor de 250mA para operar correctamente, no es recomendable hacerlo funcionar con los 3,3V de la placa Arduino UNO, (pues la conexión Wi-Fi consume eventualmente mas de lo que esa fuente puede entregar).
2. Para cargar el Firmware (programarlo) necesitaremos un convertidor USB-Serial de 3,3V (preferiblemente que use un chip con oscilador de cristal).
3. Es posible usar el convertidor USB-Serial de la placa Arduino Uno para programarlo y también la fuente de 3,3V de la placa solo para programarlo.



Instalando el ESP8266 Arduino core

1. Iniciar Arduino y abrir la ventana Preferencias. Y escribir `http://arduino.esp8266.com/stable/package_esp8266com_index.json` en el campo Gestor de URLs adicionales de tarjetas. Se pueden agregar varias URL, separándolas con comas.
2. Abrir el Gestor de tarjetas desde el menú Herramientas> Placa e instalar la plataforma esp8266.
3. Luego seleccionar la placa ESP8266 en el menú Herramientas>Placa después de la instalación, también seleccionar el puerto.

El mejor lugar para hacer preguntas relacionadas con este núcleo es ESP8266 foro de la comunidad: <http://www.esp8266.com/arduino>.