

Desarrollos de dispositivos IoT basado en el módulo ESP32

Curso Octubre 2020

Instructor:

Ing. Gabriel Jesús Pool Balam

Contenido

Introducción.....	4
1. Instalación del IDE de Arduino	5
1.1 Instalación del IDE del Arduino v1.8.13.....	6
1.2 Instalación del plugin ESP32 para IDE del Arduino v1.8.13.....	13
1.3 Ejecutando el ejemplo “blink”.....	19
1.4 Usando la ayuda del IDE del Arduino	29
1.5 Usando el monitor del puerto serie del IDE del Arduino	34
1.6 Agregando bibliotecas al IDE del Arduino del catálogo de drivers.....	39
2. Implementando códigos en IDE de Arduino.....	49
2.1 El Hardware del módulo NodeMCU-32S	50
2.2 Análisis de ejemplos considerados relevantes para la implementación de un lector de temperatura.....	55
2.2.1 Blink.ino.....	56
2.2.2 BlinkWithOutDelay.ino	62
2.2.3 Máquina de estados.....	65
2.2.4 Usando la pantalla OLED SSD1306.....	69
2.2.5 Usando el sensor de temperatura DS18B20 de fabricado por dallas semiconductor.....	74
2.2.6 Usando el sensor de temperatura DHT22 (AM2301) fabricado por AMLOGIC	79
2.2.7 Mostrando los valores del sensor en la pantalla del SSD1306	84
2.2.8 Usando el módulo Analog to Digital Converter (ADC).....	88
3. Diseño de un WebServer en NodeMCU-32S.....	93
3.1 El Servidor Web implementado en NodeMCU-32S	94
3.1.1 MDNS	94

3.1.2	Advanced Web Server.....	101
3.1.3	Paso de argumentos al servidor WEB	115
3.2	El Servidor Web con página dinámica en NodeMCU-32S.....	121
3.2.1	ESP32 Filesystem Uploader Plugin.....	121
3.2.2	SPIFFS WebServer.....	125



Centro de Investigación Científica de Yucatán A. C.

Departamento de Instrumentación

www.cicy.mx

Introducción

Este manual se enfoca a reunir la información dispersa de internet en un solo lugar con el fin de poder explotar todo el potencial del módulo ESP32 en sus diferentes versiones. También se anexan funciones adicionales que se consideran importantes para el desarrollo de una solución. Para seguir los desarrollos de este manual, se requiere de una conexión a Internet, conocimientos básicos de programación y dominar el uso de una PC.

Modulo I

1. Instalación del IDE de Arduino

Objetivo General: Descargará desde la página web, instalará y configurará el software del IDE del Arduino en su computadora, para desarrollar con el módulo ESP32.

1.1 Instalación del IDE del Arduino v1.8.13

Objetivo específico: Instalará desde el sitio web el IDE del Arduino en su computadora con las opciones del IDE que trae configuradas de manera predeterminada.

Para esta instalación se contempla que usted tiene instalado **Windows 10** en su computadora y con los **parches de Windows** actualizados al día.

Con el explorador de internet de su preferencia, escriba el siguiente link en la barra de búsqueda del navegador: <https://www.arduino.cc/en/Main/Software> presione la tecla <enter> y espere a que el navegador cargue la página web. Una vez cargada la página web, ubique la siguiente sección en la página web:

Download the Arduino IDE



The screenshot shows the Arduino IDE download page. On the left, there's a large teal circle with a white infinity symbol containing a minus and plus sign. To its right, the text "ARDUINO 1.8.13" is displayed. Below this, a paragraph describes the software as open-source and compatible with Windows, Mac OS X, and Linux. It includes a link to the "Getting Started" page. On the right side, there's a teal sidebar with various download links. A red arrow points to the first link in this sidebar: "Windows Installer, for Windows 7 and up".

Windows Installer, for Windows 7 and up
Windows ZIP file for non admin install

Windows app Requires Win 8.1 or 10
[Get](#)

Mac OS X 10.10 or newer

Linux 32 bits
Linux 64 bits
Linux ARM 32 bits
Linux ARM 64 bits

[Release Notes](#)
[Source Code](#)
[Checksums \(sha512\)](#)

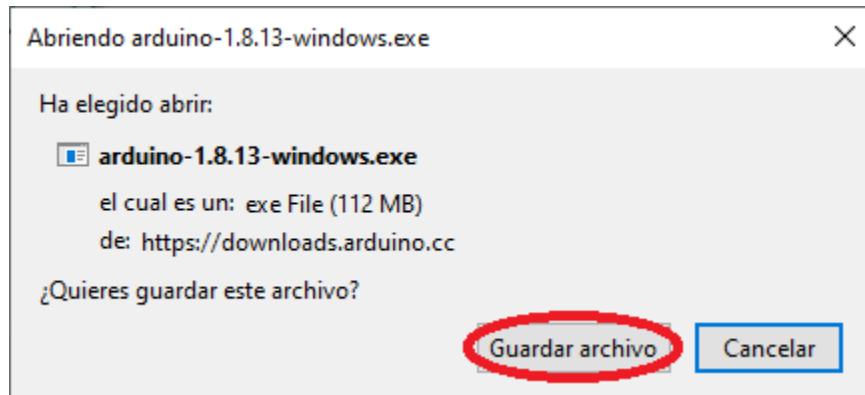
...déle un click con el botón izquierdo del mouse en el texto marcado por la flecha que se muestra en la figura anterior y espere a que la siguiente página web se cargue, a continuación, se muestra lo siguiente:

Contribute to the Arduino Software

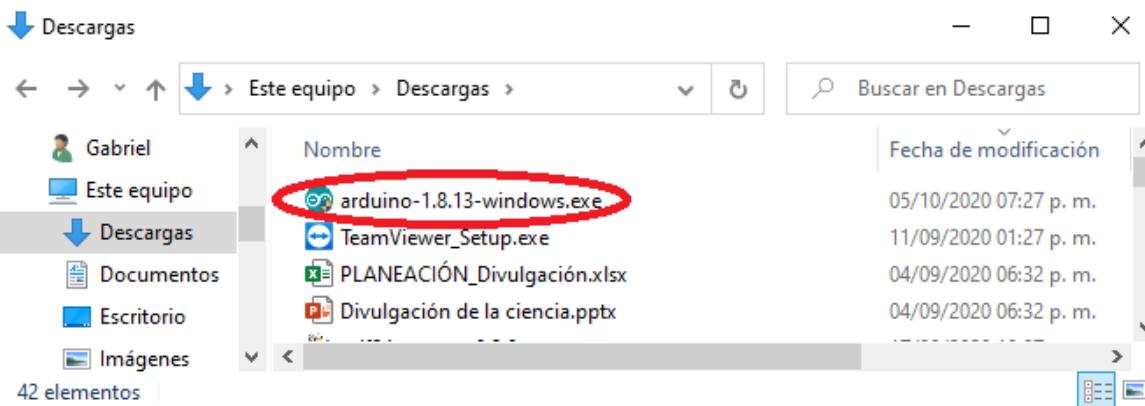
Consider supporting the Arduino Software by contributing to its development. (US tax payers, please note this contribution is not tax deductible). [Learn more on how your contribution will be used.](#)



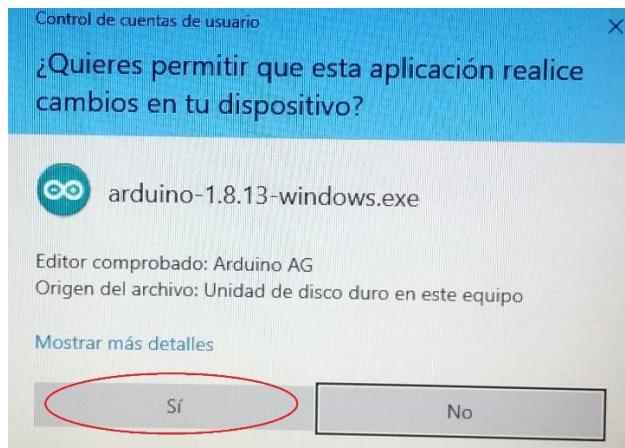
...dé un click con el botón izquierdo del mouse en el texto marcado como “just download” e inmediatamente saldrá una ventana emergente como ésta:



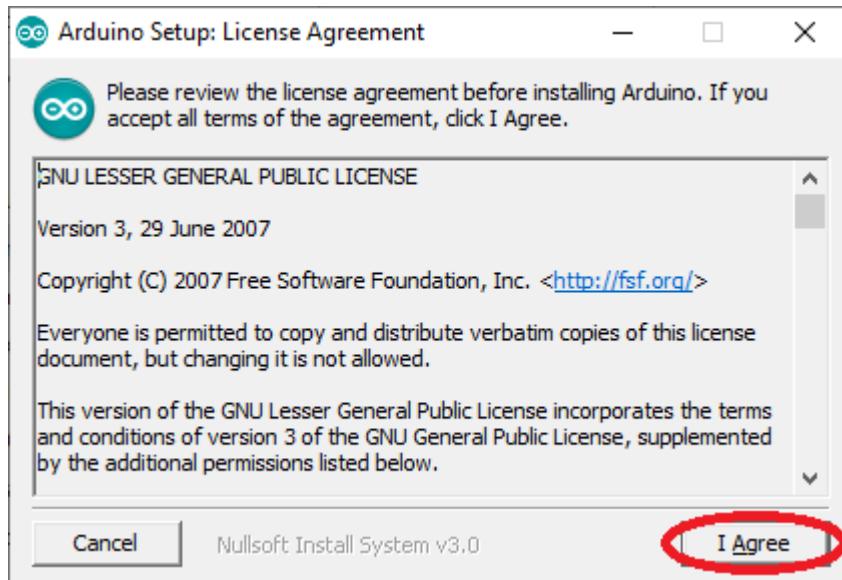
... presione el botón con el texto “Guardar archivo” y la descarga iniciará, cuando haya concluido, su navegador le avisará que la descarga ya ha terminado. Ubique el archivo descargado en la carpeta de descargas (típicamente los archivos se guardan en esta carpeta a menos que usted haya modificado la ruta de descarga). Se mostrará como sigue:



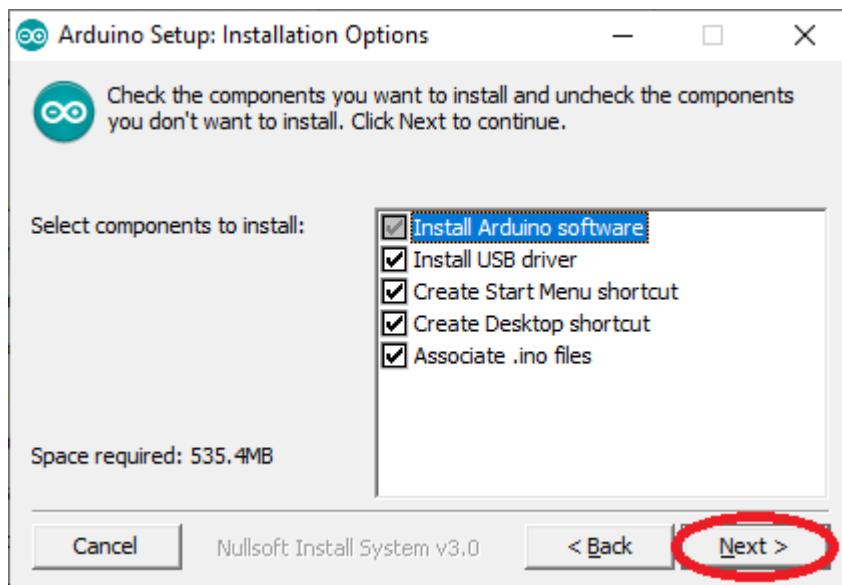
... presione doble click con el botón izquierdo del mouse al archivo recién descargado (llamado Arduino-1.8.13-windows.exe) e inmediatamente la pantalla se pondrá negra con una ventana como se muestra a continuación:



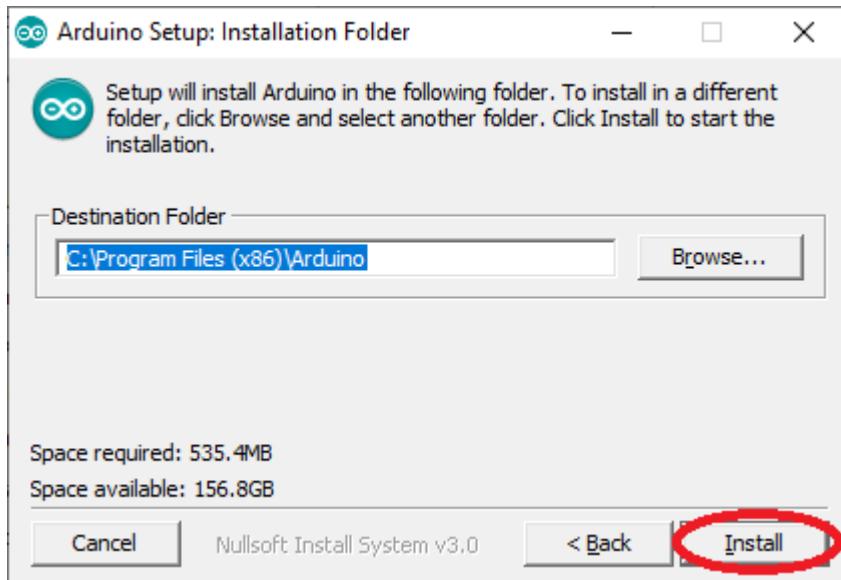
... presione el botón con el texto "si" y la instalación se ejecutará de manera automática, se mostrará una ventana como se muestra a continuación:



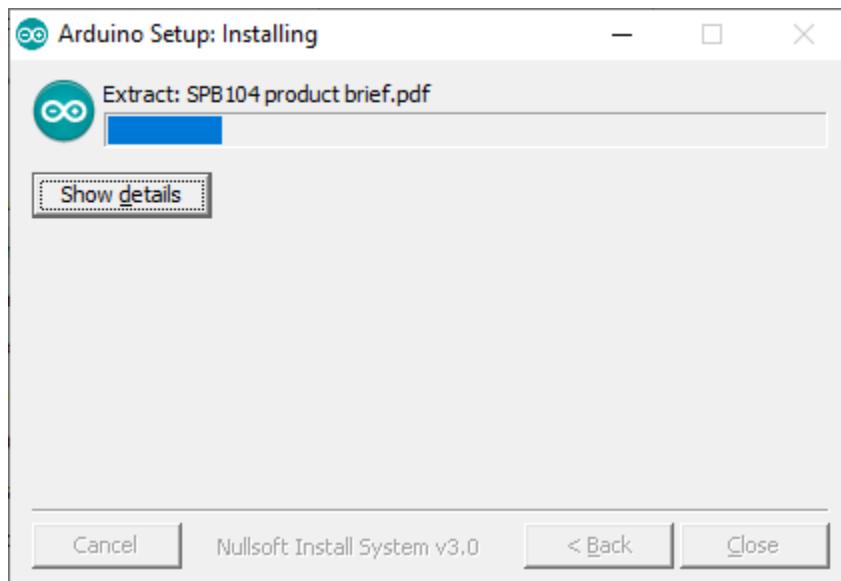
Presione el botón marcado con el texto “I Agree” y se mostrará lo siguiente:



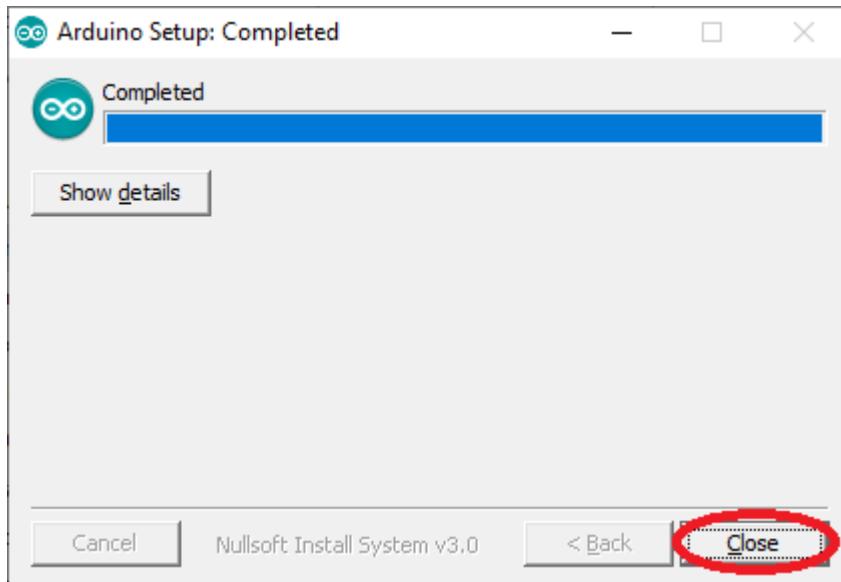
Presione el botón marcado con el texto “Next” y se mostrará lo siguiente:



Presione el botón marcado con el texto “Install” y se mostrará lo siguiente:



Espere a que la barra de progreso (de color azul) se llene y se mostrará lo siguiente:



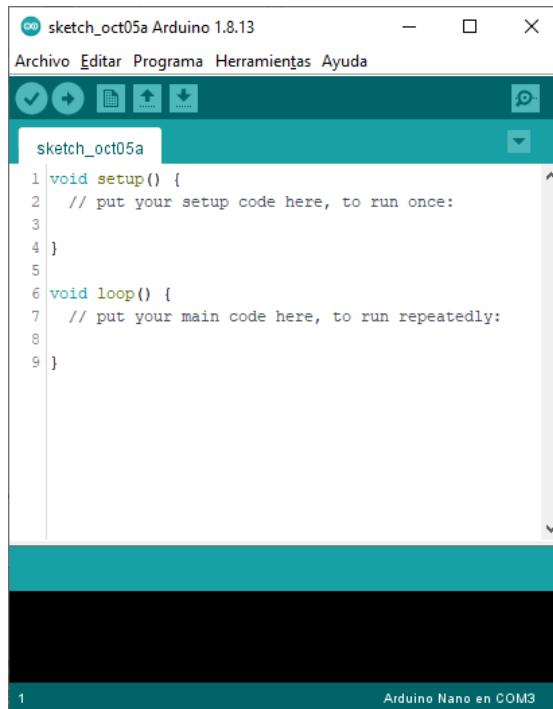
Presione el botón con el texto “Close” y la ventana anterior se cerrará. Para abrir el programa recién instalado basta con hacer doble click con el botón izquierdo del mouse al ícono ubicado en su escritorio:



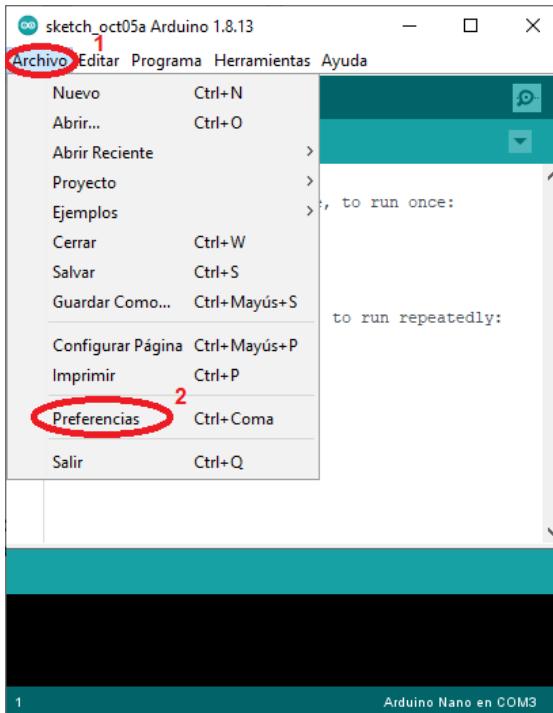
Al hacer doble click con el botón izquierdo del mouse en el ícono anterior, se abrirá como se muestra a continuación:



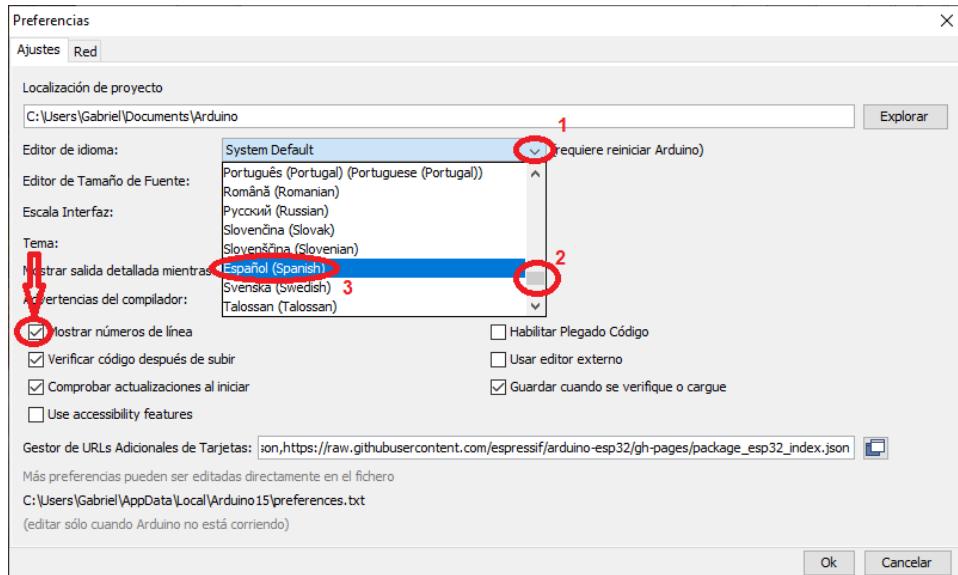
Seguidamente, se mostrará la ventana del programa Arduino:



Si por alguna razón, se muestra el menú del programa en otro idioma diferente al suyo, puede modificarlo como sigue, presione el botón izquierdo del mouse en la parte superior marcada como “Archivo” y seguidamente seleccione “Preferencias”:



Se abrirá una ventana como se muestra a continuación:

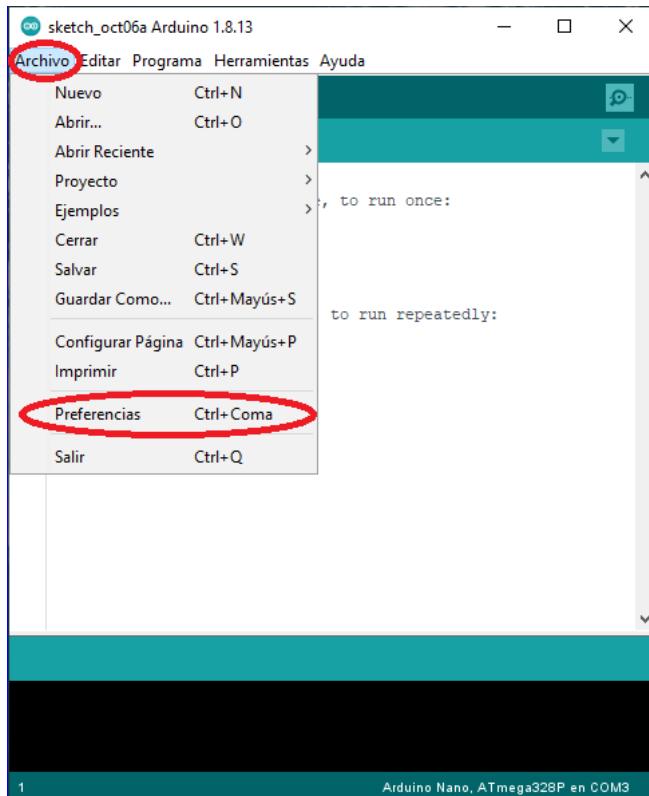


Ubique la flecha en el campo llamado “Editor de Idioma” y presione el botón izquierdo del mouse y se abrirá un menú, deslice el menú hacia abajo hasta encontrar el idioma de su preferencia, posteriormente, presione el botón izquierdo del mouse en el nombre del idioma que deseé elegir. Seguidamente, **active el casillero con la etiqueta “Mostrar números de página”**. Por último, presione el botón con el texto “Ok”. Para que los cambios surtan efecto, cierre el programa y vuélvalo a abrir.

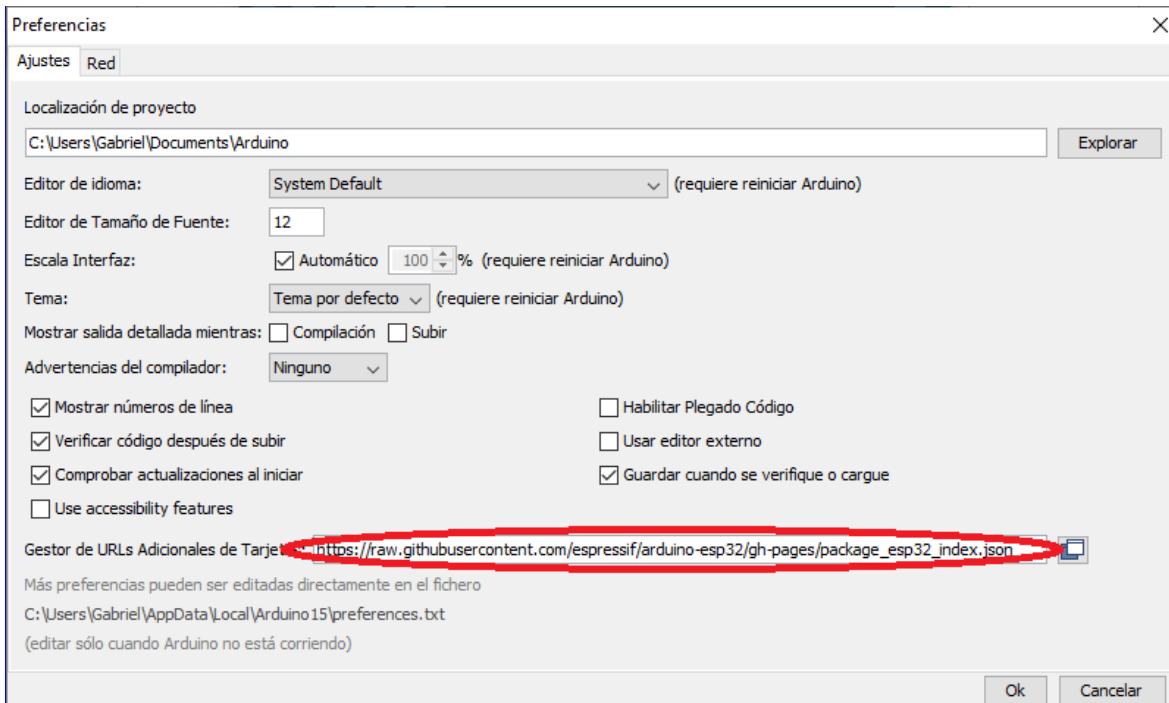
1.2 Instalación del plugin ESP32 para IDE del Arduino v1.8.13

Objetivo específico: Configurar el IDE del Arduino para trabajar con diversos modelos de módulos basados en el chip ESP32

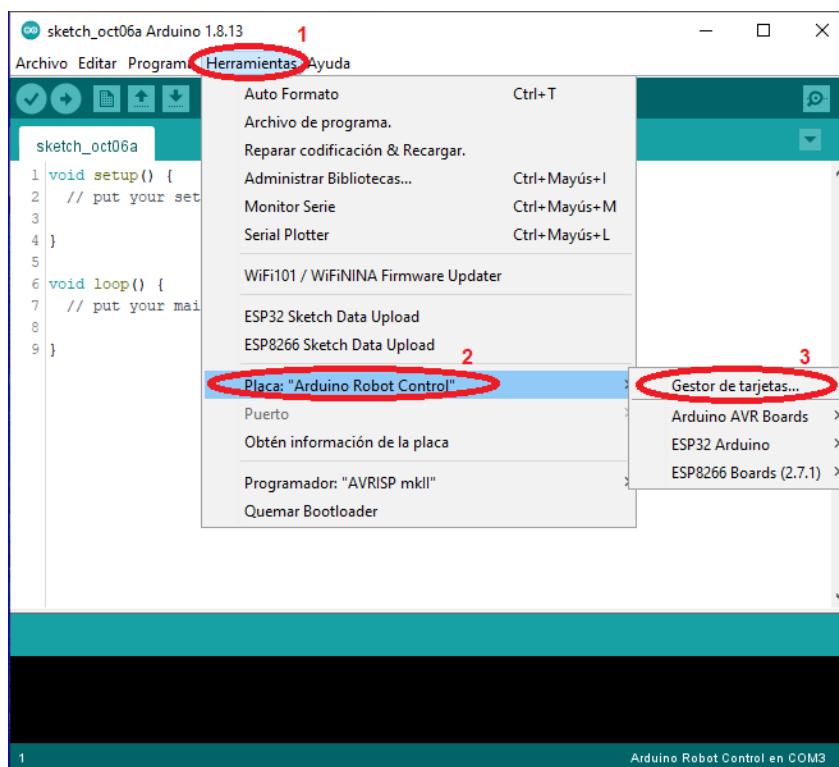
Una vez instalado el IDE del Arduino se configurará para habilitar las opciones de trabajo con los modelos de módulos EPS32. En el link: <https://github.com/espressif/arduino-esp32> se encuentran las instrucciones necesarias para cubrir este paso. Según lo anterior, abrir el IDE del Arduino, ubique la ventana de preferencias como sigue:



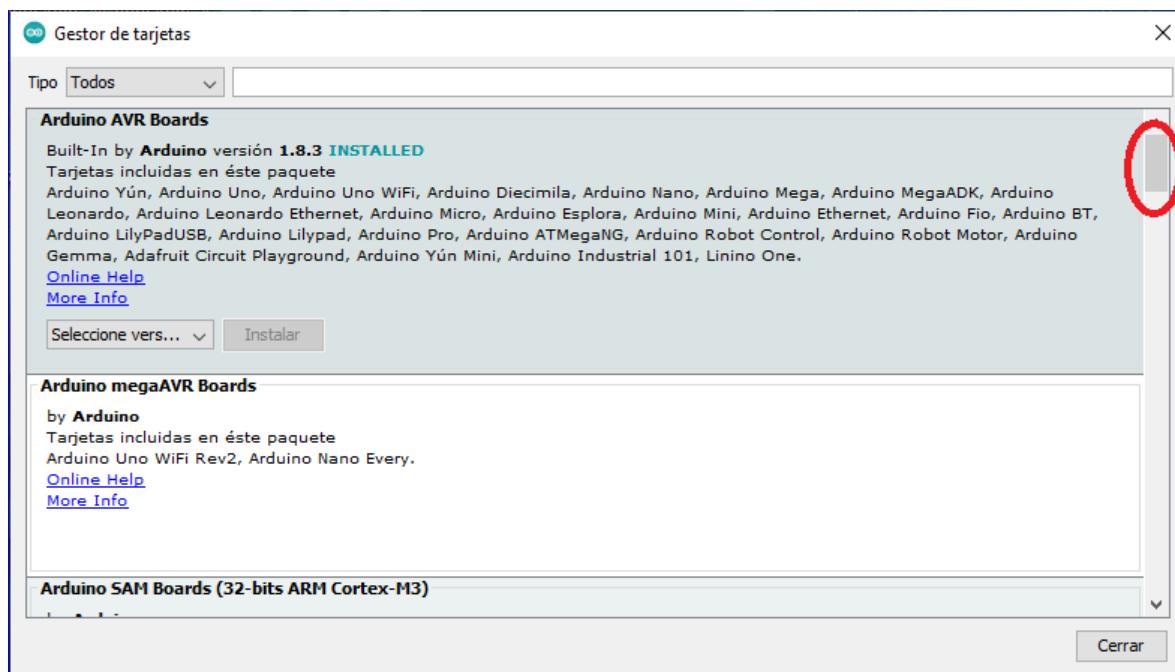
A continuación introduzca el siguiente link https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json en el cuadro de texto llamado “Gestor de URLs adicionales de tarjetas” como se muestra a continuación:



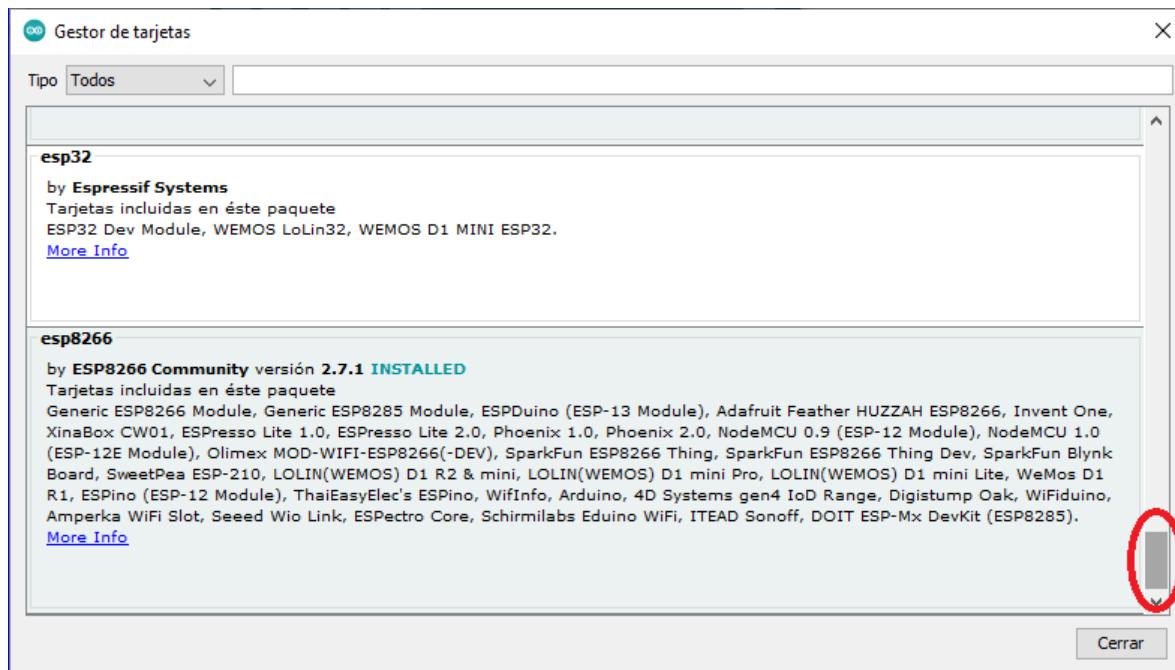
Luego, presione el botón “OK” y la ventana se cerrará. Abra la ventana del gestor de tarjetas como sigue:



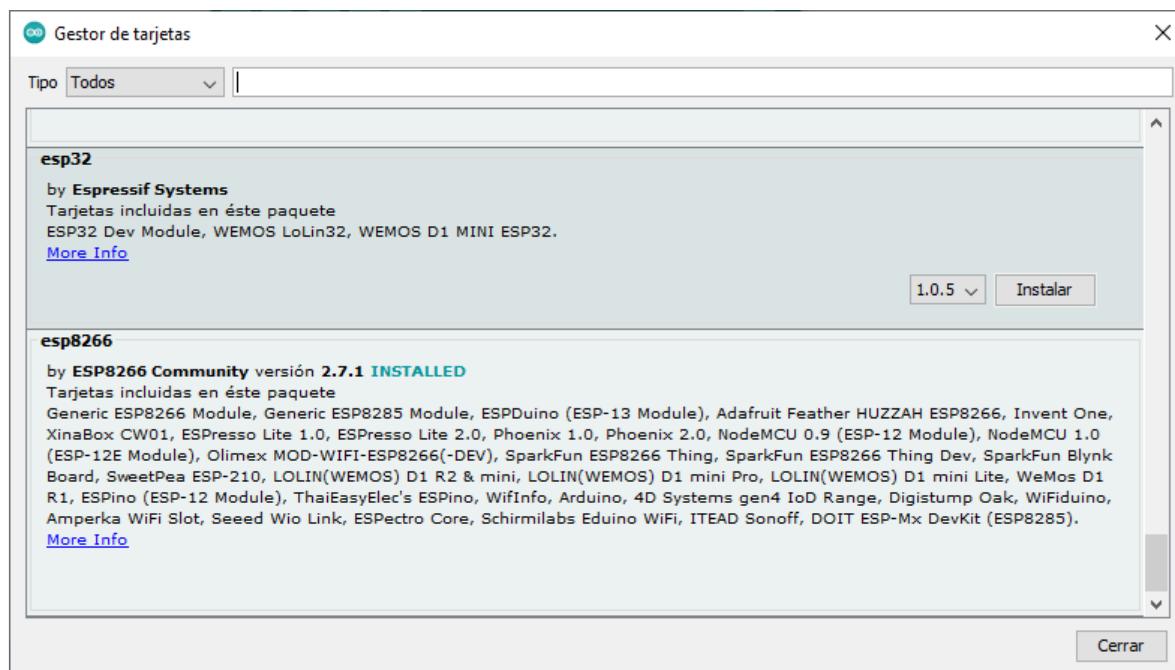
Al hacer click con el botón del mouse en la parte mostrada en la figura anterior, se abrirá la siguiente ventana:



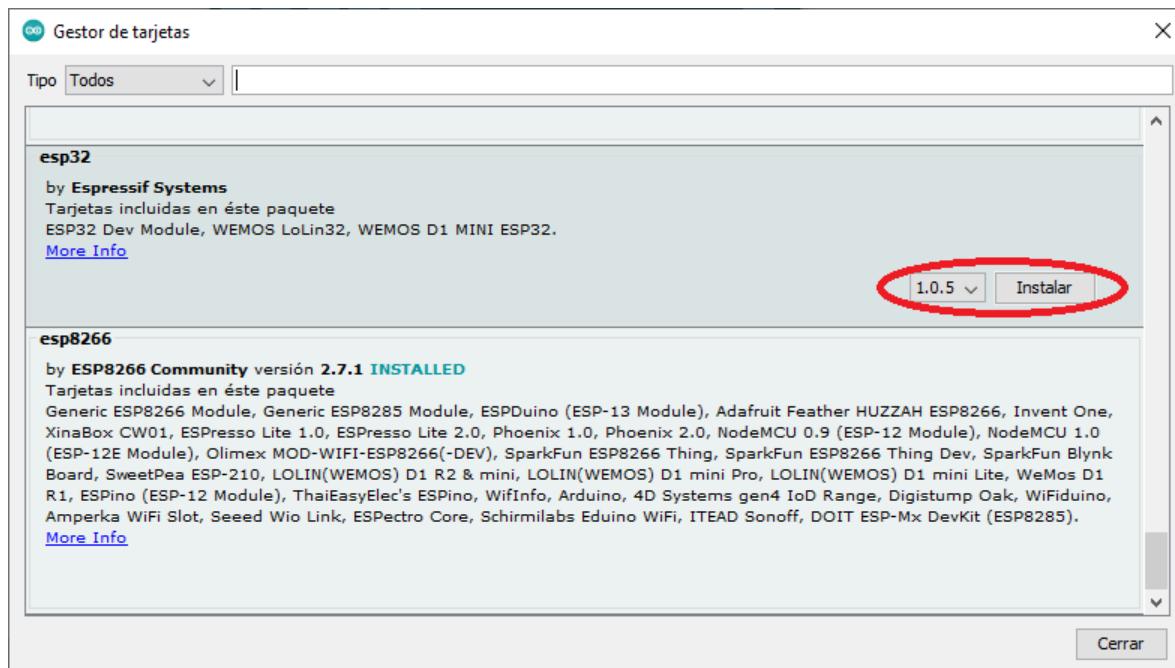
Deslice el control indicado en la figura anterior hasta el final, así como se muestra a continuación:



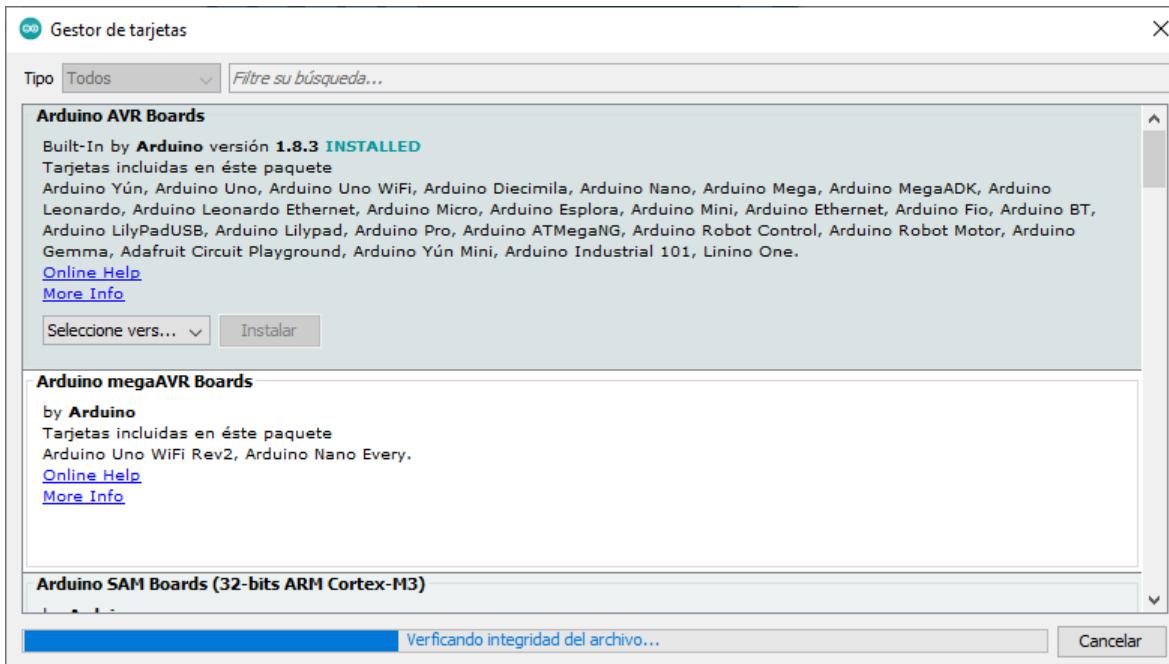
Ponga el mouse sobre el campo con el texto esp32, el área se pondrá en gris y se activarán dos botones como se muestra a continuación:



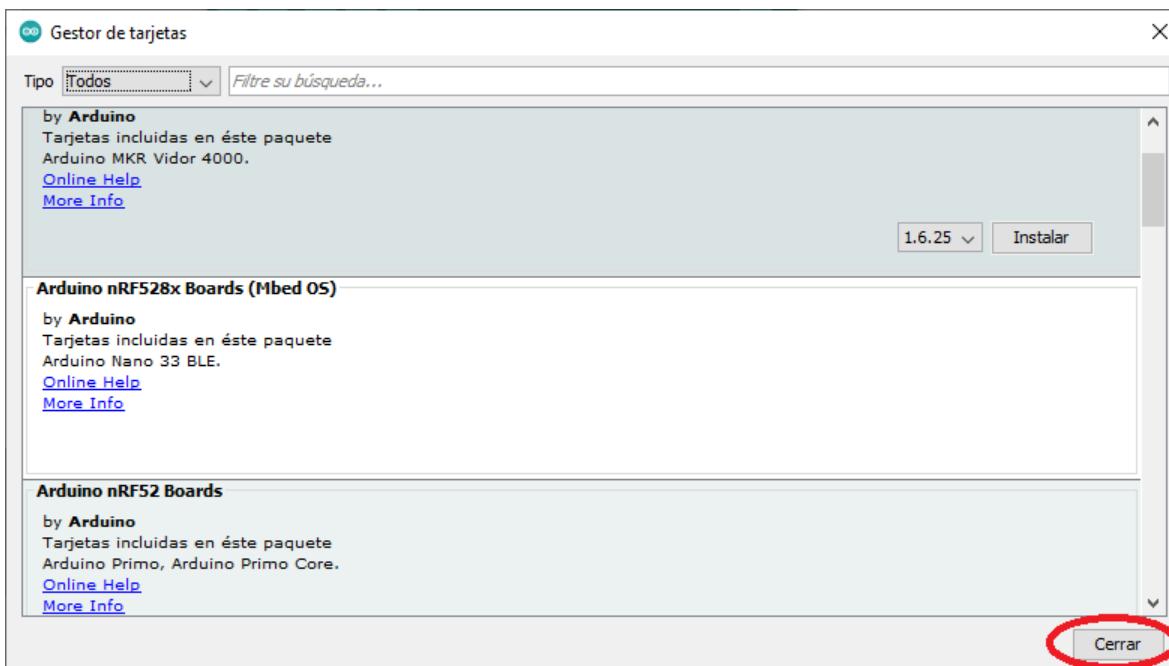
Seleccione la última versión (1.0.5) y presione el botón “Instalar” e iniciará la instalación de la versión seleccionada:



Espere a que la barra de progreso termine:



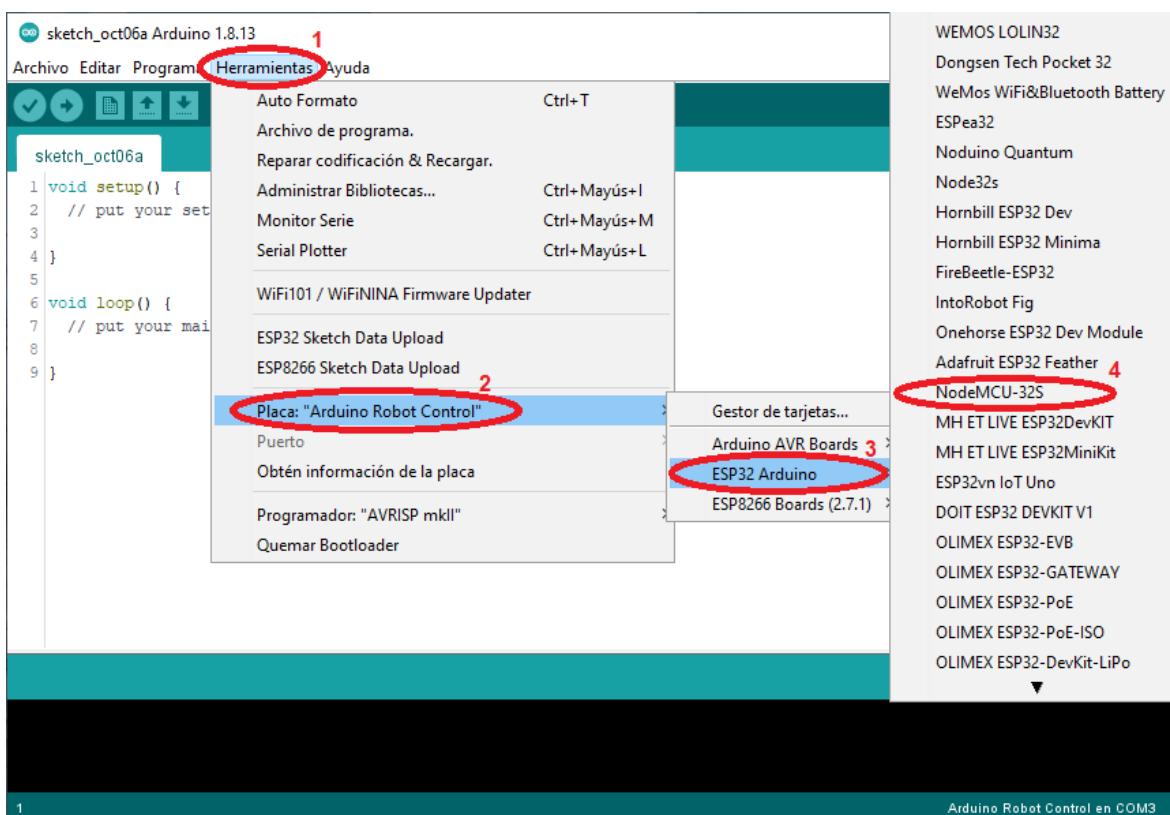
Cuando se termine de actualizar presione el botón de cerrar.



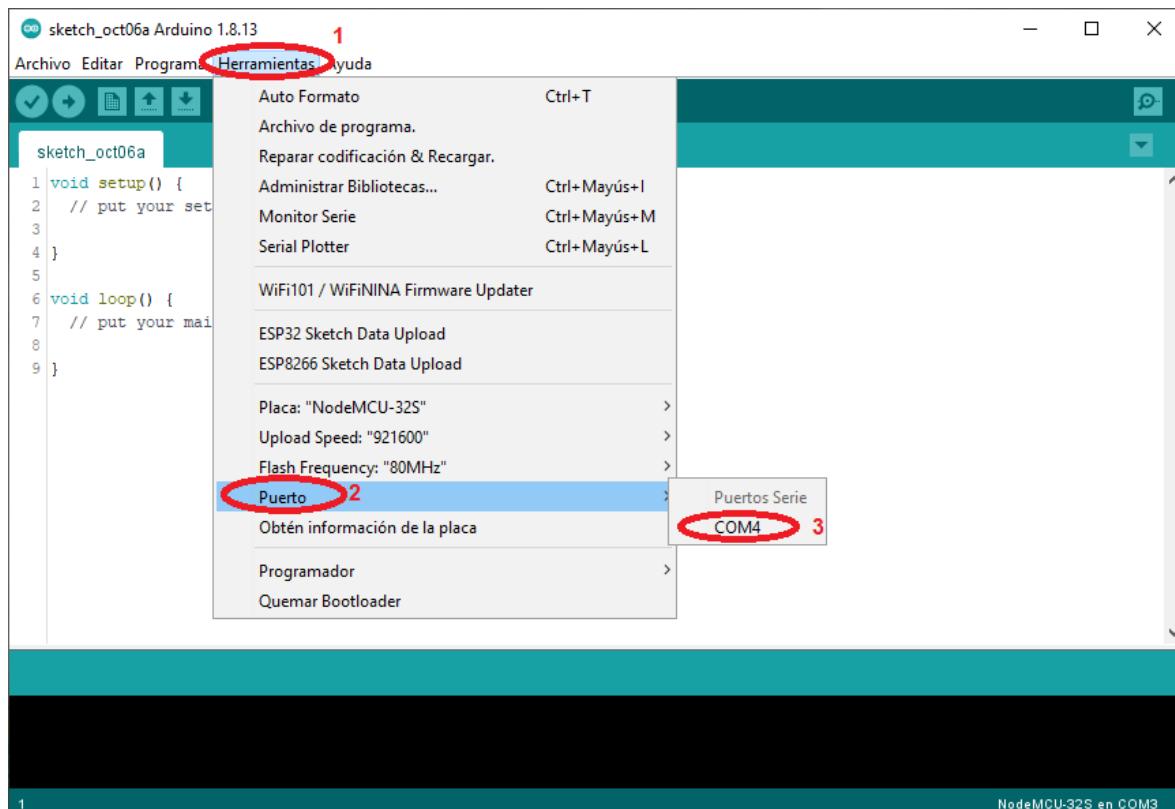
1.3 Ejecutando el ejemplo “blink”

Objetivo específico: Elegir la tarjeta de desarrollo a utilizar en el IDE del Arduino, cargar, verificar (compilar) y descargar en la tarjeta de desarrollo un código de la lista de ejemplos que nos ofrece el desarrollador.

El **PRIMER PASO** es elegir la tarjeta de desarrollo a utilizar:



Como **SEGUNDO PASO**, se requiere establecer el puerto donde se tiene conectado el dispositivo, **conecte su tarjeta** de desarrollo y verifique el número de puerto que el sistema operativo le asigna:



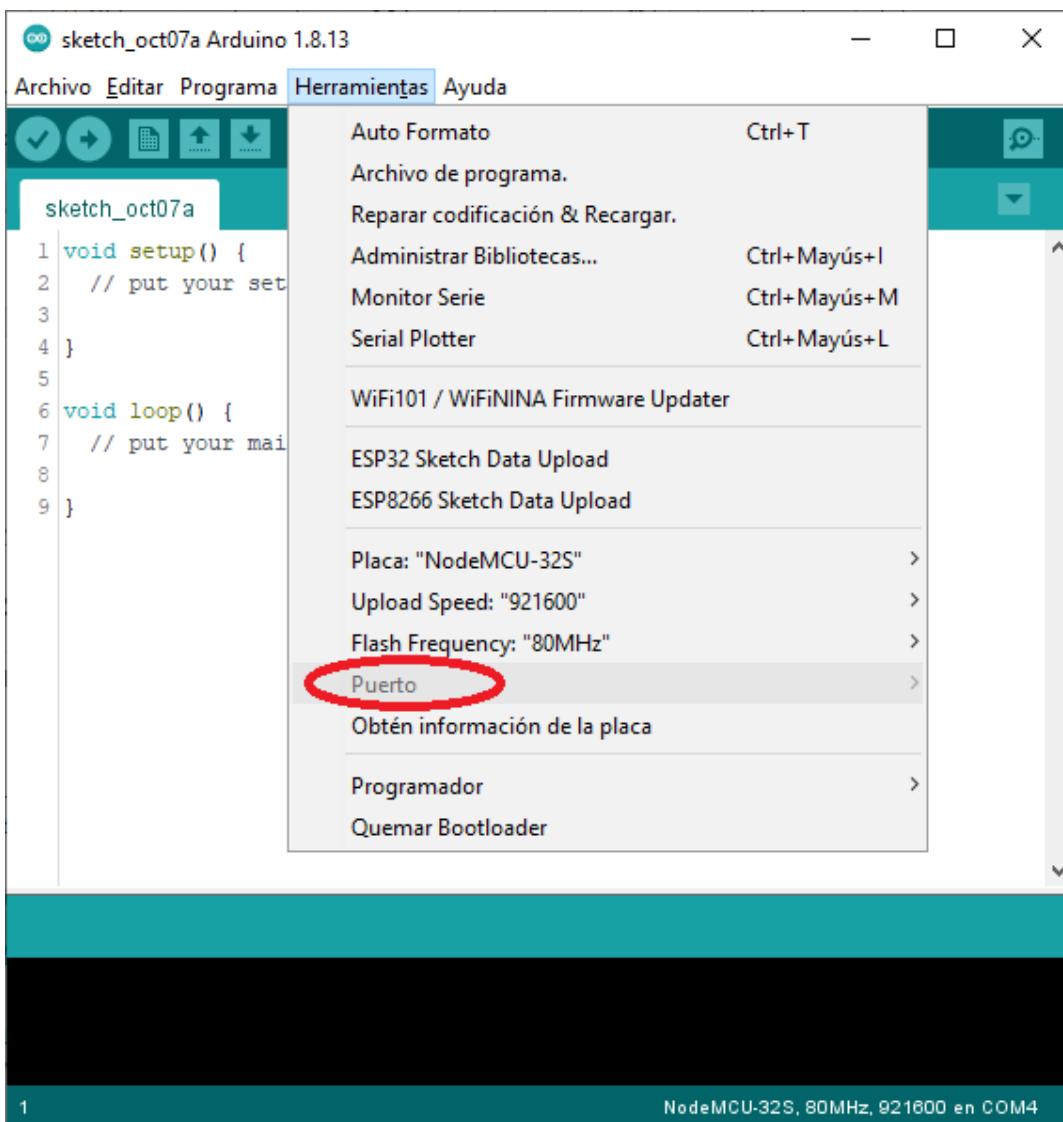
IMPORTANTE:

Antes de conectar su dispositivo, verifique que no exista otros puertos ya reconocidos por el PC, si los hay, solamente tome en cuenta que, el puerto de su dispositivo será el puerto nuevo que el sistema operativo nos muestre al conectar la tarjeta de desarrollo.

No siempre se muestra el mismo número de puerto para todas las computadoras.

Para windows 8 y 10, se requiere conectar el dispositivo y tener activado las actualizaciones de windows. Apenas windows detecte el nuevo dispositivo, éste lo INSTALARÁ AUTOMATICAMENTE. En éstas dos plataformas NO instale el driver manualmente (de la manera tradicional), ya que al momento de usar el dispositivo windows sacará la clásica pantalla azul (crashea).

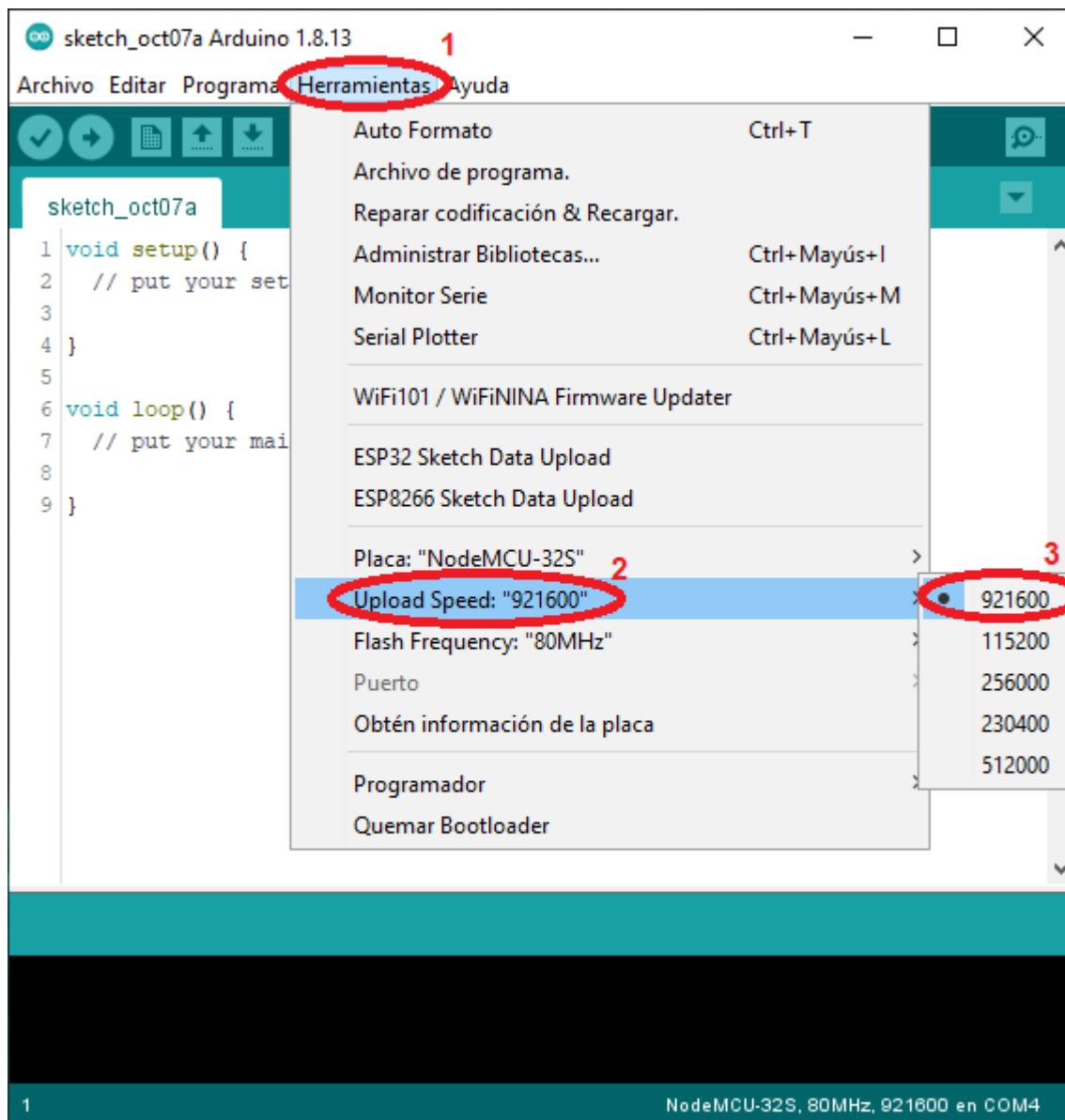
Si el puerto se muestra en gris como se indica en la siguiente figura:



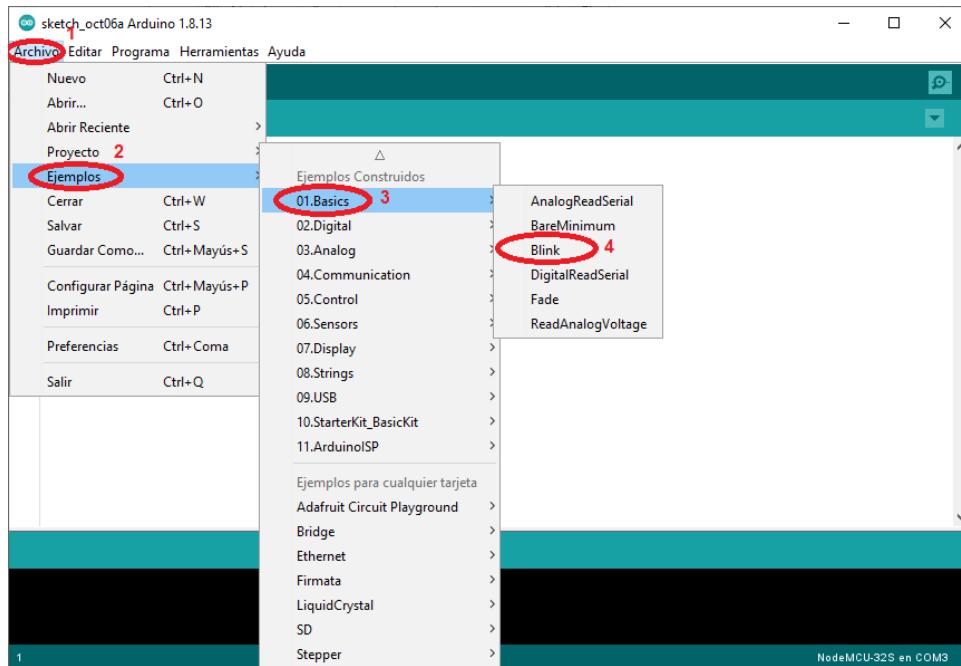
... indica que el módulo ESP32 no fue reconocido. Esto puede deberse a que:

1. La computadora no tiene acceso a Internet y no puede descargar los controladores para el dispositivo.
2. El cable USB no está firmemente conectado al módulo ESP32 ó al puerto USB de la computadora.
3. El cable USB podría estar dañado ó no es el apropiado (algunos cables USB sólo son para cargar dispositivos, no son para transferir datos).
4. El módulo ESP32 podría estar dañado (mantenga su módulo en su empaque metalizado cuando no se utilice, ya que es sensible a la electricidad estática).
5. El puerto USB de la computadora podría estar dañado.

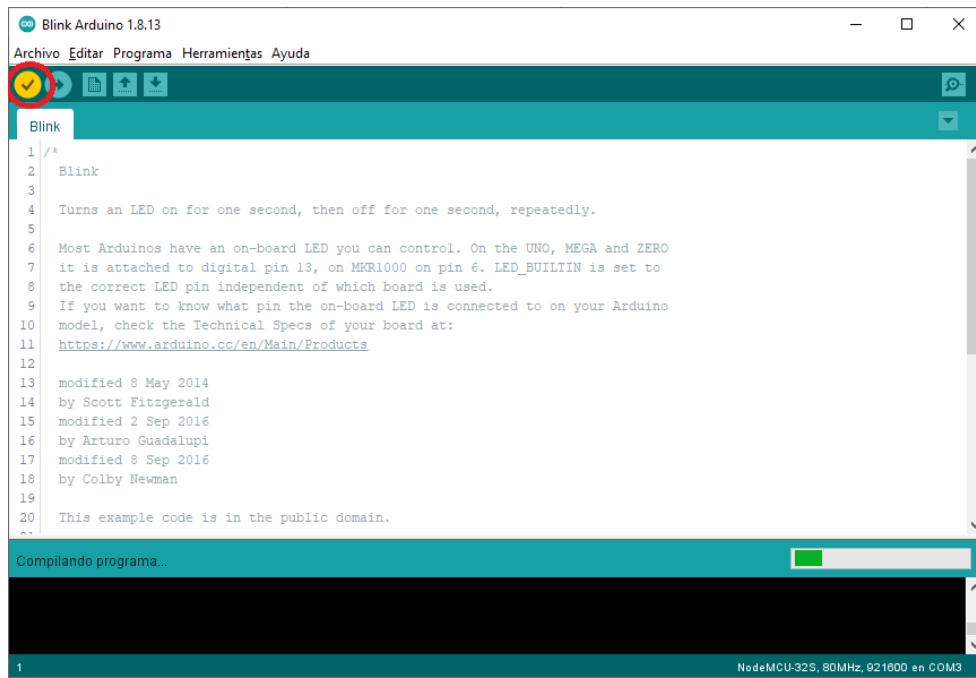
En el **TERCER PASO**, se requiere configurar la velocidad de descarga del código generado durante la compilación por el IDE del Arduino. Para eso, presionemos en la secuencia que coincide la siguiente figura:



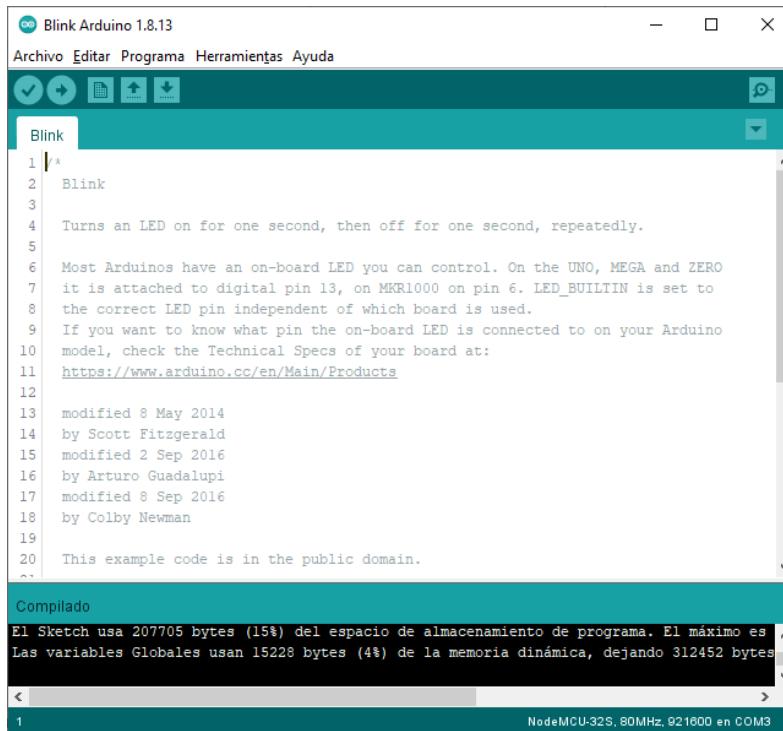
En el **CUARTO PASO**, se requiere escoger el ejemplo siguiente:



Como **QUINTO PASO** se verifica el programa presionando el botón  e indicará si hay errores:



... espere que la barra de progreso termine y nos deberá de mostrar lo siguiente:

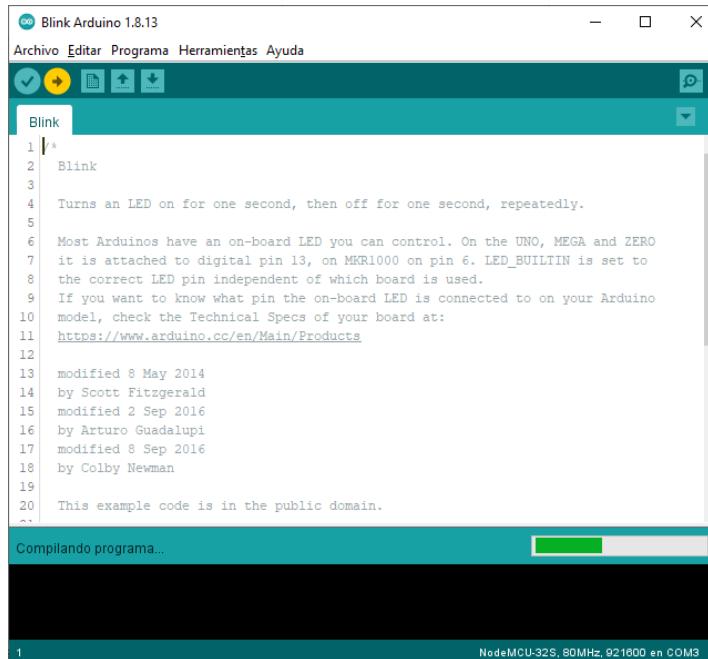


```

Blink Arduino 1.8.13
Archivo Editar Programa Herramientas Ayuda
Blink
1 /*
2  *  Blink
3
4  Turns an LED on for one second, then off for one second, repeatedly.
5
6  Most Arduinos have an on-board LED you can control. On the UNO, MEGA and ZERO
7  it is attached to digital pin 13, on MKR1000 on pin 6. LED_BUILTIN is set to
8  the correct LED pin independent of which board is used.
9  If you want to know what pin the on-board LED is connected to on your Arduino
10 model, check the Technical Specs of your board at:
11 https://www.arduino.cc/en/Main/Products
12
13 modified 8 May 2014
14 by Scott Fitzgerald
15 modified 2 Sep 2016
16 by Arturo Guadalupi
17 modified 8 Sep 2016
18 by Colby Newman
19
20 This example code is in the public domain.
-->
Compilado
El Sketch usa 207705 bytes (15%) del espacio de almacenamiento de programa. El máximo es 322560 bytes
Las variables Globales usan 15228 bytes (4%) de la memoria dinámica, dejando 312452 bytes
-->
1 NodeMCU-32S, 80MHz, 921600 en COM3

```

Como **SEXTO PASO**, se verifica y descarga el programa en su tarjeta de desarrollo presionando el botón  . Al presionar el botón se mostrará lo siguiente:

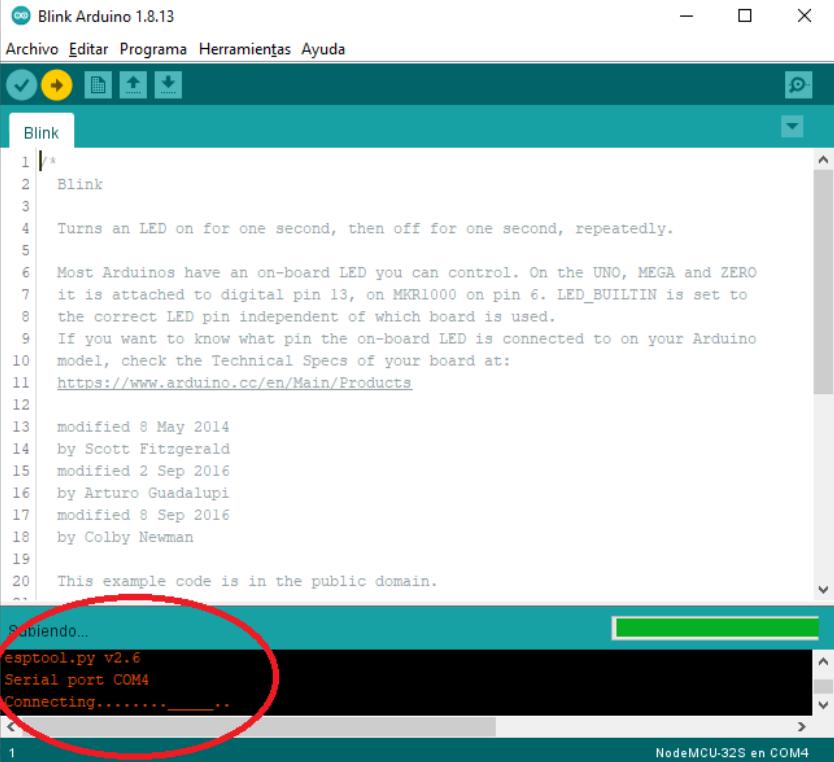


```

Blink Arduino 1.8.13
Archivo Editar Programa Herramientas Ayuda
Blink
1 /*
2  *  Blink
3
4  Turns an LED on for one second, then off for one second, repeatedly.
5
6  Most Arduinos have an on-board LED you can control. On the UNO, MEGA and ZERO
7  it is attached to digital pin 13, on MKR1000 on pin 6. LED_BUILTIN is set to
8  the correct LED pin independent of which board is used.
9  If you want to know what pin the on-board LED is connected to on your Arduino
10 model, check the Technical Specs of your board at:
11 https://www.arduino.cc/en/Main/Products
12
13 modified 8 May 2014
14 by Scott Fitzgerald
15 modified 2 Sep 2016
16 by Arturo Guadalupi
17 modified 8 Sep 2016
18 by Colby Newman
19
20 This example code is in the public domain.
-->
Compilando programa...
-->
1 NodeMCU-32S, 80MHz, 921600 en COM3

```

... espere a que la barra de progreso se llene, en algunos modelos de NodeMCU-32S no se requiere acción adicional para que el código empiece a descargarse, así que, no se mostrará la solicitud de conexión. De lo contrario, si se muestra lo siguiente:



The screenshot shows the Arduino IDE interface. The top menu bar includes Archivo, Editar, Programa, Herramientas, and Ayuda. Below the menu is a toolbar with icons for upload, download, and serial communication. The main code editor window displays the 'Blink' sketch, which turns an LED on for one second and off for one second, repeatedly. The serial monitor at the bottom shows the output of the esptool.py script connecting to the NodeMCU-32S module via COM4. A red circle highlights the 'Connecting.....' message in the serial monitor.

```

Blink Arduino 1.8.13
Archivo Editar Programa Herramientas Ayuda
Blink
1 /*
2  * Blink
3  *
4  * Turns an LED on for one second, then off for one second, repeatedly.
5  *
6  * Most Arduinos have an on-board LED you can control. On the UNO, MEGA and ZERO
7  * it is attached to digital pin 13, on MKR1000 on pin 6. LED_BUILTIN is set to
8  * the correct LED pin independent of which board is used.
9  * If you want to know what pin the on-board LED is connected to on your Arduino
10 * model, check the Technical Specs of your board at:
11 https://www.arduino.cc/en/Main/Products
12
13 modified 8 May 2014
14 by Scott Fitzgerald
15 modified 2 Sep 2016
16 by Arturo Guadalupi
17 modified 8 Sep 2016
18 by Colby Newman
19
20 This example code is in the public domain.
```
Sustiendo...
esptool.py v2.6
Serial port COM4
Connecting.....```

```

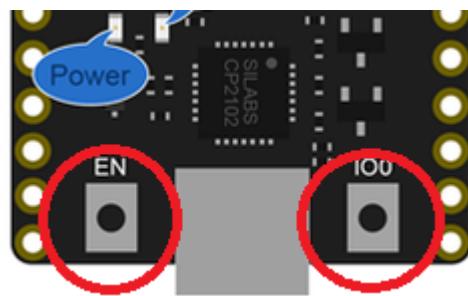
... deberá continuar con el **SEPTIMO PASO**, ya que, dependiendo del hardware adquirido, deberá probar **UNA** de las acciones siguientes:

- Presione momentáneamente el botón IO0 y el código deberá empezar a descargarse a su módulo NodeMCU-32S.

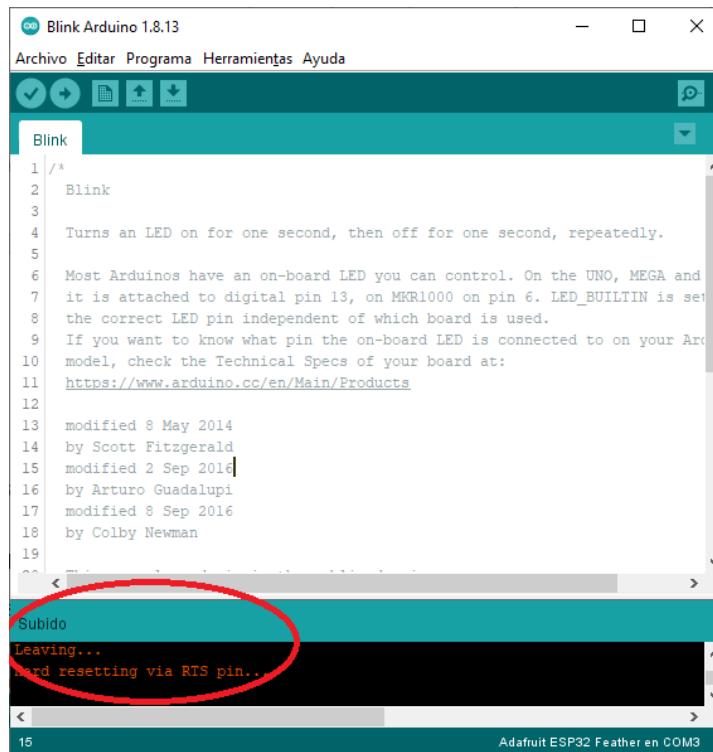


b) Presione y mantenga presionado el botón IO0, seguidamente presionar momentáneamente (presione y libere) el botón de EN ó reset y por último liberar el botón IO0. Siguiendo esto pasos el código deberá empezar a descargarse a su módulo NodeMCU-32S.

Ambos botones están ubicados a los costados del puerto microUSB de su módulo NodeMCU-32S y deberá empezar a descargar el código a su módulo NodeMCU32S.



En ese momento, el código empezará a transferirse a su tarjeta y al finalizar, el código se ejecutará de manera automática y se muestra un mensaje en la parte inferior izquierda del IDE como se muestra a continuación:



```

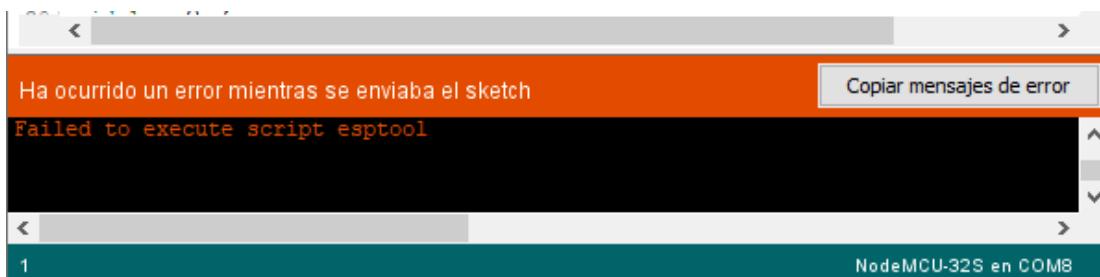
Blink Arduino 1.8.13
Archivo Editar Programa Herramientas Ayuda
Blink
/*
 Blink
 ...
 Turns an LED on for one second, then off for one second, repeatedly.
 ...
 Most Arduinos have an on-board LED you can control. On the UNO, MEGA and
 it is attached to digital pin 13, on MKR1000 on pin 6. LED_BUILTIN is set
 the correct LED pin independent of which board is used.
 If you want to know what pin the on-board LED is connected to on your Ar
 model, check the Technical Specs of your board at:
 https://www.arduino.cc/en/Main/Products
 ...
 modified 8 May 2014
 by Scott Fitzgerald
 modified 2 Sep 2014
 by Arturo Guadalupi
 modified 8 Sep 2016
 by Colby Newman
 ...
Subido
Leaving...
Board resetting via RTS pin...

```

## Errores en la descarga del código

### Falla No. 1

Si en lugar de empezar a descargar el código hacia la tarjeta NodeMCU, le presenta este error:

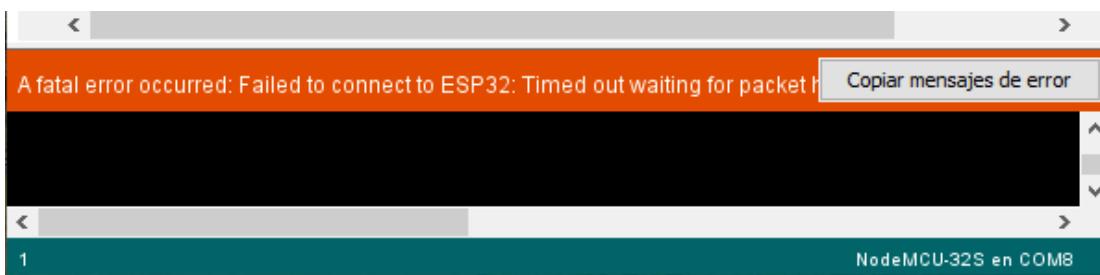


...verifique lo siguiente:

- El módulo NodeMCU **NO** se encuentra conectada a su computadora. Conecte el módulo a su PC siguiendo el procedimiento indicado en el segundo paso.
- El número de puerto COM no corresponde al módulo NodeMCU, este problema se presenta generalmente donde la computadora muestra dos o más puertos instalados. Para averiguar cuál es el puerto correcto, desconecte su módulo NodeMCU de la PC y observe cual es el número de puerto que desaparece, para eso siga la secuencia descrita en el segundo paso ya que, si deja el menú de "Herramientas" activo **NO SE MOSTRARÁ** cambio alguno si desconecta o conecta su módulo NodeMCU.
- La computadora **NO** reconoce ningún puerto COM instalado. En este caso, verifique procedimiento descrito en el segundo paso.

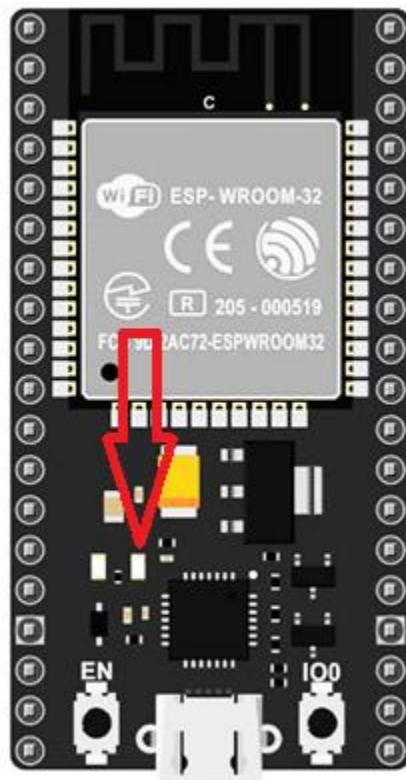
### Falla No. 2

Otro error típico es el siguiente:



Esto indica que no siguió el procedimiento descrito en el séptimo paso. Siga el procedimiento descrito en el séptimo paso.

Al terminar la descarga del código anterior en el módulo NodeMCU, observe con atención su módulo. Si todo salió bien, debería de ver un led parpadeando continuamente.

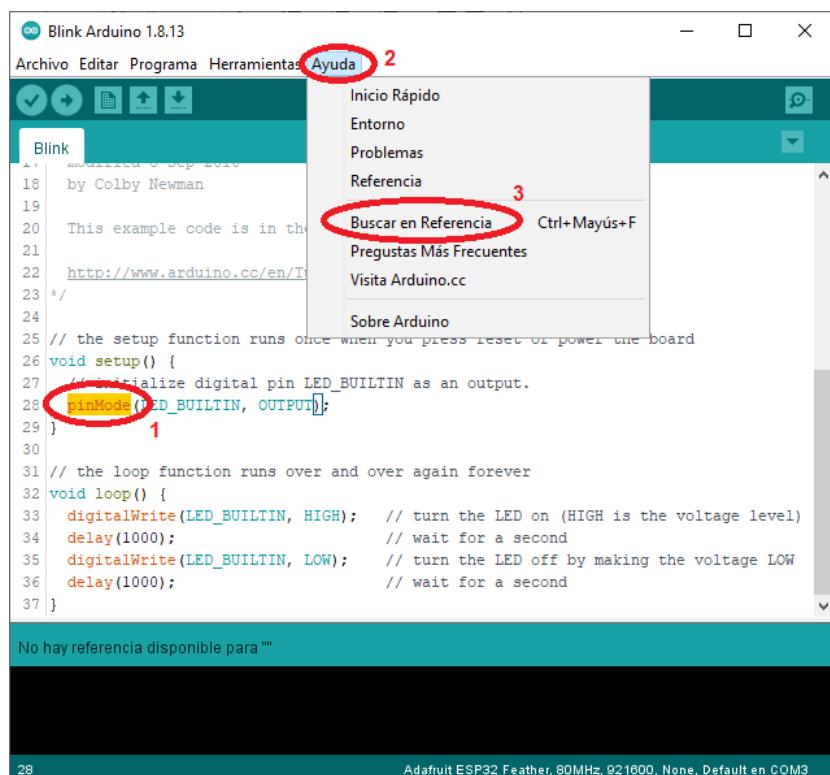


## 1.4 Usando la ayuda del IDE del Arduino

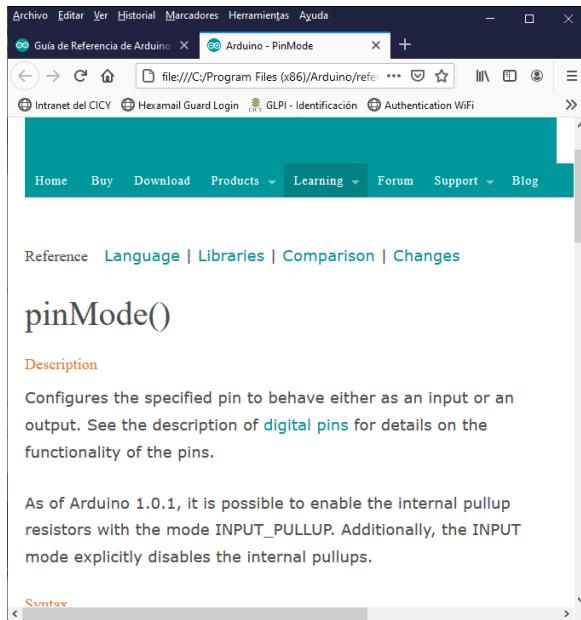
**Objetivo específico:** Aprenderá a usar la ayuda que ofrece el IDE del Arduino y usar el traductor que incorporan los navegadores Mozilla Firefox y Chrome para traducir la ayuda del inglés al español.

Para aprender a usar las sentencias es importante conocer que el IDE del Arduino nos proporciona una ayuda en el tema:

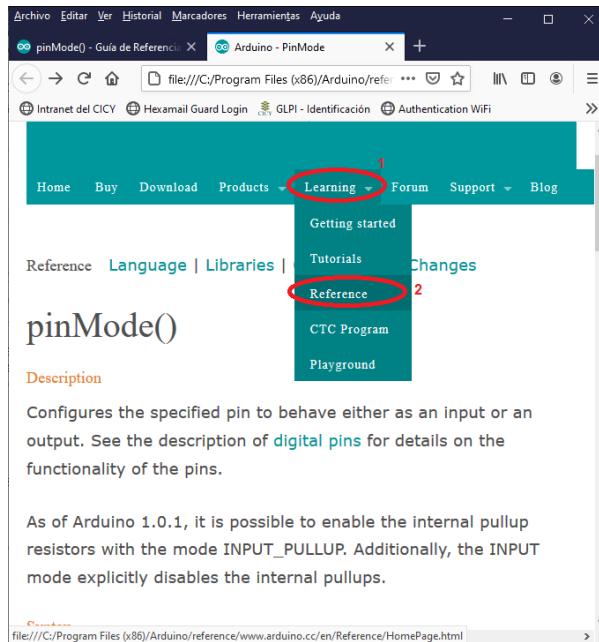
Para este punto se requiere tener abierto el ejemplo Blink.ino" (ver punto anterior), primero seleccione la sentencia que desea conocer, dé un click en la pestaña ayuda y dé un click en "Buscar en referencia", así como se muestra:



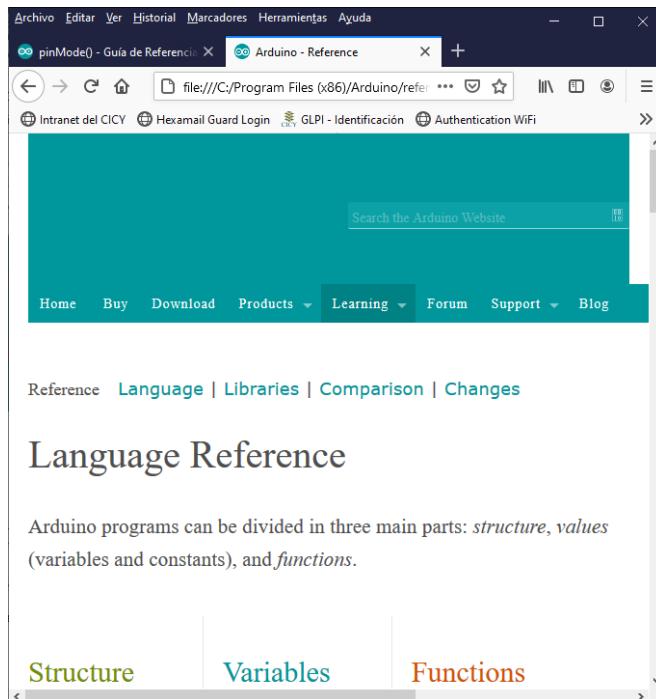
Se abrirá el navegador de internet y mostrará lo siguiente:



Desafortunadamente para algunos, la ayuda está en inglés y afortunadamente, para usar esta ayuda no se requiere internet. Si desea conocer todas las sentencias que el IDE del Arduino contiene solo presione la pestaña “Learning” y posteriormente “reference”, así como se indica:



... y se mostrará toda la ayuda disponible:

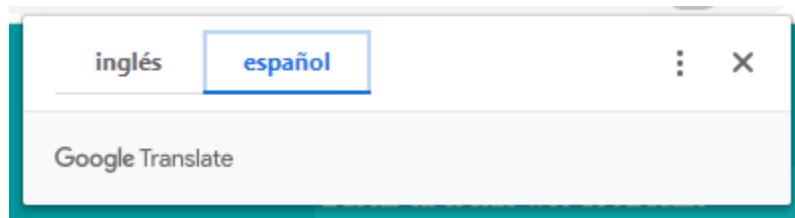


Si dispone de conectividad de internet, la ayuda la encontrará en español:

<https://www.arduino.cc/reference/es/>

... pero la descripción de las funciones estará en inglés. 😞

Una ayuda muy interesante para los aprendices es utilizar el navegador Chrome y utilizar el plugin de “Translate” (traducción):





Igualmente, al navegador Mozilla Firefox, puede descargar el plugin para la traducción:



Es importante estar pendiente de que en la traducción también se alteran las sentencias del código:

Ejemplo

```
int ledPin = 13; // LED conectado al pin digital 13

void setup()
{
 pinMode(ledPin, OUTPUT); // establece el pin digital como salida
}

void loop()
{
 digitalWrite(ledPin, HIGH); // pone el LED en
 delay(1000); // espera una segunda
 digitalWrite(ledPin, LOW); // establece el apagado del LED
 delay(1000); // espera un segundo
}
```

La ayuda es magnífica, sólo ignore la traducción de las sentencias (instrucciones). La traducción correcta sería:

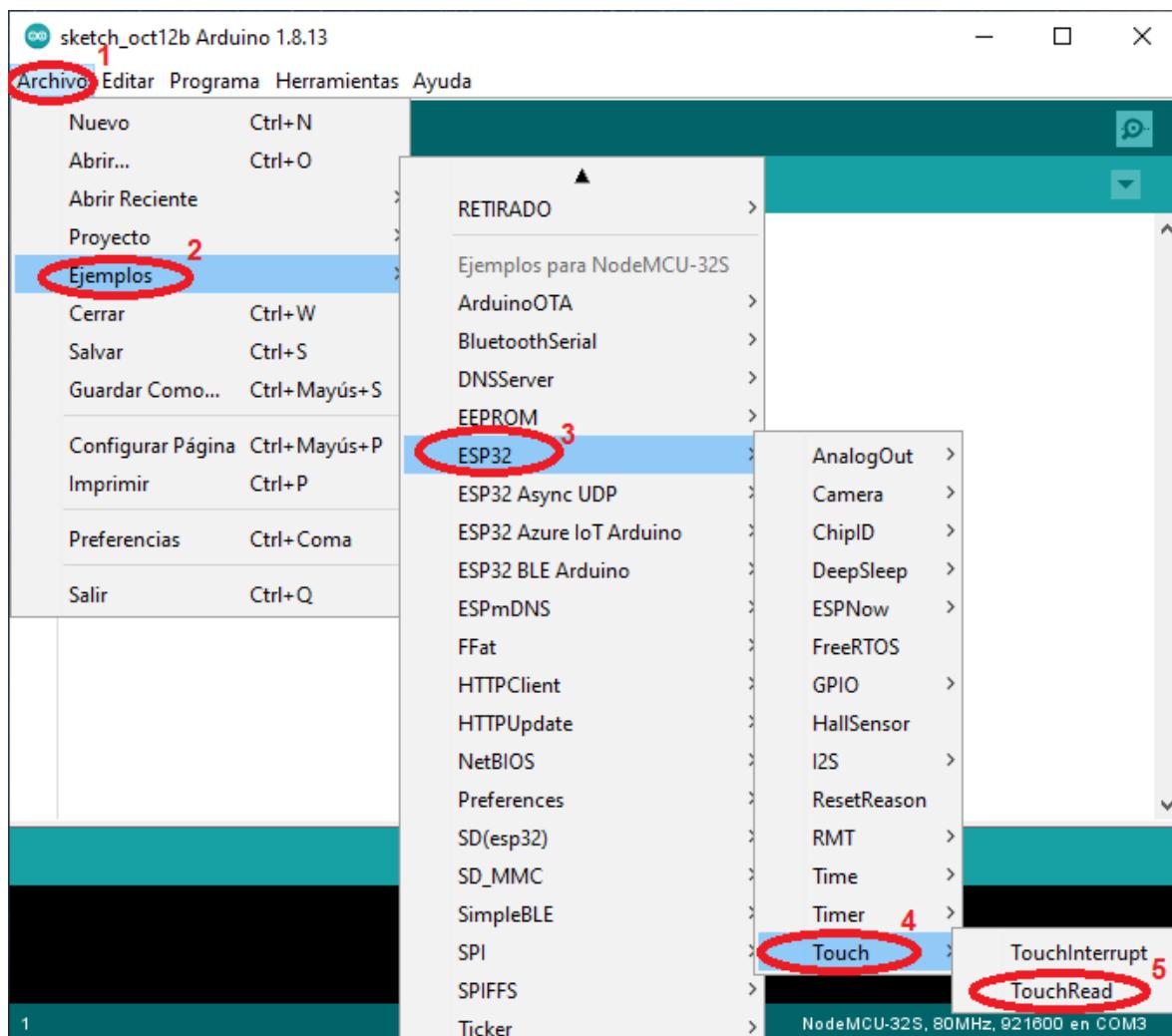
```
// la función setup() se ejecuta una vez al reiniciar o encender la placa
void setup() {
 // inicializa el pin digital llamado LED_BUILTIN como salida.
 pinMode(LED_BUILTIN, OUTPUT);
}

// la función loop() se ejecuta una y otra vez para siempre
void loop() {
 digitalWrite(LED_BUILTIN, HIGH); // enciende el LED (HIGH es el nivel de
 delay(1000); // voltaje)
 digitalWrite(LED_BUILTIN, LOW); // apaga el led haciendo el voltaje bajo
 delay(1000); // espera un segundo
}
```

## 1.5 Usando el monitor del puerto serie del IDE del Arduino

**Objetivo específico:** Utilizará el monitor del puerto serie para mostrar datos de variables.

Se abrirá el ejemplo TouchRead siguiendo la secuencia que se muestra a continuación:



No olvide que:



1. Con el botón  se verifica e inicia la solicitud de conexión con el módulo para la descarga del código a la tarjeta de desarrollo.
2. En algunos modelos de NodeMCU-32S no se requiere acción adicional para que el código empiece a descargarse, así que, no se mostrará la solicitud de conexión. El código se descargará automáticamente.

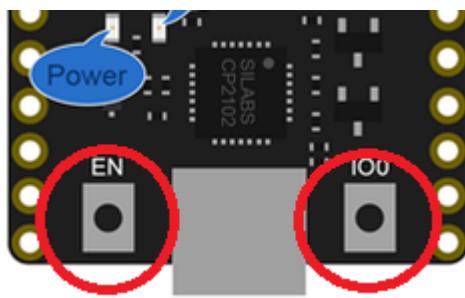
3. En otros modelos de hardware, mostrará una solicitud de conexión. En la parte inferior del IDE se mostrará la leyenda “conectando” así como se muestra en la siguiente figura:



... dependiendo del hardware adquirido, puede probar **UNA** de las acciones siguientes:

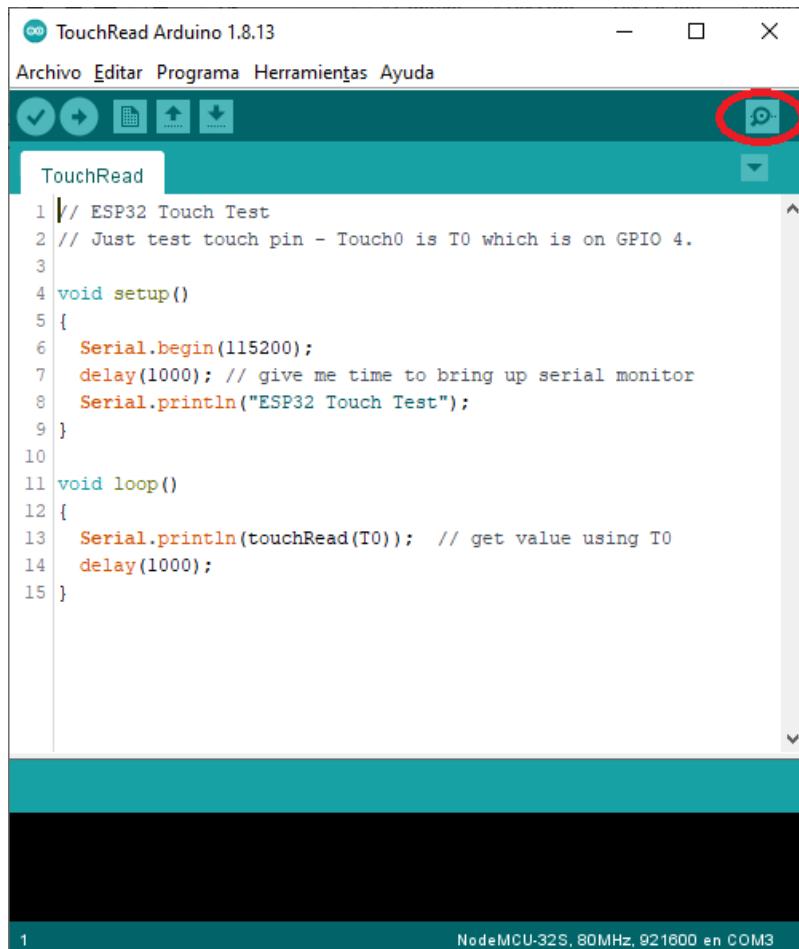
- Presione momentáneamente el botón IO0 y el código deberá empezar a descargarse a su módulo NodeMCU-32S.
- Presione y mantenga presionado el botón IO0, seguidamente presionar momentáneamente el botón de EN ó reset y por último liberar el botón IO0. Siguiendo éstos pasos el código deberá empezar a descargarse a su módulo NodeMCU-32S.

Ambos botones están ubicados a los costados del puerto microUSB de su módulo NodeMCU-32S y deberá empezar a descargar el código a su módulo NodeMCU32S.



4. Si la barra de notificación (intermedia entre la zona del código y la zona negra) del IDE cambia a naranja indica que ocurrió un error en la descarga, para corregirlo siga el procedimiento descrito al final del punto 1.3 “Errores en la descarga del código”.

Cuando el código se haya descargado, ubíquese en la parte superior derecha del IDE del Arduino y dé un click al botón del mouse, así como se muestra a continuación:



The screenshot shows the Arduino IDE interface with the title bar "TouchRead Arduino 1.8.13". The menu bar includes "Archivo", "Editar", "Programa", "Herramientas", and "Ayuda". The toolbar has icons for file operations like Open, Save, and Upload. A red circle highlights the "Upload" button icon (a blue square with a white play symbol). The code editor window contains the following sketch:

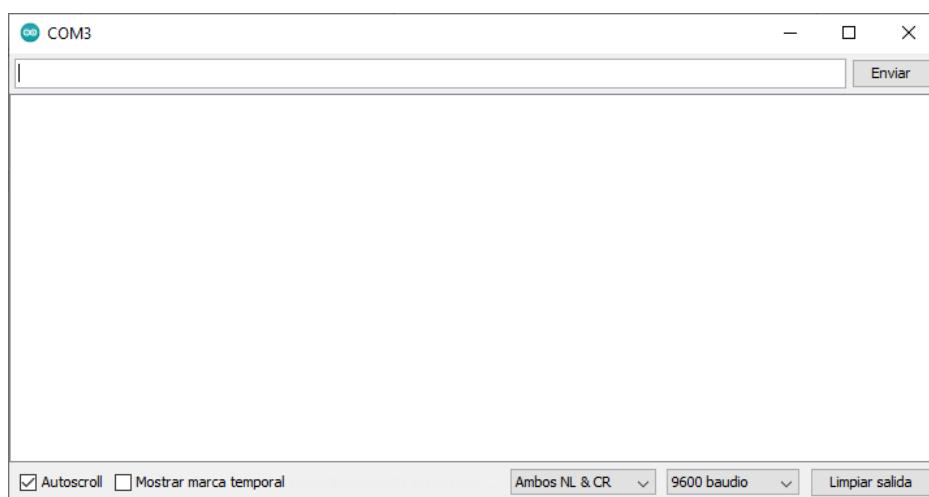
```

1 // ESP32 Touch Test
2 // Just test touch pin - Touch0 is T0 which is on GPIO 4.
3
4 void setup()
5 {
6 Serial.begin(115200);
7 delay(1000); // give me time to bring up serial monitor
8 Serial.println("ESP32 Touch Test");
9 }
10
11 void loop()
12 {
13 Serial.println(touchRead(T0)); // get value using T0
14 delay(1000);
15 }

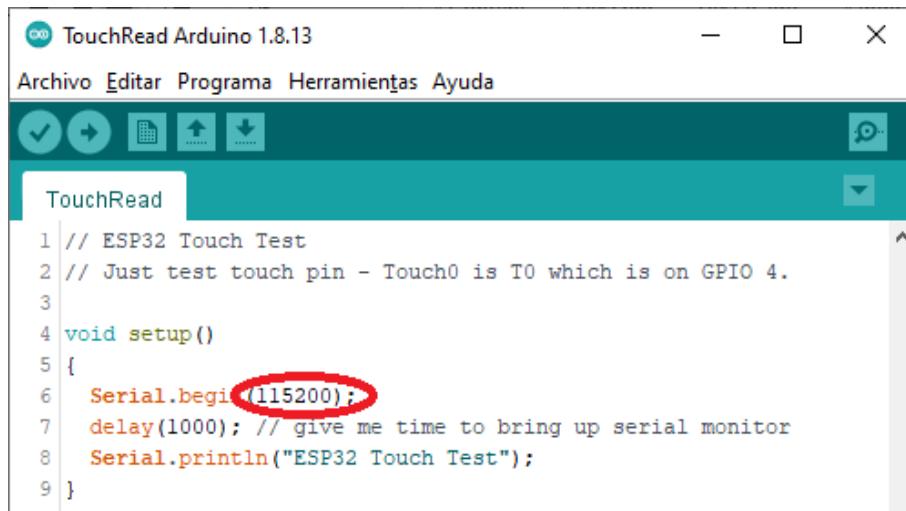
```

The status bar at the bottom indicates "NodeMCU-32S, 80MHz, 921600 en COM3".

... e inmediatamente se abrirá una ventana como ésta:

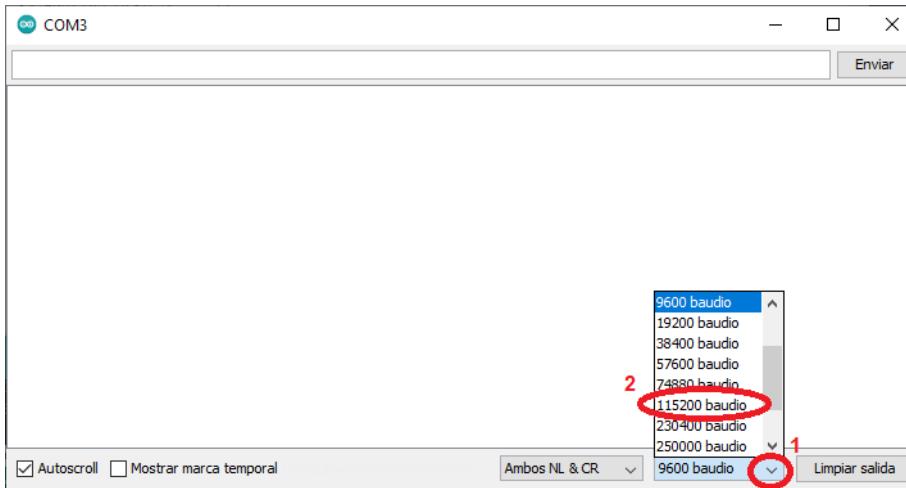


Regrese a la ventana del IDE del Arduino, observe y memorice el valor que se declara en la línea 6 del código:

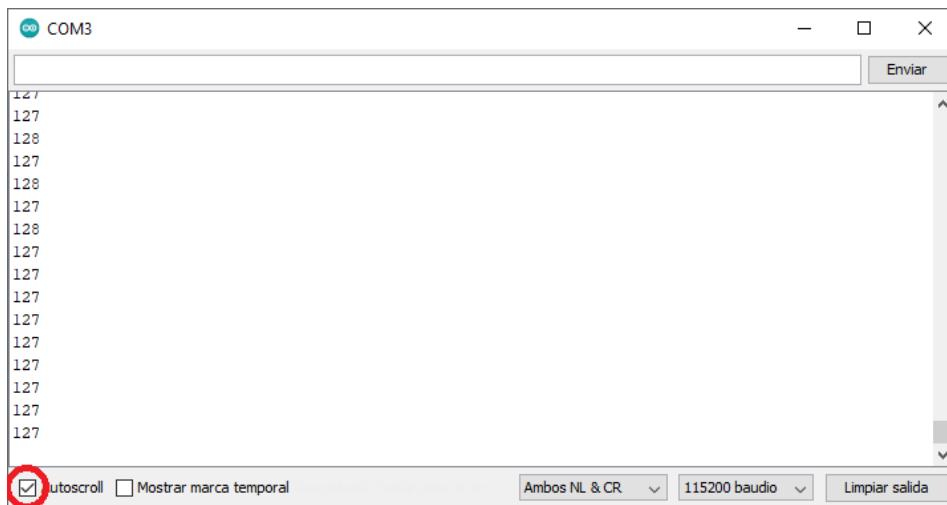


```
// ESP32 Touch Test
// Just test touch pin - Touch0 is T0 which is on GPIO 4.
void setup()
{
 Serial.begin(115200);
 delay(1000); // give me time to bring up serial monitor
 Serial.println("ESP32 Touch Test");
}
```

Posteriormente, regrese al IDE del Arduino y seleccione el valor declarado en la línea 6 del código, así como se muestra a continuación:



... y observe lo que se muestra en el monitor del puerto serie:



Asegúrese de haber activado el checkbox con la etiqueta “autoscroll”, esto con el objetivo de visualizar el último valor enviado por la tarjeta de desarrollo. Si todo salió bien, Usted deberá de tocar el pin con el dedo (así como se muestra en la figura) y al mismo tiempo, observar las lecturas en el monitor del puerto serie.

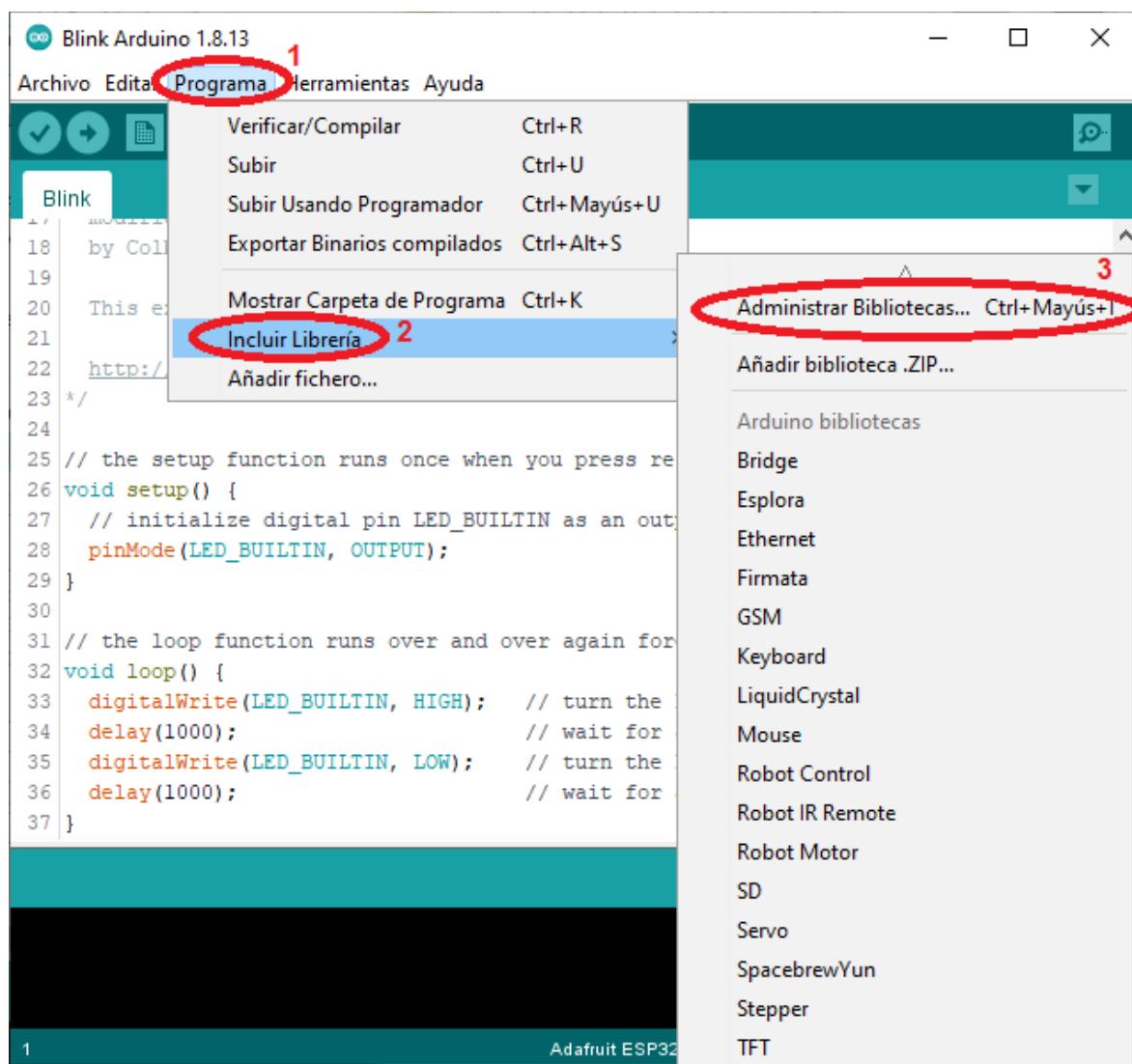


## 1.6 Agregando bibliotecas al IDE del Arduino del catálogo de drivers

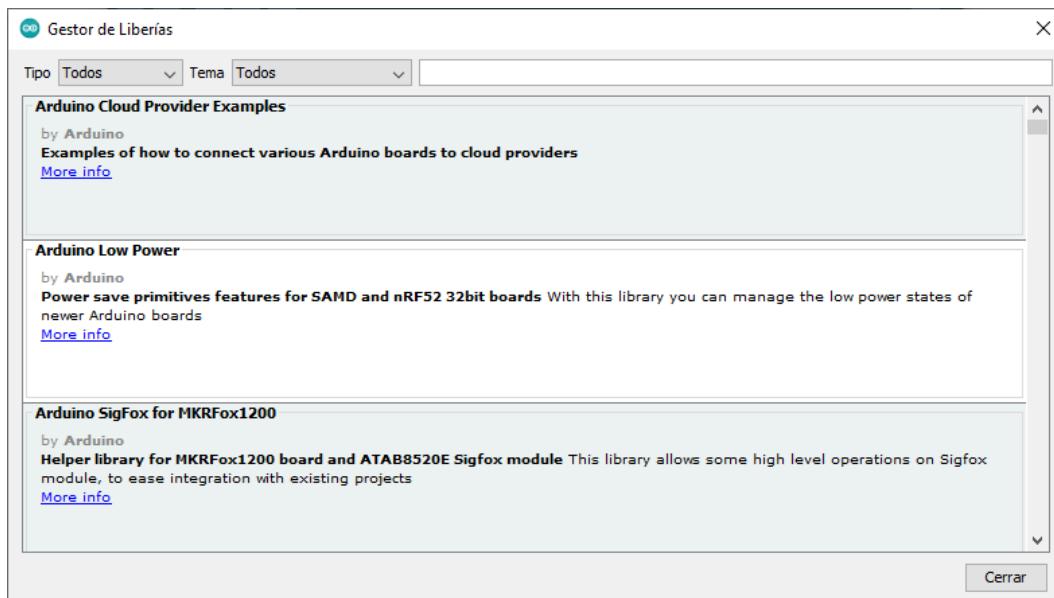
**Objetivo específico:** Aprenderá la manera de instalar las bibliotecas para el hardware que pretenda utilizar en su proyecto, en este caso: pantalla OLED Mod SSD1306, la biblioteca gfxlibrary (con sus dependencias), el sensor DS18B20 y el sensor DHT22 (AM2301).

Supongamos que se desea trabajar con la pantalla SSD1306, para ello, se requiere descargar la biblioteca del catálogo.

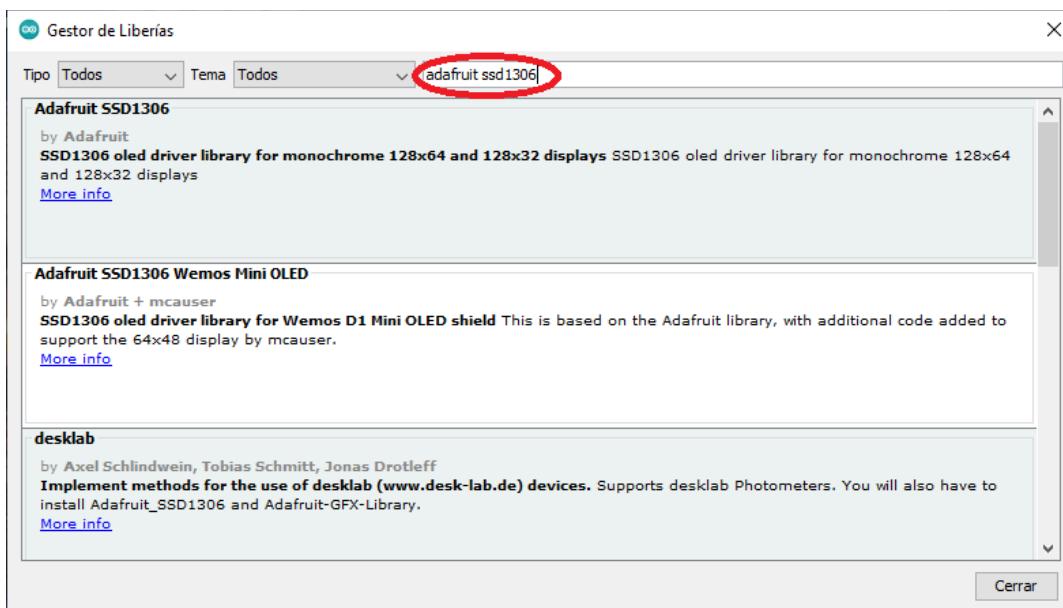
Se inicia presionando de manera secuencial las opciones que se muestran a continuación:



Y se abrirá una ventana como se muestra a continuación:

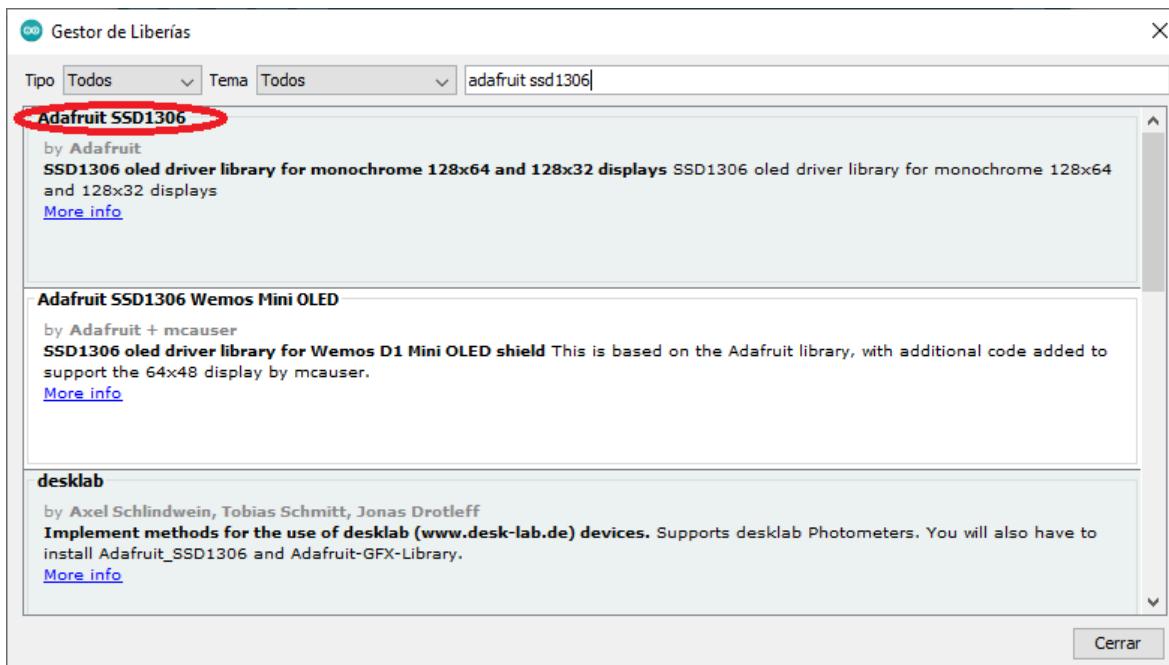


Ubique el campo de texto con la etiqueta “Tema” y escriba el nombre de la biblioteca que usted desea, en este caso escriba “adafruit ssd1306”:

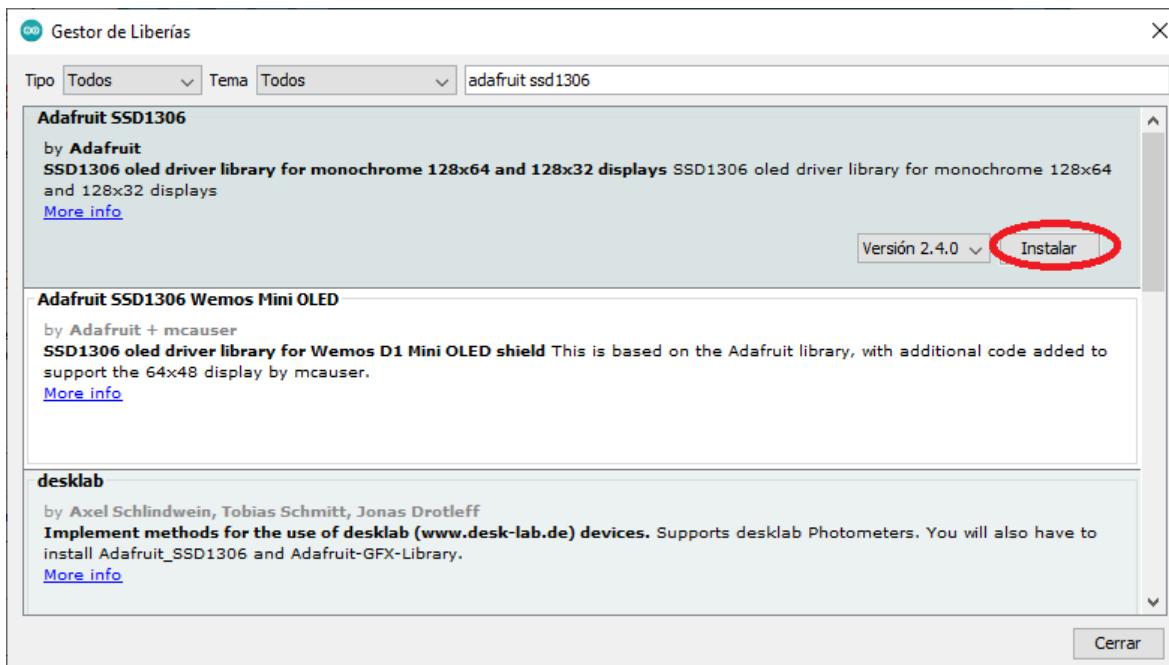


El gestor de bibliotecas automáticamente lo buscará en la base de datos y posteriormente mostrará las versiones disponibles, en este caso la biblioteca

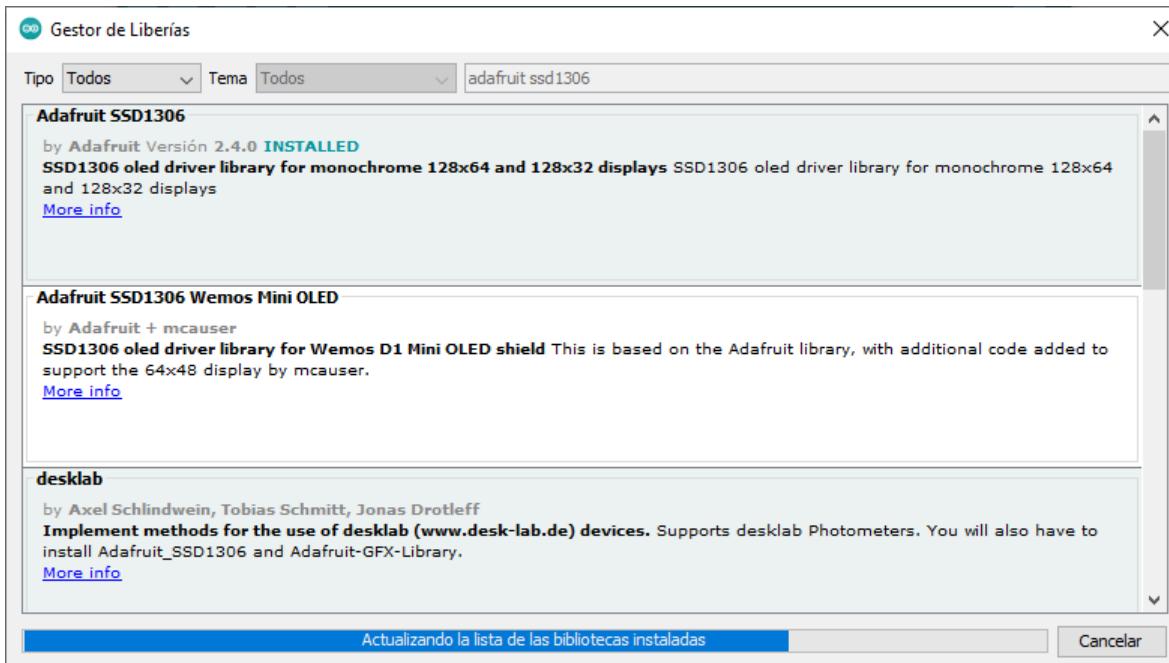
deseada, será la que se muestra primero:



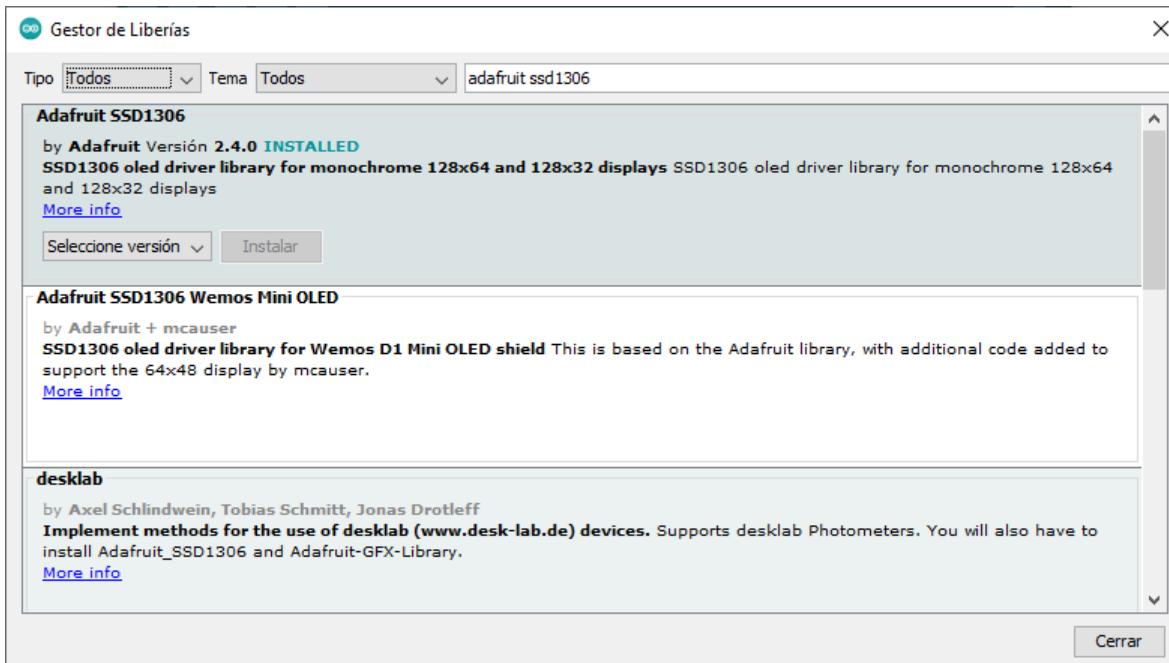
Ponga el cursor del mouse en el campo con el texto Adafruit SSD1306, se mostrarán la versión y un botón llamado “instalar”, presione el botón “Instalar”.



Espere a que la barra de progreso se complete:

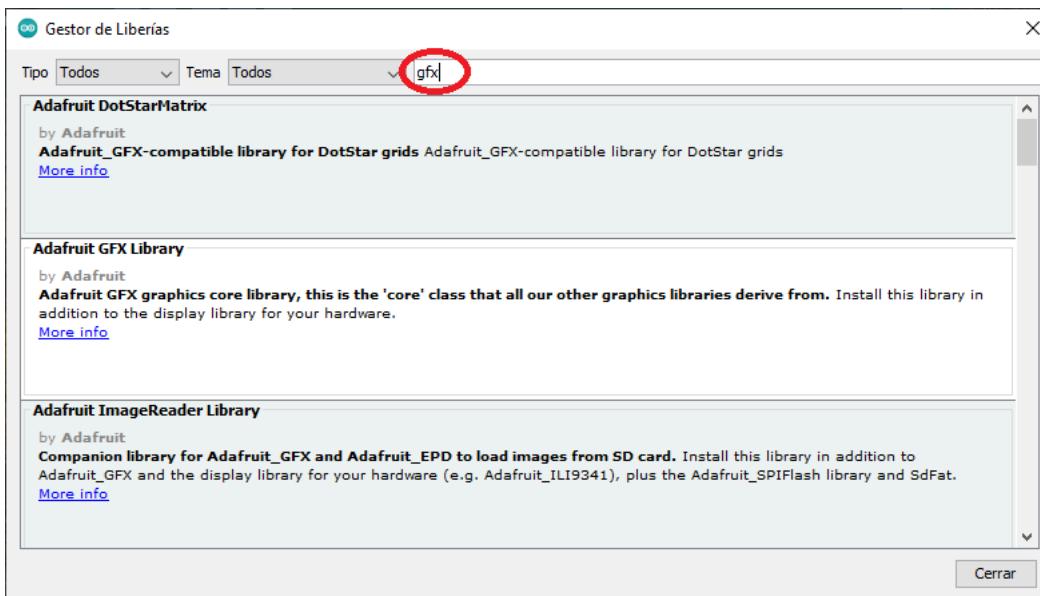


Cuando en la esquina inferior derecha se muestre el botón “cerrar”, indicará que la biblioteca ha sido instalada, sin embargo, aún no presione el botón:

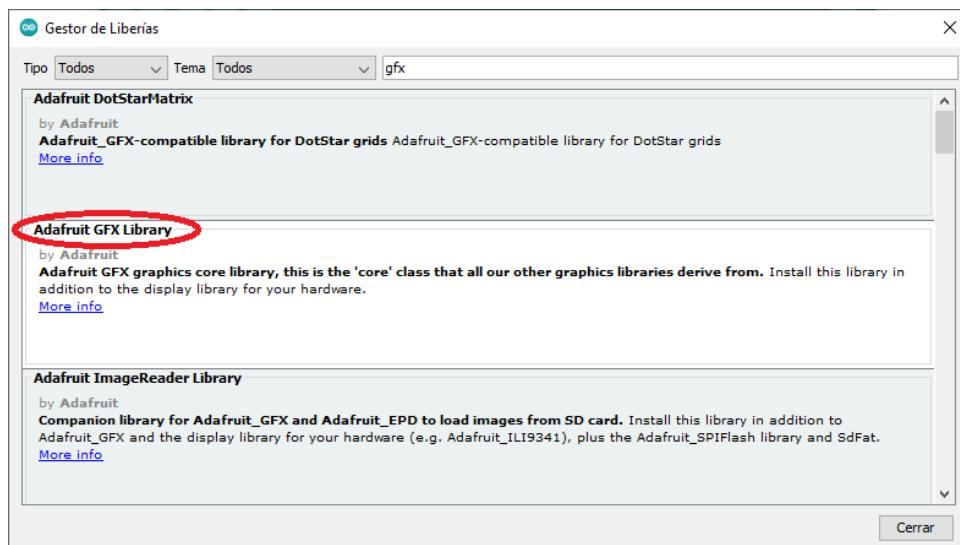


Esta biblioteca depende de otra llamada GFX Library y se instala de la misma

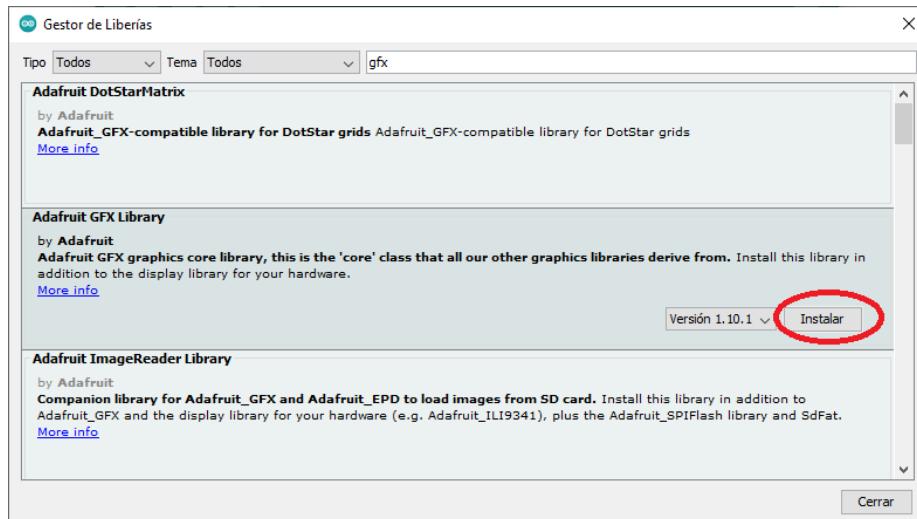
manera, borre lo anterior y teclee “gfx” en el campo de tema, así como se muestra:



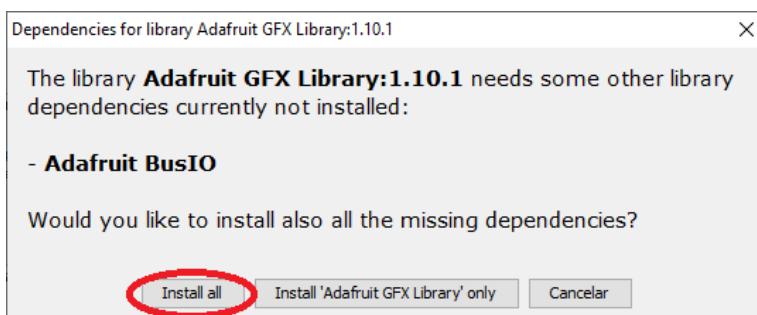
... la biblioteca deseada se muestra inmediatamente:



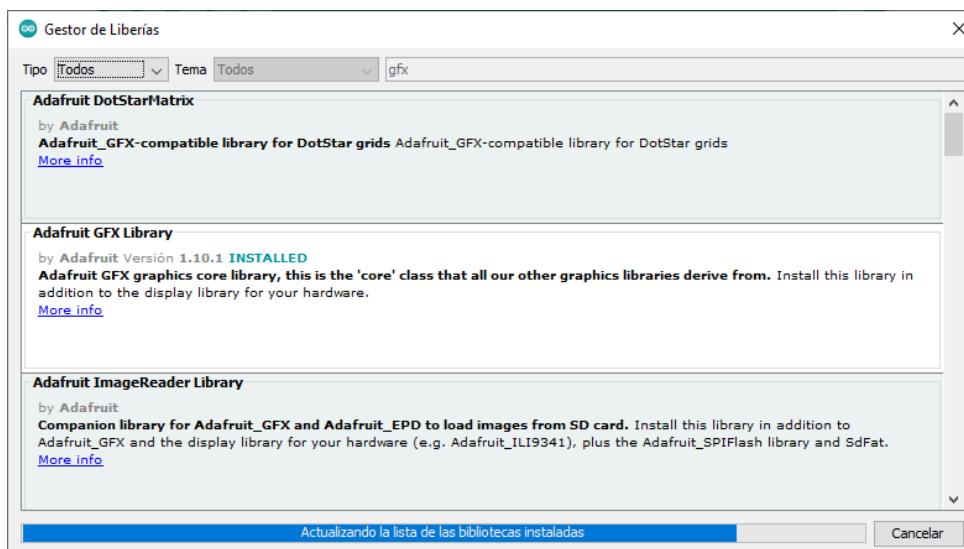
Ponga el mouse sobre el campo mostrado en la figura anterior, el cual tiene el texto “Adafruit GFX Library” y se mostrarán la versión y el botón “instalar”:



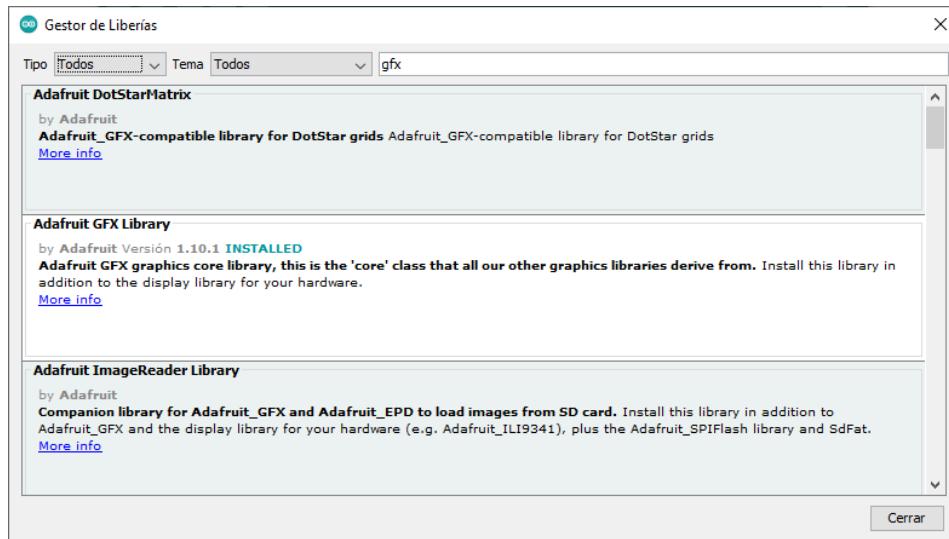
Al presionar instalar, le mostrará la siguiente ventana:



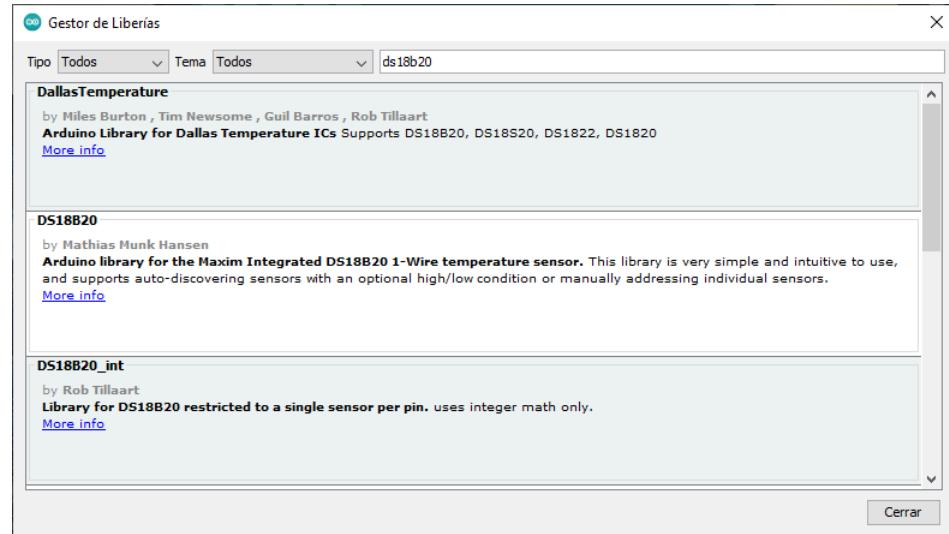
... presión el botón “Install all” y espere a que la instalación finalice:



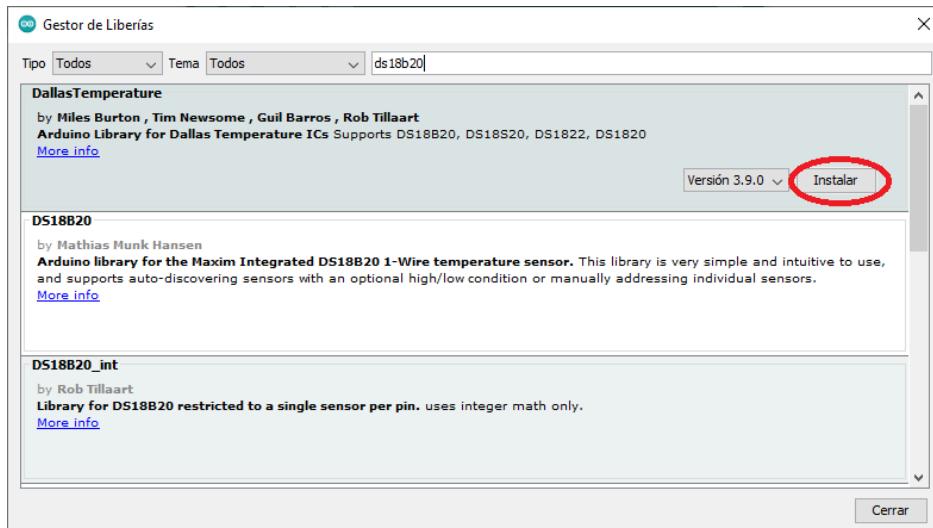
Cuando la instalación finalice, aún no presione el botón cerrar:



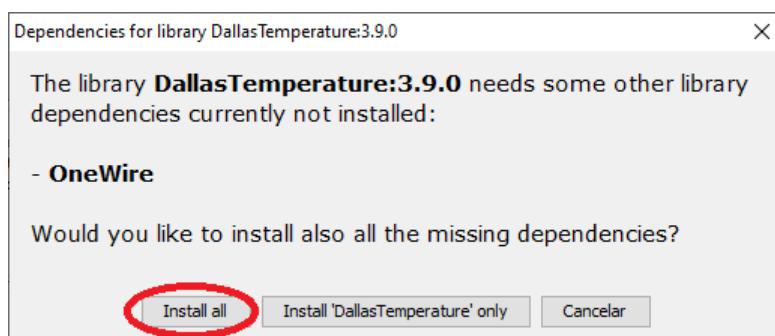
Aún queda pendiente la instalación de la biblioteca DS18B20 que se utilizará en el proyecto, así que, borre lo anterior y escriba “ds18b20” en el campo marcado como “Tema”, así como se muestra:



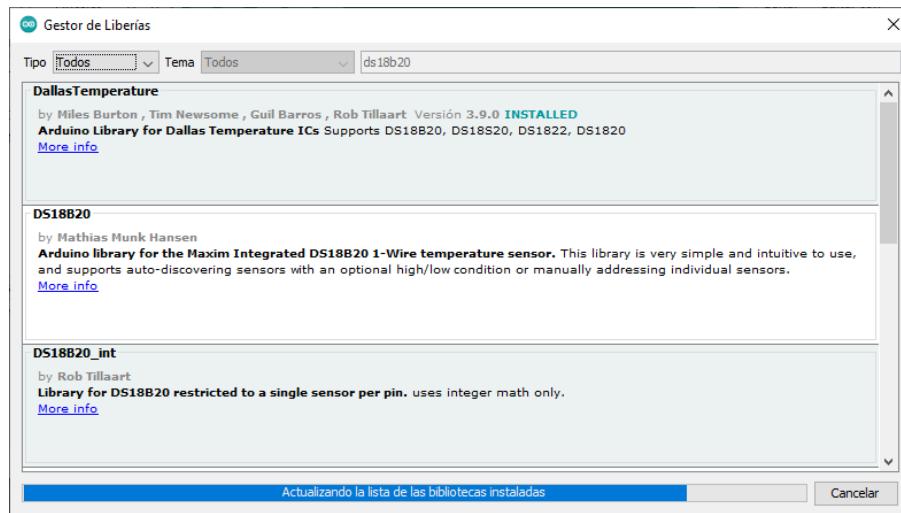
Ponga el cursor de mouse en el área del campo llamado “Dallas temperature” y se activarán los campos con la versión y el botón de instalar:



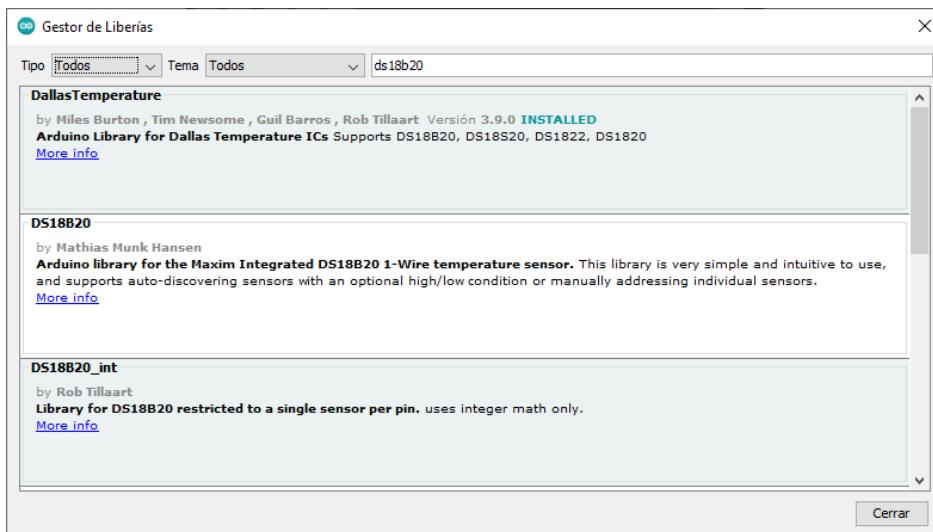
Presione el botón con el texto “instalar” y saldrá una ventana como se muestra a continuación:



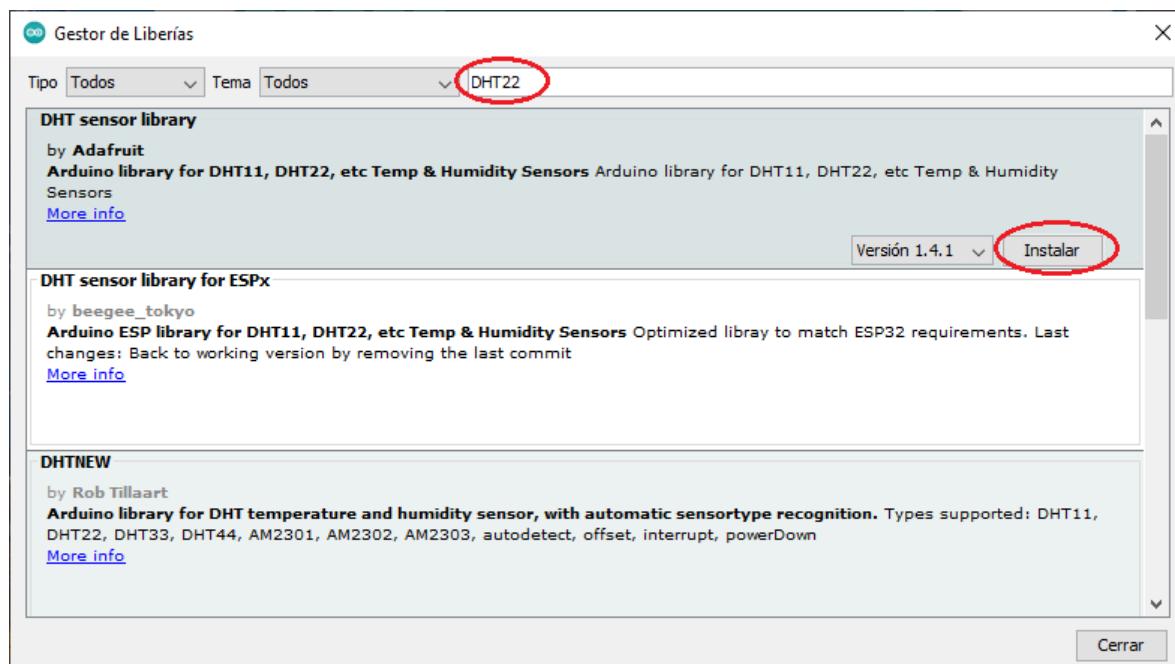
... presione el botón “Install all” y saldrá la siguiente ventana:



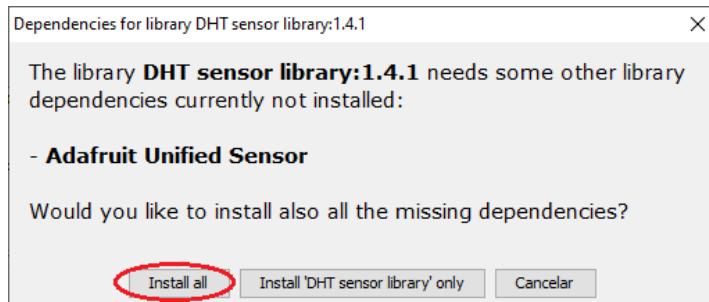
... espere a que la instalación termine y se mostrará una ventana como ésta:



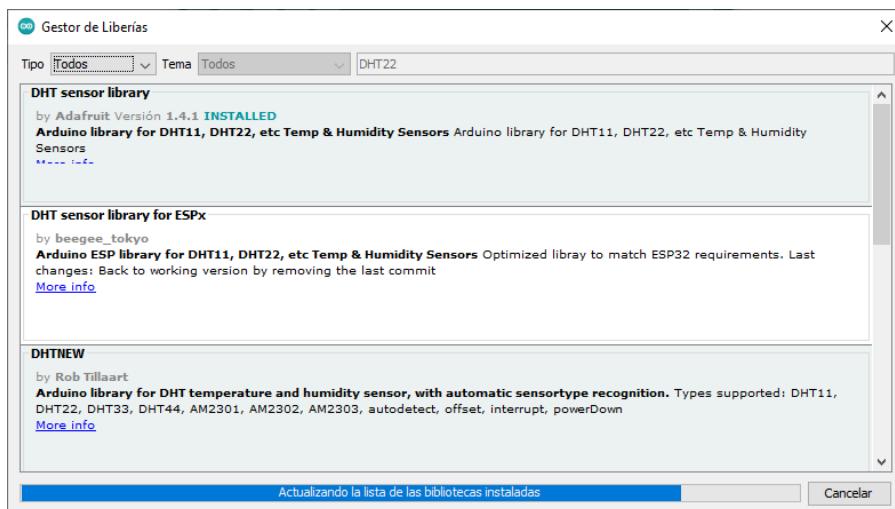
Es relevante mencionar que existe otro sensor, el cual, es ampliamente utilizado y de igual manera, se instalará su biblioteca correspondiente. El sensor mencionado anteriormente es el DHT22 (AM2301) y se inicia la instalación de la manera siguiente: borre lo escrito anteriormente y teclee DHT22 en el campo que se muestra a continuación:



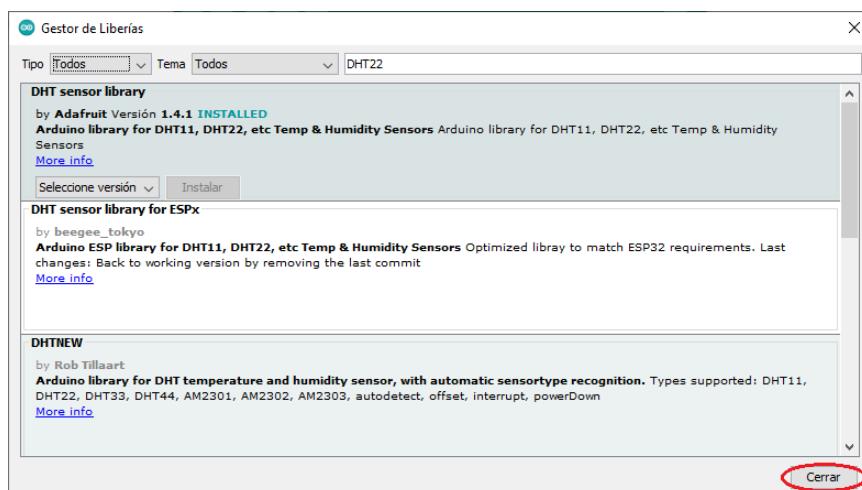
... e inmediatamente se mostrará en primer lugar la biblioteca del sensor DHT, seleccione la última versión y presione el botón "instalar" y a continuación le saldrá una ventana como se muestra:



... presione el botón “Install all” y espere a que la instalación concluya:



Cuando la barra azul se complete y se active el botón inferior derecho, notará que la leyenda del botón habrá cambiado a “cerrar”:



... presione el botón cerrar y se habrá concluido con la instalación de las bibliotecas.

## Modulo II

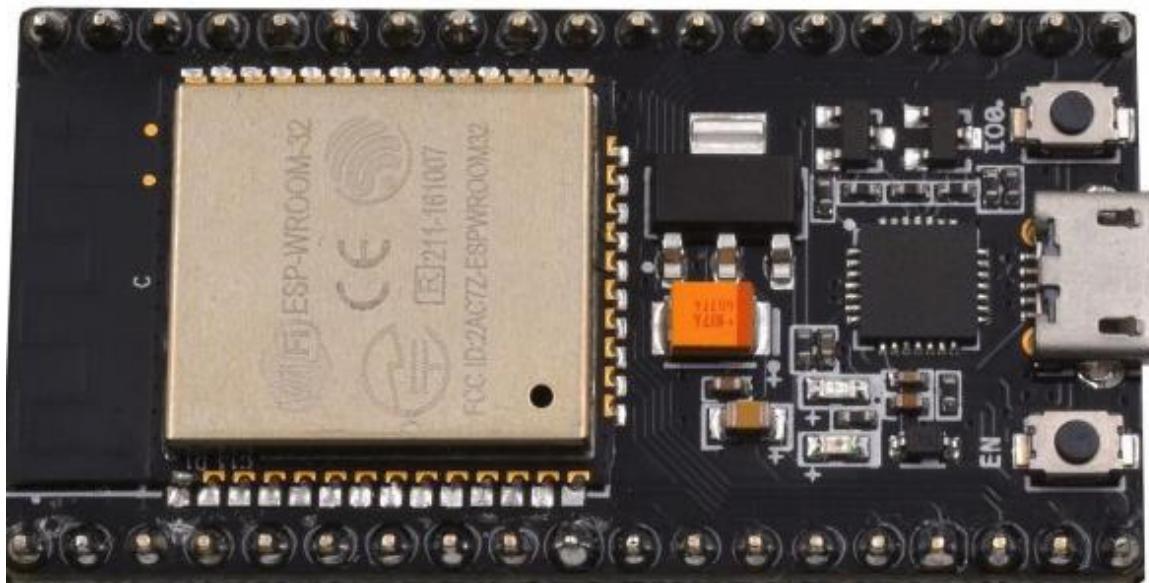
### 2. Implementando códigos en IDE de Arduino

**Objetivo General:** Conocerá la relación con el software del IDE del Arduino y el hardware del módulo NodeMCU-32S. Desarrollará e implementará el código de un medidor de temperatura.

## 2.1 El Hardware del módulo NodeMCU-32S

**Objetivo específico:** Conocer la distribución y funciones de los pines del módulo NodeMCU-32S.

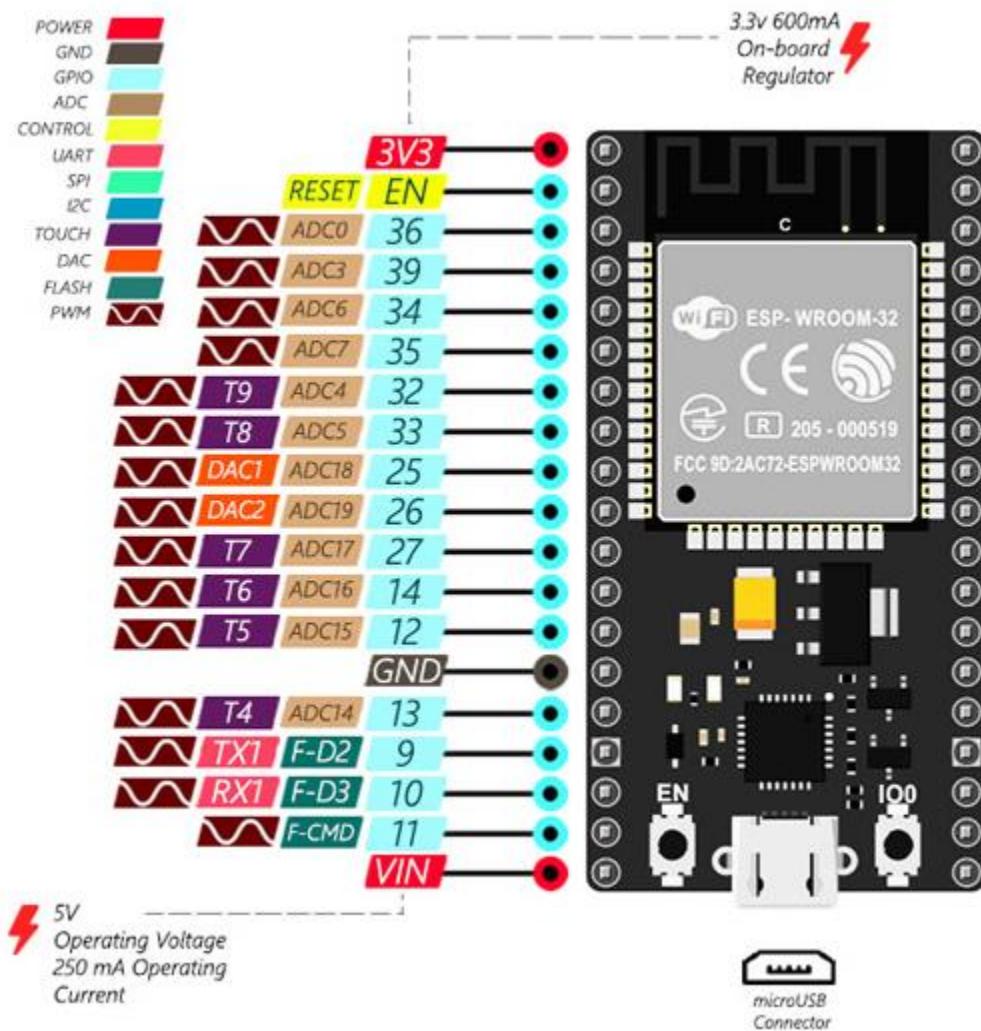
El módulo NodeMCU-32S tiene como núcleo un chip ESP32 y sólo requiere un voltaje de 3.3V. Dicho chip se compone de dos núcleos y pueden ser controlados individualmente. Se puede programar con el IDE del Arduino para la fácil implementación de dispositivos IoT. Adicionalmente, el módulo NodeMCU-32S contiene un chip CP2102, el cual, es un convertidor USB a puerto serie (RS232 TTL) fabricado por Silicon Labs y una entrada microUSB el cual, programar el módulo Huzzah y una comunicación con protocolo USB-RS232 hacia la computadora.



### Características:

- IEEE 802.11 b/g/n Wi-Fi 2.4Ghz
- Clock frequency adjustment range from 80Mhz to 240 Mhz
- Built-in 2-channel 12-bit high-precision ADC with up to 18 channels
- Soporta modos STA/AP/STA+AP
- Support UART/GPIO/ADC/DAC/SDIO/SD card/PWM/I2C/I2S interface
- Deep Sleep ultra bajo consumo < 5uA
- Memoria Flash 4 MB
- Memoria RAM para el usuario < 327KB

Pinout:

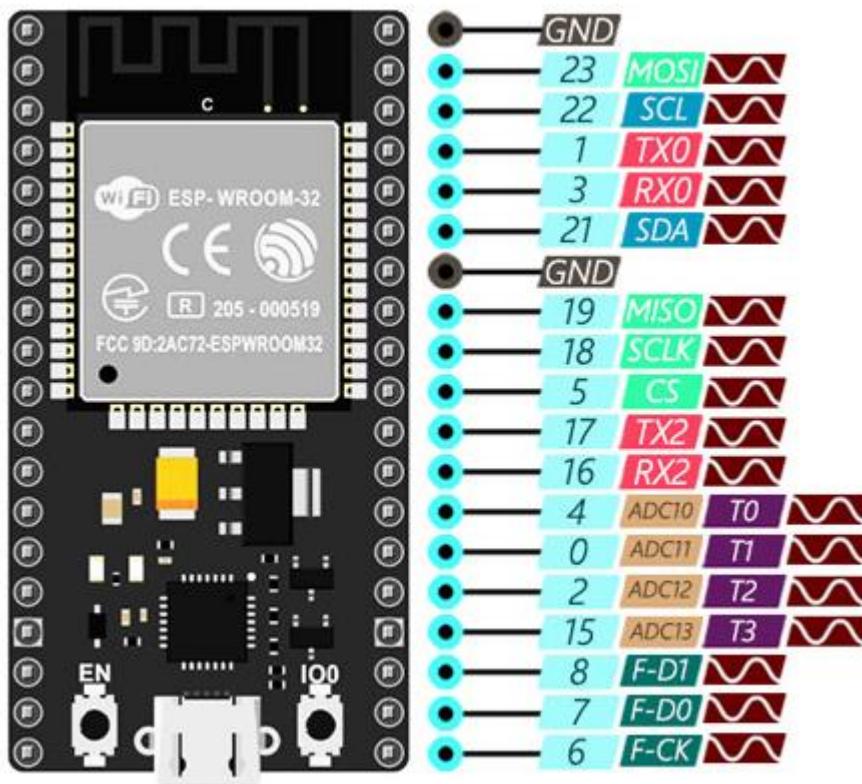


GPIO0 se encuentra conectado al pushbutton (ubicado en la esquina inferior derecha) marcado con el texto IO0.

GPIO2 se encuentra conectado al led verde para indicador para propósito general.

Los pines indicados con el cuadro color azul claro (light blue) son los que corresponden al número asignado en el IDE del Arduino.

Pinout:



Los pines indicados con el cuadro color azul claro ( ) son los que corresponden al número asignado en el IDE del Arduino.

Algunas definiciones hechas en el IDE del Arduino haciendo referencia a los pines del NodeMCU-32S para tomar en cuenta en la programación:

- LED\_BUILTIN corresponde al pin 2 (led verde en la tarjeta)
- A0 corresponde al pin 36 (en la figura se muestra los pines A's como ADC's, en el IDE del Arduino no reconoce el texto "ADC0")
- T0 corresponde al pin 4
- KEY\_BUILTIN corresponde al pushbutton marcado como IO0 (ver esquina inferior derecha)

Para más definiciones ver:

C:\Users\MiUsuario\AppData\Local\Arduino15\packages\esp32\hardware\esp32\1.0.4\variants\nodemcu-32s\pins\_Arduino.h

No olvidar que **MiUsuario** es el nombre de usuario que se le asignó a la computadora.

Los pines marcados en verde están perfectos para ser utilizados, los marcados en ámbar deben manejarse con cuidado ya que, pueden tener un comportamiento inesperado al iniciar y los marcados en rojo no son recomendable para su uso.

| <b>GPIO</b> | <b>Input</b> | <b>Output</b> | <b>Notes</b>                                                                                                                              |
|-------------|--------------|---------------|-------------------------------------------------------------------------------------------------------------------------------------------|
| <b>0</b>    | PULLED UP    | OK            | Se conecta el <b>pushbutton</b> de la tarjeta<br>Genera una señal PWM al iniciar<br>Entrará en bootloader en estado LOW después del RESET |
| <b>1</b>    | TX PIN       | OK            | Genera una señal serial al iniciar                                                                                                        |
| <b>2</b>    | OK           | OK            | Se conecta el <b>LED</b> de la tarjeta<br>Requiere estar en FLOATING para bootloader                                                      |
| <b>3</b>    | OK           | RX PIN        | Genera una señal HIGH al iniciar                                                                                                          |
| <b>4</b>    | OK           | OK            |                                                                                                                                           |
| <b>5</b>    | OK           | OK            | Debe de estar en estado HIGH al iniciar<br>Genera una señal PWM al iniciar<br>VSPI CS                                                     |
| <b>6</b>    | X            | X             | Se encuentra conectado a la SPI flash interna                                                                                             |
| <b>7</b>    | X            | X             | Se encuentra conectado a la SPI flash interna                                                                                             |
| <b>8</b>    | X            | X             | Se encuentra conectado a la SPI flash interna                                                                                             |
| <b>9</b>    | X            | X             | Se encuentra conectado a la SPI flash interna                                                                                             |
| <b>10</b>   | X            | X             | Se encuentra conectado a la SPI flash interna                                                                                             |
| <b>11</b>   | X            | X             | Se encuentra conectado a la SPI flash interna                                                                                             |
| <b>12</b>   | OK           | OK            | Debe de estar en estado LOW al iniciar<br>HSPI MISO                                                                                       |
| <b>13</b>   | OK           | OK            | HSPI MOSI                                                                                                                                 |
| <b>14</b>   | OK           | OK            | Genera una señal PWM al iniciar<br>HSPI CLK                                                                                               |
| <b>15</b>   | OK           | OK            | Debe de estar en estado HIGH al iniciar<br>Genera una señal PWM al iniciar<br>HSPI CS                                                     |
| <b>16</b>   | OK           | OK            |                                                                                                                                           |
| <b>17</b>   | OK           | OK            |                                                                                                                                           |
| <b>18</b>   | OK           | OK            | VSPI CLK                                                                                                                                  |

|           |    |    |            |
|-----------|----|----|------------|
| <b>19</b> | OK | OK | VSPI MISO  |
| <b>21</b> | OK | OK |            |
| <b>22</b> | OK | OK |            |
| <b>23</b> | OK | OK | VSPI MOSI  |
| <b>25</b> | OK | OK |            |
| <b>26</b> | OK | OK |            |
| <b>27</b> | OK | OK |            |
| <b>32</b> | OK | OK |            |
| <b>33</b> | OK | OK |            |
| <b>34</b> | OK |    | Sólo INPUT |
| <b>35</b> | OK |    | Sólo INPUT |
| <b>36</b> | OK |    | Sólo INPUT |
| <b>39</b> | OK |    | Sólo INPUT |

## 2.2 Análisis de ejemplos considerados relevantes para la implementación de un lector de temperatura.

### Consideración importante

La estructura de programación del Arduino es muy sencilla, se debe considerar lo siguiente, el método `setup()` solamente se ejecuta una vez y sirve para configurar los periféricos:

```
void setup() {
 // put your setup code here, to run once:

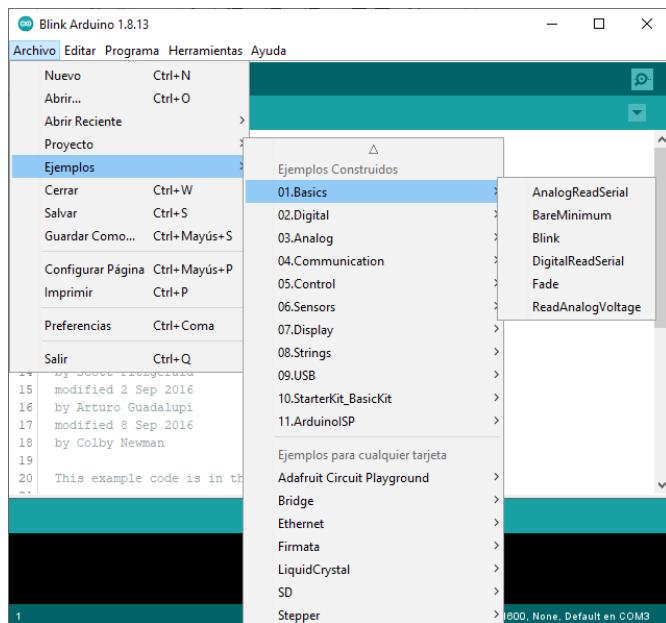
}
```

El método `loop()` se ejecuta de manera infinita, es decir, cuando llegue a la última línea de código, el microcontrolador ejecutará nuevamente la primera:

```
void loop() {
 // put your main code here, to run repeatedly:

}
```

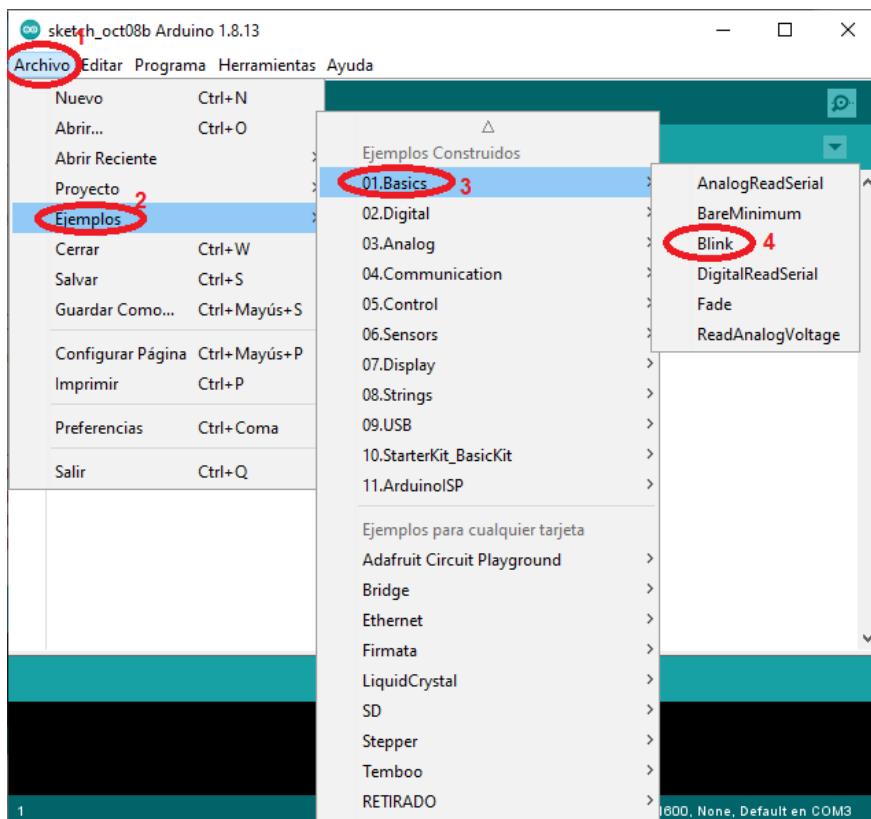
Todos los ejemplos que a continuación se mencionan se encuentran en la sección de ejemplos:



## 2.2.1 Blink.ino

**Objetivo específico:** Aprenderá a usar los pines de salida desde el IDE del Arduino.

Se inicia abriendo el ejemplo blink.ino:



Analizaremos el código de éste ejemplo:

```
// the setup function runs once when you press reset or power the board
void setup() {
 // initialize digital pin LED_BUILTIN as an output.
 pinMode(LED_BUILTIN, OUTPUT);
}
```

1. La primera y tercera línea son comentario (observe que están en inglés)
2. La segunda línea es el encabezado del método `setup()` que es el encargado de configurar nuestro periféricos y el microcontrolador SOLO lo ejecuta UNA vez.
3. La cuarta línea es una sentencia (comando ó instrucción) que se encarga de configurar los pines GPIO del microcontrolador.

4. La quinta línea le indica al compilador (convertidor de texto a lenguaje máquina) que ahí terminan las instrucciones del método setup().

Posteriormente analizamos la función loop()

```
// la función loop() se ejecuta una y otra vez para siempre
void loop() {
 digitalWrite(LED_BUILTIN, HIGH); // enciende el LED (HIGH es el nivel de
 // voltaje)
 delay(1000); // espera un segundo
 digitalWrite(LED_BUILTIN, LOW); // apaga el led haciendo el voltaje bajo
 delay(1000); // espera un segundo
}
```

Si usted recuerda, la definición “LED\_BUILTIN” (con letras azules), está hecha de la siguiente manera (OJO: No es visible desde el IDE del arduino):

```
int LED_BUILTIN = 2;
```

El programador puede cambiar por otra asignación, por ejemplo:

```
int LEDR = 2;
```

... y reescribir el código:

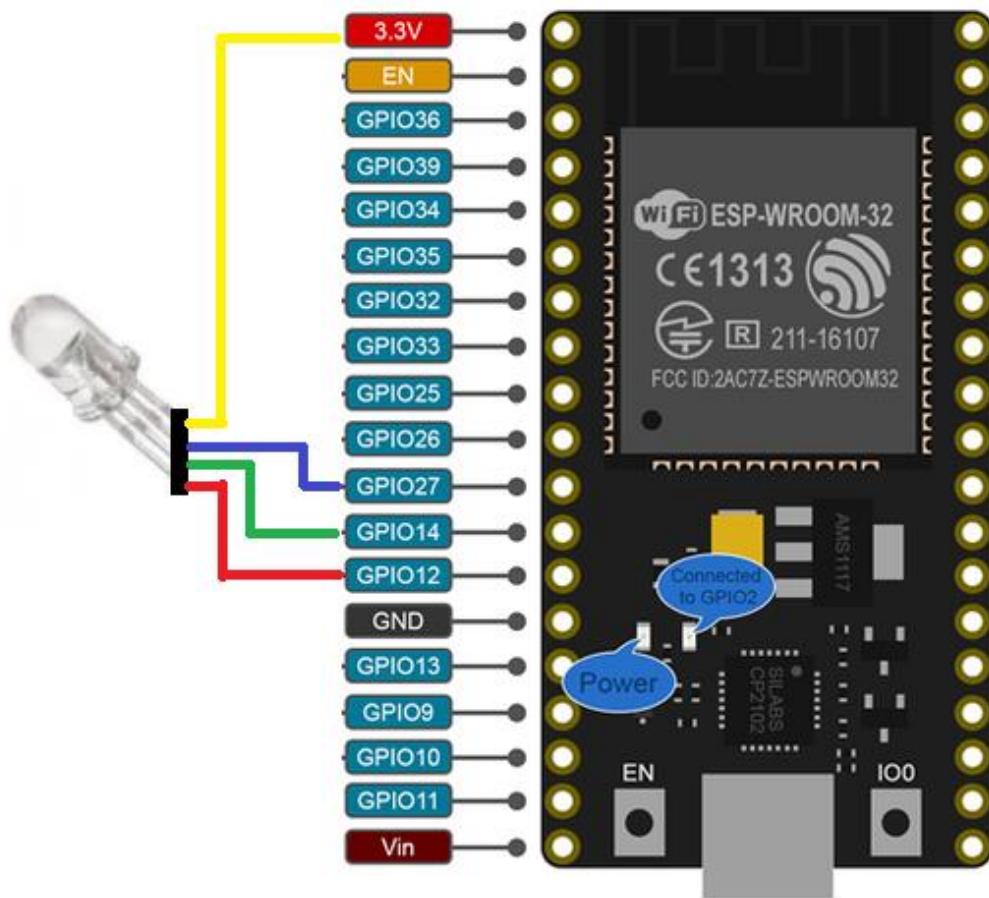
```
int LEDR = 2;
```

```
// la función setup() se ejecuta una vez al reiniciar o encender la placa
void setup() {
 // inicializa el pin digital llamado LED_BUILTIN como salida.
 pinMode(LEDR, OUTPUT);
}

// la función loop() se ejecuta una y otra vez para siempre
void loop() {
 digitalWrite(LEDR, LOW); // enciende el LED (HIGH es el nivel de voltaje)
 delay(1000); // espera un segundo
 digitalWrite(LEDR, HIGH); // apaga el led haciendo el voltaje bajo
 delay(1000); // espera un segundo
}
```

Nótese que el color de la nueva definición cambió a negro. Pero el código funciona exactamente igual al código anterior.

Ahora, con esa nueva definición... ¿Qué números le puedo asignar?, ¿Cuántos led puedo hacer parpadear al mismo tiempo?, Ahora conecte un led tricolor y haga las siguientes conexiones:



**IMPORTANTE: NO CONECTE EL LED DIRECTAMENTE AL MODULO,** utilice sólo el LED proporcionado en el curso, ya que contiene 3 resistencias limitadoras de corriente: LedR (180 ohms), LedG y LedB (100 ohms)

Cambie el valor de la variable por 12, compile y descargue su código a la tarjeta de desarrollo, posteriormente cámbiela por los valores 14 y 27. En cada caso observe los efectos:

```
int LEDR = 12;
```

... OJO, No olvide que:



1. Con el botón se verifica e inicia la solicitud de conexión con el módulo para la descarga del código a la tarjeta de desarrollo.
2. En algunos modelos de NodeMCU-32S no se requiere acción adicional para que el código empiece a descargarse, así que, no se mostrará la solicitud de conexión. El código se descargará automáticamente.

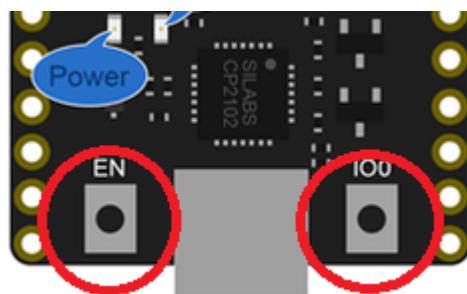
3. En otros modelos de hardware, mostrará una solicitud de conexión. En la parte inferior del IDE se mostrará la leyenda “conectando” así como se muestra en la siguiente figura:



... dependiendo del hardware adquirido, puede probar **UNA** de las acciones siguientes:

- Presione momentáneamente el botón IO0 y el código deberá empezar a descargarse a su módulo NodeMCU-32S.
- Presione y mantenga presionado el botón IO0, seguidamente presionar momentáneamente el botón de EN ó reset y por último liberar el botón IO0. Siguiendo éstos pasos el código deberá empezar a descargarse a su módulo NodeMCU-32S.

Ambos botones están ubicados a los costados del puerto microUSB de su módulo NodeMCU-32S y deberá empezar a descargar el código a su módulo NodeMCU32S.



4. Si la barra de notificación (intermedia entre la zona del código y la zona negra) del IDE cambia a naranja indica que ocurrió un error en la descarga, para corregirlo siga el procedimiento descrito al final del punto 1.3 “Errores en la descarga del código”.

¿Qué sucede si desea que parpadeen dos led's? Agregue al código anterior lo marcado en verde:

```
int LEDR = 12;
int LEDG = 14;
```

```
// la función setup() se ejecuta una vez al reiniciar o encender la placa
void setup() {
 // inicializa el pin digital llamado LEDR como salida.
 pinMode(LEDR, OUTPUT);

 // inicializa el pin digital llamado LEDG como salida.
 pinMode(LEDG, OUTPUT);
}

// la función loop() se ejecuta una y otra vez para siempre
void loop() {
 digitalWrite(LEDR, LOW); // enciende el LED (HIGH es el nivel de voltaje)
 delay(1000); // espera un segundo
 digitalWrite(LEDR, HIGH); // apaga el led haciendo el voltaje bajo
 delay(1000); // espera un segundo

 // agregamos ésta sección de código para que parpadee ahora el led verde
 digitalWrite(LEDG, LOW); // enciende el LED (HIGH es el nivel de voltaje)
 delay(1000); // espera un segundo
 digitalWrite(LEDG, HIGH); // apaga el led haciendo el voltaje bajo
 delay(1000); // espera un segundo
}
```

¿y en el caso de tres led's? Agregue al código lo marcado en azul:

```
int LEDR = 12;
int LEDG = 14;
int LEDB = 27;

// la función setup() se ejecuta una vez al reiniciar o encender la placa
void setup() {
 // inicializa el pin digital llamado LEDR como salida.
 pinMode(LEDR, OUTPUT);

 // inicializa el pin digital llamado LEDG como salida.
 pinMode(LEDG, OUTPUT);

 // inicializa el pin digital llamado LEDB como salida,
 pinMode(LEDB, OUTPUT);

}

// la función loop() se ejecuta una y otra vez para siempre
void loop() {
 digitalWrite(LEDR, LOW); // enciende el LED (HIGH es el nivel de voltaje)
 delay(1000); // espera un segundo
 digitalWrite(LEDR, HIGH); // apaga el led haciendo el voltaje bajo
 delay(1000); // espera un segundo
```

```
// agregamos ésta sección de código para que parpadee ahora el led verde
digitalWrite(LEDG, LOW); // enciende el LED (HIGH es el nivel de voltaje)
delay(1000); // espera un segundo
digitalWrite(LEDG, HIGH); // apaga el led haciendo el voltaje bajo
delay(1000); // espera un segundo

// agregamos ésta sección de código para que parpadee ahora el led azul
digitalWrite(LEDB, LOW); // enciende el LED (HIGH es el nivel de voltaje)
delay(1000); // espera un segundo
digitalWrite(LEDB, HIGH); // apaga el led haciendo el voltaje bajo
delay(1000); // espera un segundo
}
```

### Actividades:

Ahora implemente un código que simule un semáforo real:

1. El led Verde encendido por 3 segundos, luego que parpadee cada medio segundo tres veces.
2. Luego el led Azul encienda durante 1 segundo.
3. Por último que el led Rojo se quede encendido 3 segundos.

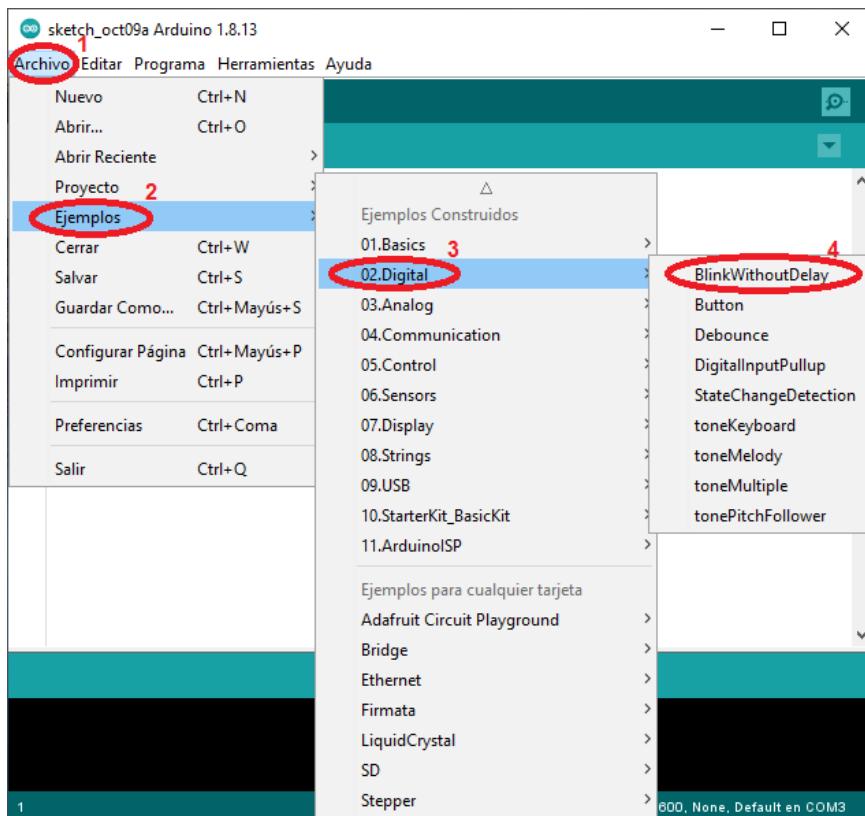
Considere que `delay(1000);` el valor de 1000 corresponde a 1 segundo.

## 2.2.2 BlinkWithOutDelay.ino

**Objetivo específico:** Aprenderá a programar prescindiendo de la sentencia `delay()`. Adicionalmente, conocerá la función `millis()`.

La sentencia `delay()` es muy útil cuando se desea que el procesador espere por algún tiempo, pero encadena al microcontrolador a realizar una sola función conocida como NOP (No Operation) y no se puede estar pendiente de otros eventos. Dicha instrucción (`delay`) NO puede usarse junto con el stack de WiFi ni con otras funciones que dependen de eventos programados.

Se analizará el siguiente ejemplo, cuya conexión del led está basado en el punto 2.2.1 de éste manual:



Compile y descargue el código a su módulo NodeMCU.

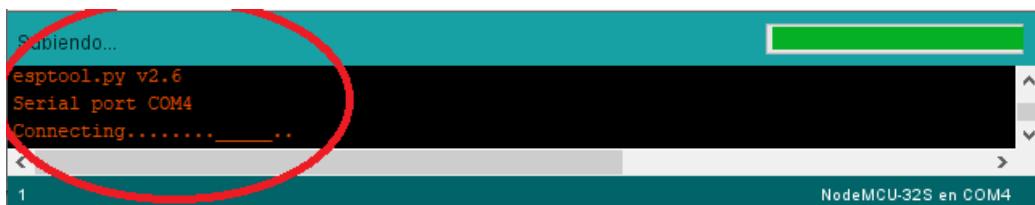
... **OJO**, No olvide que:



1. Con el botón  se verifica e inicia la solicitud de conexión con el módulo para la descarga del código a la tarjeta de desarrollo.

2. En algunos modelos de NodeMCU-32S no se requiere acción adicional para que el código empiece a descargarse, así que, no se mostrará la solicitud de conexión. El código se descargará automáticamente.

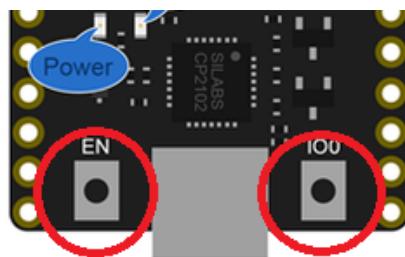
3. En otros modelos de hardware, mostrará una solicitud de conexión. En la parte inferior del IDE se mostrará la leyenda “conectando” así como se muestra en la siguiente figura:



... dependiendo del hardware adquirido, puede probar **UNA** de las acciones siguientes:

- Presione momentáneamente el botón IO0 y el código deberá empezar a descargarse a su módulo NodeMCU-32S.
- Presione y mantenga presionado el botón IO0, seguidamente presionar momentáneamente el botón de EN ó reset y por último liberar el botón IO0. Siguiendo éstos pasos el código deberá empezar a descargarse a su módulo NodeMCU-32S.

Ambos botones están ubicados a los costados del puerto microUSB de su módulo NodeMCU-32S y deberá empezar a descargar el código a su módulo NodeMCU32S.



4. Si la barra de notificación (intermedia entre la zona del código y la zona negra) del IDE cambia a naranja indica que ocurrió un error en la descarga, para corregirlo siga el procedimiento descrito al final del punto 1.3 “Errores en la descarga del código”.

... y ahora analicemos el siguiente ejemplo:

```
// Las constantes no pueden cambiarse. Usado para asignar el número de pin:
const int ledPin = LED_BUILTIN;// el número del pin donde está el LED
```

```
// Las variables cambiarán:
int ledState = HIGH; // El estado del led es ajustado

// Generalmente, debería usar el tipo "unsigned long" para variables que guardan
el tiempo
// El valor de esta variable rápidamente crecerá convirtiéndose muy grande para
ser almacenado en un tipo "int"
unsigned long previousMillis = 0; // Almacenará el último valor de tiempo
en que el led fue actualizado de estado

// constantes no cambiarán:
const long interval = 1000; // lapso de tiempo en la cual cambiará de
estado (valor en milisegundos)

void setup() {
 // ajusta el pin como salida:
 pinMode(ledPin, OUTPUT);
}

void loop() {
 // Aquí es donde Usted pone el código que necesita estar corriendo todo el
tiempo.

 // Checa si es tiempo de cambiar el estado del LED, esto es, si la
 // diferencia entre el tiempo actual y el último tiempo en que el led
 // cambió de estado es más grande que el intervalo que usted ajustó
 // previamente, cambiará de estado el LED.
 unsigned long currentMillis = millis(); // Se lee el tiempo actual y se
asigna en una variable del tipo entero de 32 bits

 // Mediante una operación aritmética, se calcula la diferencia del tiempo
transcurrido (currentMillis - previousMillis) y dicho resultado, se compara con
el tiempo establecido arbitrariamente en la variable "interval". Si la
diferencia del tiempo calculado es mayor que la establecida en la variable
"interval" se ejecutarán las sentencias que se ubican entre los corchetes.
 if (currentMillis - previousMillis >= interval) {
 // Se almacena el último valor de tiempo en que el LED cambió de estado
 previousMillis = currentMillis;

 // si el LED está apagado se encenderá y viceversa:
 if (ledState == HIGH) {
 ledState = LOW;
 } else {
 ledState = HIGH;
 }

 // ajusta el estado con el valor almacenado en la variable ledState:
 digitalWrite(ledPin, ledState);
 }
}
```

## Actividades:

1. Cambie el valor de la variable por 12, compile y descargue su código a la tarjeta de desarrollo, posteriormente cámbiela por los valores 14 y 27. En cada caso observe los efectos:

```
const int ledPin = 12;
```

2. Ahora, cambie el tiempo almacenado en la variable, pruebe con valores 500, 250 y 100. En cada caso observe los efectos:

```
const long interval = 500;
```

### 2.2.3 Máquina de estados

**Objetivo específico:** Implementará una máquina de estados usando el módulo ESP32.

Este ejemplo enseña a usar las máquinas de estado, los cuales son indispensable para el uso del stack del WiFi. Observe en el ejemplo que no se utiliza la sentencia `delay()`, adicionalmente, se observa que el valor de **una variable controla una secuencia**, o sea, un estado. De ahí proviene el nombre de **máquinas de estado**, ya que el valor de la variable `estadoLed` determina que color de LED enciende.

```
int estadoLed = 0;

estadoLed++; // Esta variable controla los estados y cada vez que se ejecuta ésta
// sentencia la variable estadoLed se incrementa en uno
// (estadoLed = estadoLed + 1;)
```

La manera óptima de manejar las secuencias de la máquina de estados es el uso de la secuencia `switch` y `case`:

```
switch (estadoLed) {
 case 0: // todos los leds apagados
 // Secuencia a ejecutar cuando estadoled = 0
 break;
 case 1: // sólo el led rojo se enciende
 // Secuencia a ejecutar cuando estadoled = 1
 break;
 case 2: // sólo el led verde se enciende
 // Secuencia a ejecutar cuando estadoled = 2
 break;
 case 3: // sólo el led azul se enciende
 // Secuencia a ejecutar cuando estadoled = 3
 break;
 default:
 // Secuencia a ejecutar cuando estadoled tiene valor diferente a los anteriores
 break;
}
```

Para realizar éste ejemplo se usaron los demos: Digital => `BlinkWithOutDelay`, Control => `IfStatementConditional` y Control => `SwitchCase`.

Partiendo de este ejemplo, desarrolle el ejemplo del semáforo visto anteriormente.

```
// Inicio del código de la máquina de estados
// constants won't change. Used here to set a pin number:
// Conecte los leds como sigue:
const int ledR = 12;// the number of the LED pin (rojo)
const int ledG = 14;// the number of the LED pin (verde)
const int ledB = 27;// the number of the LED pin (azul)

// Generally, you should use "unsigned long" for variables that hold time
// The value will quickly become too large for an int to store
unsigned long previousMillis = 0; // will store last time LED was updated

// constants won't change:
const long interval = 1000; // interval at which to blink (milliseconds)
int estadoLed = 0;

void setup() {
 // set the digital pin as output:
 pinMode(ledR, OUTPUT);
 pinMode(ledG, OUTPUT);
 pinMode(ledB, OUTPUT);
 digitalWrite(ledR, HIGH);
 digitalWrite(ledG, HIGH);
 digitalWrite(ledB, HIGH);
}

void loop() {
 // here is where you'd put code that needs to be running all the time.

 // check to see if it's time to blink the LED; that is, if the difference
 // between the current time and last time you blinked the LED is bigger than
 // the interval at which you want to blink the LED.
 unsigned long currentMillis = millis();

 if (currentMillis - previousMillis >= interval) {
 // save the last time you blinked the LED
 previousMillis = currentMillis;
 estadoLed++; // Esta variable controla los estados y cada vez que se ejecuta ésta
 // sentencia la variable estadoLed se incrementa en uno
 // (estadoLed = estadoLed + 1)
 if (estadoLed > 3) // Los valores que puede tomar la variable
 estadoLed = 1; // estadoLed son: 1, 2 y 3. (Se descarta el valor 0)

 // do something different depending on the range value:
 switch (estadoLed) {
 case 0: // todos los leds apagados
 digitalWrite(ledR, HIGH);
 digitalWrite(ledG, HIGH);
 digitalWrite(ledB, HIGH);
 break;
 case 1: // sólo el led rojo se enciende
 digitalWrite(ledR, LOW);
 digitalWrite(ledG, HIGH);
 digitalWrite(ledB, HIGH);
 break;
 case 2: // sólo el led verde se enciende
 digitalWrite(ledR, HIGH);
 digitalWrite(ledG, LOW);
 digitalWrite(ledB, HIGH);
 break;
 case 3: // sólo el led azul se enciende
 digitalWrite(ledR, HIGH);
 digitalWrite(ledG, HIGH);
 break;
 }
 }
}
```

```
 digitalWrite(ledB, LOW);
 break;
default: // Sólo se ejecuta cuando 0 > estadoLed > 3
 digitalWrite(ledR, HIGH);
 digitalWrite(ledG, HIGH);
 digitalWrite(ledB, HIGH);
 estadoLed = 0;
 break;
}

}
} //Fin del código de la maquina de estados
```

El ejemplo está disponible en: <https://github.com/gpoolb/ESP32> en la carpeta “MaquinaDeEstados” y está basado en la conexión del LED que se describe en el punto 2.2.1 de éste manual.

Descargue el ejemplo del link, compile y descargue el código a su módulo NodeMCU.

... **OJO**, No olvide que:



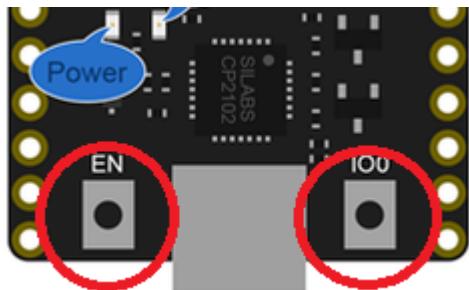
1. Con el botón  se verifica e inicia la solicitud de conexión con el módulo para la descarga del código a la tarjeta de desarrollo.
2. En algunos modelos de NodeMCU-32S no se requiere acción adicional para que el código empiece a descargarse, así que, no se mostrará la solicitud de conexión. El código se descargaría automáticamente.
3. En otros modelos de hardware, mostrará una solicitud de conexión. En la parte inferior del IDE se mostrará la leyenda “conectando” así como se muestra en la siguiente figura:



... dependiendo del hardware adquirido, puede probar **UNA** de las acciones siguientes:

- a) Presione momentáneamente el botón IO0 y el código deberá empezar a descargarse a su módulo NodeMCU-32S.
- b) Presione y mantenga presionado el botón IO0, seguidamente presionar momentáneamente el botón de EN ó reset y por último liberar el botón IO0. Siguiendo éstos pasos el código deberá empezar a descargarse a su módulo NodeMCU-32S.

Ambos botones están ubicados a los costados del puerto microUSB de su módulo NodeMCU-32S y deberá empezar a descargar el código a su módulo NodeMCU32S.



4. Si la barra de notificación (intermedia entre la zona del código y la zona negra) del IDE cambia a naranja indica que ocurrió un error en la descarga, para corregirlo siga el procedimiento descrito al final del punto 1.3 “Errores en la descarga del código”.

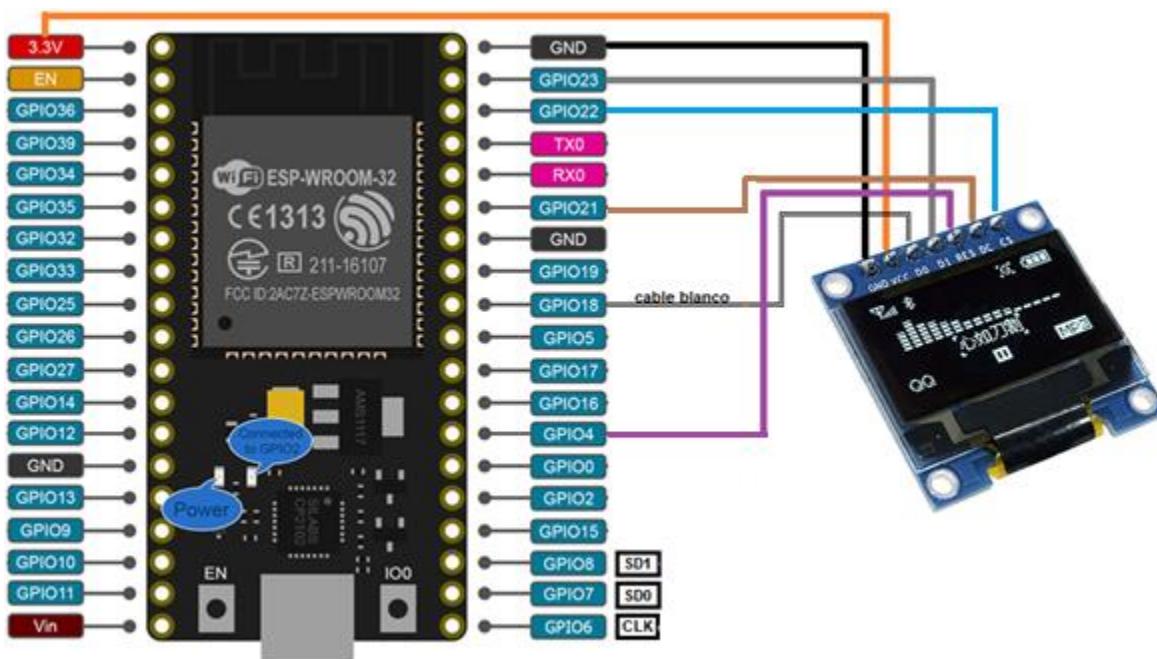
#### Actividades:

¿podría agregar los estados requeridos para que el led verde parpadee dos veces antes de cambiar al led azul?

## 2.2.4 Usando la pantalla OLED SSD1306

**Objetivo específico:** Ejecutará el ejemplo que permite verificar el funcionamiento de la pantalla OLED SSD1306 usando el módulo ESP32.

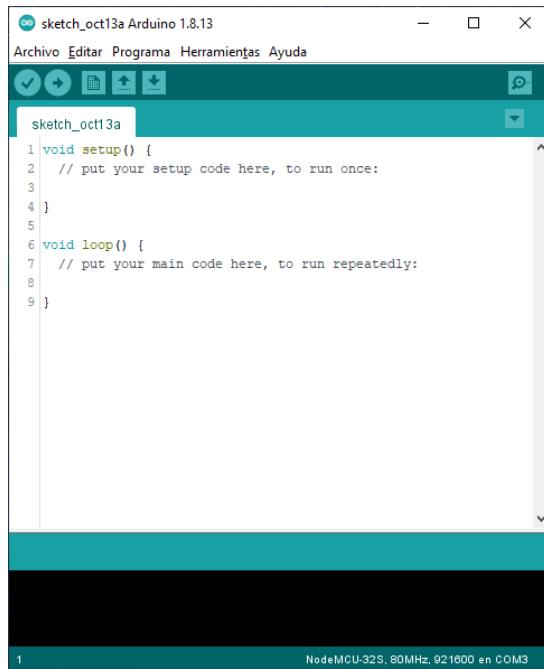
Para iniciar con este punto, se requiere a ver unas conexiones a la pantalla OLED y al módulo NodeMCU-32S así como se muestra en la pantalla:



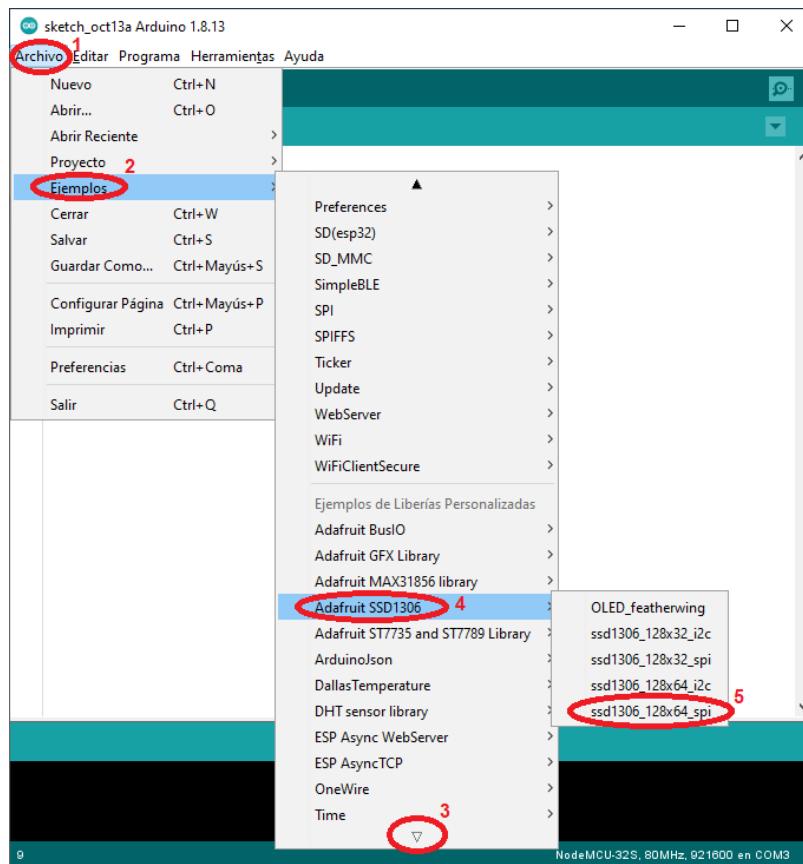
**IMPORTANTE:** Si el conexionado de la pantalla no se hace correctamente, puede dañarse irremediablemente. **Las conexiones que más debe cuidar son las de GND y VCC** (las dos primeras contando de izquierda a derecha) que corresponden a los cables **negro y naranja**.

Adicionalmente, tiene que considerar que la instalación de las bibliotecas (descritas en el punto 1.6 de este manual) ha sido completadas con éxito.

Al terminar las conexiones, se requiere abrir el IDE del Arduino como sigue:



Proceda a abrir el ejemplo siguiente la siguiente secuencia:



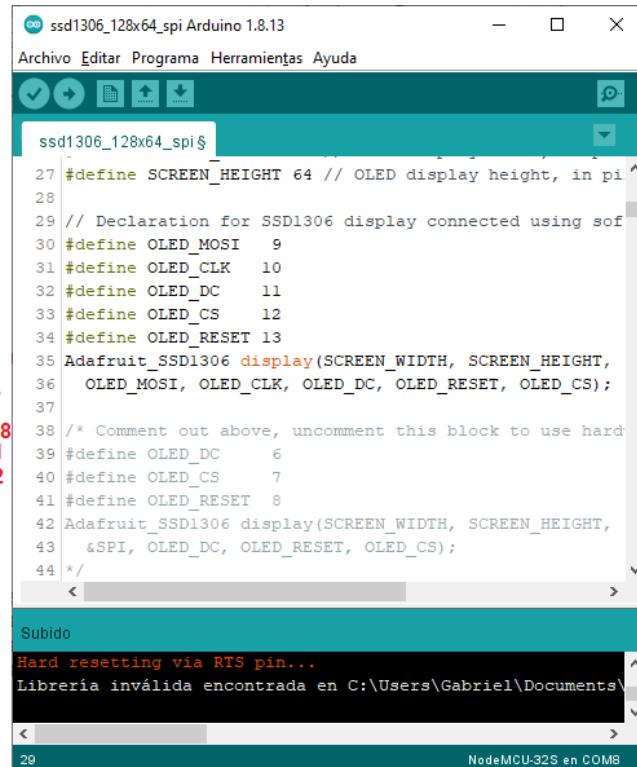
Antes de ejecutar el ejemplo, se requiere hacer unas modificaciones al código, así como se muestra a continuación:

1. Agregue /\* en la linea 28

2. Agregue \*/ en la linea 37

3. Elimine el /\* y cambiela por // en la linea 38
4. Cambie el valor de ésta definición por 21
5. Cambie el valor de ésta definición por 22
6. Cambie el valor de ésta definición por 4

7. Elimine el \*/ de la linea 44



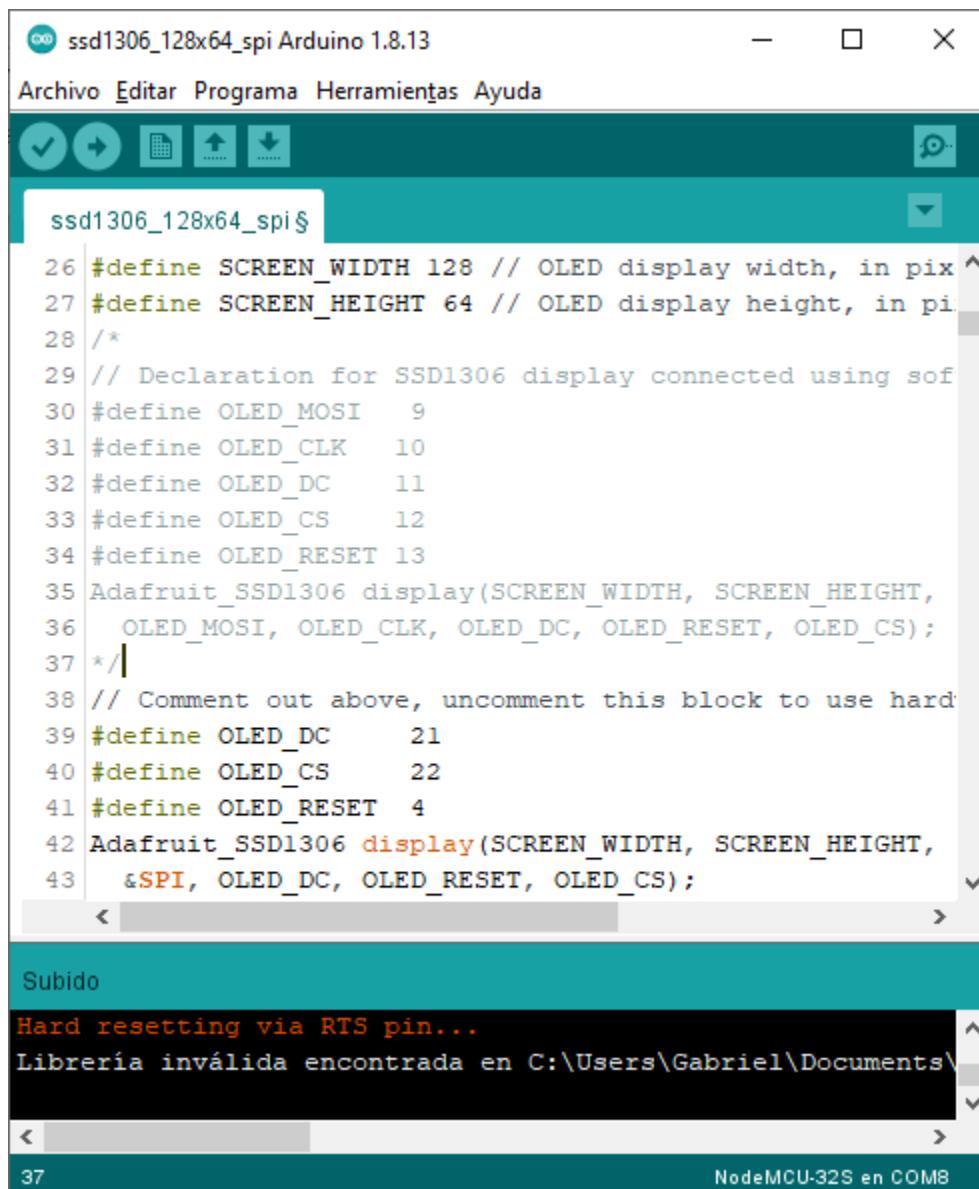
```
ssd1306_128x64_spi Arduino 1.8.13
Archivo Editar Programa Herramientas Ayuda

ssd1306_128x64_spi$

27 #define SCREEN_HEIGHT 64 // OLED display height, in pixels
28 // Declaration for SSD1306 display connected using software SPI
29 #define OLED莫斯I 9
30 #define OLEDCLK 10
31 #define OLED_DC 11
32 #define OLED_CS 12
33 #define OLED_RESET 13
34 Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT,
35 OLED莫斯I, OLEDCLK, OLED_DC, OLED_RESET, OLED_CS);
36
37 /* Comment out above, uncomment this block to use hardware SPI
38 #define OLED_DC 6
39 #define OLED_CS 7
40 #define OLED_RESET 8
41 Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT,
42 &SPI, OLED_DC, OLED_RESET, OLED_CS);
43 */
44 */

Subido
Hard resetting via RTS pin...
Librería inválida encontrada en C:\Users\Gabriel\Documents\Arduino\libraries\Adafruit_SSD1306\SSD1306.h:1:2: fatal error: 'Adafruit_GFX.h' file not found
#include "Adafruit_GFX.h"
 ^~~~~~
compilation terminated.
```

El código debe quedar, como se muestra a continuación:



The screenshot shows the Arduino IDE interface. The top menu bar includes Archivo, Editar, Programa, Herramientas, and Ayuda. Below the menu is a toolbar with icons for upload, download, and other functions. The main code editor window has the title "ssd1306\_128x64\_spi Arduino 1.8.13". It contains the following code:

```
ssd1306_128x64_spi$
26 #define SCREEN_WIDTH 128 // OLED display width, in pixels
27 #define SCREEN_HEIGHT 64 // OLED display height, in pixels
28 /*
29 // Declaration for SSD1306 display connected using software SPI
30 #define OLED_MOSI 9
31 #define OLED_CLK 10
32 #define OLED_DC 11
33 #define OLED_CS 12
34 #define OLED_RESET 13
35 Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT,
36 OLED_MOSI, OLED_CLK, OLED_DC, OLED_RESET, OLED_CS);
37 */
38 // Comment out above, uncomment this block to use hardware SPI
39 #define OLED_DC 21
40 #define OLED_CS 22
41 #define OLED_RESET 4
42 Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT,
43 &SPI, OLED_DC, OLED_RESET, OLED_CS);
```

The bottom half of the screen shows a terminal window with the following output:

```
Subido
Hard resetting via RTS pin...
Librería inválida encontrada en C:\Users\Gabriel\Documents\
```

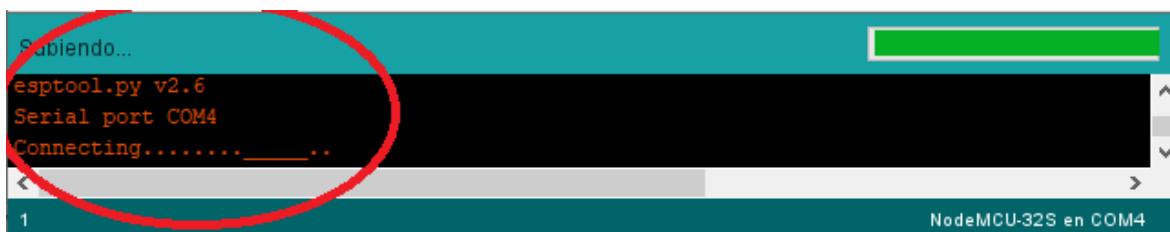
The status bar at the bottom right indicates "NodeMCU-32S en COM8".

Compile y descargue a su tarjeta de desarrollo.

... OJO, No olvide que:

1. Con el botón  se verifica e inicia la solicitud de conexión con el módulo para la descarga del código a la tarjeta de desarrollo.
2. En algunos modelos de NodeMCU-32S no se requiere acción adicional para que el código empiece a descargarse, así que, no se mostrará la solicitud de conexión. El código se descargará automáticamente.
3. En otros modelos de hardware, mostrará una solicitud de conexión. En la parte

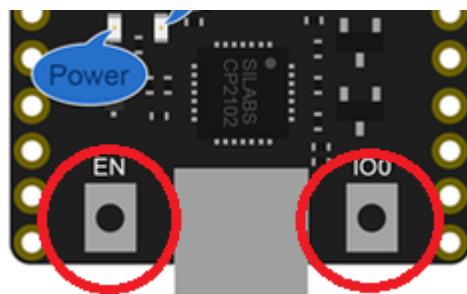
inferior del IDE se mostrará la leyenda “conectando” así como se muestra en la siguiente figura:



... dependiendo del hardware adquirido, puede probar **UNA** de las acciones siguientes:

- Presione momentáneamente el botón IO0 y el código deberá empezar a descargarse a su módulo NodeMCU-32S.
- Presione y mantenga presionado el botón IO0, seguidamente presionar momentáneamente el botón de EN ó reset y por último liberar el botón IO0. Siguiendo éstos pasos el código deberá empezar a descargarse a su módulo NodeMCU-32S.

Ambos botones están ubicados a los costados del puerto microUSB de su módulo NodeMCU-32S y deberá empezar a descargar el código a su módulo NodeMCU32S.



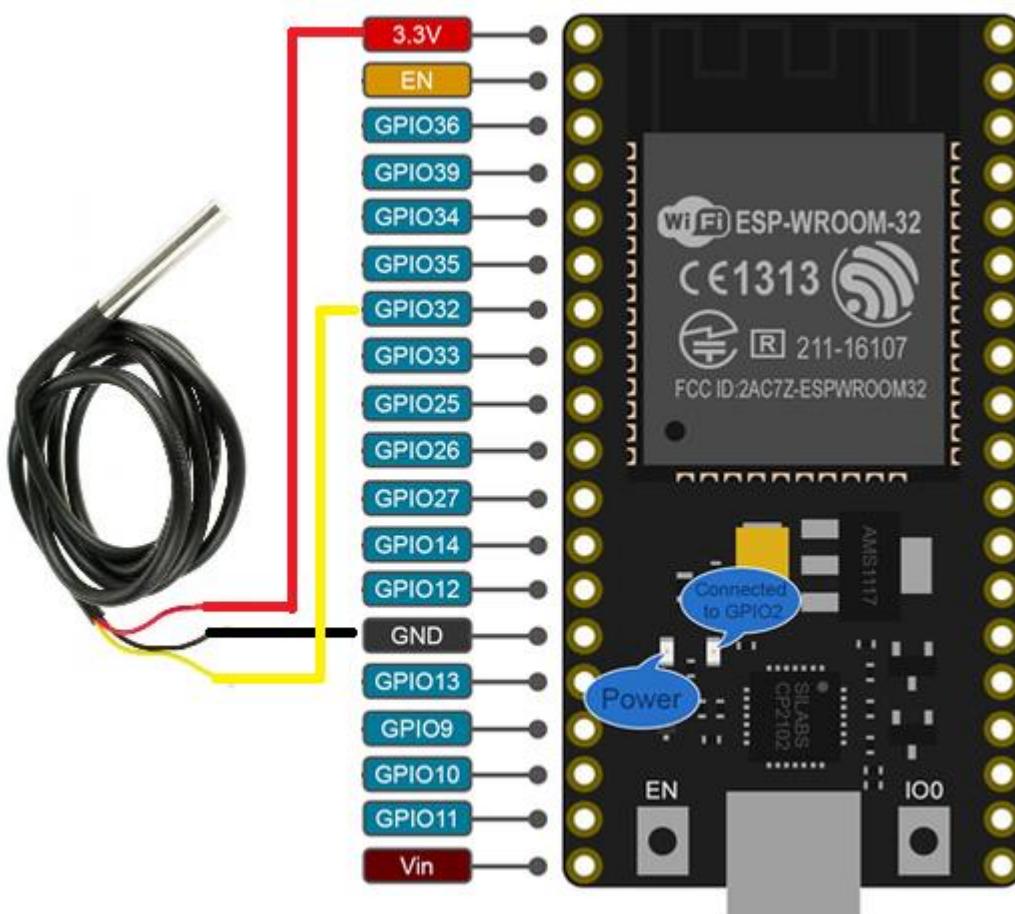
- Si la barra de notificación (intermedia entre la zona del código y la zona negra) del IDE cambia a naranja indica que ocurrió un error en la descarga, para corregirlo siga el procedimiento descrito al final del punto 1.3 “Errores en la descarga del código”.

¿Qué observa en la pantalla?

## 2.2.5 Usando el sensor de temperatura DS18B20 de fabricado por dallas semiconductor

**Objetivo específico:** Ejecutará el ejemplo que permite verificar el funcionamiento del sensor de temperatura usando el módulo ESP32.

Antes de iniciar, se requiere del conexionado del sensor como se muestra en la figura:



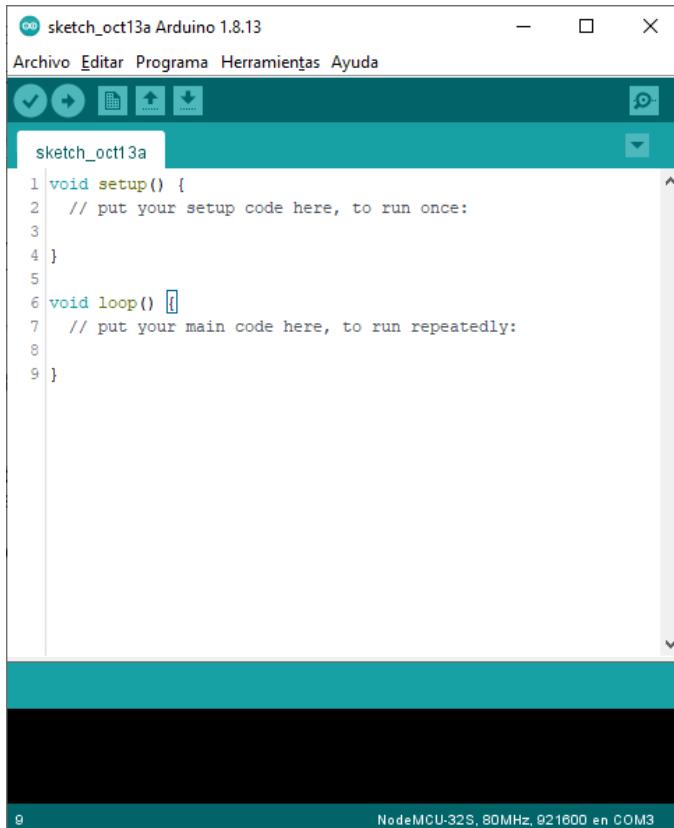
### IMPORTANTE:

**USE EL SENSOR PROPORCIONADO EN EL CURSO,** ya que contiene una resistencia de 4.7 Kohms conectado entre los cables rojo y amarillo, los cuales son necesarios para su funcionamiento.

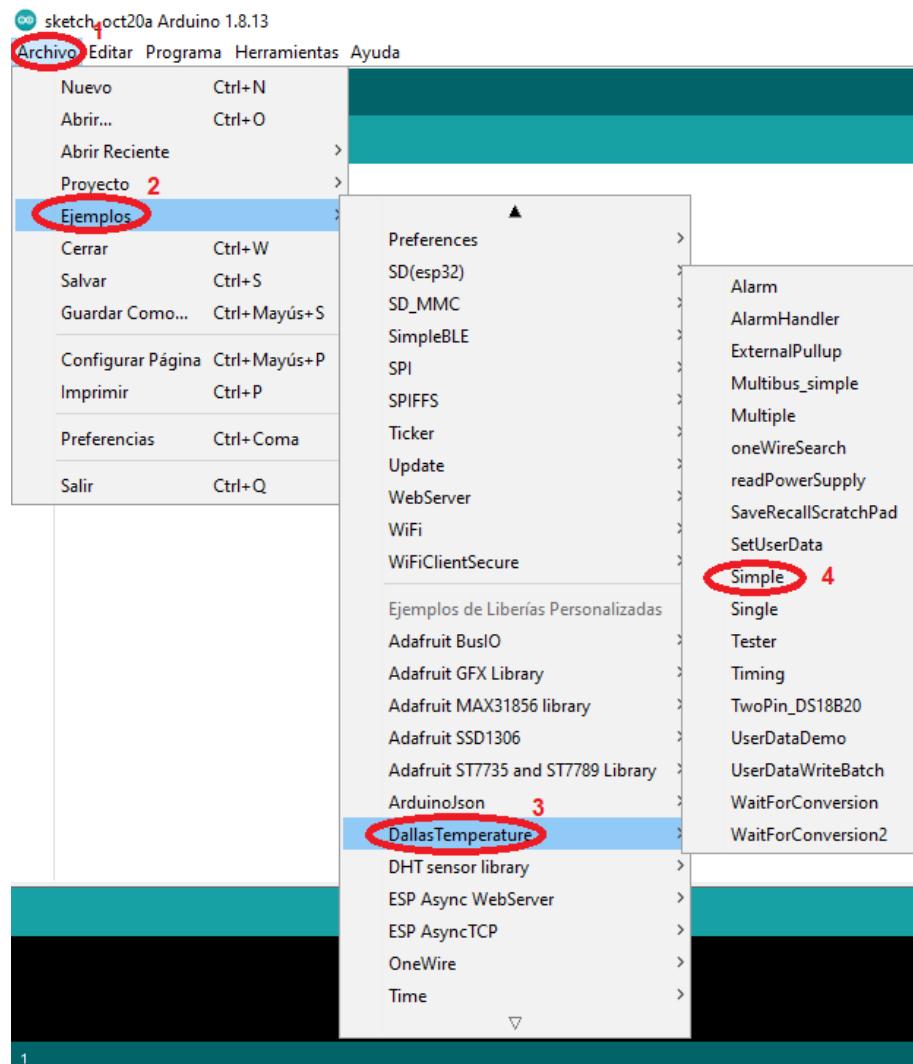
**El dispositivo PUEDE DAÑARSE si es conectado de manera incorrecta.** Los pines críticos son el GND y 3.3V (cable Negro y rojo)

Adicionalmente, tiene que considerar que la instalación de las bibliotecas (descritas en el punto 1.6 de este manual) ha sido completadas con éxito.

Seguidamente, se requiere abrir el IDE del Arduino como sigue:



Ahora siga la secuencia marcada como se muestra a continuación:



Ubique la línea 6 del código:

```

4
5 // Data wire is plugged into port 2 on the Arduino
6 #define ONE_WIRE_BUS 2
7

```

... y cámbiela como sigue:

```
#define ONE_WIRE_BUS 32
```

Compile y descargue su código a la tarjeta de desarrollo.

**... OJO,** No olvide que:

1. Con el botón  se verifica e inicia la solicitud de conexión con el módulo para

la descarga del código a la tarjeta de desarrollo.

2. En algunos modelos de NodeMCU-32S no se requiere acción adicional para que el código empiece a descargarse, así que, no se mostrará la solicitud de conexión. El código se descargará automáticamente.

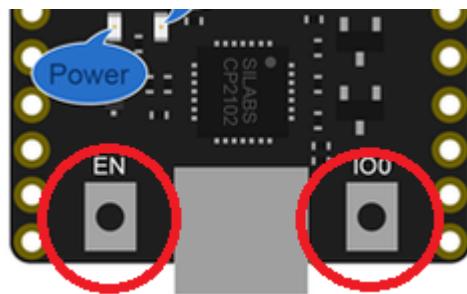
3. En otros modelos de hardware, mostrará una solicitud de conexión. En la parte inferior del IDE se mostrará la leyenda “conectando” así como se muestra en la siguiente figura:



... dependiendo del hardware adquirido, puede probar **UNA** de las acciones siguientes:

- Presione momentáneamente el botón IO0 y el código deberá empezar a descargarse a su módulo NodeMCU-32S.
- Presione y mantenga presionado el botón IO0, seguidamente presionar momentáneamente el botón de EN ó reset y por último liberar el botón IO0. Siguiendo éstos pasos el código deberá empezar a descargarse a su módulo NodeMCU-32S.

Ambos botones están ubicados a los costados del puerto microUSB de su módulo NodeMCU-32S y deberá empezar a descargar el código a su módulo NodeMCU32S.



4. Si la barra de notificación (intermedia entre la zona del código y la zona negra) del IDE cambia a naranja indica que ocurrió un error en la descarga, para corregirlo siga el procedimiento descrito al final del punto 1.3 “Errores en la descarga del código”.

Una vez descargado en su tarjeta de desarrollo se requiere abrir el monitor serial y ajustar el baud rate tal como se describe en el punto 1.5 de este manual:

Observe y memorice el valor señalado a continuación:

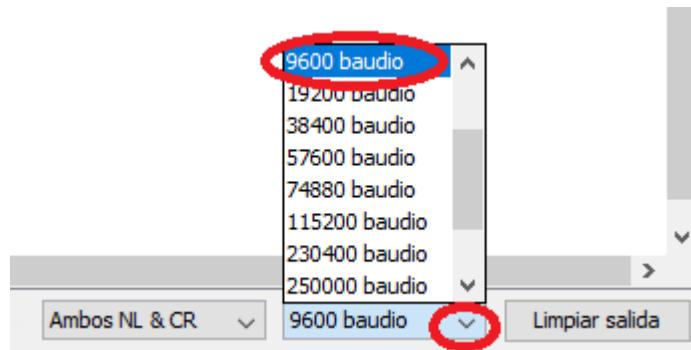
```
17 void setup(void)
18 {
19 // start serial port
20 Serial.begin(9600);
```



Abra el monitor del puerto serie presionando el botón marcado en la siguiente figura:



Ajuste los baudios como se indica a continuación:



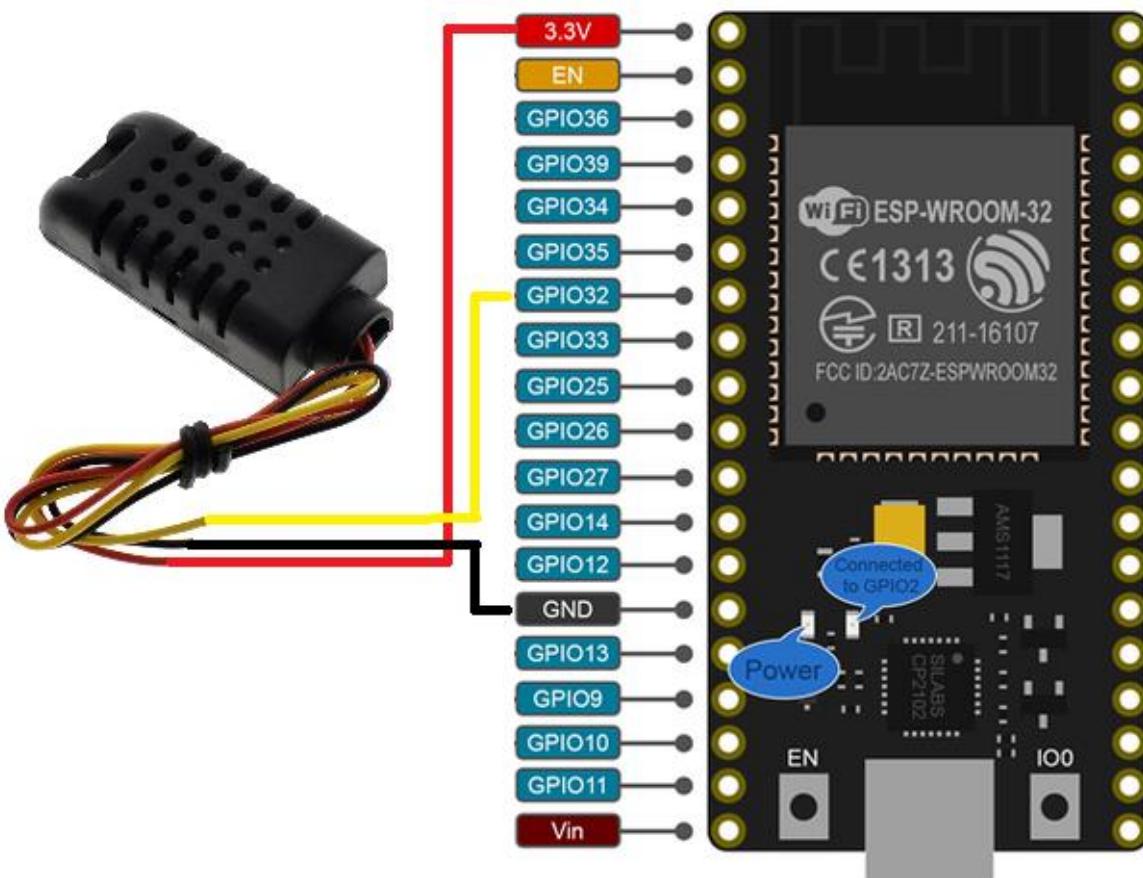
Ahora se observa el monitor de puerto serie mostrando los datos del sensor:

¿Qué datos se aprecian?  
¿Puede ubicar el dato de la temperatura?

## 2.2.6 Usando el sensor de temperatura DHT22 (AM2301) fabricado por AMLOGIC

**Objetivo específico:** Ejecutará el ejemplo que permite verificar el funcionamiento del sensor de temperatura usando el módulo ESP32.

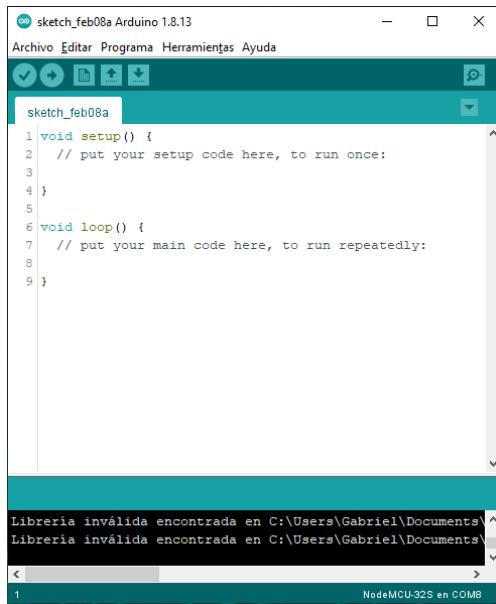
Antes de iniciar, se requiere del conexionado del sensor como se muestra en la figura:



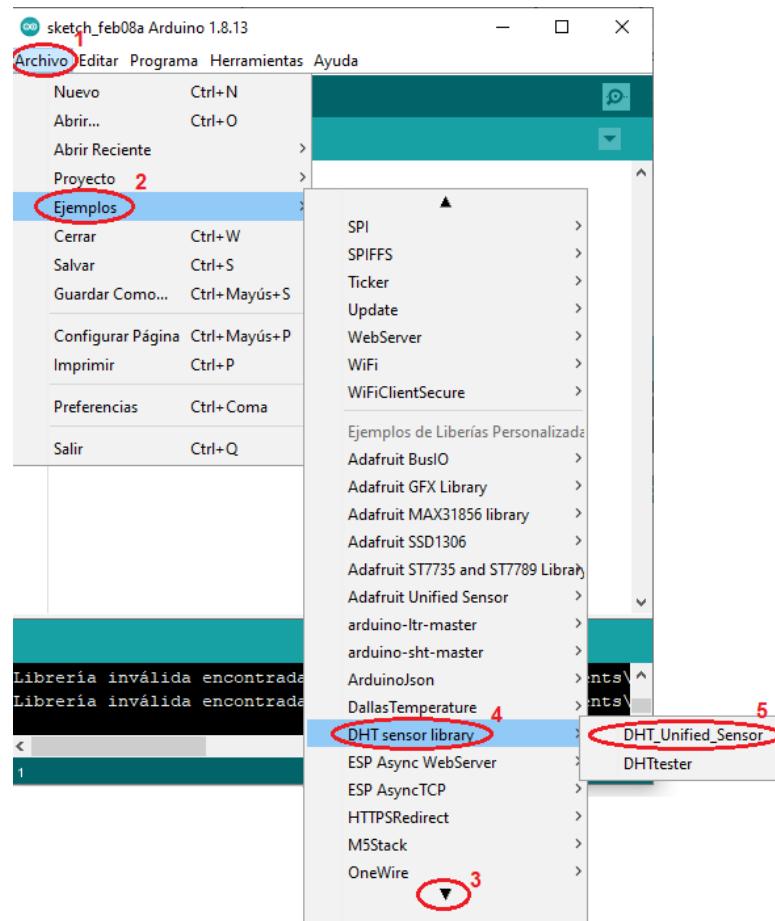
**IMPORTANTE:** El dispositivo puede dañarse si es conectado de manera incorrecta. Los pines críticos son el GND y 3.3V (cable Negro y rojo)

Adicionalmente, tiene que considerar que la instalación de las bibliotecas (descritas en el punto 1.6 de este manual) ha sido completadas con éxito.

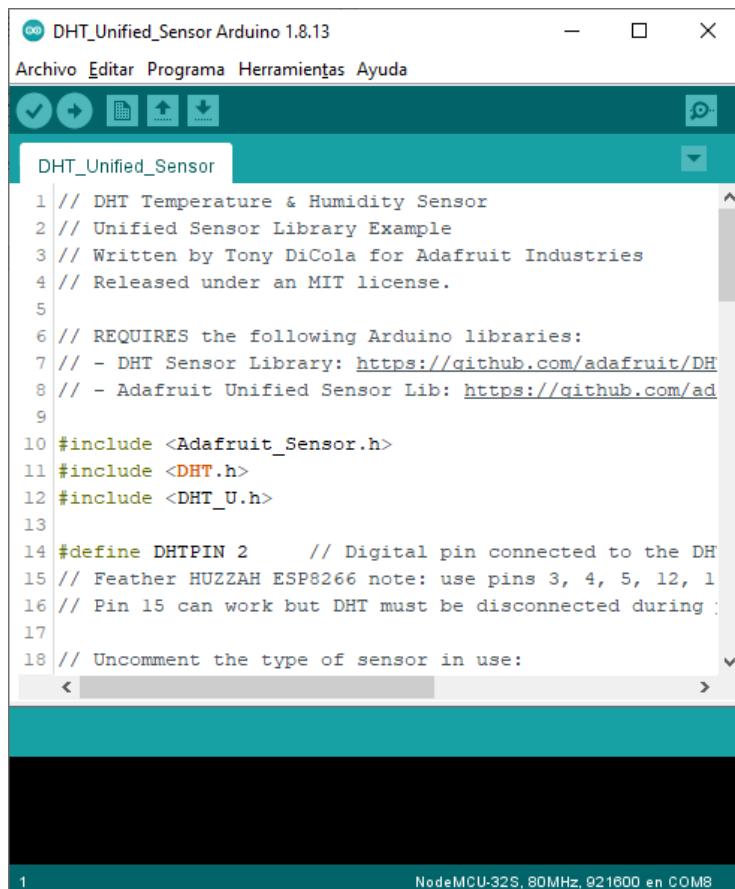
Seguidamente, se requiere abrir el IDE del Arduino como sigue:



Presione en la secuencia indicada los siguientes menús:



... y se abrirá el ejemplo siguiente:



The screenshot shows the Arduino IDE interface with the title bar "DHT\_Unified\_Sensor Arduino 1.8.13". The menu bar includes "Archivo", "Editar", "Programa", "Herramientas", and "Ayuda". Below the menu is a toolbar with icons for file operations like Open, Save, and Print. The main window displays the "DHT\_Unified\_Sensor" example code. The code is a C++ program that includes headers for Adafruit\_Sensor.h, DHT.h, and DHT\_U.h. It defines a digital pin (DHTPIN) connected to the DHT sensor. The code also includes comments about the required Arduino libraries and the type of sensor in use. At the bottom of the code editor, there is a status bar showing "NodeMCU-32S, 80MHz, 921600 en COM8".

Ubique la línea 14 y cámbiela como sigue:

```
#define DHTPIN 32 // Digital pin connected to the DHT sensor
```

Compile y descargue su código a la tarjeta de desarrollo.

... OJO, No olvide que:



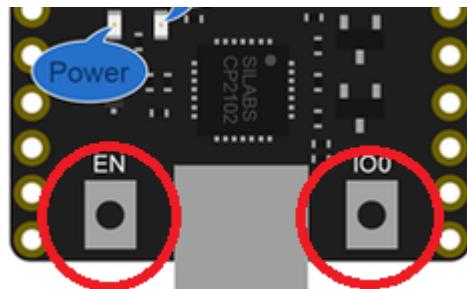
1. Con el botón  se verifica e inicia la solicitud de conexión con el módulo para la descarga del código a la tarjeta de desarrollo.
2. En algunos modelos de NodeMCU-32S no se requiere acción adicional para que el código empiece a descargarse, así que, no se mostrará la solicitud de conexión. El código se descargará automáticamente.
3. En otros modelos de hardware, mostrará una solicitud de conexión. En la parte inferior del IDE se mostrará la leyenda “conectando” así como se muestra en la siguiente figura:



... dependiendo del hardware adquirido, puede probar **UNA** de las acciones siguientes:

- Presione momentáneamente el botón IO0 y el código deberá empezar a descargarse a su módulo NodeMCU-32S.
- Presione y mantenga presionado el botón IO0, seguidamente presionar momentáneamente el botón de EN ó reset y por último liberar el botón IO0. Siguiendo éstos pasos el código deberá empezar a descargarse a su módulo NodeMCU-32S.

Ambos botones están ubicados a los costados del puerto microUSB de su módulo NodeMCU-32S y deberá empezar a descargar el código a su módulo NodeMCU32S.



- Si la barra de notificación (intermedia entre la zona del código y la zona negra) del IDE cambia a naranja indica que ocurrió un error en la descarga, para corregirlo siga el procedimiento descrito al final del punto 1.3 “Errores en la descarga del código”.

Una vez descargado en su tarjeta de desarrollo se requiere abrir el monitor serial y ajustar el baud rate tal como se describe en el punto 1.5 de este manual:

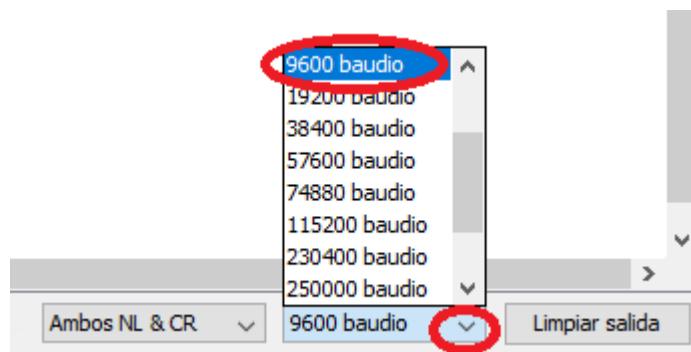
Observe y memorice el valor señalado a continuación:

```
30 void setup() {
31 Serial.begin(9600);
```

Abra el monitor del puerto serie presionando el botón marcado en la siguiente figura:



Ajuste los baudios como se indica a continuación:



Ahora se observa el monitor de puerto serie mostrando los datos del sensor:

- ¿Qué datos se aprecian?
- ¿Puede ubicar el dato de la temperatura?
- ¿Puede ubicar el dato de la humedad?

## 2.2.7 Mostrando los valores del sensor en la pantalla del SSD1306

**Objetivo específico:** Ejecutará y analizará el ejemplo que permite mostrar los valores del sensor de temperatura DHT22 (AM2301) usando el módulo ESP32.

Antes de iniciar, se requiere descargar el ejemplo del sitio: <https://github.com/gpoolb/ESP32> en la carpeta “ssd1306\_con\_DHT22”. Dicho ejemplo contiene la integración de ambos Hardware basado en los puntos 2.2.4 y 2.2.6 de este manual. Revise los comentarios, ya que, ahí se detalla paso a paso el desarrollo del programa. Lo más relevante a revisar es esta instrucción:

```
// Se muestra el nombre del sensor en la parte superior de la pantalla
// display.fillRect(CoordEjeX, CoordEjeY, AnchoCaracter * NumCaracter * TamanoTexto,
AlturaCaracter * TamanoTexto, Color);
/* No olvidar que el tamaño del texto standart es 5 * 7 pixeles,
* se anade un pixel adicional por la separación de caracteres
* quedando en 6 pixeles de ancho * 8 pixeles de alto
*/
display.fillRect(22, 0, 6 * 14 * 1, 8 * 1, SSD1306_BLACK); // Se borra el texto anterior de 14
caracteres (paso 1)
//display.setFont(&FreeMono9pt7b);
display.setTextSize(1); // Se elige el tamaño del texto (3X) (paso 2)
display.setTextColor(SSD1306_WHITE); // Se elige el color del texto (blanco) (paso 3)
display.setCursor(22, 0); // Se elige las coordenadas donde se coloca el texto (paso 4)

display.print("SENSOR DIGITAL"); // Se coloca en memoria el texto (paso 5)
display.display(); // Se muestra el texto en pantalla
```

... cada vez que se requiera mostrar algo en la pantalla, se necesitan esas instrucciones (No olvidar que el texto en gris son comentarios hechos por el autor y no tienen efecto en el código):

**1. Borre** el campo donde se desea mostrar la información usando la función de rectángulo relleno (fillRect), esta instrucción requiere 5 parámetros:

- La coordenada en el eje ‘X’* donde empieza la parte izquierda del rectángulo, recuerde que la pantalla es de 128 x 64 pixeles, o sea, ‘X’ puede tener un valor máximo de 127
- La coordenada en el eje ‘Y’* donde empieza la parte superior del rectángulo, recuerde que la pantalla es de 128 x 64 pixeles, o sea, ‘Y’ puede tener hasta 63

c) *El ancho del rectángulo*, cuide que este valor no exceda de 127 y se calcula mediante la siguiente ecuación:

AnchoCaracter: Se maneja una fuente de 5 (Ancho) x 7 (Alto) pixeles (px) adicionalmente, se considera 1px adicional de separación tanto en el ancho como en la altura, resumiendo, este valor es de 6.

NumCaracter: Es la cantidad de caracteres que desea mostrar en esa línea.

TamanoTexto: Es el tamaño del texto que desea mostrar (1, 2, 3, ... etc)

- La altura del rectángulo*, cuide que este valor no exceda de 63 y se calcula con

la siguiente ecuación:

AlturaCaracter, esta es la altura del carácter, si considera al punto anterior (el texto de 5x7) y su debida separación (1px) su valor es de 6.

TamanoTexto: Es el tamaño del texto que desea mostrar (1, 2, 3, ... etc)

e) Color, esta pantalla sólo tiene dos colores disponibles: SSD1306\_BLACK y SSD1306\_WHITE, por lo que, con el color black se “borra” la pantalla y con el color blanco se muestra el contenido en pantalla.

**2. Ajuste el tamaño del texto** que desea mostrar los valores podrían ser 1, 2, 3, ... etc., la función que permite hacer esto es setTextSize.

**3. Ajuste el color** del contenido que desea mostrar, recuerde sólo tiene dos colores disponibles: SSD1306\_BLACK y SSD1306\_WHITE, por lo que, con el color blanco se muestra el contenido en pantalla. La función que permite hacer esto es setTextColor. La función que permite hacer esto es setTextColor.

**4. Fije las coordenadas** donde se desea mostrar el contenido, recuerde que la pantalla es de 128 x 64 pixeles, o sea, ‘X’ puede tener un valor entre 0 ~ 127 y el eje ‘Y’ tiene valores de 0 ~ 63 (el valor ‘0,0’ representa la esquina superior izquierda del display). La función que permite hacer esto es setCursor.

**5. Ponga en el buffer** el contenido a mostrar, la función que permite hacer esto es print. Observe que también se puede utilizar variables con contenido numérico (Tipo int, long, etc.) ó texto (Tipo String).

**6. Mostrar texto** en pantalla, la función que permite hacer esto es display. La instrucción anterior, envía el comando para transferir el contenido almacenado en memoria al display.

Compile y descargue el código a su módulo NodeMCU.

... OJO, No olvide que:



1. Con el botón se verifica e inicia la solicitud de conexión con el módulo para la descarga del código a la tarjeta de desarrollo.

2. En algunos modelos de NodeMCU-32S no se requiere acción adicional para que el código empiece a descargarse, así que, no se mostrará la solicitud de conexión. El código se descargará automáticamente.

3. En otros modelos de hardware, mostrará una solicitud de conexión. En la parte inferior del IDE se mostrará la leyenda “conectando” así como se muestra en la siguiente figura:



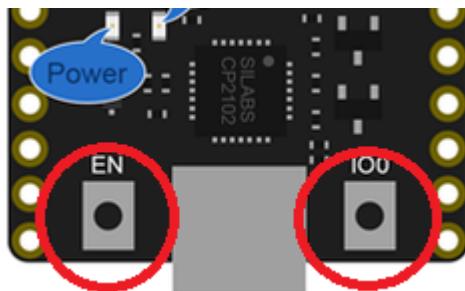
The screenshot shows a terminal window with a red circle highlighting the status bar at the bottom which reads "NodeMCU-32S en COM4". The main text area displays the following output:

```
Subiendo...
esptool.py v2.6
Serial port COM4
Connecting.....
```

... dependiendo del hardware adquirido, puede probar **UNA** de las acciones siguientes:

- Presione momentáneamente el botón IO0 y el código deberá empezar a descargarse a su módulo NodeMCU-32S.
- Presione y mantenga presionado el botón IO0, seguidamente presionar momentáneamente el botón de EN ó reset y por último liberar el botón IO0. Siguiendo éstos pasos el código deberá empezar a descargarse a su módulo NodeMCU-32S.

Ambos botones están ubicados a los costados del puerto microUSB de su módulo NodeMCU-32S y deberá empezar a descargar el código a su módulo NodeMCU32S.



- Si la barra de notificación (intermedia entre la zona del código y la zona negra) del IDE cambia a naranja indica que ocurrió un error en la descarga, para corregirlo siga el procedimiento descrito al final del punto 1.3 “Errores en la descarga del código”.

#### Actividades:

Compare entre los códigos mostrados en los ejemplos del DHT22 y SSD1306, cuáles son las partes extraídas para implementar este ejemplo.

¿Podría mostrar su nombre en la parte inferior de la pantalla?

#### Sugerencias:

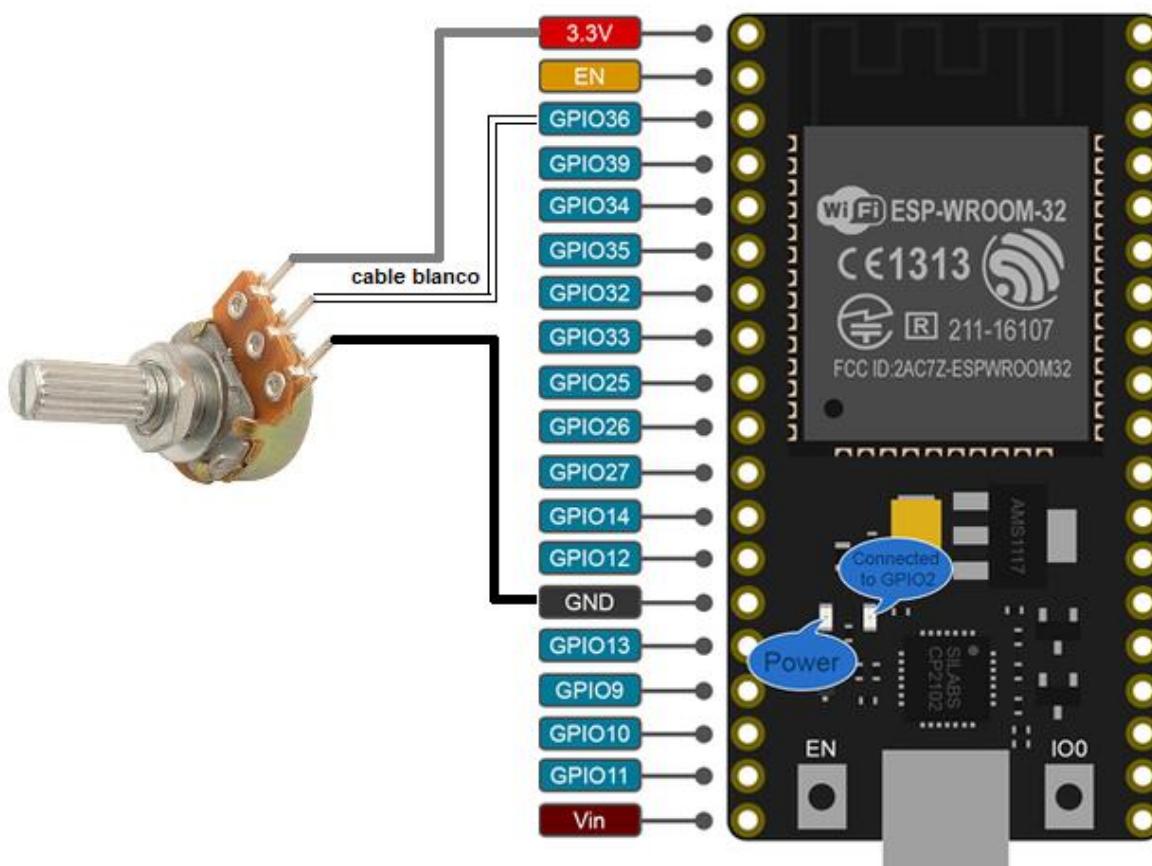
- Use la coordenada (0,55)

- b) Utilice el tamaño de texto con el valor de 1
- c) Sólo es necesario ejecutar las sentencias una vez, ya que, el nombre no cambia.
- d) No es necesario ejecutar la sentencia que borra el texto (fillRect()).

## 2.2.8 Usando el módulo Analog to Digital Converter (ADC)

**Objetivo específico:** Ejecutará el ejemplo que permite verificar el funcionamiento del ADC usando el módulo ESP32.

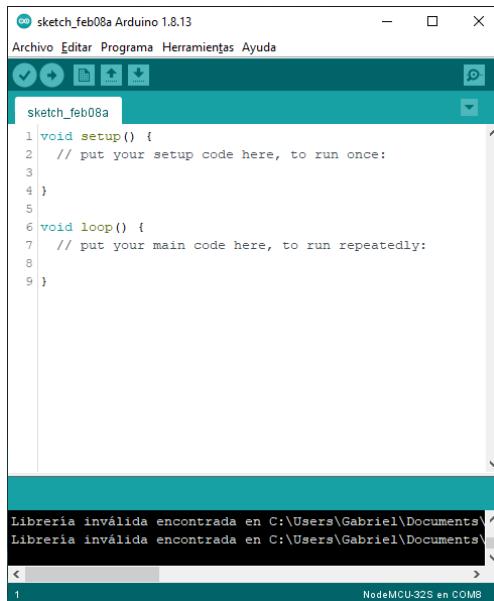
Antes de iniciar, se requiere del conexionado del potenciómetro como se muestra en la figura:



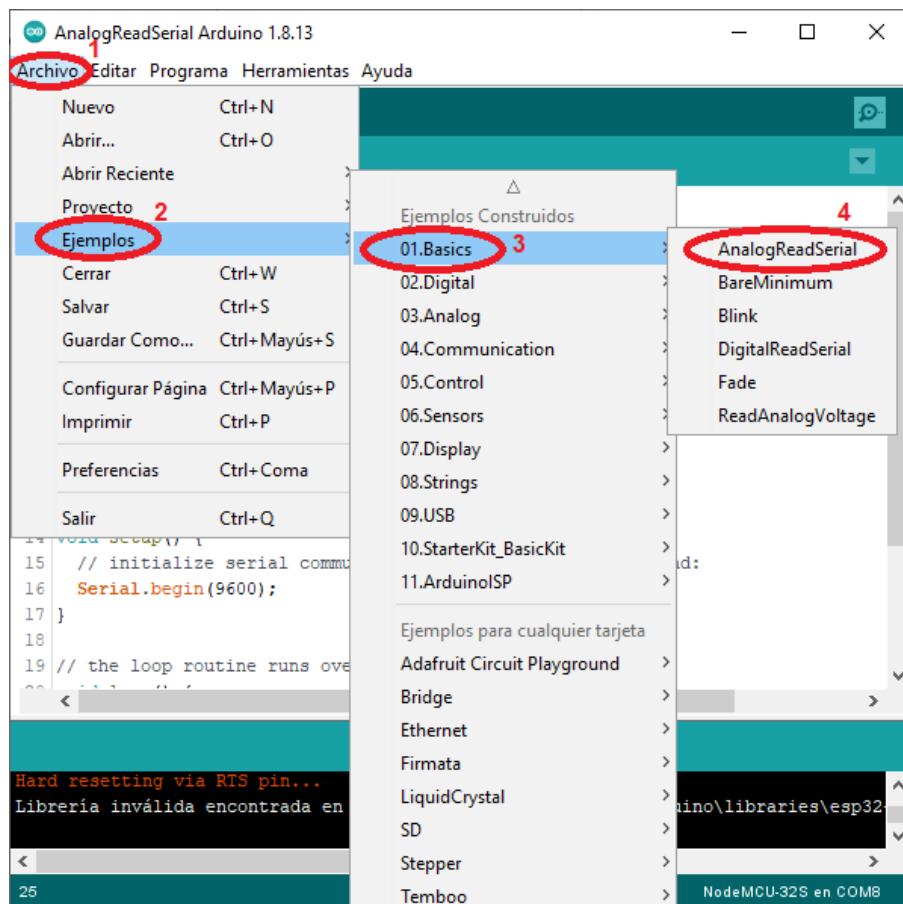
### IMPORTANTE:

**El dispositivo PUEDE DAÑARSE si es conectado de manera incorrecta.** Los pines críticos son el GND y 3.3V (cable Negro y gris).

Seguidamente, se requiere abrir el IDE del Arduino como sigue:



Presione en la secuencia indicada los siguientes menús:



Compile y descargue el código a su módulo NodeMCU.

... **OJO**, No olvide que:



1. Con el botón **se verifica e inicia la solicitud de conexión con el módulo para la descarga del código a la tarjeta de desarrollo.**
2. En algunos modelos de NodeMCU-32S no se requiere acción adicional para que el código empiece a descargarse, así que, no se mostrará la solicitud de conexión. El código se descargará automáticamente.
3. En otros modelos de hardware, mostrará una solicitud de conexión. En la parte inferior del IDE se mostrará la leyenda “conectando” así como se muestra en la siguiente figura:

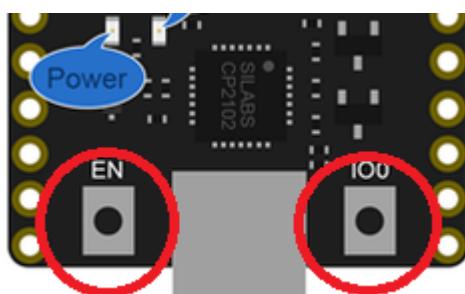


The screenshot shows the Arduino IDE's Serial Monitor window. The text "Subiendo..." is at the top, followed by "esptool.py v2.6", "Serial port COM4", and "Connecting.....". A red circle highlights the "Connecting....." text. At the bottom, it says "NodeMCU-32S en COM4".

... dependiendo del hardware adquirido, puede probar **UNA** de las acciones siguientes:

- Presione momentáneamente el botón IO0 y el código deberá empezar a descargarse a su módulo NodeMCU-32S.
- Presione y mantenga presionado el botón IO0, seguidamente presionar momentáneamente el botón de EN ó reset y por último liberar el botón IO0. Siguiendo éstos pasos el código deberá empezar a descargarse a su módulo NodeMCU-32S.

Ambos botones están ubicados a los costados del puerto microUSB de su módulo NodeMCU-32S y deberá empezar a descargar el código a su módulo NodeMCU32S.



- Si la barra de notificación (intermedia entre la zona del código y la zona negra)

del IDE cambia a naranja indica que ocurrió un error en la descarga, para corregirlo siga el procedimiento descrito al final del punto 1.3 “Errores en la descarga del código”.

Una vez descargado en su tarjeta de desarrollo se requiere abrir el monitor serial y ajustar el baud rate tal como se describe en el punto 1.5 de este manual:

Observe y memorice el valor señalado a continuación:

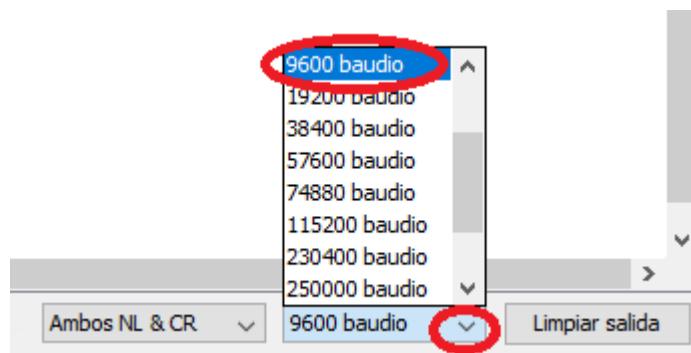
```
13 // the setup routine runs once when you press reset:
14 void setup() {
15 // initialize serial communication at 9600 bits per second:
16 Serial.begin(9600);
17 }
```



Abra el monitor del puerto serie presionando el botón marcado en la siguiente figura:



Ajuste los baudios como se indica a continuación:



Ahora se observa el monitor de puerto serie mostrando los datos del sensor:

Actividades:

Mueva el potenciómetro de izquierda a derecha y viceversa.

Observe las lecturas en el puerto serie.

Determine cuál es el valor máximo y mínimo que se obtiene al mover el

potenciómetro.

Podría mostrar esta lectura en la pantalla OLED descrito en el punto 2.2.7?

Podría condicionar que, para el primer tercio del rango del potenciómetro encienda el LedB, el segundo tercio del rango de lecturas apague el LedB y encienda el LedG y por último en el tercio de lecturas más alto encienda el LedR y apague los LedB y LedG, basándose del código descrito en el punto 2.2.3?

Sugerencia:

Declare los valores constantes como sigue:

```
// El ADC es de 12 bits por lo que el valor
// máximo es 2^12 = 4096, por lo que,
// ese valor se divide entre tres.
const int primerLimite = 1365; // 4096 * (1 / 3)
const int segundoLimite = 2730; // 4096 * (2 / 3)
```

Luego, en la rutina principal loop() puede hacer las comparaciones pertinentes:

```
if ((0 <= sensorValue) && (sensorValue < primerLimite)){
 // Encienda el ledR y apague ledG y ledB
} else if ((primerLimite <= sensorValue) && (sensorValue < segundoLimite)){
 // Encienda el ledG y apague ledR y ledB
} else if ((segundoLimite <= sensorValue) && (sensorValue < 4096)){
 // Encienda el ledB y apague ledR y ledG
}
```

Como algunas declaraciones son redundantes, se reescribe el código:

```
if (sensorValue < primerLimite){
 // 0 < sensorValue < primerLimite
 // Encienda el ledR y apague ledG y ledB
} else if ((primerLimite <= sensorValue) && (sensorValue < segundoLimite)){
 // primerLimite <= sensorValue < segundoLimite
 // Encienda el ledG y apague ledR y ledB
} else {
 // segundoLimite <= sensorValue < 4096
 // Encienda el ledB y apague ledR y ledG
}
```

Como podrá observar algunas comparaciones fueron eliminadas sin embargo, ambos segmentos de código hacen la misma función.

## **Modulo III**

### **3. Diseño de un WebServer en NodeMCU-32S.**

**Objetivo General:** Aprenderá el procedimiento para publicar una página web en la red local.

### 3.1 El Servidor Web implementado en NodeMCU-32S

En este punto se desarrollará una página web estática la cual, mostrará una gráfica con datos aleatorios y podrá manipular el estado de un led desde la página web.

#### 3.1.1 MDNS

**Objetivo específico:** Usará las herramientas existentes para el protocolo mDNS y encontrará al servidor en la red por nombre.

El sistema de nombres de dominio (o DNS, del inglés domain name system) podría considerarse una extensa guía telefónica: mediante este servicio, los usuarios introducen la dirección web en el navegador y el propio sistema identifica la dirección IP relacionada. En este proceso de resolución de nombre, el propio dispositivo envía la solicitud al servidor DNS correspondiente, donde hay una lista en la que cada nombre de equipo (es decir, la dirección web) tiene asignada la dirección IP correcta. En cambio, el multicast DNS sigue otro sistema. ¿Cómo funciona esta alternativa al DNS clásico?

El multicast DNS (mDNS) es un servicio diseñado para llevar a cabo la resolución de nombres en redes más pequeñas. Para ello, mDNS sigue un sistema distinto al conocido DNS: en lugar de enviar la solicitud a un servidor de nombres, todos los participantes de la red reciben la solicitud. El cliente correspondiente envía un multicast a la red y, de este modo, pregunta con qué integrante de la red coincide el nombre del equipo. El multicast o multidifusión es una forma especial de comunicación en la que un solo mensaje se envía a un grupo de destinatarios. El grupo puede consistir, por ejemplo, en toda la red o en una subred.

Resumiendo, es posible ubicar al servidor por nombre, pero requiere de la instalación de software adicional en la computadora como sigue:

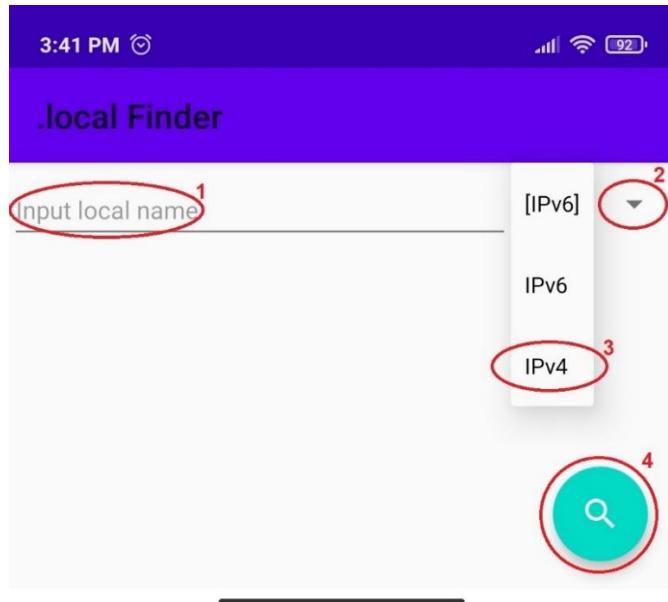
En Android se requiere instalar el programa “.local Finder” disponible en Play Store:



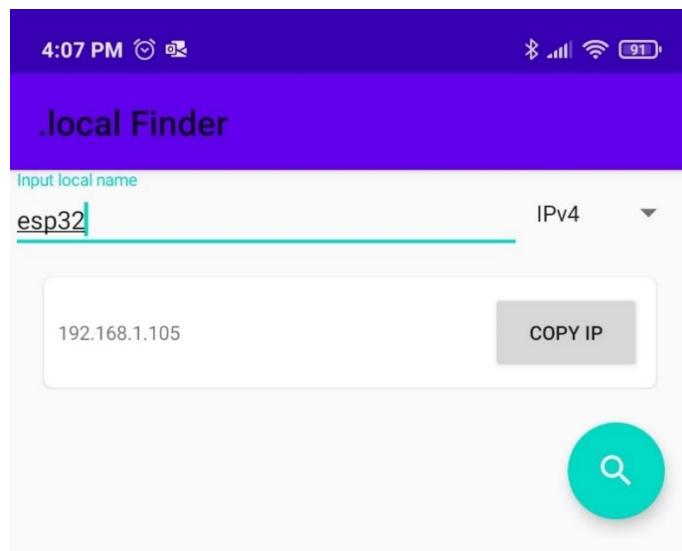
... una vez instalado el logo cambia a color azul oscuro:



Para utilizarlo, se requiere escribir el nombre en el campo: “Input local name”, seleccionar IPv4 y presionar el botón con forma de lupa:



**Suponiendo que Ud. tiene un NodeMCU funcionando como servidor** en su red local (se implementará en el punto 3.1.2), al presionar el botón con forma de lupa nos dará este resultado:



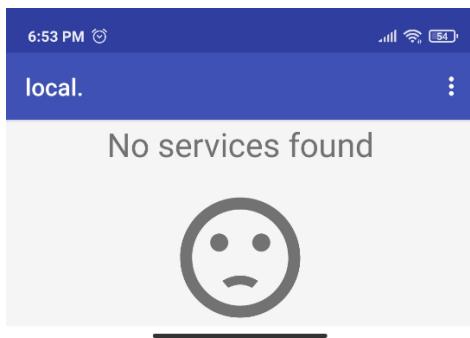
Para utilizar esa información, sólo bastaría con presionar el botón “COPY IP” y pegarlo en la barra de direcciones de su navegador web en Android.

En caso de que se active la búsqueda de Google e intente buscar su dirección web en internet, agregue a la dirección ip anterior lo siguiente: <http://> y al final de la dirección ip ponga el carácter de diagonal (/), quedando de esta manera: <http://192.168.1.105/>

Existe otro programa interesante para Android y se encarga de encontrar todos los servicios ubicados en la red, en su dispositivo Android, descargue el programa “Service Browser” ubicado en la Play Store que tiene el logo que se muestra:



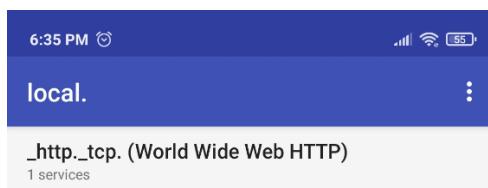
Instale, abra el programa y se deberá mostrar lo siguiente:



**Suponiendo que Ud. tiene un NodeMCU funcionando como servidor en su red local (se implementará en el punto 3.1.2), adicionalmente le ha agregado la siguiente línea al final del método setup() en su código:**

```
MDNS.addService("http", "tcp", 80);
} // fin del método setup()
```

Al abrir nuevamente el programa se mostrará lo siguiente:

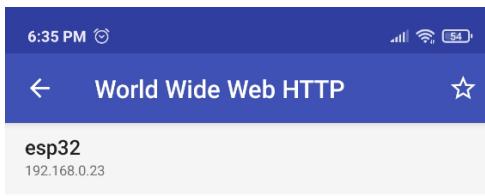


96/134

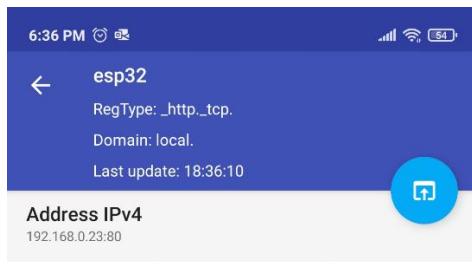
Es importante recalcar que, si se encuentra en un sitio con muchos servicios publicados en la red, le saldrá una lista en ocasiones muy extensa, para ello siempre ubique el nombre de este servicio y con el dedo, presione el área del texto mostrado en la imagen (el área se marca en un rectángulo rojo):



Y observará lo siguiente:



Nuevamente, presione el área de texto marcado con el nombre del servidor que usted trata de encontrar (en caso de salir una lista de nombres), en este caso “esp32” y se mostrará lo siguiente:

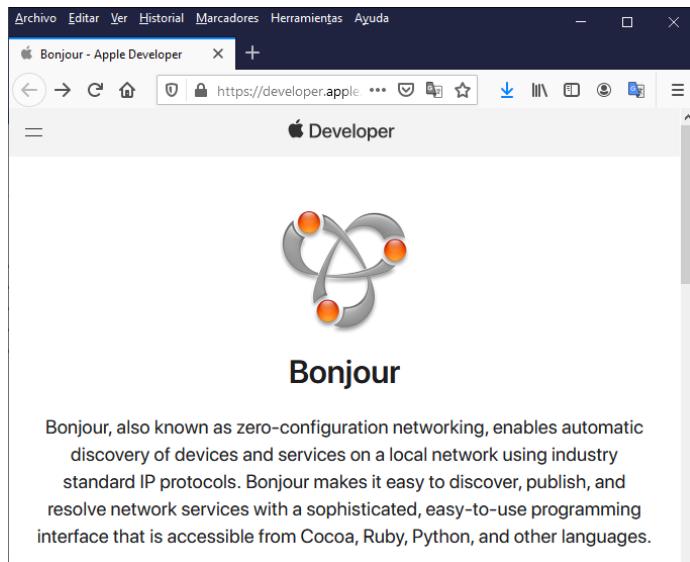


En caso de NO mostrar el botón azul, regrese un paso anterior (presionando la flecha blanca que indica hacia la izquierda) y repita el paso anterior. Ahora presione el botón azul claro con forma de circulo e incluye un cuadro con flecha hacia arriba y observará lo siguiente:

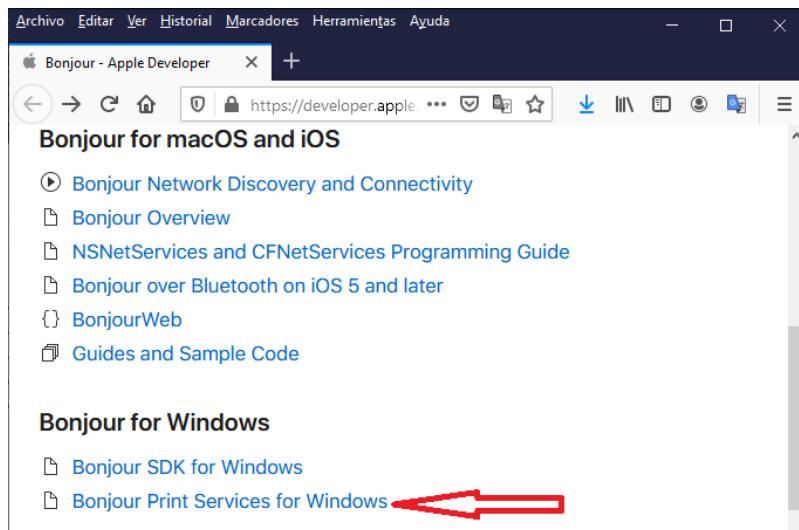


... el cual, muestra el contenido del servidor que se encuentra publicando en la red.

En **Windows** se requiere instalar el Bonjour Print Services for Windows ubicado en el sitio de apple: <https://developer.apple.com/bonjour/>:

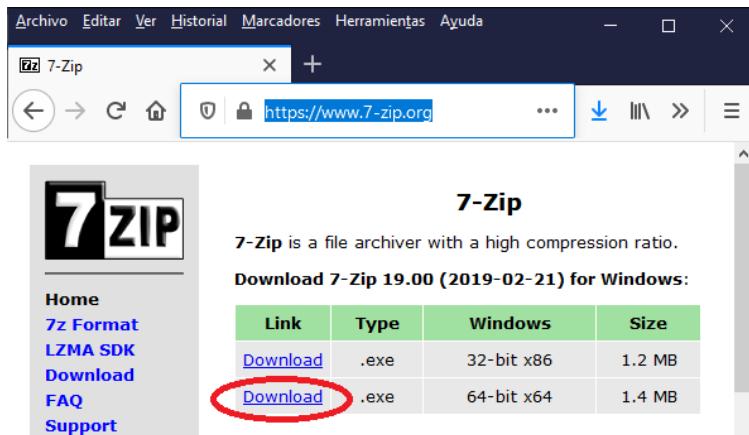


Deslice la página web hacia abajo hasta encontrar lo siguiente:



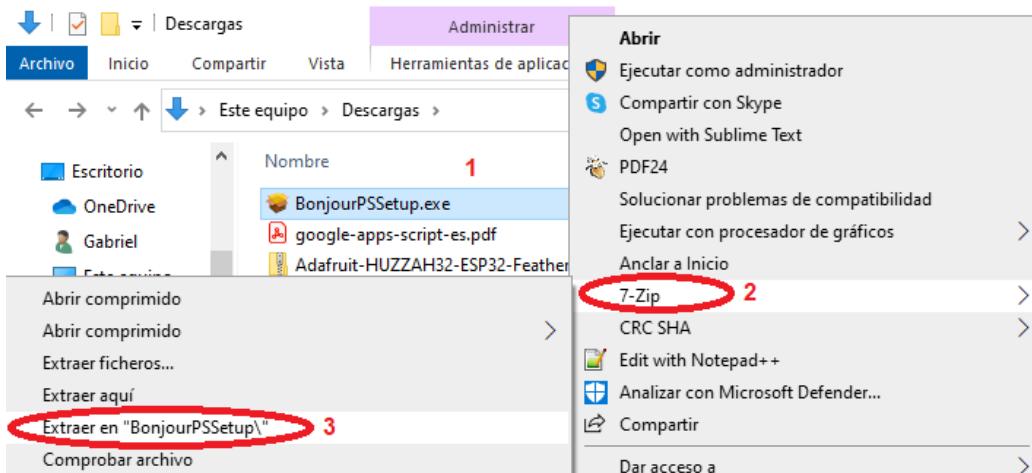
... déle un click al texto que se encuentra marcado en la figura anterior y descargue el archivo.

Para no tener que instalar todos los programas que se incluyen en el paquete, se requiere que descargue el programa 7zip en el sitio <https://www.7-zip.org/> así como sigue:



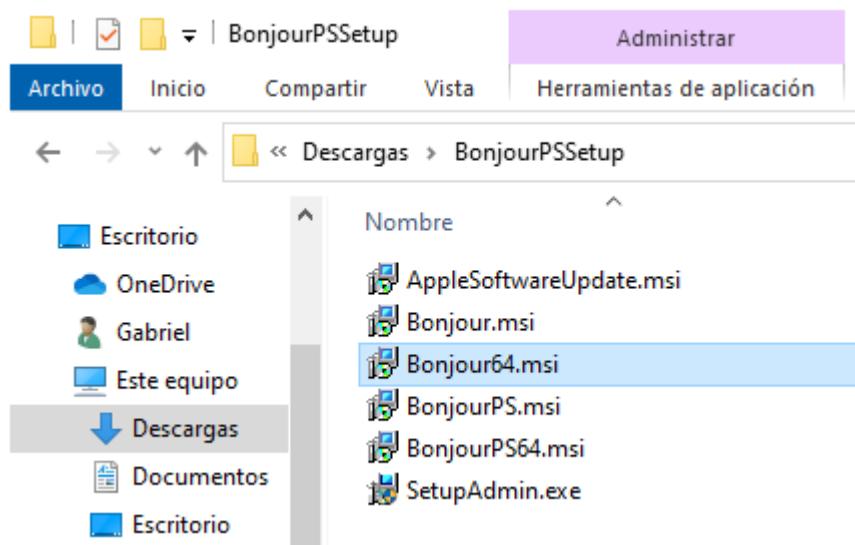
Suponiendo que su Windows sea de la versión de 64 bits descargue el programa en el link marcado en la figura anterior, si desconoce su versión de Windows instalada, de todas maneras, descargue esta versión y si su sistema operativo le informa de algún error, y no permite instalarlo, entonces pruebe con la versión de 32 bits.

Descargue e instale el 7zip en su computadora con las opciones que trae ya predeterminadas. Una vez instalado el 7zip y haber descargado el Bonjour desde el sitio web, descomprima como se muestra en la figura, presione el **botón derecho** del mouse sobre el nombre del archivo y en el menú que sale a continuación, siga la secuencia indicada en la siguiente figura:



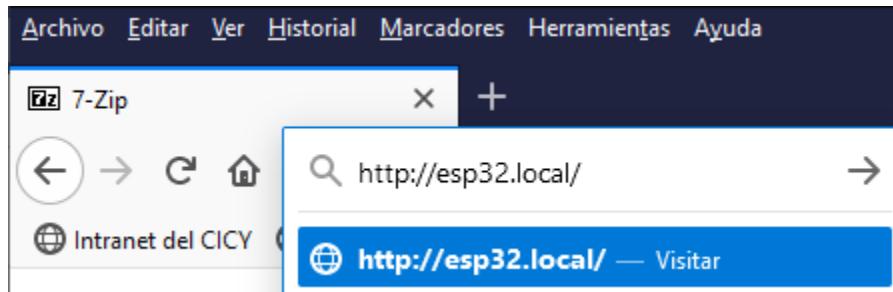
Entre a la carpeta recién creada por el 7zip en el punto anterior:

99/134



... e instale el Bonjour64.msi, en caso de que su sistema operativo le informe de algún error, y no permite instalarlo, entonces pruebe con el Bonjour.msi

**Suponiendo que Ud. tiene un NodeMCU funcionando como servidor** en su red local (se implementará en el punto 3.1.2), abra su navegador web y teclee lo siguiente:



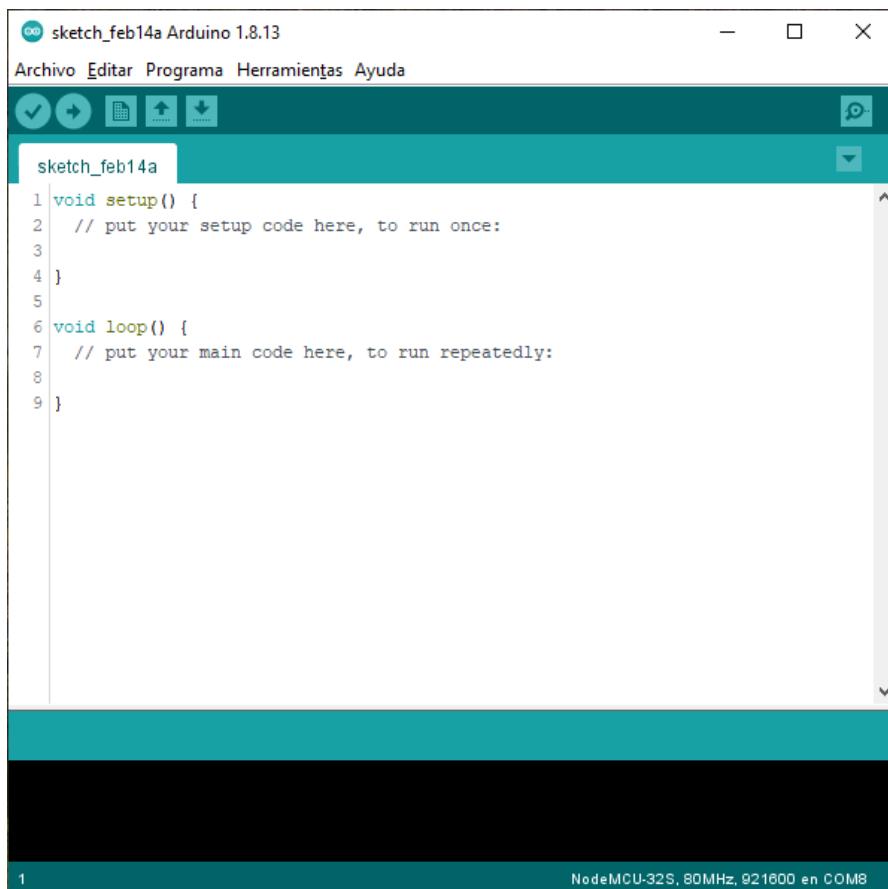
En Linux se requiere la instalación del programa llamado avahi, que, por lo general, ya lo trae de manera nativa, es decir, no se requiere instalar programa adicional.

Ahora, para poder dar uso a lo anterior, se implementará el servidor en el NodeMCU en el siguiente punto.

### 3.1.2 Advanced Web Server

**Objetivo específico:** Aprenderá a obtener los recursos configurados en un servidor implementado en un NodeMCU-32S, identificará los métodos asociado al recurso.

El primer ejemplo relevante para conocer la estructura de un servidor web implementado en el NodeMCU-32S es el que se verá a continuación, partimos del sketch en el IDE del Arduino:

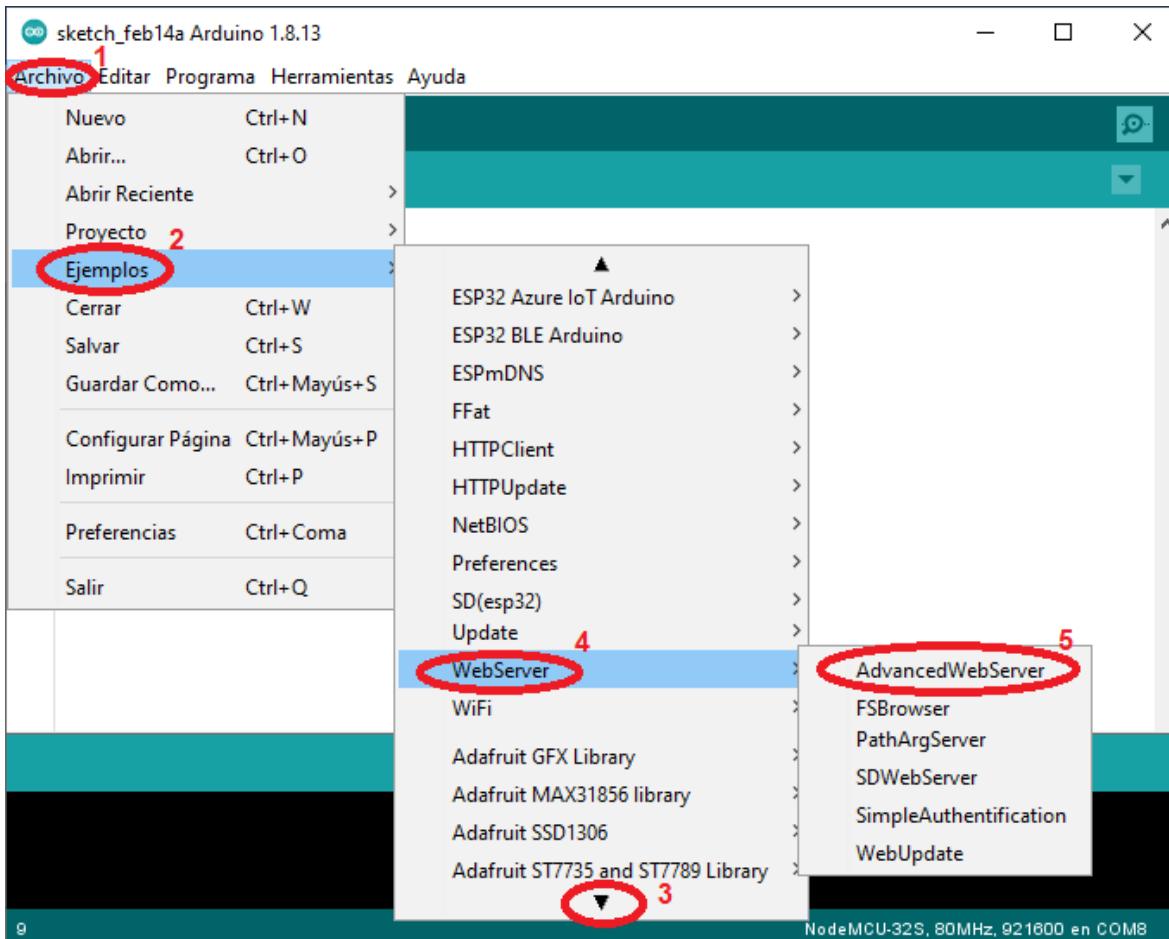


The screenshot shows the Arduino IDE interface. The title bar reads "sketch\_feb14a Arduino 1.8.13". The menu bar includes "Archivo", "Editar", "Programa", "Herramientas", and "Ayuda". Below the menu is a toolbar with icons for file operations. The main code editor window displays the following sketch:

```
1 void setup() {
2 // put your setup code here, to run once:
3
4 }
5
6 void loop() {
7 // put your main code here, to run repeatedly:
8 }
9 }
```

The status bar at the bottom indicates "NodeMCU-32S, 80MHz, 921600 en COM8".

... abra el ejemplo siguiendo la secuencia indicada en la siguiente figura:



Antes de descargar este código al NodeMCU es necesario completar las siguientes configuraciones, cambie las definiciones de las líneas indicadas a continuación:

```

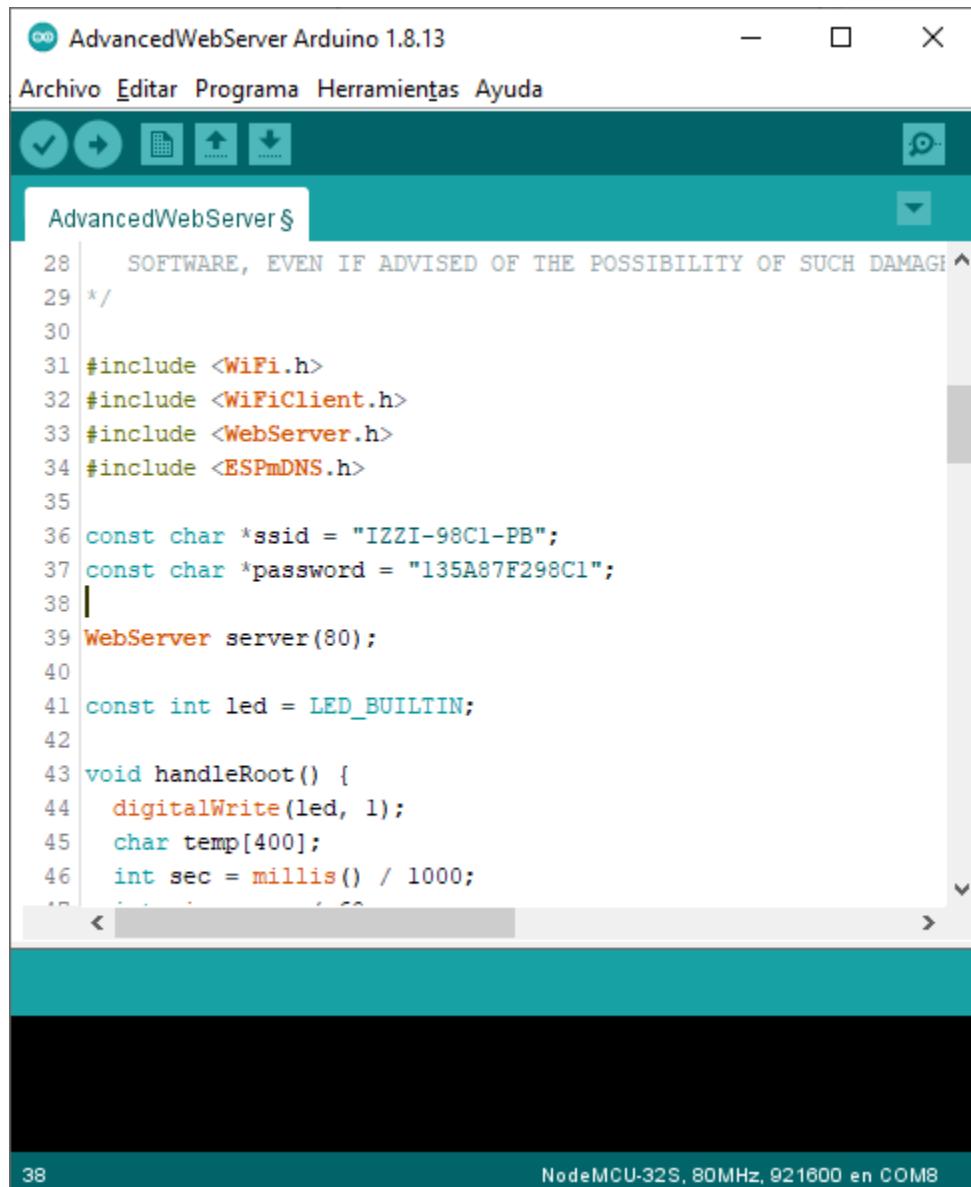
36 | const char *ssid = "YourSSIDHere"; // Establezca el nombre del WiFi (Internet)
37 | const char *password = "YourPSKHere"; // Establezca la clave del WiFi (Internet)

41 | const int led = 13; // Cambie el valor por 2 ó LED_BUILTIN
42 |

```

Compile y descargue el programa a su NodeMCU siguiendo los pasos descritos en la sección 2.2.1 a partir del sexto paso.

**ES IMPORTANTE** considerar que, si por alguna razón omite un punto y coma, alguna comilla o no respeta la estructura de las definiciones anteriores, el código **NO COMPILARA** y genera errores. Para evitar eso, respete el modo de escritura. En este caso, el código queda así:



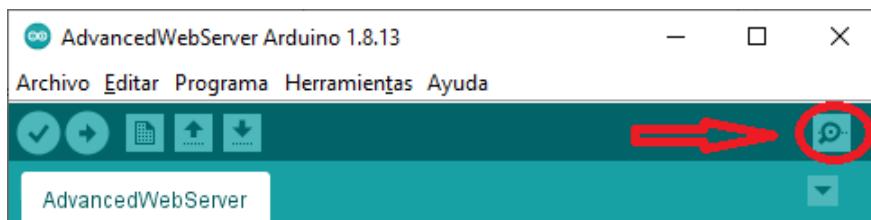
The screenshot shows the AdvancedWebServer Arduino IDE interface. The title bar reads "AdvancedWebServer Arduino 1.8.13". The menu bar includes "Archivo", "Editar", "Programa", "Herramientas", and "Ayuda". Below the menu is a toolbar with icons for file operations. The main area displays the following C++ code:

```
28 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE
29 */
30
31 #include <WiFi.h>
32 #include <WiFiClient.h>
33 #include <WebServer.h>
34 #include <ESPmDNS.h>
35
36 const char *ssid = "IZZI-98C1-PB";
37 const char *password = "135A87F298C1";
38
39 WebServer server(80);
40
41 const int led = LED_BUILTIN;
42
43 void handleRoot() {
44 digitalWrite(led, 1);
45 char temp[400];
46 int sec = millis() / 1000;
47 sprintf(temp, "HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n
48 <html>
49 <head>
50 <title>NodeMCU-32S, 80MHz, 921600 en COM8</title>
51 </head>
52 <body>
53 <h1>NodeMCU-32S, 80MHz, 921600 en COM8</h1>
54 <p>Connected to %s</p>
55 <p>IP: %s</p>
56 <p>LED: %d</p>
57 <p>Time: %d sec</p>
58 </body>
59 </html>
60 }
```

The status bar at the bottom indicates "NodeMCU-32S, 80MHz, 921600 en COM8".

... compile y descargue el código a su NodeMCU, posteriormente, al subir el código a su NodeMCU, abra el puerto serie (los pasos detallados se describen en la última sección del punto 1.5):

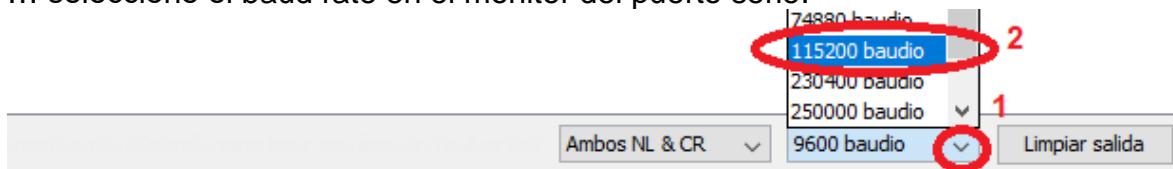
Abra el monitor del puerto serie:



... antes de abrir el monitor de puerto serie, memorice la velocidad de transferencia del puerto serie (baud rate) como sigue:

```
94 | digitalWrite(led, 0);
95 | Serial.begin(115200); ← Red arrow pointing here
96 | WiFi.mode(WIFI_STA);
```

... seleccione el baud rate en el monitor del puerto serie:



... presione el botón de RESET ó EN de su tarjeta NodeMCU y observe el monitor del puerto serial:

```
Connected to 192.168.1.105
IP address: 192.168.1.105 ← Circled with red
MDNS responder started
HTTP server started
```

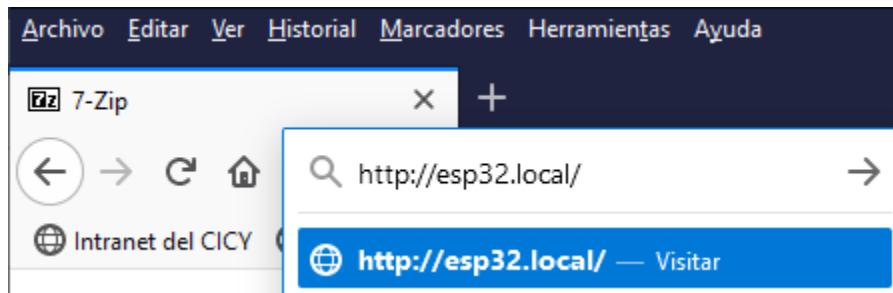
Se observa una serie de números marcadas con la leyenda “IP address”, copie el número y péguelo en su navegador de internet con el siguiente formato <http://192.168.1.105/>, así como se muestra:



.. y presione la tecla “ENTER” y espere a que el navegador muestre su contenido:



... describa lo que observa en pantalla. Haciendo uso del programa instalado en el punto anterior (3.1.1) teclee en su navegador web lo siguiente:

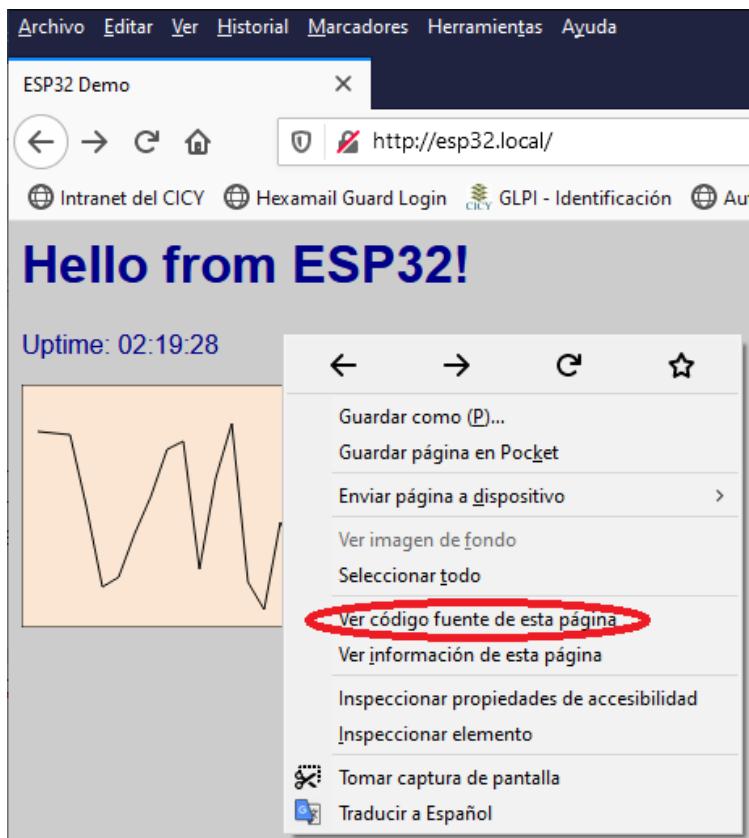


... y observe nuevamente el resultado:



A partir de este punto ya no se harán referencias a la dirección IP, ya que, en cada caso es diferente y puede prestarse a confusiones. Ahora, con las herramientas instaladas en el punto 3.1.1 se hará referencia al servidor por nombre (este no cambia en todos los casos)

Ahora presione el botón derecho del mouse en el área fuera de la gráfica y en el menú que sale a continuación seleccione “ver código fuente de la página”:



... y se mostrará lo siguiente:

```
1 <html> <head> <meta http-equiv='refresh' content='5' />
```

Esto es una línea bastante larga que contiene el texto que envía el servidor embebido en el NodeMCU-32S, para evitar eso, necesitamos agregarle al código el carácter \n a cada línea, **ES IMPORTANTE** considerar que, si por alguna razón omite un punto y coma, alguna comilla o no respeta la estructura de las definiciones anteriores, el código **NO COMPILARA** y genera errores. Para evitar eso, respete el modo de escritura. El código debe quedar, así como se muestra a continuación:

AdvancedWebServerConComentarios Arduino 1.8.13

Archivo Editar Programa Herramientas Ayuda

AdvancedWebServerConComentarios

```

43 void handleRoot() { // Método que envía la página WEB al cliente que lo solicite
44 digitalWrite(led, 1); // Enciende el Led incorporado en su módulo NodeMCU
45 char temp[400]; // Se define una variable la cual, contendrá la página WEB en forma c
46 int sec = millis() / 1000; // Se define una variable que almacene los segundos a partir
47 int min = sec / 60; // Se define una variable que almacene el cálculo de los minutos
48 int hr = min / 60; // Se define una variable que almacene el cálculo de las horas
49 sprintf(temp, 400, \
50 /******INICIA LA PAGINA WEB *****/
51 "<html>\n\
52 <head>\n\
53 <meta http-equiv='refresh' content='5' />\n\
54 <title>ESP32 Demo</title>\n\
55 <style>\n\
56 body { background-color: #cccccc; font-family: Arial, Helvetica, Sans-Serif; Color: \
57 </style>\n\
58 </head>\n\
59 <body>\n\
60 <h1>Hello from ESP32!</h1>\n\
61 <p>Uptime: %02d:%02d:%02d</p>\n\
62 \n\
63 </body>\n\
64 </html>\n",
65 /******TERMINA LA PAGINA WEB *****/
66 hr, min % 60, sec % 60 // Aquí se calcula del tiempo en que inicia la ejecución
67);
68 server.send(200, "text/html", temp); // Sentencia que envía la página web al cliente
69 digitalWrite(led, 0); // Se apaga el led incorporado en su módulo NodeMCU
70 } // Termina método que envía la página WEB al cliente que lo solicita

```

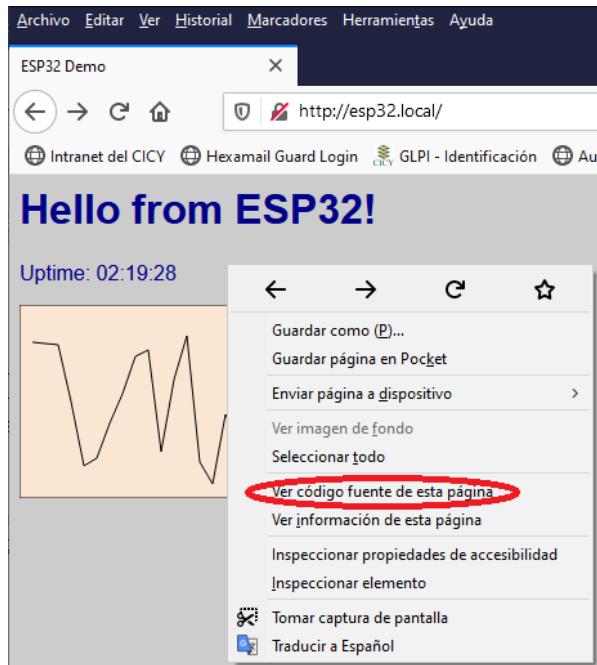
Subido

Hard resetting via RTS pin...

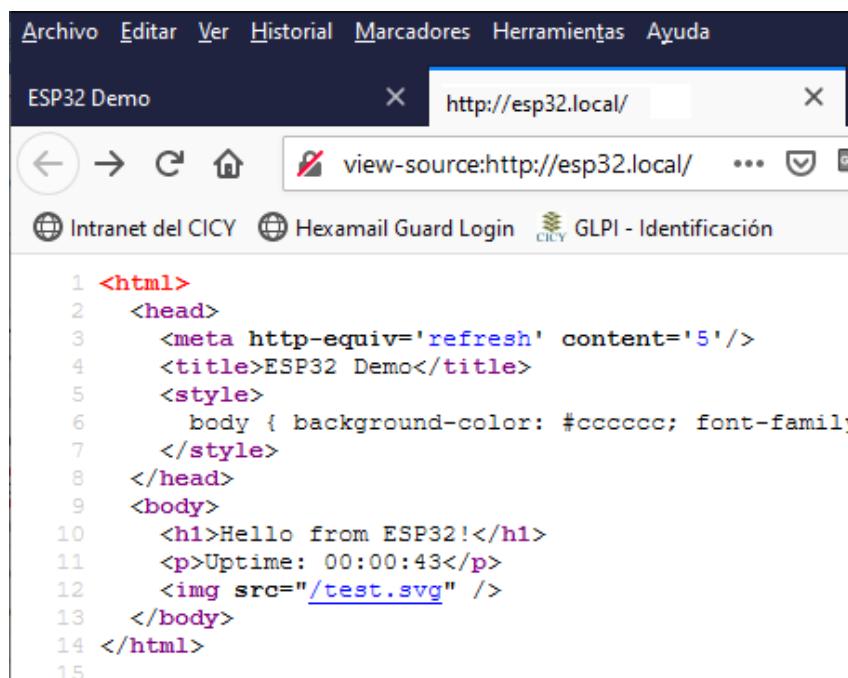
Librería inválida encontrada en C:\Users\Gabriel\Documents\Arduino\libraries\esp32-camera: No e

64 NodeMCU-32S en COM8

... compile y descargue nuevamente el código a su módulo NodeMCU. **ES NECESARIO CERRAR LAS PESTANAS, BORRAR DATOS E HISTORIAL** del navegador web para observar el cambio. Una vez borrado el historial, teclee en el navegador web la dirección anterior <http://esp32.local/> y obtendrá la misma página. Nuevamente, **vea el código fuente de la página**, así como sigue:



... e inmediatamente observará:



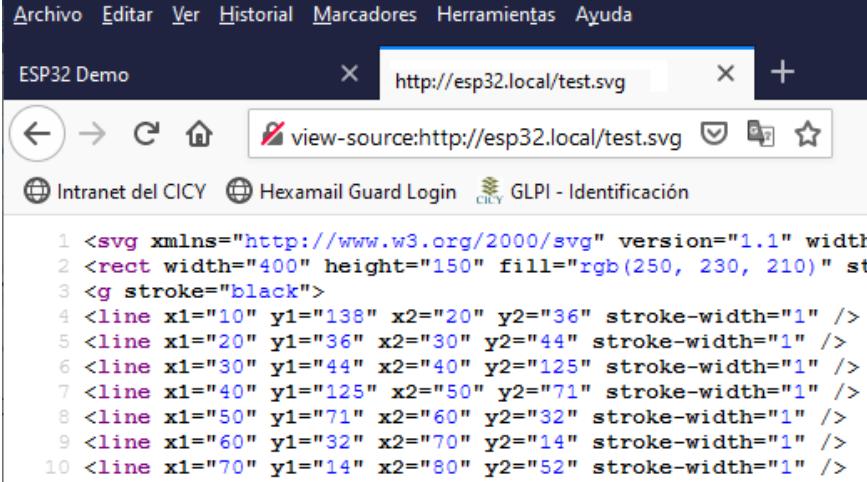
The screenshot shows a web browser window displaying the raw HTML code of the page. The address bar shows "http://esp32.local/" followed by "view-source:". The code is as follows:

```

1 <html>
2 <head>
3 <meta http-equiv='refresh' content='5' />
4 <title>ESP32 Demo</title>
5 <style>
6 body { background-color: #cccccc; font-family:
7 </style>
8 </head>
9 <body>
10 <h1>Hello from ESP32!</h1>
11 <p>Uptime: 00:00:43</p>
12
13 </body>
14 </html>
15

```

... con el salto de línea implementado en el código, ahora se observa mejor la estructura de la página web, ahora ponga el cursor del mouse en **el hipertexto marcado como "/test.svg"**, dele **un click con el mouse** y observe lo que sucede:



The screenshot shows a web browser window with the title "ESP32 Demo". The address bar displays "http://esp32.local/test.svg". Below the address bar, there are standard browser controls: back, forward, search, and refresh. The main content area shows the source code of an SVG file. The code consists of 10 numbered lines, each containing an XML command for drawing lines and rectangles. The XML uses attributes like xmlns, width, height, fill, stroke, and stroke-width.

```
1 <svg xmlns="http://www.w3.org/2000/svg" version="1.1" width="400" height="150" fill="rgb(250, 230, 210)" stroke="black">
2 <rect width="400" height="150" fill="rgb(250, 230, 210)" stroke="black"/>
3 <g stroke="black">
4 <line x1="10" y1="138" x2="20" y2="36" stroke-width="1" />
5 <line x1="20" y1="36" x2="30" y2="44" stroke-width="1" />
6 <line x1="30" y1="44" x2="40" y2="125" stroke-width="1" />
7 <line x1="40" y1="125" x2="50" y2="71" stroke-width="1" />
8 <line x1="50" y1="71" x2="60" y2="32" stroke-width="1" />
9 <line x1="60" y1="32" x2="70" y2="14" stroke-width="1" />
10 <line x1="70" y1="14" x2="80" y2="52" stroke-width="1" />
```

... sólo observamos 10 líneas de 44. Este último código corresponde a la gráfica, el cual, es una gráfica vectorial y su contenido son comandos de líneas, rectángulos, etc., en formato de texto.

No olvide el concepto de HyperText Markup Language (HTML), es decir, Lenguaje de Marcas de Hipertexto el cual, es en el que se basan los navegadores web para construir la página que muestran al usuario.

Ahora se analiza el código:

Se han añadido comentarios que explican cada línea del código y está disponible en: <https://github.com/gpoolb/ESP32> en la carpeta "AdvancedWebServerConComentarios", aún así es relevante explicar cómo funciona el programa a nivel de ejecución de métodos:

### 1er Recurso:

Según la línea 115, se configura al servidor para que ejecute al método llamado handleRoot, el cual, es el encargado de enviar la página web la cual, ya hemos observado anteriormente, adicionalmente, contiene la gráfica. Para ser redundantes, este primer recurso es el que llama el navegador al momento de teclear solamente la dirección ip: <http://esp32.local/> y puede observarlo en el código como sigue:

```

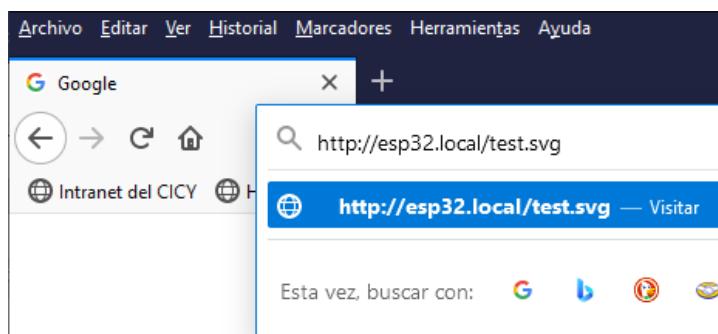
111 server.on("/", handleRoot); // Se establece el nombre de
112 server.on("/test.svg", drawGraph); // Se establece el nombre del recurso
113 server.on("/inline", []() { // Se establece el método a ejecutar
114 server.send(200, "text/plain", "this works as well");
115 });
116 // Termina el método a ejecutar cuando el cliente indica que no
117 server.onNotFound(handleNotFound); // Se establece el nombre del recurso
118 server.begin(); // Se inicia las funciones del servidor

```

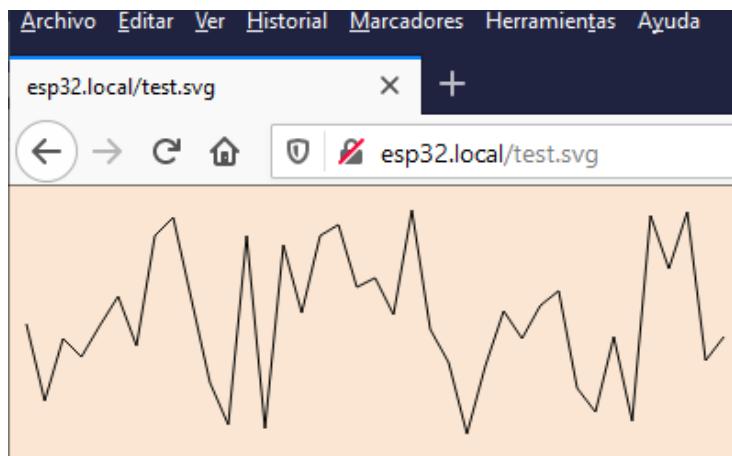
En la figura anterior se puede apreciar los recursos disponibles en el servidor, ahí se observan 4 recursos declarados. El primero ya se ha observado y se llama “/”. Cuando se teclea la dirección IP ó el nombre del servidor seguido de “.local” en el navegador WEB, pero... ¿Cómo se hacen las peticiones a los demás recursos?

## 2º Recurso

Ahora en el navegador WEB teclee <http://esp32.local/test.svg> así como se muestra:



... observe el resultado:



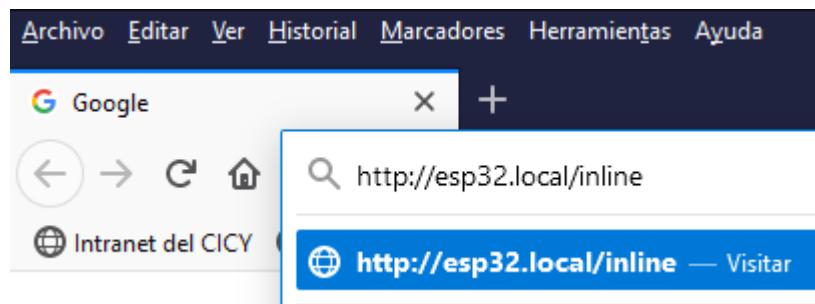
¿Qué ha pasado con la página Web? ¿Por qué luce diferente?, ¿Qué método fue el que se ejecutó al llamar a la gráfica?

```
115 server.on("/", handleRoot); // Se establece el nombre de
116 server.on("/test.svg", drawGraph); // Se establece el nom
117 server.on("/inline", []() { // Se establece el método a
118 server.send(200, "text/plain", "this works as well");
119 });
120 server.onNotFound(handleNotFound); // Se establece el n
121 server.begin(); // Se inicia las funciones del servidor
```

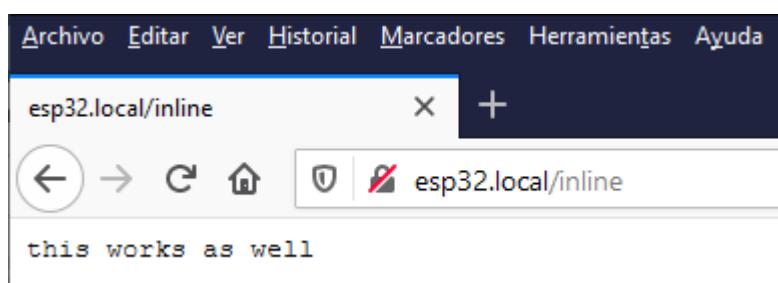
... en la figura anterior, podemos observar la línea 116, la cual, contiene el código que llama al método drawGraph y es el que se encarga de construir la gráfica y enviarlo al cliente solicitante.

### 3er Recurso

Para llamar al tercer recurso, sólo se tiene que teclear en el navegador web el nombre del servidor más el nombre de dominio (.local) seguido de /inline como se muestra a continuación:



... al presionar la tecla de “enter” se observa lo siguiente:



Sólo se aprecia el texto: “this works as well”. Observe con atención la línea 116, donde se declara la función embebida le envía al cliente.

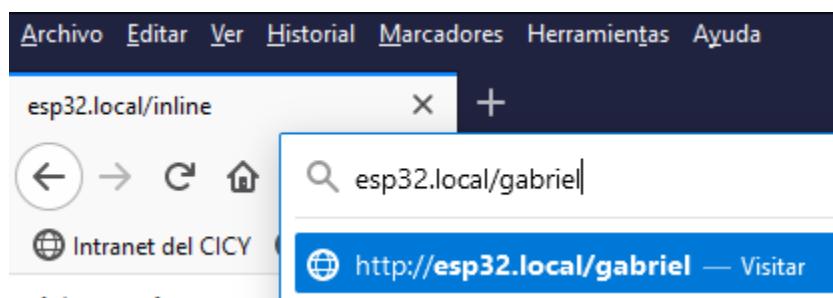
```

115 server.on("/", handleRoot); // Se establece el nombre de
116 server.on("/test.svg", drawGraph); // Se establece el nom
117 server.on("/inline", []() { // Se establece el método a
118 server.send(200, "text/plain", "this works as well");
119 }); // Termina el método a ejecutar cuando el cliente i
120 server.onNotFound(handleNotFound); // Se establece el n
121 server.begin(); // Se inicia las funciones del servidor

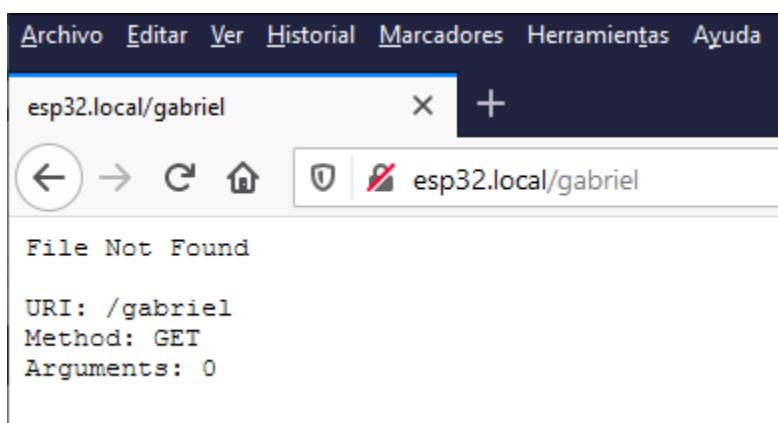
```

## 4o Recurso

Ahora se pretende solicitar al servidor el último recurso, pero no tiene un texto asociado... ¿Cómo se le puede llamar?, ¿qué texto se tiene que utilizar?. El último recurso se llama automáticamente al hacer una petición en la dirección ip del servidor, pero con un texto diferente a los declarados anteriormente. Un ejemplo, ponga la dirección ip correspondiente al servidor seguido de su nombre:



... presione la tecla “enter” y observe lo siguiente:



... puede apreciar que sólo se muestra un texto, y si analizamos la función de la línea 120 se puede apreciar que ahí se invoca al método handleNotFound:

```
115 server.on("/", handleRoot); // Se establece el nombre de
116 server.on("/test.svg", drawGraph); // Se establece el nom
117 server.on("/inline", []() { // Se establece el método a
118 server.send(200, "text/plain", "this works as well");
119 }); // Termina el método a ejecutar cuando el cliente i
120 server.onNotFound(handleNotFound); // Se establece el n
121 server.begin(); // Se inicia las funciones del servidor
```

Este método es relevante conocerlo, ya que tiene implementada la extracción de argumentos que el cliente le envía. Los argumentos son las “variables” que se pretende enviar al servidor para que éste lo procese, este tema se analizará en el punto 3.1.3 de este manual.

## mDNS

En el punto 3.1.1 de este manual, se instaló un software (en Windows y Android) que permitía encontrar en la red local al servidor instalado en su NodeMCU por nombre (no se requiere saber la dirección IP). La sección de código que hace posible esta acción es la que se muestra a continuación:

```
110
111 if (MDNS.begin("esp32")) { // Se configura la instancia para difusión del n
112 Serial.println("MDNS responder started"); // Si la configuración fué un é
113 } // Si la configuración falló no se mostrará lo anterior en el monitor del
114
```

En la línea 111 de la figura anterior se declara el nombre “esp32” el cual, es el nombre que usted elige de manera arbitraria. Elija preferentemente nombres en minúsculas, sin caracteres especiales (\$%&, etc.) y que le sea fácil recordar. El nombre de dominio se declara de manera implícita (.local) y sirve para “crear” un grupo en la red de trabajo y compartir los nombres entre los integrantes del grupo. Es por esa razón que al teclear en su navegador web tiene que usar esta nomenclatura: <http://esp32.local/>

Para que el programa “Service Browser” instalado en Android encuentre a su dispositivo en la red se requiere que agregue la línea al código justo al final del método setup() así como se muestra a continuación.

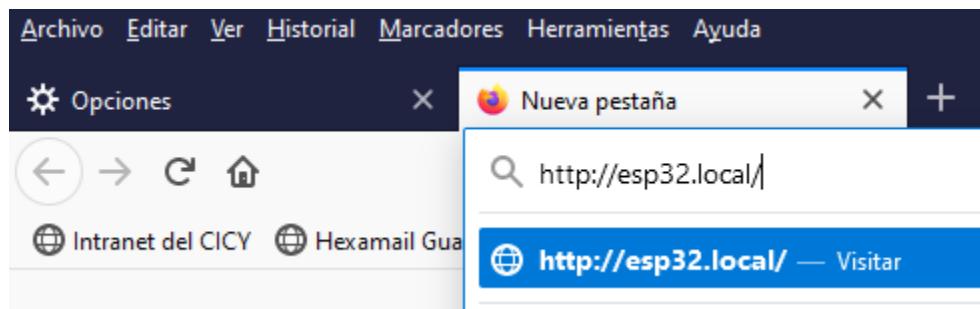
```
138 | Serial.println("HTTP server started"); // Se imprime la leyenda
139 |
140 | MDNS.addService("http", "tcp", 80);
141 |
142 |
```

Compile y descargue el código en su NodeMCU, ahora en su diapositivo Android, abra el programa “Service Browser” y ubique su servidor, siguiendo los pasos que se describen al final del punto 3.1.1.

### 3.1.3 Paso de argumentos al servidor WEB

**Objetivo específico:** Transferirá argumentos entre la página web (cliente) y su NodeMCU (servidor).

Antes de iniciar, se requiere descargar el ejemplo del sitio: <https://github.com/gpoolb/ESP32> en la carpeta “PasoDeArgumentos”. Obtenga del sitio web, compile y descargue el código a su módulo NodeMCU. ES **NECESARIO CERRAR LAS PESTANAS, BORRAR DATOS E HISTORIAL** del navegador web para observar el cambio. Una vez borrado el historial, teclee en el navegador web la dirección anterior <http://esp32.local> y obtendrá la misma página del ejemplo anterior con algunos cambios.



... presione la tecla “enter” y observa la página web del ejemplo:



Como podrá observar se han añadido dos botones, uno para encender y otro para apagar el LED incorporado en el NodeMCU. Presione los botones y observe su comportamiento. Ahora...

¿Qué cambios se le hizo al código anterior para lograr esto?

El primer cambio para lograr esto fue agregar los botones en la página web:

```

49 /*****INICIA LA PAGINA WEB *****
50 "<html>\n\
51 <head>\n\
52 <meta http-equiv='refresh' content='5;URL=/'>\n\
53 <title>ESP32 Demo</title>\n\
54 <style>\n\
55 body { background-color: #cccccc; font-family: Arial, Helvetica, Sans-Serif; }
56 </style>\n\
57 </head>\n\
58 <body>\n\
59 <h1>Hello from ESP32!</h1>\n\
60 <p>Uptime: %02d:%02d:%02d</p>\n\
61 \n\
62 <p></p>\n\
63 <form method=\"GET\" action=\"/led\" id=\"ledForm\" />\n\
64 <input type=\"submit\" form=\"ledForm\" name=\"ledBuiltIn\" value=\"on\">\n\
65 <input type=\"submit\" form=\"ledForm\" name=\"ledBuiltIn\" value=\"off\">\n\
66 </body>\n\
67 </html>\n",
68 *****TERMINA LA PAGINA WEB *****

```

... adicionalmente se cambia la declaración “content” para evitar que la página envíe el mismo argumento varias veces.

El segundo cambio, se requiere aumentar el tamaño del buffer de memoria, tanto de la variable “temp” y de la sentencia sprintf, así como se muestra:

```

42
43 void handleRoot() { // Método que envía la página WEB al client
44 char temp[800]; // Se define una variable la cual, contendrá
45 int sec = millis() / 1000; // Se define una variable que almacene el tiempo en milisegundos
46 int min = sec / 60; // Se define una variable que almacene el tiempo en minutos
47 int hr = min / 60; // Se define una variable que almacene el tiempo en horas
48 sprintf(temp, 800) \

```

En tercer lugar, se requiere de construir el método que controla el estado del led incorporado del NodeMCU:

```
76
77 void controlLed (void){ // Metodo que controla el estado del led incorporado
78 for (uint8_t i = 0; i < server.args(); i++) { // Se analiza el número de argumentos
79 Serial.print (server.argName (i)); // Se imprime en el monitor serial el nombre del argumento
80 Serial.print (": ");
81 Serial.println (server.arg (i)); // Se imprime al valor del argumento
82 if (server.argName (i).equalsIgnoreCase("ledBuiltIn")) { // Si el nombre es el establecido
83 if (server.arg (i).equalsIgnoreCase("on")) // Si nombre del argumento es "on"
84 digitalWrite (led, LOW); // Si el nombre y el valor de argumento coinciden
85 else
86 digitalWrite (led, HIGH); // Si el nombre y el valor de argumento no coinciden
87 }
88 }
89 handleRoot(); // El servidor nos devuelve a la página principal.
90 }
91
92 void handleNotFound() { //Metodo que devuelve al cliente el aviso de que
```

En cuarto lugar, se requiere eliminar la sentencia que está al inicio de los métodos handleRoot y handleNotFound:

```
digitalWrite (led, 1); // Enciende el Led incorporado en su módulo NodeMCU
```

... y la sentencia que está al final de los mismos métodos mencionados anteriormente:

```
digitalWrite (led, 0); // Enciende el Led incorporado en su módulo NodeMCU
```

... ya que interfieren con el propósito propuesto.

Por último, se tiene que declarar el nuevo recurso en la sección de configuración del servidor:

```
132
133 server.on("/", handleRoot); // Se establece el nombre de la página principal
134 server.on("/test.svg", drawGraph); // Se establece el nombre de la página de gráficos
135 server.on("/inline", []() { // Se establece el método a ejecutar
136 server.send(200, "text/plain", "this works as well");
137 }); // Termina el método a ejecutar cuando el cliente indica
138 server.on("/led", controlLed); // Se añade un nuevo recurso
139 server.onNotFound(handleNotFound); // Se establece el nombre de la página de error
```

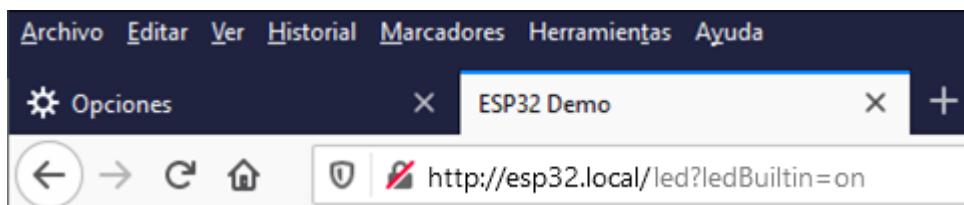
**NO OLVIDE QUE** para que el programa “Service Browser” instalado en Android encuentre a su dispositivo en la red se requiere que agregue la línea al código justo al final del método setup() así como se muestra a continuación.

```
MDNS.addService("http", "tcp", 80);
} // fin del método setup()
```

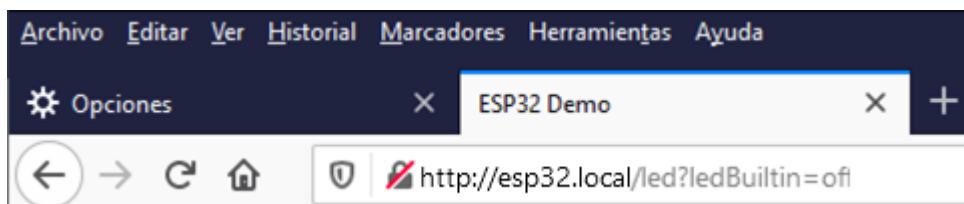
Compile y descargue el código en su NodeMCU, ahora en su dispositivo Android, abra el programa “Service Browser” y ubique su servidor, siguiendo los pasos que se describen al final del punto 3.1.1.

Ahora... ¿cómo funciona?, en el momento de presionar cualquier botón (“on” o “off”) observe con atención la barra de dirección del navegador WEB:

Para el botón on:



Para el botón off:



¿qué significan los caracteres que le siguen al nombre de servidor y nombre de dominio?

Ya se sabe que posterior al nombre de servidor y nombre de dominio sigue el recurso a solicitar, en este caso es: “/led”, eso le indica al servidor qué método va a ejecutar.

Ahora... ¿Qué significan los caracteres que siguen posterior al símbolo de interrogación?

El símbolo de interrogación es utilizado para separar el recurso de los argumentos. Los argumentos son las variables que enviamos al servidor para que las procese. Su notación es:

?nombre1=valor1, nombre2=valor2

En este caso es:

?ledBuiltIn=on para encender el led y,  
?ledBuiltIn=off para apagar el led

¿Cuántos argumentos se puede enviar?

Para el método GET que es el que se está utilizando se puede enviar un máximo de un paquete de 2048 caracteres, o sea, si los nombres y valores de los argumentos son largos, reducirá el número de argumentos a enviar.

¿Es posible enviar más argumentos?

Si, es posible enviar más argumentos, pero se requiere usar el método POST ya que los argumentos son enviados en paquetes separados de la dirección ip y el recurso solicitado. Es decir, en el primer paquete se envía la dirección y el recurso solicitado, cuando el servidor lo reciba, inmediatamente se le envían los argumentos, y éstos pueden ser tantos como se requiera.

Actividades:

¿Podría implementar el método POST en ambos botones?

Sugerencia:

En la sección de la página web cambie la forma en la que envía la petición los botones on y off:

```
<form method="POST\" action="/led\" id="ledForm\" />\n<input type="submit" form="ledForm" name="ledBuiltIn" value="on">\n<input type="submit" form="ledForm" name="ledBuiltIn" value="off">\n
```

Posteriormente, declare en la configuración del servidor, el método a utilizar:

```
server.on("/led", HTTP_POST, controlLed); // Se añade un nuevo recurso para controlar el led incorporado del NodeMCU
```

¿Podría implementar los botones para controlar el Led RGB descrito en el punto 2.2.1?

Sugerencias:

Incremente el valor del buffer y del espacio de la instrucción sprintf:

```
char temp[1050];
```

```
snprintf(temp, 1050, \
```

Implemente los botones en la página web:

```
<p></p>\n<input type=\"submit\" form=\"ledForm\" name=\"ledR\" value=\"on\">\n<input type=\"submit\" form=\"ledForm\" name=\"ledR\" value=\"off\">\n<p></p>\n<input type=\"submit\" form=\"ledForm\" name=\"ledG\" value=\"on\">\n<input type=\"submit\" form=\"ledForm\" name=\"ledG\" value=\"off\">\n<p></p>\n<input type=\"submit\" form=\"ledForm\" name=\"ledB\" value=\"on\">\n<input type=\"submit\" form=\"ledForm\" name=\"ledB\" value=\"off\">\n\n
```

Declare las variables al inicio del código que correspondan a los pines asignado al led RGB

```
const int led = LED_BUILTIN;
const int ledR = 12;
const int ledG = 14;
const int ledB = 27;
```

En el método setup() configure los pines de los led's como salida:

```
pinMode(ledR, OUTPUT);
digitalWrite(ledR, 1); // Se apaga el led incorporado en su módulo NodeMCU
pinMode(ledG, OUTPUT);
digitalWrite(ledG, 1); // Se apaga el led incorporado en su módulo NodeMCU
pinMode(ledB, OUTPUT);
digitalWrite(ledB, 1); // Se apaga el led incorporado en su módulo NodeMCU
```

Posteriormente, implemente el código para controlar el estado de los tres colores de led:

```
if (server.argName (i).equalsIgnoreCase("ledBuiltIn")) {
 if (server.arg (i).equalsIgnoreCase("on"))
 digitalWrite(led, LOW);
 else
 digitalWrite(led, HIGH);
} else if (server.argName (i).equalsIgnoreCase("ledR")) {
 if (server.arg (i).equalsIgnoreCase("on"))
 digitalWrite(ledR, LOW);
 else
 digitalWrite(ledR, HIGH);
} else if (server.argName (i).equalsIgnoreCase("ledG")) {
 if (server.arg (i).equalsIgnoreCase("on"))
 digitalWrite(ledG, LOW);
 else
 digitalWrite(ledG, HIGH);
} else if (server.argName (i).equalsIgnoreCase("ledB")) {
 if (server.arg (i).equalsIgnoreCase("on"))
 digitalWrite(ledB, LOW);
 else
 digitalWrite(ledB, HIGH);
}
```

## 3.2 El Servidor Web con página dinámica en NodeMCU-32S

En este punto se desarrollará una página web dinámica la cual, mostrará el valor del sensor de temperatura y podrá manipular el estado de un led desde la página web.

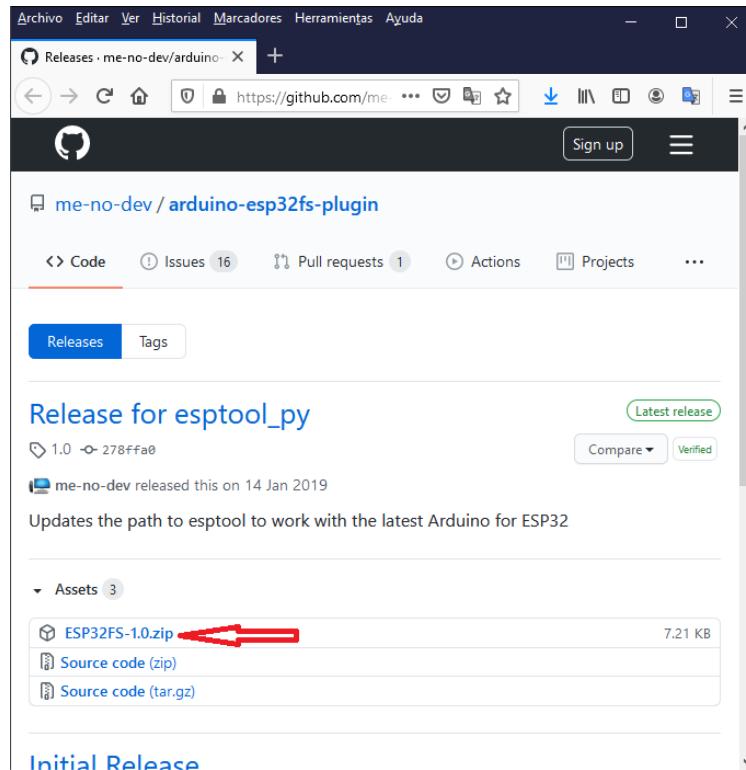
### 3.2.1 ESP32 Filesystem Uploader Plugin

**Objetivo específico:** Instalará la herramienta que permite subir archivos y carpetas en la memoria Flash del ESP32.

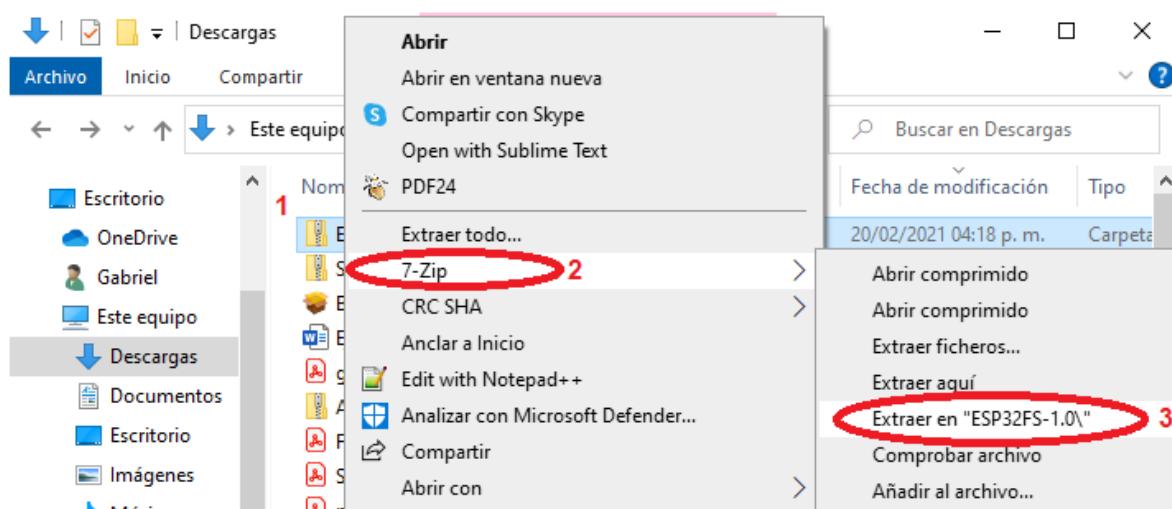
El ESP32 contiene un Sistema de Archivos en la memoria Flash Interna con Interfaz Serial (SPIFFS). SPIFFS es un sistema de archivos muy ligero creado para microcontroladores con una memoria flash. Usted puede leer, escribir, cerrar y borrar archivos. En la literatura menciona que no soporta directorios, pero se ha probado introducir archivos en una carpeta y ha funcionado. Adicionalmente, tiene la limitante de utilizar nombre de 8 caracteres como máximo y tres caracteres como extensión.

En el sitio <https://randomnerdtutorials.com/install-esp32-filesystem-uploader-arduino-ide/> se describen los pasos para la instalación del plugin, según lo anterior, se requiere seguir los siguientes pasos:

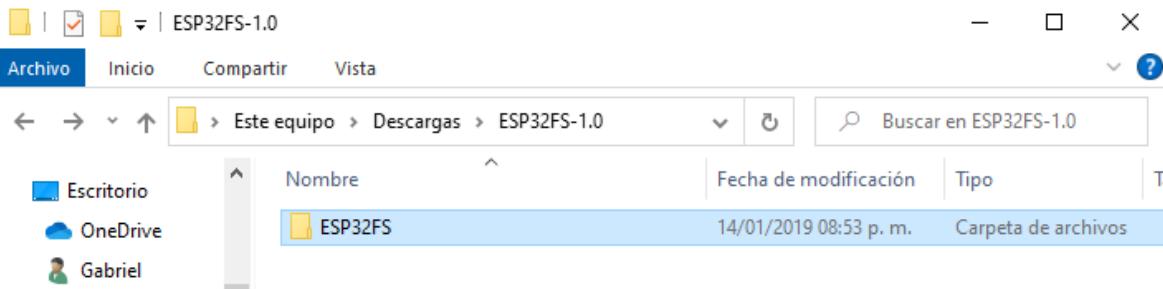
- 1) Obtenga el plugin del sitio <https://github.com/me-no-dev/arduino-esp32fs-plugin/releases/> y descargue el archivo con terminación .zip así como se muestra:



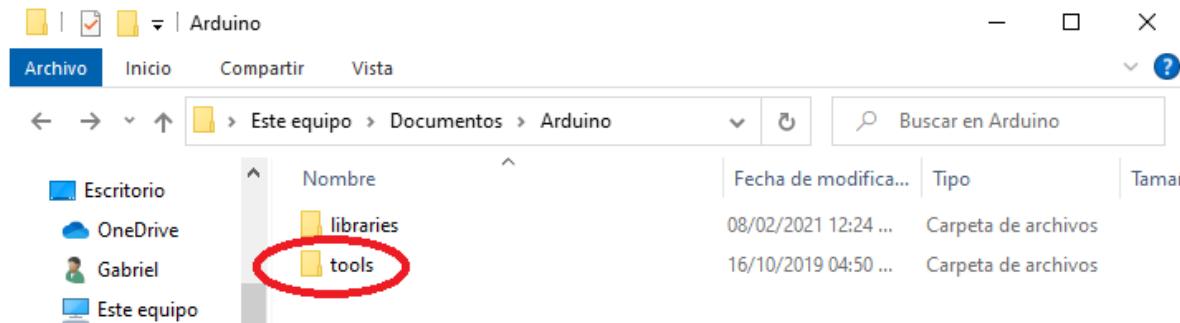
**2)** Al dar un click con el mouse en el sitio marcado en la figura anterior, iniciará la descarga. Ahora, con el archivo descargado, se requiere descomprimirlo para esto, dé un click con el botón derecho del mouse sobre el archivo recién descargado y siga la secuencia en el menú que sale a continuación, así como se muestra (se requiere tener instalado el programa 7zip vista en el punto 3.1.1):



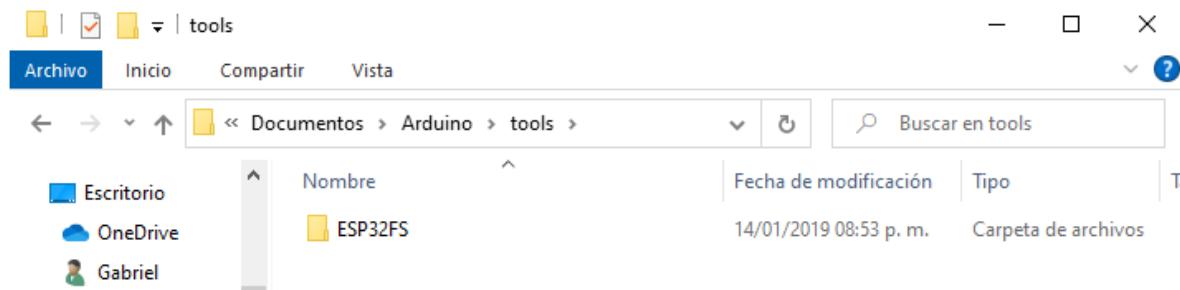
**3)** Al terminar de descomprimir, se genera una nueva carpeta llamada “ESP32FS-1.0”, entre a esa carpeta y verá otra llamada “ESP32FS”



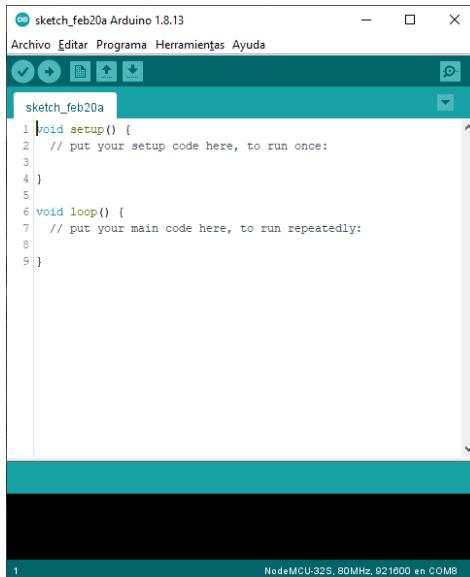
**4)** Ahora abra OTRA ventana del explorador de archivos, ubíquese en Mis Documentos, entre a la carpeta de Arduino y en el interior de la carpeta Arduino cree otra carpeta llamada “tools” así como se muestra:



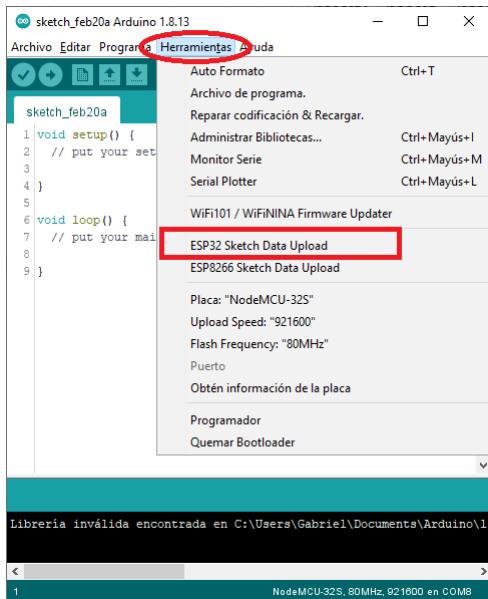
**5)** Entre a la carpeta “tools” recién creada y copie la carpeta ESP32FS (vista en el punto 3) así como se muestra:



**6)** Cierre ambas ventanas del explorador y abra el IDE del Arduino (Si lo tenía abierto, cierre el IDE del Arduino y vuelva a abrirlo):



**7)** Al presionar el menú de herramientas en el IDE del Arduino, podrá observar que ahora se tiene una nueva opción llamada “ESP32 Sketch Data Upload”, el cual, es el que se utilizará para la descarga de archivos al NodeMCU.



Por el momento NO presione esa opción, ya que, se requiere de la creación de la carpeta “data” en el proyecto, que, contiene los archivos que serán transferidos al NodeMCU. Esta opción se utilizará en el proyecto que se verá más adelante.

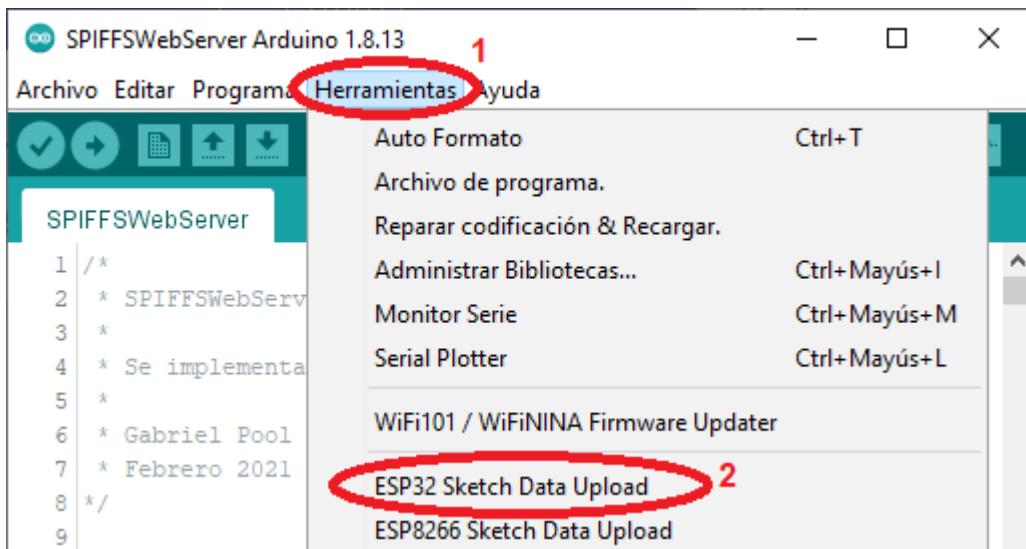
### 3.2.2 SPIFFS WebServer

**Objetivo específico:** Implementará una página web dinámica en el módulo ESP32.

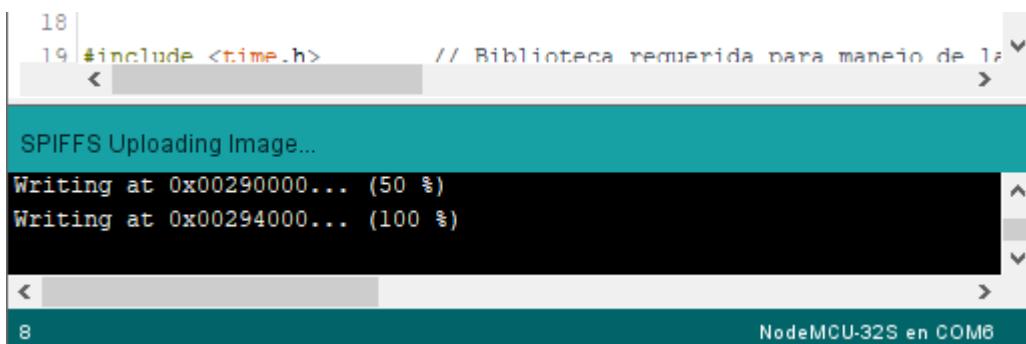
Antes de iniciar, se requiere descargar el ejemplo del sitio: <https://github.com/gpoolb/ESP32> en la carpeta “SPIFFSWebServer”. Obtenga del sitio web, compile y descargue el código a su módulo NodeMCU de la misma manera que se ha manejado en los ejemplos anteriores:



Al terminar de descargar su código anterior, es necesario, ejecutar un paso adicional, utilizando la herramienta instalada en el punto 3.2.1, se requiere descargar en el módulo NodeMCU los archivos incluidos en la carpeta “data” que se encuentra en la carpeta del proyecto, así como se muestra:

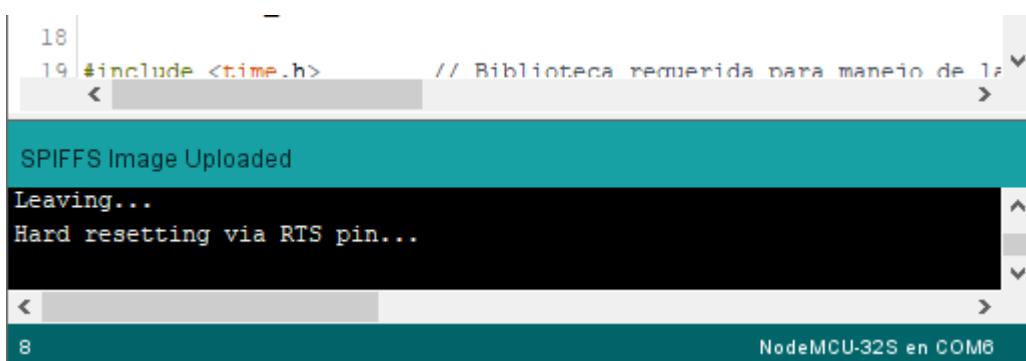


... y en la parte inferior del IDE mostrará lo siguiente:



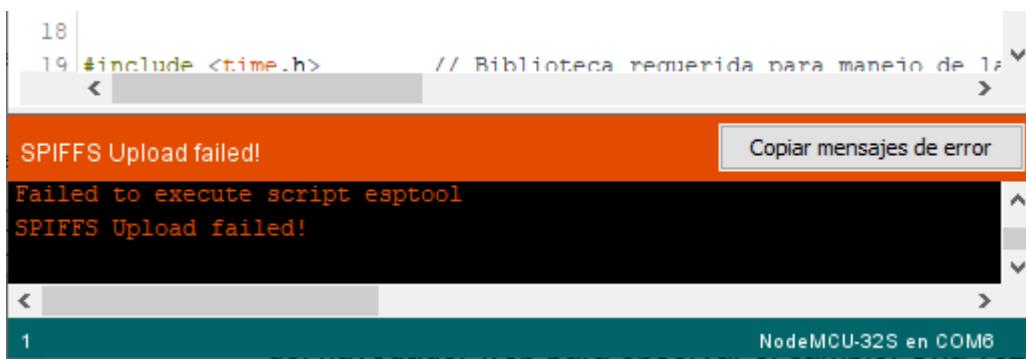
The screenshot shows the Arduino IDE's Serial Monitor window. The text output is:  
18  
19 #include <time.h> // Biblioteca requerida para manejo de tiempos  
SPIFFS Uploading Image...  
Writing at 0x00290000... (50 %)  
Writing at 0x00294000... (100 %)  
8  
NodeMCU-32S en COM6

Espere a que la barra azul muestre lo siguiente:



The screenshot shows the Arduino IDE's Serial Monitor window. The text output is:  
18  
19 #include <time.h> // Biblioteca requerida para manejo de tiempos  
SPIFFS Image Uploaded  
Leaving...  
Hard resetting via RTS pin...  
8  
NodeMCU-32S en COM6

Si al momento de utilizar esta herramienta, muestra el siguiente error:



The screenshot shows the Arduino IDE's Serial Monitor window. The text output is:  
18  
19 #include <time.h> // Biblioteca requerida para manejo de tiempos  
SPIFFS Upload failed!  
Failed to execute script esptool  
SPIFFS Upload failed!  
1  
NodeMCU-32S en COM6

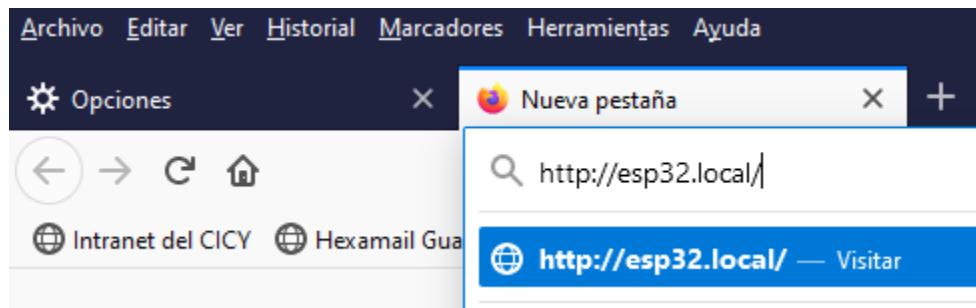
A red box highlights the error message "SPIFFS Upload failed!" in the serial monitor.

... se debe a que tiene abierto la ventana del monitor serial. Es necesario cerrar dicha ventana para que la herramienta pueda transferir los archivos al módulo NodeMCU.

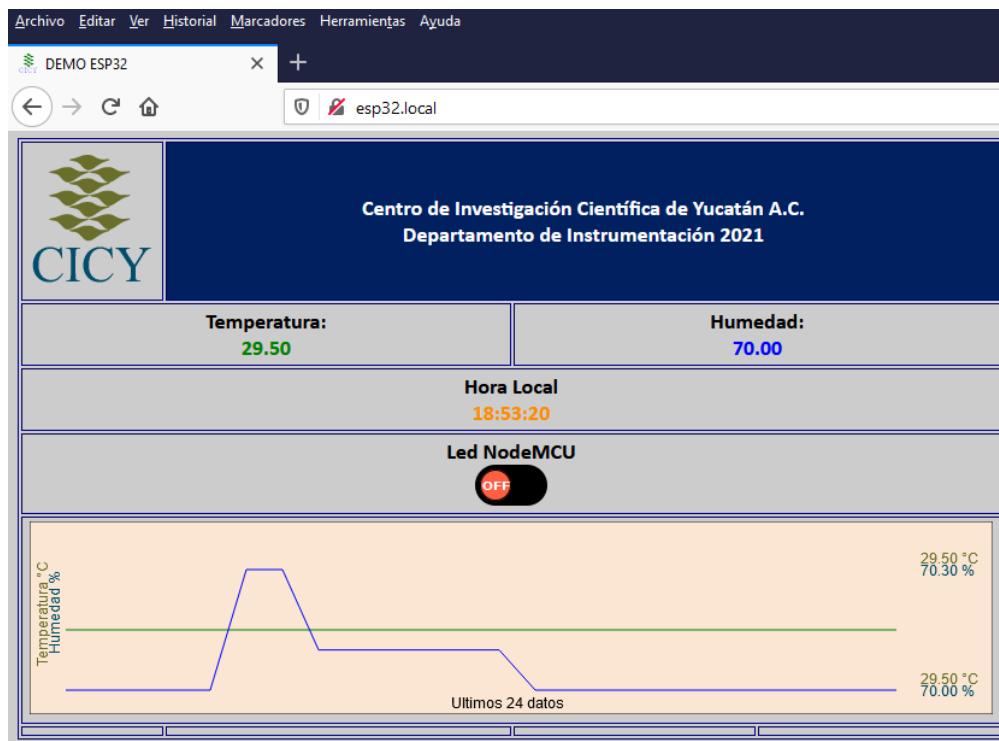
El contenido guardado en la memoria flash por esta herramienta (ESP32 Data Upload), es independiente del código de programa que se compila y se descarga en el módulo NodeMCU, es decir, no es necesario compilar y descargar de nuevo

el código de programa al módulo NodeMCU cuando la herramienta sea utilizada (ESP32 Data Upload).

Una vez descargados ambos (código y archivos) en el NodeMCU, ES **NECESARIO CERRAR LAS PESTAÑAS, BORRAR DATOS E HISTORIAL** del navegador web para observar el cambio. Una vez borrado el historial, teclee en el navegador web la dirección anterior <http://esp32.local>, cabe aclarar que se está considerando que usa Windows en su PC y tiene instalado el Bonjour descrito en el punto 3.1.1, para Android use el programa Service Browser que también se describe en el punto 3.1.1:



... presione la tecla “enter” y observa la página web del ejemplo:

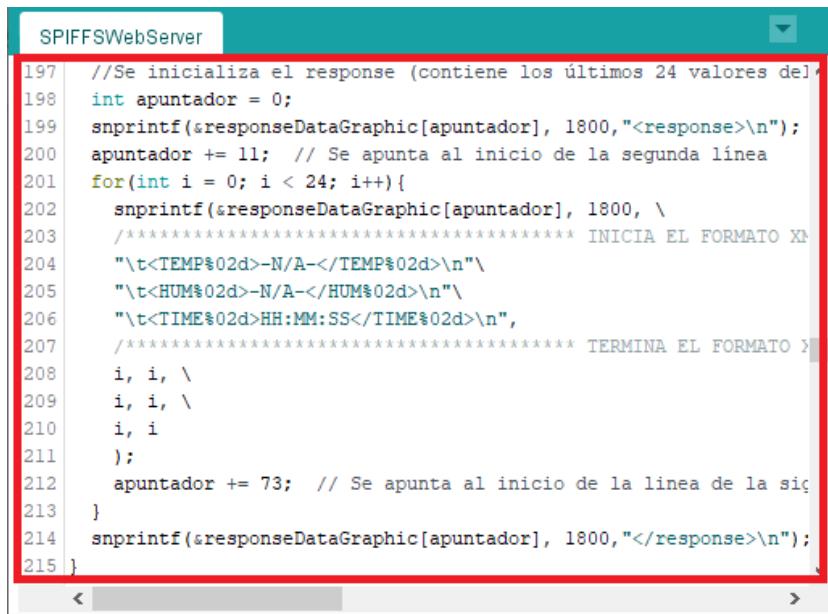


Como podrá observar de manera más relevante: el botón del LED cambió su apariencia, la gráfica se desplaza horizontalmente y se ha añadido un logo. Presione el botón y observe su comportamiento. Los textos y la gráfica se han vuelto dinámicos. Ahora... ¿Qué cambios se le hizo al código anterior para lograr esto?

Todo el código de programa está comentado, así que analizaremos por secciones las partes más relevantes:

```
169 |
170 | Serial.println("Contacting Time Server");
171 | configTime(3600*timezone, daysavetime*3600, "time.nist.gov", "0.
172 | delay(2000);
173 | tmstruct.tm_year = 0;
174 | getLocalTime(&tmstruct, 5000); // Se obtiene el tiempo actual
175 | Serial.printf("\nNow is : %d-%02d-%02d %02d:%02d:%02d\\:", (tmstru
176 | Serial.println("");
177 |
178 | if(!SPIFFS.begin()){
179 | Serial.println("Card Mount Failed");
180 | return;
181 | }
182 | }
```

En el método `setup()`, está declarado la configuración para la obtención de la hora y fecha, es posible tener la hora y fecha actual desde un servidor que se especializa en proporcionar ése servicio. Adicionalmente, se configura al módulo ESP32 para tener acceso al sistema de archivos en su memoria flash (es equivalente a tener una microSD para almacenar archivos, aunque de capacidad muy limitada).



```
SPIFFSWebServer
197 //Se inicializa el response (contiene los últimos 24 valores de
198 int apuntador = 0;
199 snprintf(&responseDataGraphic[apuntador], 1800,"<response>\n");
200 apuntador += 11; // Se apunta al inicio de la segunda linea
201 for(int i = 0; i < 24; i++){
202 snprintf(&responseDataGraphic[apuntador], 1800, \
203 /***** INICIA EL FORMATO XML *****
204 "\t<TEMP%02d>-N/A-</TEMP%02d>\n"
205 "\t<HUM%02d>-N/A-</HUM%02d>\n"
206 "\t<TIME%02d>HH:MM:SS</TIME%02d>\n",
207 /***** TERMINA EL FORMATO XML *****
208 i, i, \
209 i, i, \
210 i, i
211 };
212 apuntador += 73; // Se apunta al inicio de la linea de la siq
213 }
214 snprintf(&responseDataGraphic[apuntador], 1800,"</response>\n");
215 }
```

En esta sección (Lin 197) se construye un archivo xml (Extensible Markup Language) los cuales se componen de etiquetas en este caso están de forma **individual**. La estructura que se genera es ésta:

```
<response>
 <TEMP00>XX.XX</TEMP00>
 <HUM00>XX.XX</HUM00>
 <TIME00>HH:MM:SS</TIME00>
 <TEMP01>XX.XX</TEMP01>
 <HUM01>XX.XX</HUM01>
 <TIME01>HH:MM:SS</TIME01>
 .
 .
 .
 <TEMP23>XX.XX</TEMP23>
 <HUM23>XX.XX</HUM23>
 <TIME23>HH:MM:SS</TIME23>
</response>
```

Este archivo se utiliza para enviar los datos de la gráfica al cliente, contiene 24 series de 3 datos: Humedad, Temperatura y el Tiempo con formato HH:MM:SS. Observe que tienen etiquetas de apertura y de cierre, en medio de estos se encapsula la información a enviar. El cliente al recibirla, por el nombre del tag ya sabe donde asignar el valor en el lugar que le corresponde.

```
222 void sendDataGraphic() { // Método que dibuja una gráfica tipo Scal
223
224 server.send(200, "text/xml", responseDataGraphic); // Sentencia
225 //Serial.println("Response: ");
226 //Serial.println(responseDataGraphic);
```

Es importante declarar al momento del envío el tipo de archivo, ya que el navegador estará esperando un archivo con formato xml (el cual se declara en el ajax.js, específicamente, el método pollAJAX())

```
230
231 void leeSensor (void) {
232
```

En la método leeSensor(), se encarga de leer los datos del sensor e ir insertando los datos en el archivo xml cada segundo. Cada vez que se lee un nuevo dato, este será insertado en la primera posición (00) y el dato que ocupaba la posición '00' es desplazado a la posición '01' y así sucesivamente hasta llegar a la posición '23', el último dato (que ocupaba la posición '23') es desecharido.

```

52 void sendDataTemp() { // Método que envía la página WEB al cliente
53 char temp[150]; // Se define una variable la cual, contendrá
54 //get time
55 getLocalTime(&tmstruct, 0); // Se obtiene el tiempo local almacenado
56
57 sprintf(temp, 150, \
58 /***** INICIA EL FORMATO XML *****
59 "<response>\n" \
60 "\t<TEMP>%s</TEMP>\n" \
61 "\t<HUM>%s</HUM>\n" \
62 "\t<ledBuiltIn>%s</ledBuiltin>\n" \
63 "\t<TIME>%02d:%02d:%02d</TIME>\n" \
64 "</response>\n",
65 /***** TERMINA EL FORMATO XML *****
66 /***** INICIA LISTA DE VARIABLES A SER ENVIADAS *****
67 charTemp, \
68 charHum, \
69 estadoLed, \
70 tmstruct.tm_hour, \
71 tmstruct.tm_min, \
72 tmstruct.tm_sec // Aquí se calcula el tiempo en que inicia la
73 /***** TERMINA INICIA LISTA DE VARIABLES A SER ENVIADAS *****
74);
75 server.send(200, "text/xml", temp); // Sentencia que envía la página
76 } // Termina método que envía la página WEB al cliente que lo solicita

```

En la rutina sendDataTemp(), se encarga de construir otro archivo xml más pequeño que el anterior, éste se utiliza para enviar los valores instantáneos del sensor (mas recientes), la hora y el estado del LED.

```

<response>
 <TEMP>XX.XX</TEMP>
 <HUM>XX.XX</HUM>
 <ledBuiltIn>false</ledBuiltin>
 <TIME>HH:MM:SS</TIME>
</response>

```

... al igual que en el archivo anterior, es importante especificar el tipo de archivo que se envía al cliente (ver línea 76 de la imagen anterior) ya que el mismo método pollAJAX() se encarga de recibirla.

```

 78 void controlled (void){ // Método que controla el estado del led

```

La pregunta que surge es... ¿porqué enviar dos archivos xml cuando se pueden enviar todos en un solo archivo? La respuesta es porque los valores de la gráfica pueden ser actualizados en tiempos diferentes de acuerdo con el tiempo que decida el programador.

El método controlLed() ya se había analizado y se encarga de controlar el led de acuerdo a los parámetros recibidos, aquí existe un cambio con respecto al ejemplo anterior, el valor del parámetro ha cambiado por ‘true’ o ‘false’ ya que se envía el valor del checkbox declarado en el archivo index.htm

```
187 server.on("/dataGraphics.xml", HTTP_GET, sendDataGraph); // Se
188 server.on("/temp.xml", HTTP_GET, sendDataTemp);
189 server.on("/led", HTTP_GET, controlLed); // Se añade un nuevo r
190 server.onNotFound(handleNotFound); // Se establece el nombre de
191 server.begin(); // Se inicia las funciones del servidor.
```

Los recursos declarados en el servidor son sólo cuatro:

1. El recurso para obtener los valores de la gráfica
2. El recurso para obtener los valores instantáneos de la temperatura, humedad, hora y estado del Led.
3. El recurso que controla el estado del led
4. Por último, las peticiones que no se han declarado pasan por este punto.

Pregunta: ¿Dónde se lee la página web y sus dependencias?

```
134 void handleNotFound() { //Metodo que devuelve al cliente el aviso
135 if(loadFromSdCard(server.uri())) return;
136 String message = "File Not Found\n\n"; // Definición de la vari
```

Todas las peticiones que no se han declarado pasan obligatoriamente en el cuarto recurso (handleNotFound()), en dicho método, en la línea 135 se declara un método llamado loadFromSDCard() al cual se le proporciona como parámetro el recurso solicitado por el cliente. Así que, el programa salta nuevamente a la ejecución de la línea 98:

```
98 bool loadFromSdCard(String path){
99 String dataType = "text/plain";
100 if(path.endsWith("/")) path += "index.htm";
101
102 if(path.endsWith(".src")) path = path.substring(0, path.lastIndexOf('.'));
103 else if(path.endsWith(".htm")) dataType = "text/html";
104 else if(path.endsWith(".css")) dataType = "text/css";
105 else if(path.endsWith(".js")) dataType = "application/javascript";
106 else if(path.endsWith(".png")) dataType = "image/png";
107 else if(path.endsWith(".gif")) dataType = "image/gif";
108 else if(path.endsWith(".jpg")) dataType = "image/jpeg";
109 else if(path.endsWith(".ico")) dataType = "image/x-icon";
110 else if(path.endsWith(".xml")) dataType = "text/xml";
111 else if(path.endsWith(".pdf")) dataType = "application/pdf";
112 else if(path.endsWith(".zip")) dataType = "application/zip";
113 else if(path.endsWith(".svg")) dataType = "image/svg+xml";
```

En dicho método (extraído del ejemplo SDWebServer) se compara la extensión para que le sea asignado el tipo de archivo a enviar al cliente (esto es necesario, recuerde que el explorador web NO aceptará otro tipo de archivo diferente al que espera), posteriormente, en la línea 118 se gestiona en el sistema de archivos (SPIFFS) el recurso solicitado por el cliente:

```
118 File dataFile = SPIFFS.open(path.c_str(), "r");
119
120 if (!dataFile)
121 return false;
122
123 if (server.hasArg("download")) dataType = "application/octet-sti:
124
125 if (server.streamFile(dataFile, dataType) != dataFile.size()) {
126 Serial.println("Sent less data than expected!");
127 }
128
129 dataFile.close();
130 return true;
131 }
```

Recuerde que, hasta este punto, sólo llegan los recursos que no se declararon anteriormente, así que, se esperan TODOS los nombres de archivos que se han guardado en el SPIFFS con la herramienta (ESP32 Data Upload), en este caso son:

index.htm  
ajax.js  
ajax2.js  
logo.svg  
test.svg  
favicon.ico

Si llegara otro diferente a los anteriores, el método terminará en la línea 121 e indicará que no encontró el recurso solicitado con un valor ‘false’.

Si el recurso es encontrado, será enviado al cliente en la línea 125, el sistema de archivos será cerrado en la línea 129 y el método finalizará devolviendo un valor ‘true’ que indica que la respuesta fue enviada exitosamente.

Hasta este punto, se ha descrito el funcionamiento del servidor web por parte del módulo NodeMCU ESP32, ahora se describirá a nivel de métodos cómo funciona en la parte del cliente (navegador WEB).

### El cliente

Al momento de teclear la dirección IP ó el nombre MDNS, el navegador le envía el carácter '/' al servidor, en respuesta, el servidor envía el archivo index.htm. Típicamente, el primer archivo que el navegador recibe de un servidor es el index.html, así que, el ESP32 se apega al estándar. Al recibir el index.htm del NodeMCU, el navegador lo analiza e inmediatamente, gestiona las dependencias con el servidor, en este caso son los demás archivos (ya mencionados anteriormente), el cual, cuando el servidor recibe las peticiones envía como respuesta los archivos solicitados.

En el momento de recibir todas las dependencias, el navegador web (cliente) inicia la construcción de la página web que muestra al usuario e inmediatamente empieza la ejecución de los códigos en lenguaje JavaScript (archivos con extensión .js) los cuales, no son visibles al usuario.

Si abre con el editor de texto (preferentemente Notepad++) el archivo index.htm, se puede identificar los códigos JavaScript por estar entre los siguientes tags:

```
<script src="*.js" TYPE="text/javascript" ></script>
```

Las imágenes que se utilizan en este proyecto están en formato svg y se identifican por estar entre los siguientes tags:

```
<iframe id="myImage" type="image/svg+xml" src = "*.svg" frameborder= "0" height=" " width=" " >
</iframe>
```

... los cuales están embebidos en una tabla.

Los estilos declarados para la página web y el botón (on-off) se delimitan por los siguientes tags:

```
<style> </style>
```

La página web fue construida de la manera más sencilla posible para poder agregar más ítems. Por ejemplo, si desea agregar un texto, tiene que declararlo en la página web como:

```
<text id="miText"></text>
```

Y desde el servidor, agregue en el response un tag de la siguiente manera:

```
"\t<miText>Hola Mundo</miText>\n"\
```

... cuando el navegador lo reciba, asignará el texto "Hola Mundo" en el sitio donde declaró los tags.

Se construyó un método para gestionar recursos desde la página web, sólo requiere dos parámetros:

```
EnviaDatos(recurso,parámetros);
```

Si quisiera enviar la instrucción de apagar led sería de la siguiente manera:

```
EnviaDatos('/led?','ledBuitin=false');
```

En este caso, se requería enviar el estado del checkbox, así que se utilizan las propiedades de éste para poder enviar su nombre y su estado:

```
onclick="EnviaDatos('./led?', this.id + '=' + this.checked)"
```

... el cual:

this.id es el equivalente a leer el nombre asignado en la propiedad id = "ledBuiltin"  
this.checked es el equivalente a leer el estado del checkbox: 'true' o 'false'

... en pocas palabras, esta última declaración podría ser usado para cualquier checkbox, lo único que tendría que cuidar es que el nombre asignado en la propiedad 'Id' sea diferente.

Si desea que el botón indique si el led esté encendido o apagado (o siendo manipulado por otro cliente), es necesario agregar en el response que envía el servidor lo siguiente:

```
"\t<ledBuiltin>true</ledBuiltin>\n"\n0"\t<ledBuiltin>false</ledBuiltin>\n"\n
```

... el cual indica si el led está encendido ó apagado en el servidor. En este caso particular, el estado del led está almacenado en una variable del tipo string llamada: estadoLed y al usar %s en el código del NodeMCU se indica que lea la variable como una del tipo string (estudie la sentencia sprintf y snprintf usada en la línea 59 del código en el NodeMCU).

Se agregan comentarios a todos los archivos que dependen de la página web de este proyecto para su mejor comprensión. Considere estudiar lenguaje HTML, JavaScript y CSS. Existen alternativas interesantes como el Bootstrap para una mejor presentación en la página web.