El gran libro
por ESP32forth

versión 1.1 - 21. octubre 2023

Autor

- Marc PETREMANN petremann@arduino-forth.com

Colaboradores

- Vaclav POSSELT
-

Indice

Introducción

Desde 2019 he gestionado varios sitios web dedicados al desarrollo del lenguaje FORTH para placas ARDUINO y ESP32, así como la versión web eForth :

- ARDUINO : https://arduino-forth.com/
- ESP32 : https://esp32.arduino-forth.com/
- eForth web : https://eforth.arduino-forth.com/

Estos sitios están disponibles en dos idiomas, francés e inglés. Cada año pago por el alojamiento del sitio principal. **arduino-forth.com**.

Tarde o temprano –y lo más tarde posible– sucederá que ya no podré garantizar la sostenibilidad de estos lugares. La consecuencia será que la información difundida por estos sitios desaparecerá.

Este libro es la recopilación del contenido de mis sitios web. Se distribuye gratuitamente desde un repositorio de Github. Este método de distribución permitirá una mayor sostenibilidad que los sitios web.

De paso, si algunos lectores de estas páginas desean hacer su aporte, son bienvenidos. :

- para sugerir capítulos ;
- para informar errores o sugerir cambios ;
- para ayudar con la traducción...

Ayuda de traducción

Google Translate te permite traducir textos fácilmente, pero con errores. Por eso pido ayuda para corregir las traducciones.

En la práctica, proporciono los capítulos ya traducidos en formato LibreOffice. Si desea ayudar con estas traducciones, su función será simplemente corregir y devolver estas traducciones.

Corregir un capítulo lleva poco tiempo, de una a unas pocas horas.

Para contactar conmigo : petremann@arduino-forth.com

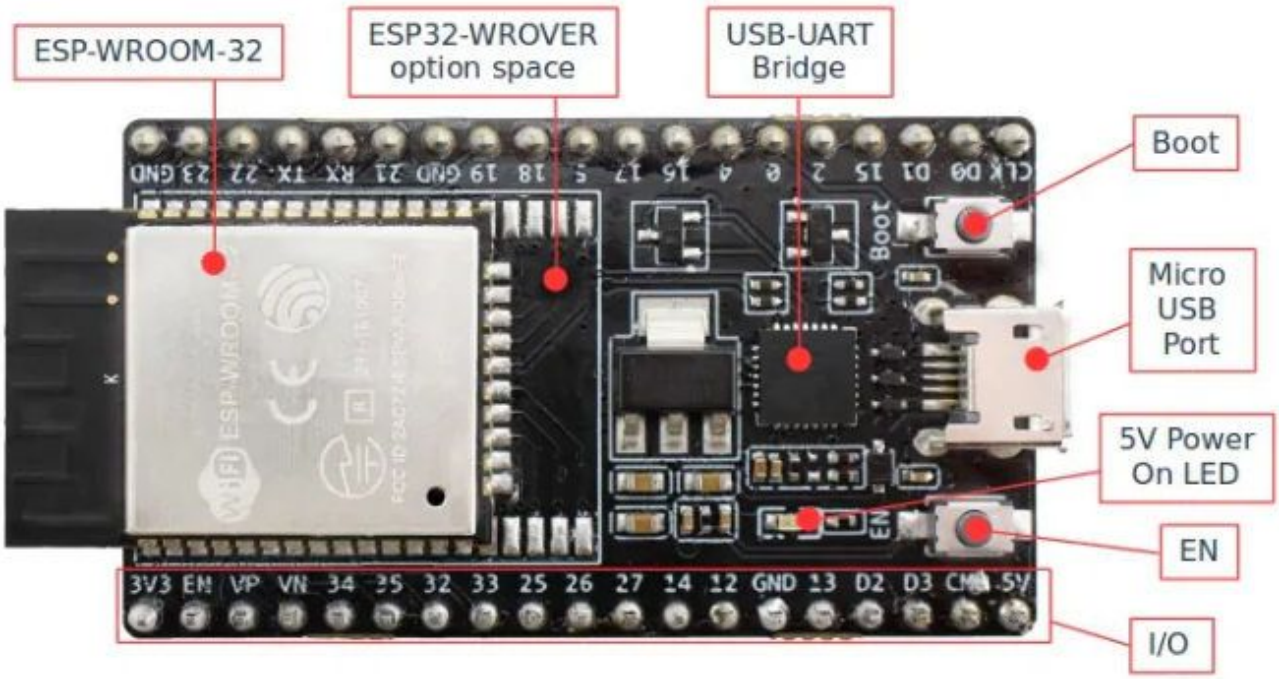
Descubrimiento de la tarjeta ESP32

Presentación

La placa ESP32 no es una placa ARDUINO. Sin embargo, las herramientas de desarrollo aprovechan ciertos elementos del ecosistema ARDUINO, como ARDUINO IDE.

Puntos fuertes

En cuanto al número de puertos disponibles, la tarjeta ESP32 se sitúa entre un ARDUINO



NANO y un ARDUINO UNO. El modelo básico tiene 38 conectores :

Los dispositivos ESP32 incluyen :

- 18 canales de convertidor analógico a digital. (ADC)
- 3 interfaces SPI
- 3 interfaces UART
- 2 interfaces I2C
- 16 canales de salida PWM
- 2 convertidores de digital a analógico (DAC)
- 2 interfaces I2S

- 10 GPIO de detección capacitiva

La funcionalidad ADC (convertidor analógico a digital) y DAC (convertidor digital a analógico) están asignadas a pines estáticos específicos. Sin embargo, puedes decidir qué pines son UART, I2C, SPI, PWM, etc. Sólo necesitas asignarlos en el código. Esto es posible gracias a la función de multiplexación del chip ESP32.

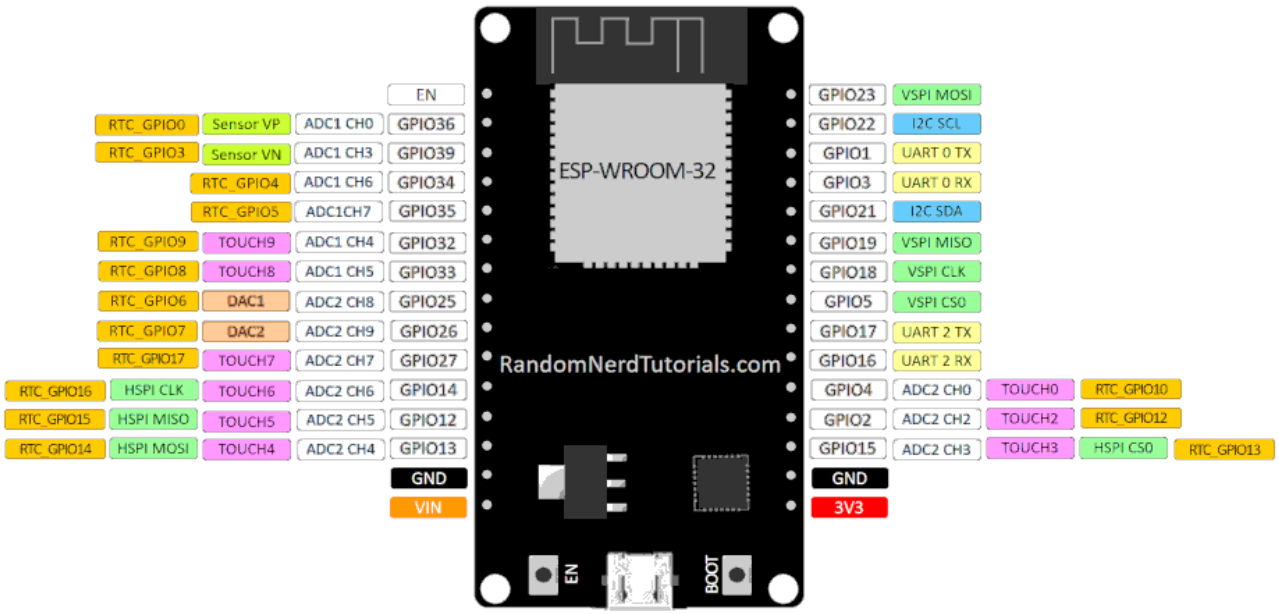
La mayoría de los conectores tienen múltiples usos.

Pero lo que distingue a la placa ESP32 es que está equipada de serie con soporte WiFi y Bluetooth, algo que las placas ARDUINO sólo ofrecen en forma de extensiones.

Entradas/salidas GPIO en ESP32

Aquí, en foto, la tarjeta ESP32 desde la que explicaremos el papel de las diferentes entradas/salidas GPIO :

La posición y la cantidad de E/S GPIO pueden cambiar según la marca de la tarjeta. Si este es el caso, sólo son auténticas las indicaciones que aparecen en el mapa físico. En la foto, fila inferior, de izquierda a derecha : CLK, SD0, SD1, G15, G2, G0, G4, G16.....G22, G23, GND.



En este diagrama, vemos que la fila inferior comienza con 3V3 mientras que en la foto, esta E/S está al final de la fila superior. Por lo tanto, es muy importante no confiar en el diagrama y, en su lugar, verificar la correcta conexión de los periféricos y componentes en la tarjeta física ESP32.

Las placas de desarrollo basadas en un ESP32 generalmente tienen 33 pines aparte de los de la fuente de alimentación. Algunos pines GPIO tienen funciones un tanto particulares :

GPIO	Posibles nombres
6	SCK/CLK
7	SCK/CLK
8	SDO/SD0
9	SDI/SD1
10	SHD/SD2
11	CSC/CMD

Si tu tarjeta ESP32 tiene E/S GPIO6, GPIO7, GPIO8, GPIO9, GPIO10, GPIO11, definitivamente no debes usarlas porque están conectadas a la memoria flash del ESP32. Si los usas el ESP32 no funcionará.

Las E/S GPIO1(TX0) y GPIO3(RX0) se utilizan para comunicarse con la computadora en UART a través del puerto USB. Si los utilizas, ya no podrás comunicarte con la tarjeta.

GPIO36(VP), GPIO39(VN), GPIO34, GPIO35 I/O se pueden utilizar solo como entrada. Tampoco tienen resistencias pullup y pulldown internas incorporadas.

El terminal EN le permite controlar el estado de encendido del ESP32 a través de un cable externo. Está conectado al botón EN de la tarjeta. Cuando el ESP32 está encendido, está a 3,3 V. Si conectamos este pin a tierra el ESP32 se apaga. Puedes usarlo cuando el ESP32 está en una caja y quieres poder encenderlo/apagarlo con un interruptor.

Periféricos ESP32

Para interactuar con módulos, sensores o circuitos electrónicos, el ESP32, como cualquier microcontrolador, dispone de multitud de periféricos. Hay más que en una placa Arduino clásica.

ESP32 tiene los siguientes periféricos:

- 3 interfaces UART
- 2 interfaces I2C
- 3 interfaces SPI
- 16 salidas PWM
- 10 sensores capacitivos
- 18 entradas analógicas (ADC)
- 2 salidas DAC

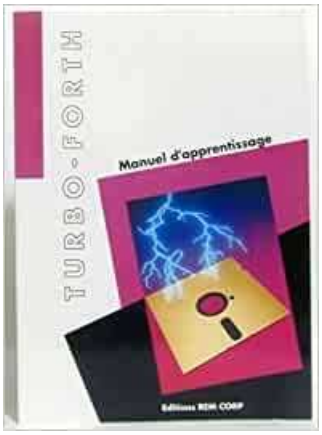
ESP32 ya utiliza algunos periféricos durante su funcionamiento básico. Por lo tanto, hay menos interfaces posibles para cada dispositivo.

¿Por qué programar en lenguaje FORTH en ESP32?

Preámbulo

Llevo programando en FORTH desde 1983. Dejé de programar en FORTH en 1996. Pero nunca he dejado de seguir la evolución de este lenguaje. Reanudé la programación en 2019 en ARDUINO con FlashForth y luego ESP32forth.

Soy coautor de varios libros sobre el idioma FORTH:



- Introduction au ZX-FORTH (ed Eyrolles - 1984 - ASIN:B0014IGOXO)
- Tours de FORTH (ed Eyrolles - 1985 - ISBN-13: 978-2212082258)
- FORTH pour CP/M et MSDOS (ed Loisitech - 1986)
- TURBO-Forth, manuel d'apprentissage (ed Rem CORP - 1990)
- TURBO-Forth, guide de référence (ed Rem CORP - 1991)

Programar en el lenguaje FORTH siempre fue un hobby hasta que en 1992 me contactó el gerente de una empresa que trabajaba como subcontratista para la industria del automóvil. Tenían inquietudes por el desarrollo de software en lenguaje C. Necesitaban encargar un autómata industrial.

Los dos diseñadores de software de esta empresa programaron en lenguaje C: TURBO-C de Borland para ser precisos. Y su código no podía ser lo suficientemente compacto y rápido como para caber en los 64 kilobytes de memoria RAM. Corría el año 1992 y no existían las ampliaciones de tipo memoria flash. ¡En estos 64 KB de RAM teníamos que meter MS-DOS 3.0 y la aplicación!

Durante un mes, los desarrolladores del lenguaje C habían estado dando vuelta al problema en todas direcciones, incluso aplicando ingeniería inversa con SOURCER (un desensamblador) para eliminar partes no esenciales del código ejecutable.

Analicé el problema que se me presentó. Partiendo de cero, creé, solo, en una semana, un prototipo perfectamente operativo y que cumplía con las especificaciones. Durante tres años, de 1992 a 1995, creé numerosas versiones de esta aplicación que se utilizó en las líneas de montaje de varios fabricantes de automóviles.

Límites entre lenguaje y aplicación

Todos los lenguajes de programación se comparten de la siguiente manera:

- un intérprete y código fuente ejecutable: BASIC, PHP, MySQL, JavaScript, etc... La aplicación está contenida en uno o más archivos que serán interpretados cuando sea necesario. El sistema debe alojar permanentemente al intérprete que ejecuta el código fuente;
- un compilador y/o ensamblador: C, Java, etc. Algunos compiladores generan código nativo, es decir ejecutable específicamente sobre un sistema. Otros, como Java, compilan código ejecutable en una máquina Java virtual.

El lenguaje FORTH es una excepción. Integra:

- un intérprete capaz de ejecutar cualquier palabra en el CUARTO idioma
- un compilador capaz de ampliar el diccionario de CUARTAS palabras

¿Qué es una CUARTA palabra?

Una FORTH palabra designa cualquier expresión de diccionario compuesta por caracteres ASCII y utilizable en interpretación y/o compilación: palabras le permite enumerar todas las palabras en el FORTH diccionario.

Ciertas palabras FORTH solo se pueden usar en la compilación: **`if else then`** por ejemplo.

Con el lenguaje FORTH, el principio esencial es que no creamos una aplicación. ¡En ADELANTE, ampliamos el diccionario ! Cada nueva palabra que defina será una parte tan importante del diccionario FORTH como todas las palabras predefinidas cuando se inicie FORTH. Ejemplo :

```
: typeToLoRa ( -- )
    0 echo !    \ desactive l'echo d'affichage du terminal
    ['] serial2-type is type
  ;
: typeToTerm ( -- )
    ['] default-type is type
    -1 echo !   \ active l'echo d'affichage du terminal
  ;
```

Creamos dos nuevas palabras: **`typeToLoRa`** y **`typeToTerm`** que completarán el diccionario de palabras predefinidas.

¿Una palabra es una función?

Si y no. De hecho, una palabra puede ser una constante, una variable, una función... Aquí, en nuestro ejemplo, la siguiente secuencia:

`: typeToLoRa ...código... ;`

tendría su equivalente en lenguaje C:

`void typeToLoRa() { ...código... }`

En FORTH idioma, no hay límite entre el idioma y la aplicación.

En FORTH, como en el lenguaje C, puede utilizar cualquier palabra ya definida en la definición de una nueva palabra.

Sí, pero entonces ¿por qué FORTH en lugar de C?

Estaba esperando esta pregunta.

En lenguaje C, solo se puede acceder a una función a través de la función principal `main()` . Si esta función integra varias funciones adicionales, resulta difícil encontrar un error de parámetro en caso de un mal funcionamiento del programa.

Por el contrario, con FORTH es posible ejecutar - a través del intérprete - cualquier palabra predefinida o definida por usted, sin tener que pasar por la palabra principal del programa.

Se puede acceder inmediatamente al intérprete FORTH en la tarjeta ESP32 a través de un programa tipo terminal y un enlace USB entre la tarjeta ESP32 y la PC.

La compilación de programas escritos en lenguaje FORTH se realiza en la tarjeta ESP32 y no en el PC. No hay carga. Ejemplo :

```
: >gray (n -- n')
    dup 2/ xor \ n' = n xor ( 1 desplazamiento lógico a la derecha )
  ;
```

Esta definición se transmite copiando/pegando en el terminal. El intérprete/compilador FORTH analizará la secuencia y compilará la nueva palabra `>gray` .

En la definición de `>gray` , vemos la secuencia `dup 2/ xor` . Para probar esta secuencia, simplemente escríbala en la terminal. Para ejecutar `>gray` , simplemente escriba esta palabra en la terminal, precedida por el número a transformar.

Lenguaje FORTH comparado con el lenguaje C

Esta es la parte que menos me gusta. No me gusta comparar el lenguaje FORTH con el lenguaje C. Pero como casi todos los desarrolladores usan el lenguaje C, voy a probar el ejercicio.

Aquí hay una prueba con `if()` en lenguaje C:

```
if(j > 13){              // Si tous les bits sont recus
    rc5_ok = 1;          // Le processus de decodage est OK
    detachInterrupt(0);  // Desactiver l'interruption externe (INT0)
    return;
}
```

Pruebe con `if` en el lenguaje FORTH (fragmento de código):

```
var-j @ 13 >        \ Si tous les bits sont recus
    if
        1 rc5_ok !  \ Le processus de decodage est OK
        di          \ Desactiver l'interruption externe (INT0)
        exit
    then
```

Aquí está la inicialización de registros en lenguaje C :

```
void setup() {
  //  Configuration du module Timer1
  TCCR1A = 0;
  TCCR1B = 0;          // Desactive le module Timer1
  TCNT1  = 0;          // Definit valeur préchargement Timer1 sur 0
(reset)
  TIMSK1 = 1;          // activer interruption de debordement Timer1
}
```

La misma definición en CUARTO idioma:

```
: setup ( -- )
  \ Configuration du module Timer1
  0 TCCR1A !
  0 TCCR1B !     \ Desactive le module Timer1
  0 TCNT1  !     \ Définit valeur préchargement Timer1 sur 0 (reset)
  1 TIMSK1 !     \ activer interruption de debordement Timer1
;
```

Qué le permite hacer FORTH en comparación con el lenguaje C

Entendemos que FORTH da acceso inmediatamente a todas las palabras del diccionario, pero no sólo eso. A través del intérprete también accedemos a toda la memoria de la tarjeta ESP32. Conéctese a la placa ARDUINO que tiene instalado FlashForth, luego simplemente escriba:

```
hex here 100 dump
```

Deberías encontrar esto en la pantalla del terminal :

```
3FFEE964             DF DF 29 27 6F 59 2B 42 FA CF 9B 84
3FFEE970 39 4E 35 F7 78 FB D2 2C A0 AD 5A AF 7C 14 E3 52
3FFEE980 77 0C 67 CE 53 DE E9 9F 9A 49 AB F7 BC 64 AE E6
3FFEE990 3A DF 1C BB FE B7 C2 73 18 A6 A5 3F A4 68 B5 69
3FFEE9A0 F9 54 68 D9 4D 7C 96 4D 66 9A 02 BF 33 46 46 45
3FFEE9B0 45 39 33 33 2F 0D 08 18 BF 95 AF 87 AC D0 C7 5D
3FFEE9C0 F2 99 B6 43 DF 19 C9 74 10 BD 8C AE 5A 7F 13 F1
3FFEE9D0 9E 00 3D 6F 7F 74 2A 2B 52 2D F4 01 2D 7D B5 1C
3FFEE9E0 4A 88 88 B5 2D BE B1 38 57 79 B2 66 11 2D A1 76
3FFEE9F0 F6 68 1F 71 37 9E C1 82 43 A6 A4 9A 57 5D CA 9A
3FFEEA00 4C AD 03 F1 F8 AF 2E 1A B4 67 9C 71 25 98 E1 A0
3FFEEA10 E6 29 EE 2D EF 6F C7 06 10 E0 33 4A E1 57 58 60
3FFEEA20 08 74 C6 70 BD 70 FE 01 5D 9D 00 9E F7 B7 E0 CA
```

```
3FFEEA30 72 6E 49 16 0E 7C 3F 23 11 8D 66 55 CE F6 18 01
3FFEEA40 20 E7 48 63 D1 FB 56 77 3E 9A 53 7D B6 A7 A5 AB
3FFEEA50 EA 65 F8 21 3D BA 54 10 06 16 E6 9E 23 CA 87 25
3FFEEA60 E7 D7 C4 45
```

Esto corresponde al contenido de la memoria flash.

¿Y el lenguaje C no podía hacer eso?

Sí, pero no tan simple e interactivo como en el lenguaje FORTH.

Veamos otro caso que destaca la extraordinaria compacidad del lenguaje FORTH...

Pero ¿por qué una pila en lugar de variables?

La pila es un mecanismo implementado en casi todos los microcontroladores y microprocesadores. Incluso el lenguaje C aprovecha una pila, pero no tienes acceso a ella.

Sólo el lenguaje FORTH brinda acceso completo a la pila de datos. Por ejemplo, para hacer una suma, apilamos dos valores, ejecutamos la suma, mostramos el resultado: `2 5 + .` muestra `7 `.

Es un poco desestabilizador, pero cuando comprendes el mecanismo de la pila de datos, aprecias enormemente su formidable eficiencia.

La pila de datos permite pasar datos entre palabras ADELANTE mucho más rápidamente que procesando variables como en el lenguaje C o cualquier otro lenguaje que use variables.

¿Estás convencido?

Personalmente, dudo que este único capítulo lo convierta irremediablemente a programar en el lenguaje FORTH. Cuando buscas dominar las placas ESP32, tienes dos opciones :

- programar en lenguaje C y utilizar las numerosas bibliotecas disponibles. Pero permanecerá encerrado en las capacidades de estas bibliotecas. Adaptar códigos al lenguaje C requiere conocimientos reales de programación en lenguaje C y dominar la arquitectura de las tarjetas ESP32. Desarrollar programas complejos siempre será un problema.
- prueba la FORTH aventura y explora un mundo nuevo y emocionante. Por supuesto, no será fácil. Necesitará comprender en profundidad la arquitectura de las tarjetas ESP32, los registros y las banderas de registro. A cambio, tendrás acceso a una programación perfectamente adaptada a tus proyectos.

¿Hay alguna solicitud profesional escrita en FORTH?

¡Oh sí! Empezando por el telescopio espacial HUBBLE, algunos de cuyos componentes fueron escritos en lenguaje FORTH.

El TGV ICE alemán (Intercit y Express) utiliza procesadores RTX2000 para controlar motores mediante semiconductores de potencia. El lenguaje de máquina del procesador RTX2000 es el lenguaje FORTH.

Este mismo procesador RTX2000 se utilizó para la sonda Philae que intentó aterrizar en un cometa.

La elección del lenguaje FORTH para aplicaciones profesionales resulta interesante si consideramos cada palabra como una caja negra. Cada palabra debe ser simple, por lo tanto tener una definición bastante corta y depender de pocos parámetros.

Durante la fase de depuración, resulta fácil probar todos los valores posibles procesados por esta palabra. Una vez convertida en perfectamente fiable, esta palabra se convierte en una caja negra, es decir, una función en la que tenemos absoluta confianza en su correcto funcionamiento. De palabra en palabra, es más fácil hacer que un programa complejo sea confiable en FORTH que en cualquier otro lenguaje de programación.

Pero si nos falta rigor, si construimos plantas de gas, también es muy fácil que una aplicación funcione mal, ¡o incluso que falle por completo!

Finalmente, es posible, en FORTH idioma, escribir las palabras que definas en cualquier idioma humano. Sin embargo, los caracteres utilizables están limitados al conjunto de caracteres ASCII entre 33 y 127. Así es como podríamos reescribir simbólicamente las palabras alto y bajo:

```
\ Pin de puerto activo, no cambie otros.
: __/ ( pinmask portadr -- )
    mset
  ;
\ Deshabilite un pin de puerto, no cambie los demás.
: \__ ( pinmask portadr -- )
    mclr
  ;
```

A partir de este momento, para encender el LED, puedes escribir :

```
_O_ __/ \ luces LED
```

¡Sí! ¡ La secuencia `_O_ __/` está en FORTH idioma!

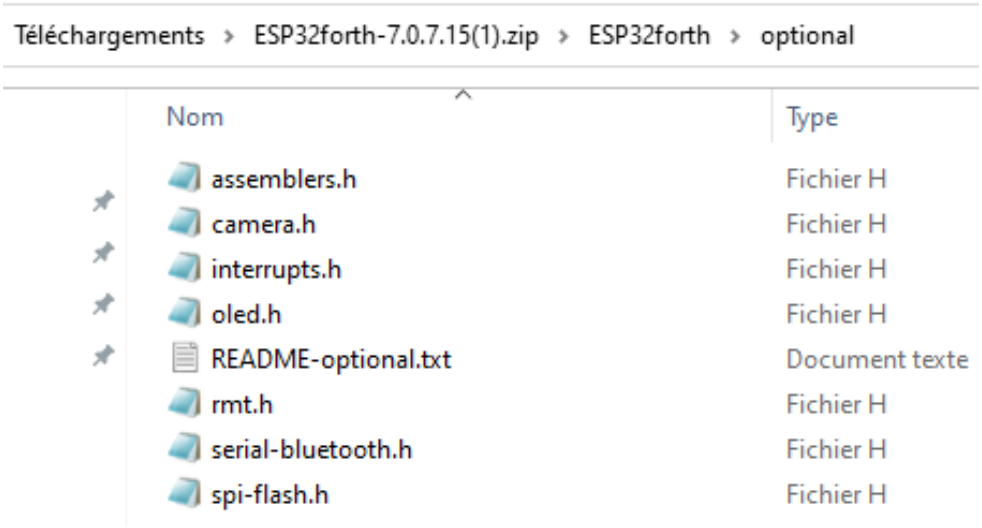
Con ESP32forth, aquí están todos los caracteres a tu disposición que pueden componer una FORTH palabra:

```
~}|{zyxwvutsrqponmlkjihgfedcba`_
^]\[ZYXWVUTSRQPONMLKJIHGFEDCBA@?
>=<;:9876543210/.-,++)('&%$#"!
```

Buena programación.

Instalación de la biblioteca OLED para SSD1306

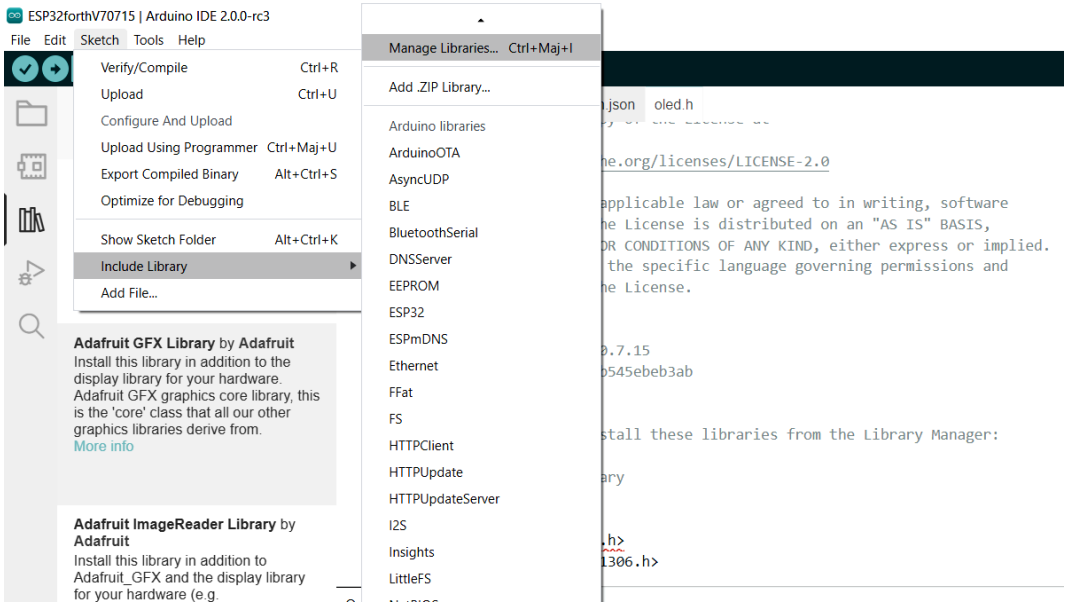
Desde ESP32 en adelante versión 7.0.7.15, las opciones están disponibles en la carpeta **optional**:



Para tener el vocabulario `oled`, copie el archivo **oled.h** a la carpeta que contiene el archivo **ESP32forth.ino**.

Luego inicie ARDUINO IDE y seleccione el archivo **ESP32forth.ino** más reciente.

Si la biblioteca OLED no se ha instalado, en ARDUINO IDE, haga clic en *Sketch* y seleccione *Include*, luego seleccione *Manage Libraries*.



En la barra lateral izquierda, busque la biblioteca **Adafruit SSD1306 by Adafruit**.

Vous pouvez maintenant lancer la compilation du croquis en cliquant sur *Sketch* et en sélectionnant *Upload*.

Ahora puede comenzar a compilar el boceto haciendo clic en *Sketch* y seleccionando *Upload*.

Una vez que el boceto esté cargado en la placa ESP32, inicie la terminal TeraTerm. Compruebe que el vocabulario **oled** esté presente:

```
oled vlist   \ display:
OledInit SSD1306_SWITCHCAPVCC SSD1306_EXTERNALVCC WHITE BLACK
OledReset
HEIGHT WIDTH OledAddr OledNew OledDelete OledBegin OledHOME OledCLS
OledTextc
OledPrintln OledNumln OledNum OledDisplay OledPrint OledInvert
OledTextsize
OledSetCursor OledPixel OledDrawL OledCirc OledCircF OledRect
OledRectF
OledRectR OledRectRF oled-builtins
```

Recursos

En inglés

- **ESP32forth** Página mantenida por Brad NELSON, el creador de ESP32forth. Allí encontrarás todas las versiones (ESP32, Windows, Web, Linux...)
  https://esp32forth.appspot.com/ESP32forth.html
-

En francés

- **ESP32 Forth** sitio en dos idiomas (francés, inglés) con muchos ejemplos
  https://esp32.arduino-forth.com/

GitHub

- **Ueforth** Recursos mantenidos por Brad NELSON. Contiene todos los archivos fuente en lenguaje Forth y C para ESP32forth
  https://github.com/flagxor/ueforth
- **ESP32forth** códigos fuente y documentación para ESP32 en adelante. Recursos mantenidos por Marc PETREMANN
  https://github.com/MPETREMANN11/ESP32forth
- **ESP32forthStation** recursos mantenidos por Ulrich HOFFMAN. Computadora Forth independiente con computadora de placa única LillyGo TTGO VGA32 y ESP32forth
  https://github.com/uho/ESP32forthStation
- **ESP32Forth** recursos mantenidos por F. J. RUSSO
  https://github.com/FJRusso53/ESP32Forth
- **esp32forth-addons** recursos mantenidos por Peter FORTH
  https://github.com/PeterForth/esp32forth-addons
- **Esp32forth-org** Repositorio de código para miembros de los grupos Forth2020 y ESp32forth
  https://github.com/Esp32forth-org
-

índice léxico