Comentario técnico: CTC-085
Componente: **ESP32(-WROOM-32)**
Autor: Sergio Caprile, Senior R&D Engineer

Revisiones	Fecha	Comentarios
0	03/08/18	

Les presentamos el módulo ESP32 de Espressif, un combo WiFi con Bluetooth y Bluetooth Low Energy

Para comenzar, y reiterando al alerta técnico CTA-020, aclaramos a los amantes del módulo ESP-WROOM-02 (basado en el ESP8266) que esto es un mundo totalmente diferente: este módulo requiere sí o sí que se le grabe algún firmware.

Sin entrar en los apabullantes detalles y características técnicas que hacen que el ESP32 sea muy superior, nos centraremos en las alternativas de uso y las diferencias fundamentales.

Desde el punto de vista del "usuario final" (por llamarlo de algún modo), refiriéndonos a quien emplea el módulo a través de su puerto serie controlándolo mediante comandos AT, observamos lo siguiente:

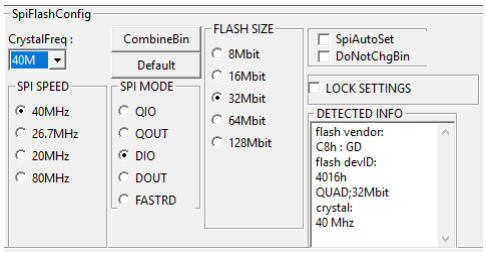
- el stack TCP/IP presente en el firmware AT del ESP-WROOM-02 está basado en una versión arcana de lwIP[1] tomada del git head en algún punto de su evolución posterior a la versión 1.4.1. El fabricante ha realizado modificaciones sobre el mismo.
- el stack TCP/IP con que trabaja el entorno del ESP32 es el actual de lwIP, actualizado periódicamente, y no contiene modificaciones importantes del fabricante.
- un módulo ESP32-WROOM-32 (o ESP-WROOM-32, el nombre anterior) viene de fábrica con un firmware de prueba. El mismo está pensado para que un desarrollador cargue en él su programa, por lo que para poder usarlo con comandos AT es necesario cargarle el firmware correspondiente. El mismo puede obtenerse en la página del fabricante, o en Cika, donde como siempre lo probamos y desmalezamos el camino hacia la documentación.
  - para grabar dicho firmware utilizaremos la herramienta de programación por el puerto serie, conectándonos al módulo por su puerto 0 (pines 35 y 34) y conectando brevemente a masa el GPIO0 al aplicar la alimentación.[2]
  - cargamos los siguientes archivos en las siguientes direcciones:

0x1000	bootloader/bootloader.bin
0x20000	at_customize.bin
0x21000	customized_partitions/ble_data.bin
0x24000	customized_partitions/server_cert.bin
0x26000	customized_partitions/server_key.bin
0x28000	customized_partitions/server_ca.bin
0xf000	phy_init_data.bin
0x100000	esp-at.bin
0x8000	partitions_at.bin

1 lwIP: LightWeight IP, un desarrollo open source sueco que podríamos decir es un standard del mercado en procesadores de estas características, inclusive en ARM Cortex-M. El autor original es Adam Dunkels, el mismo de uIP y el Contiki OS (entre los relacionados).

2 Un ejemplo del hardware necesario lo podemos encontrar en el esquemático de la herramienta de desarrollo, el ESP32-GROVER

- y configuramos:



- como bien menciona el alerta técnico CTA-020, el puerto serie 0 sólo se utiliza para cargar firmware y para debugging, los comandos AT operan sobre el puerto 1 (GPIO16 y GPIO17, pines 27 y 28)

Desde el punto de vista del "usuario avanzado" (nuevamente, por llamarlo de algún modo), disponemos de varias opciones. En principio el fabricante entrega un entorno de trabajo, el IDF (IoT Development Framework), el cual podemos mantener actualizado sincronizándolo mediante git (https://github.com/espressif/esp-idf/releases). Existen tutoriales en la página web del fabricante guiándonos sobre cómo instalar todo el entorno y llegar a correr un ejemplo (https://esp-idf.readthedocs.io/en/latest/get-started/); cumplen con su función y lo hacen muy bien.

Sin embargo, dado el nivel de complejidad con el que estamos trabajando, y la naturaleza de desarrollo de software, sistemas de comunicaciones, y software de aplicación, no nos es posible dar soporte en ninguna de estas áreas, más allá de ofrecer nuestros servicios de desarrollo y consultoría. A tal fin, y en base a nuestra experiencia, nos permitimos sugerir el uso de Mongoose-OS (https://mongoose-os.com/) como alternativa al IDF del fabricante.

Mongoose-OS corre como una tarea sobre el IDF (que a su vez emplea freeRTOS) y nos presenta un entorno event-driven en el cual podemos desarrollar nuestra aplicación aprovechando un poderoso web server y un framework que nos permite abstraernos de los detalles de la comunicación, reduciéndolos a eventos que podemos atender si nuestra aplicación lo necesita, o dejarlos al OS. Una ventaja adicional es que ofrece un intérprete de una versión reducida de JavaScript; esto nos permite desarrollar nuestras pruebas de concepto prototipando rápidamente una mini-aplicación en un lenguaje orientado a eventos de alto nivel, y luego ir portando paulatinamente una a una las funciones necesarias a C, ya que el entorno JavaScript permite llamar a funciones del usuario en C.

Todo lo que puede hacerse en el IDF, puede hacerse con Mongoose-OS; de hecho este último lo utiliza (excepto donde provee alternativas de mayores prestaciones como por ejemplo el web server).

La única desventaja es que la compilación es lenta, se realiza en el sitio del proveedor[3]. La ventaja es que no necesitamos instalar el compilador ni el Eclipse, el IDE es extremadamente simple y opera como una página web en cualquier navegador[4].

Este entorno se ofrece en una licencia Apache 2.0, lo cual permite su uso en aplicaciones comerciales sin necesidad de abrir el código. Eligiendo pagar una licencia comercial se accede a algunos beneficios que simplifican la actualización de firmware y el manejo de flotas de dispositivos.

Si bien existen otras opciones, no consideramos que estén a la altura de lo que un desarrollador profesional requiere, siendo tal vez excelentes para hobbyistas o una curiosidad interesante.

3 nuestro código viaja por la Internet; encriptado en SSL, pero es visto por el fabricante.

4 corremos una aplicación en nuestra computadora y ésta arranca un servidor web y abre una ventana en el navegador. A través de esto interactuamos. Sí, corre sobre GNU/Linux sin problemas, también puede ser manzanita y ventanitas.