



Centro de Investigación Científica de Yucatán A. C.
Departamento de Instrumentación
www.cicy.mx

Desarrollos de dispositivos IoT basado en el módulo ESP32

Curso Octubre 2020

Instructor:

Ing. Gabriel Jesús Pool Balam



Contenido

Introducción	3
1. Instalación del IDE de Arduino	4
1.1 Instalación del IDE del Arduino v1.8.13	5
1.2 Instalación del plugin ESP32 para IDE del Arduino v1.8.13	12
1.3 Ejecutando el ejemplo “blink”	18
1.4 Usando la ayuda del IDE del Arduino	26
1.5 Usando el monitor del puerto serie del IDE del Arduino	32
1.6 Agregando bibliotecas al IDE del Arduino del catálogo de drivers	36
2. Implementando códigos en IDE de Arduino	46
2.1 El Hardware del módulo NodeMCU 32S	47
2.2 Análisis de ejemplos considerados relevantes para la implementación de un lector de temperatura	52
2.2.1 Blink.ino	53
2.2.2 BlinkWithoutDelay.ino	58
2.2.3 Máquina de estados	60
2.2.3 Usando la pantalla OLED SSD1306	64
2.2.4 Usando el sensor de temperatura DS18B20 de fabricado por dallas semiconductor	68
2.2.5 Usando el sensor de temperatura DHT22 (AM2301) fabricado por AMLOGIC	72
2.2.6 Mostrando los valores del sensor en la pantalla del SSD1306	76



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



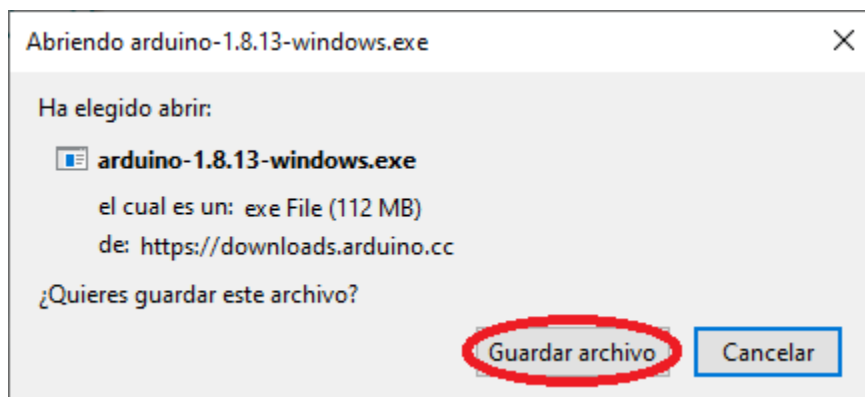
...déle un click con el botón izquierdo del mouse en el texto marcado por la flecha que se muestra en la figura anterior y espere a que la siguiente página web se cargue, a continuación, se muestra lo siguiente:

Contribute to the Arduino Software

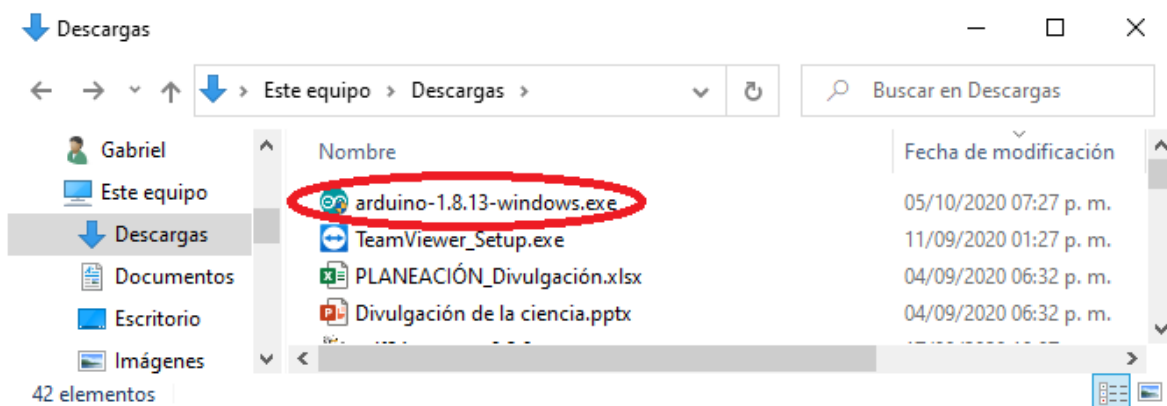
Consider supporting the Arduino Software by contributing to its development. (US tax payers, please note this contribution is not tax deductible). [Learn more on how your contribution will be used.](#)



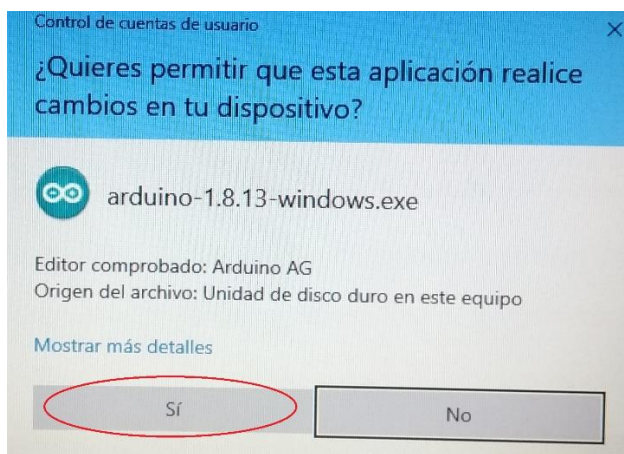
...dé un click con el botón izquierdo del mouse en el texto marcado como “just download” e inmediatamente saldrá una ventana emergente como ésta:



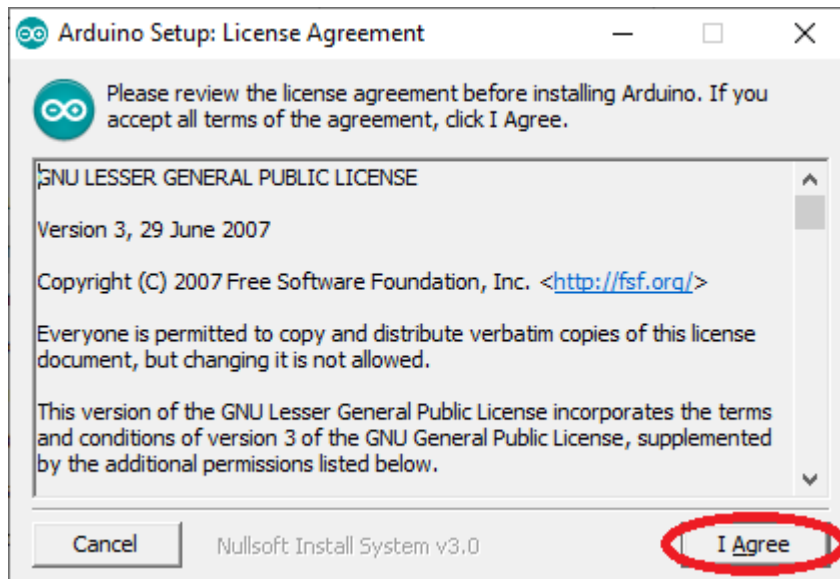
... presione el botón con el texto “Guardar archivo” y la descarga iniciará, cuando haya concluido, su navegador le avisará que la descarga ya ha terminado. Ubique el archivo descargado en la carpeta de descargas (típicamente los archivos se guardan en esta carpeta a menos que usted haya modificado la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



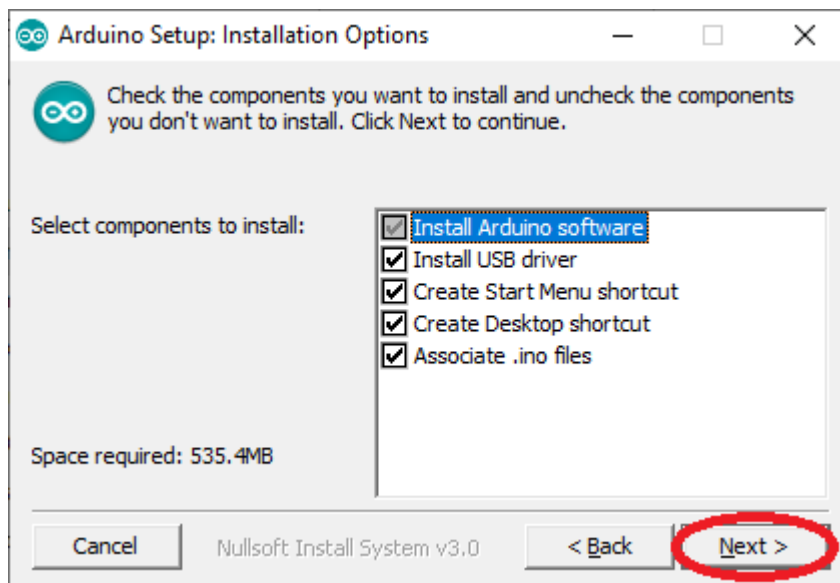
... presione doble click con el botón izquierdo del mouse al archivo recién descargado (llamado Arduino-1.8.13-windows.exe) e inmediatamente la pantalla se pondrá negra con una ventana como se muestra a continuación:



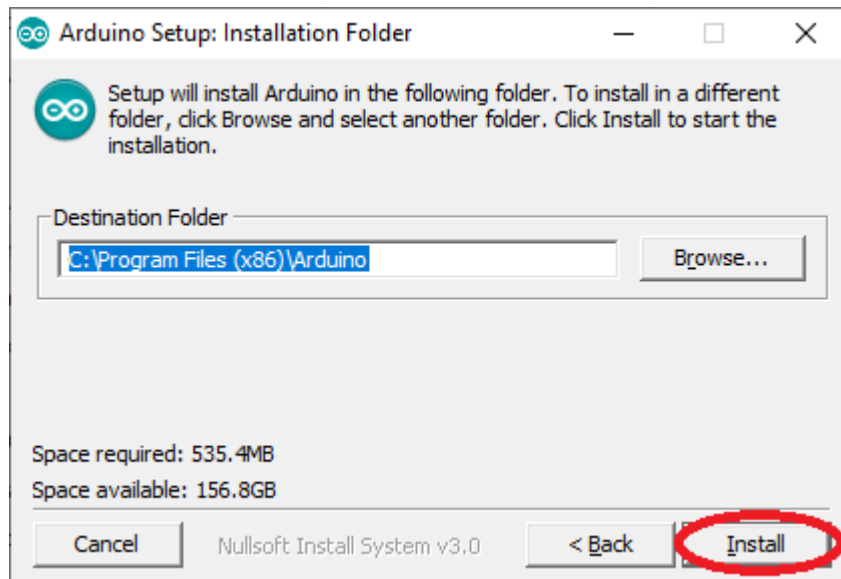
... presione el botón con el texto “sí” y la instalación se ejecutará de manera automática, se mostrará una ventana como se muestra a continuación:



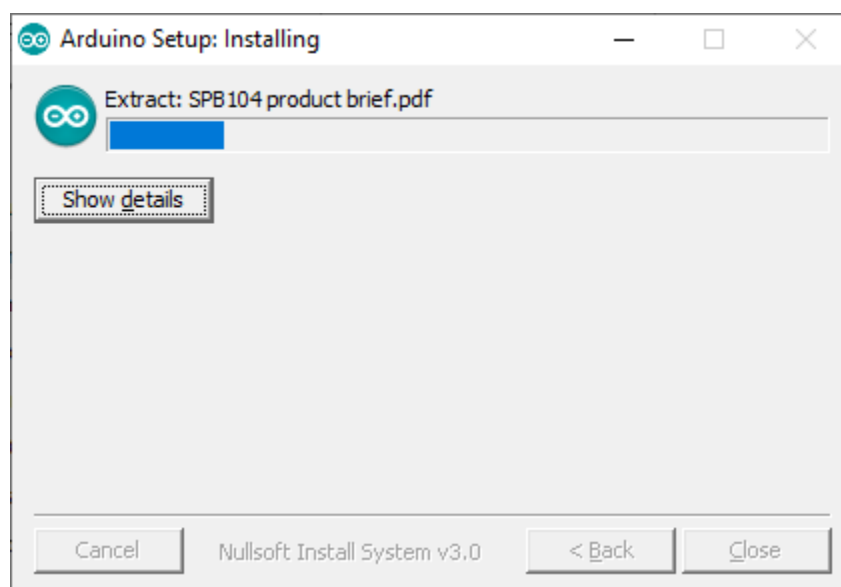
Presione el botón marcado con el texto “I Agree” y se mostrará lo siguiente:



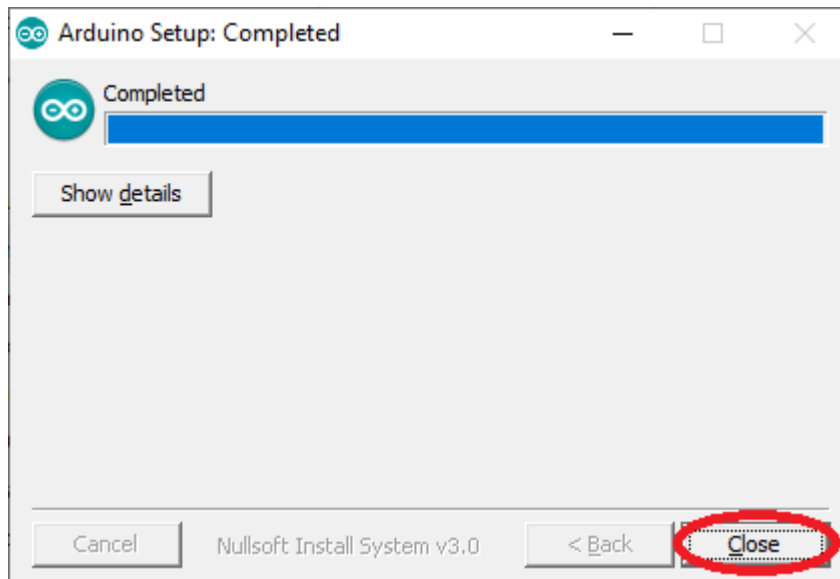
Presione el botón marcado con el texto “Next” y se mostrará lo siguiente:



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



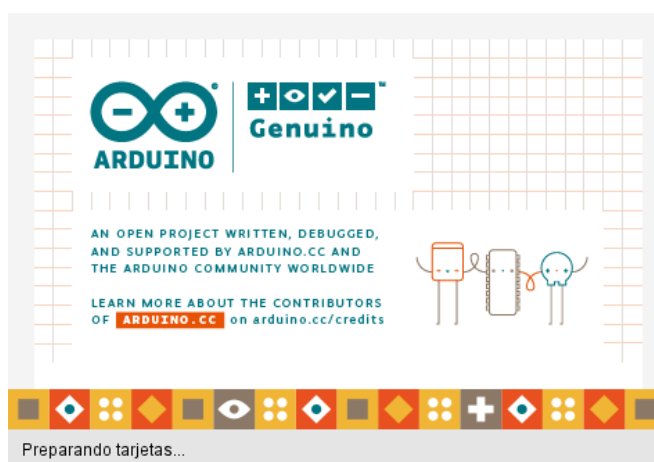
Espere a que la barra de progreso (de color azul) se llene y se mostrará lo siguiente:



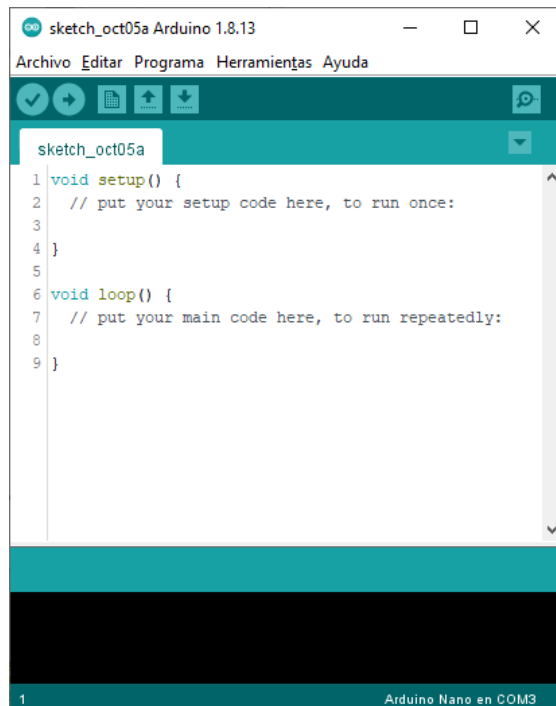
Presione el botón con el texto “Close” y la ventana anterior se cerrará. Para abrir el programa recién instalado basta con hacer doble click con el botón izquierdo del mouse al ícono ubicado en su escritorio:



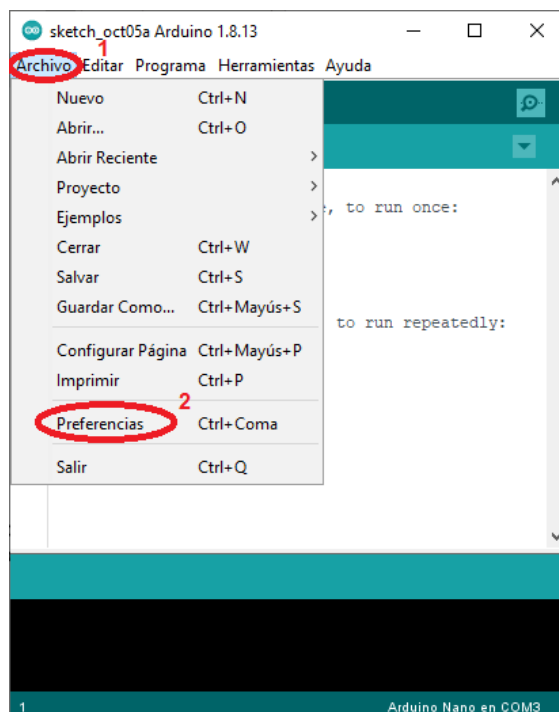
Al hacer doble click con el botón izquierdo del mouse en el ícono anterior, se abrirá como se muestra a continuación:



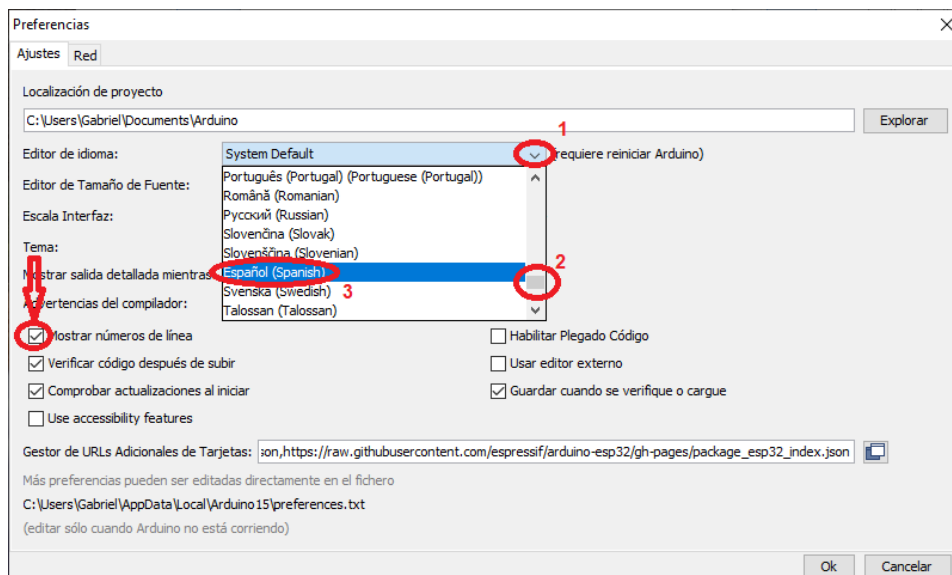
Seguidamente, se mostrará la ventana del programa Arduino:



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:

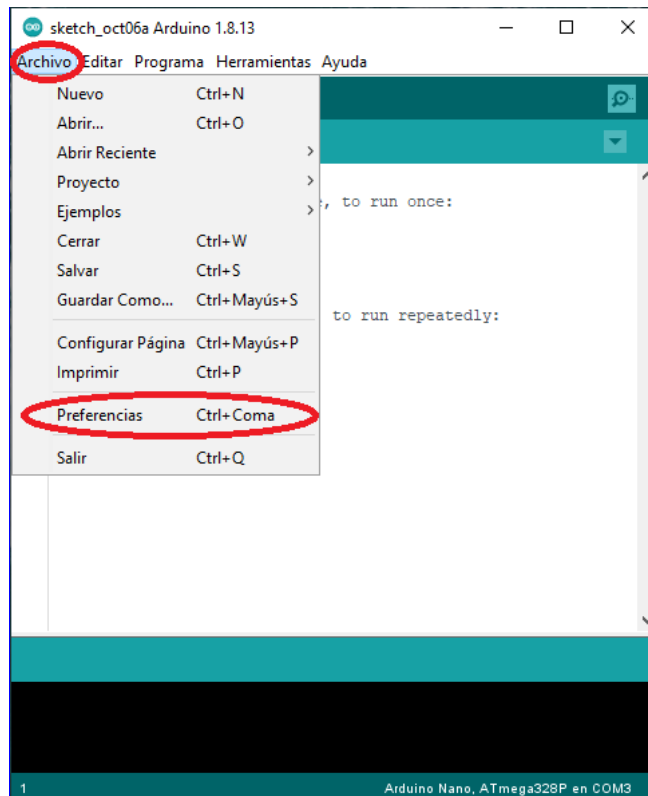


Ubique la flecha en el campo llamado “Editor de Idioma” y presione el botón izquierdo del mouse y se abrirá un menú, deslice el menú hacia abajo hasta encontrar el idioma de su preferencia, posteriormente, presione el botón izquierdo del mouse en el nombre del idioma que desee elegir. Seguidamente, **active 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.

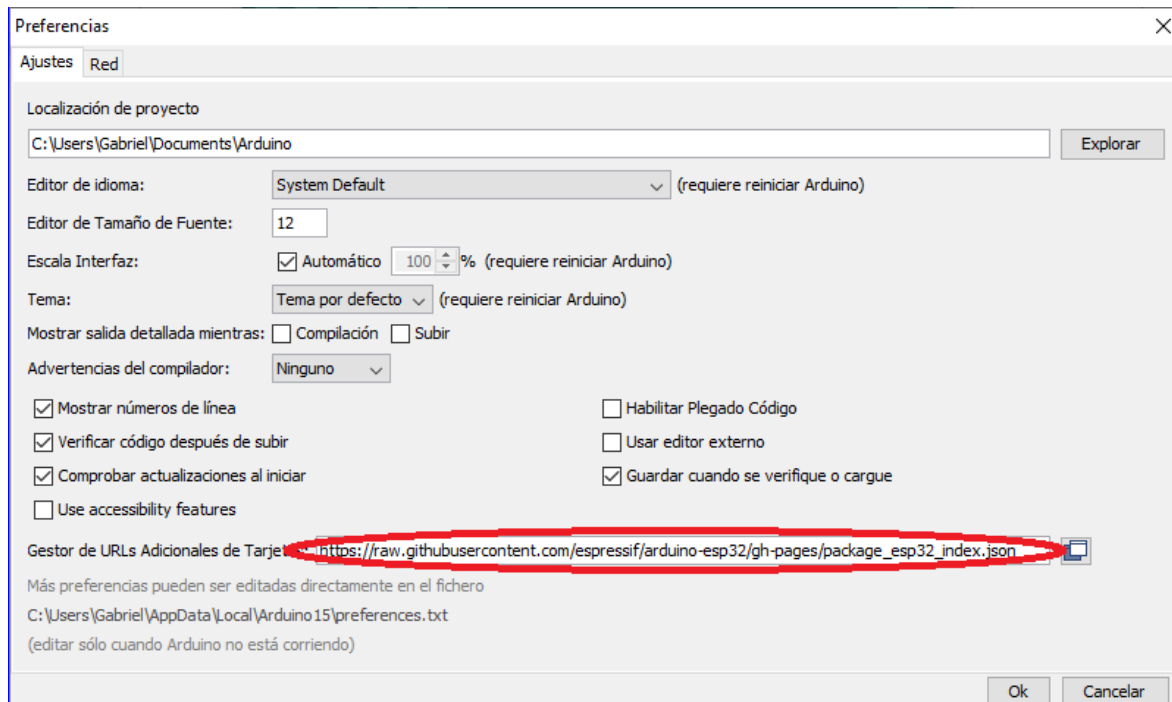
1.2 Instalación del plugin ESP32 para IDE del Arduino v1.8.13

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

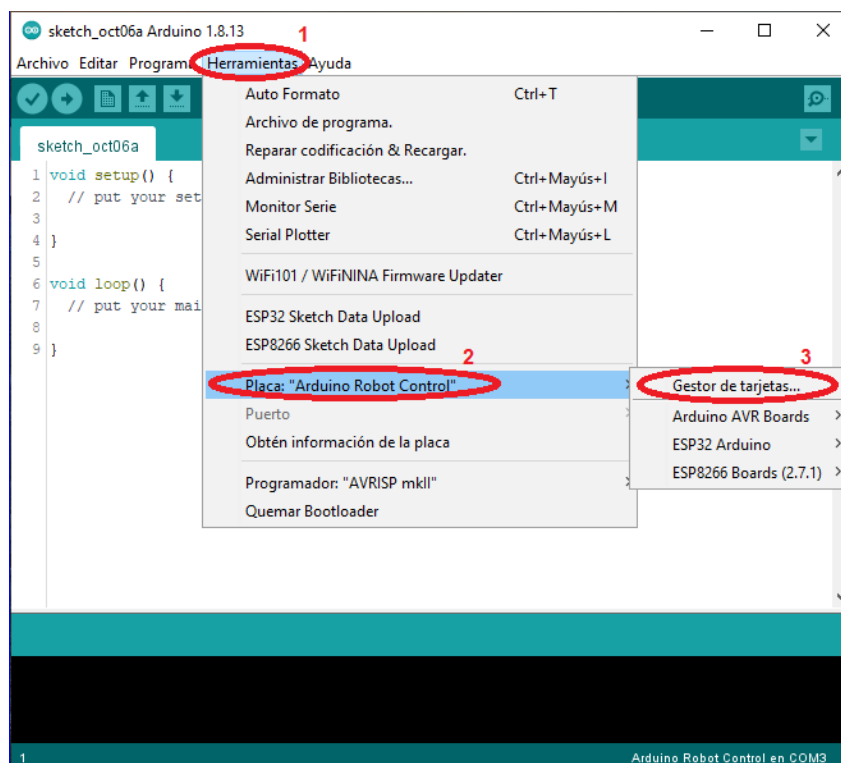
Una vez instalado el IDE del Arduino se configurará para habilitar las opciones de trabajo con los modelos de módulos EPS32. En el link: <https://github.com/espressif/arduino-esp32> se encuentran las instrucciones necesarias para cubrir este paso. Según lo anterior, abrir el IDE del Arduino, ubique la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



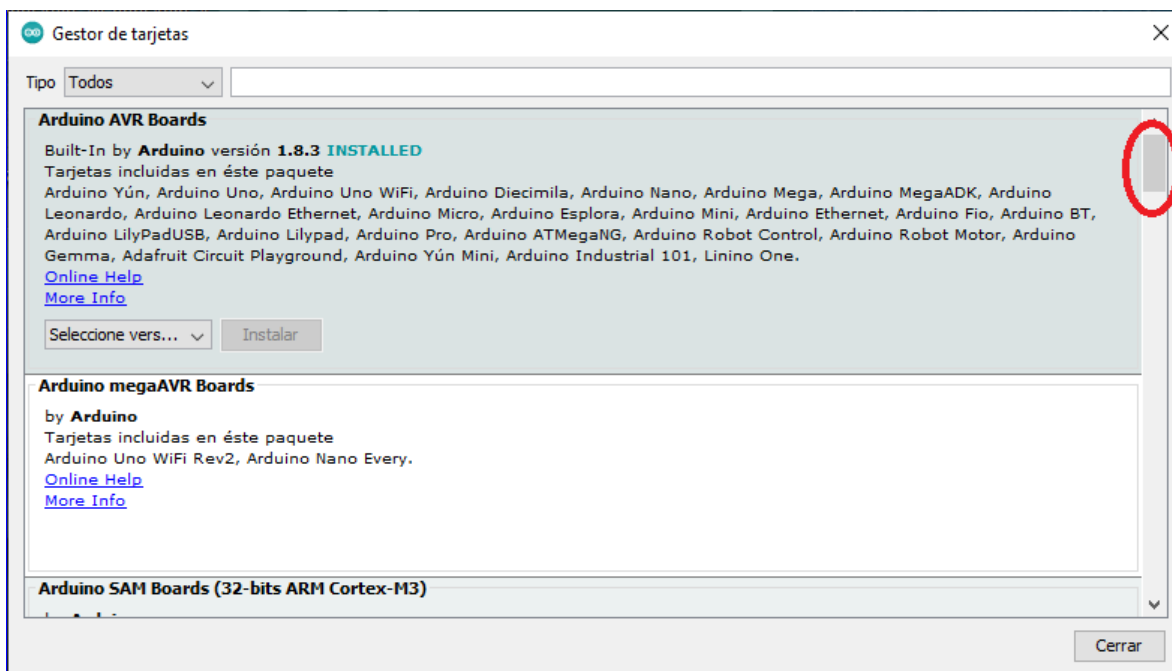
A continuación introduzca el siguiente link
https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json
en el cuadro de texto llamado “Gestor de URLs adicionales de tarjetas” como se muestra a continuación:



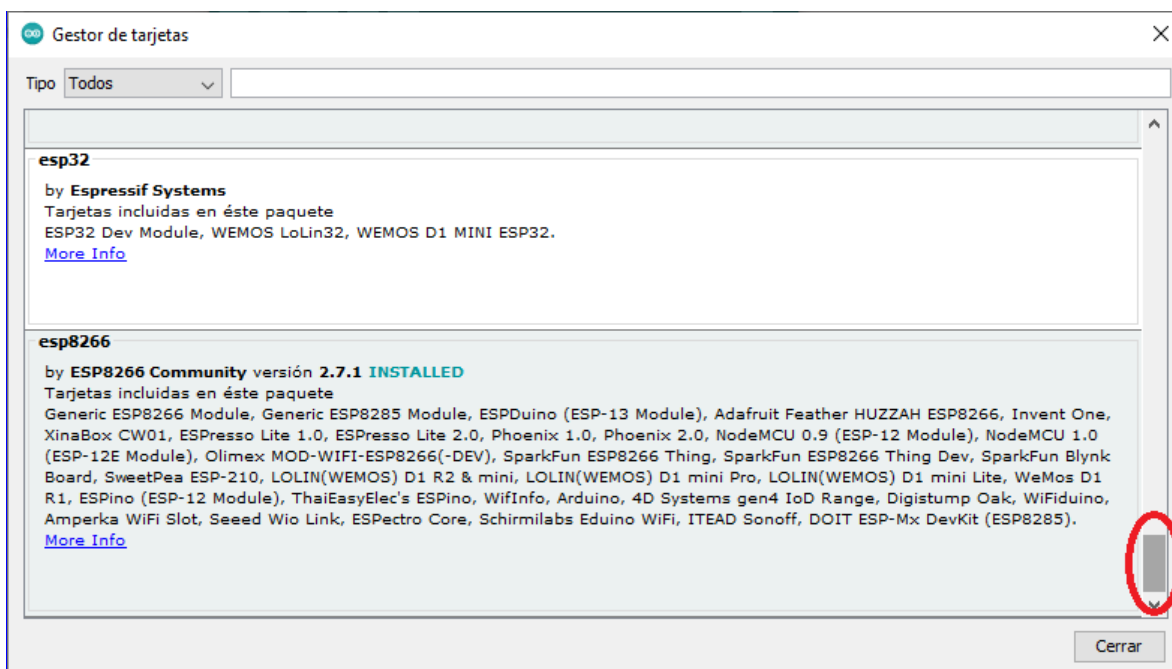
Luego, presione el botón “OK” y la ventana se cerrará. Abra la ventana del gestor de tarjetas como sigue:



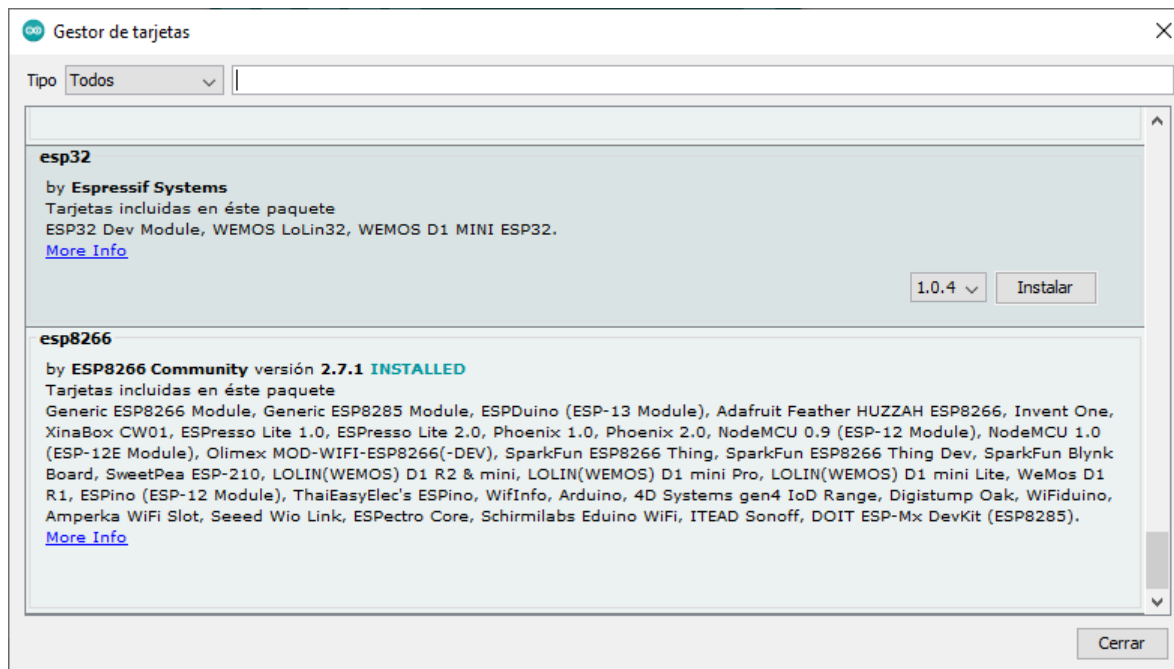
Al hacer click con el botón del mouse en la parte mostrada en la figura anterior, se abrirá la siguiente ventana:



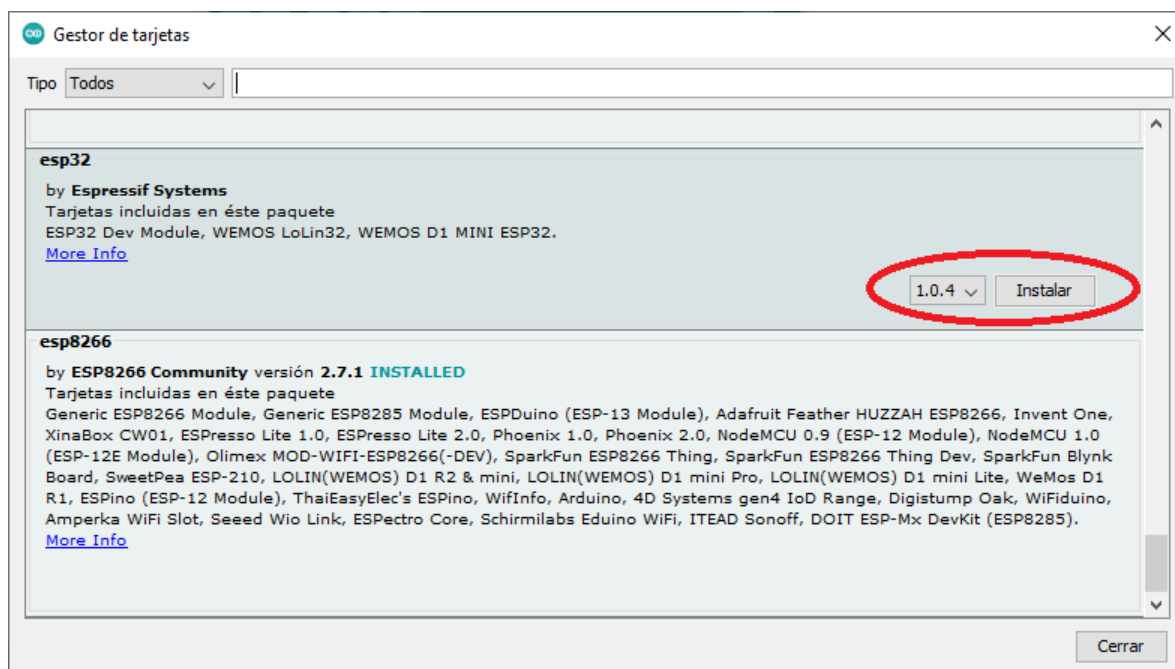
Deslice el control indicado en la figura anterior hasta el final, así como se muestra a continuación:



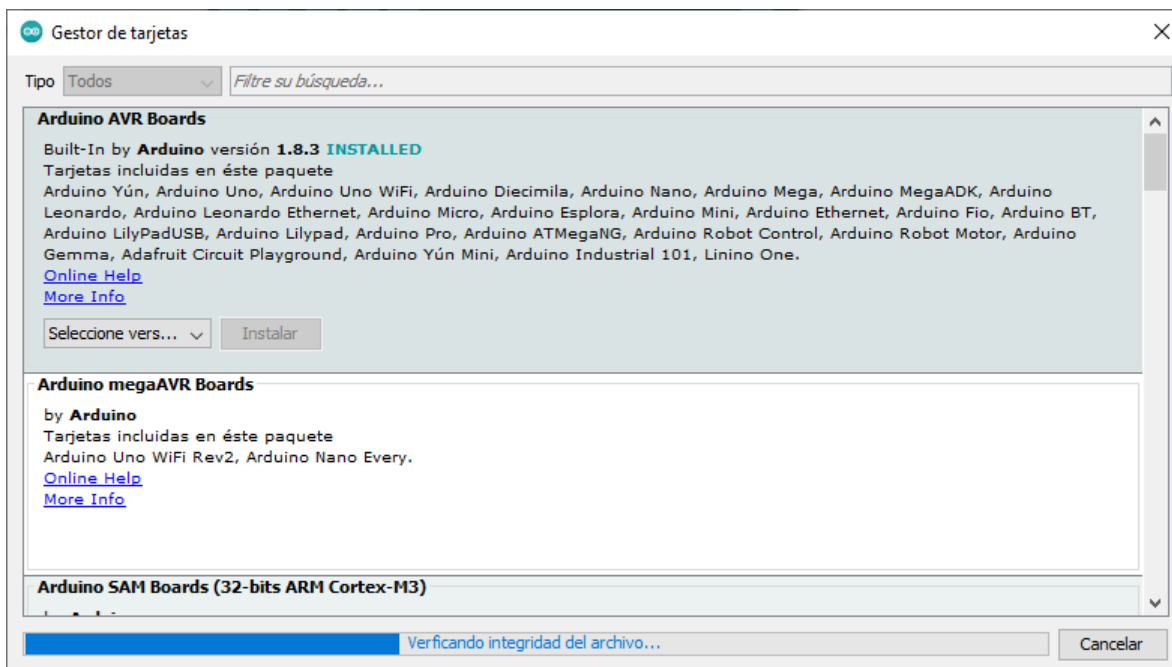
Ponga el mouse sobre el campo con el texto esp32, el área se pondrá en gris y se activarán dos botones como se muestra a continuación:



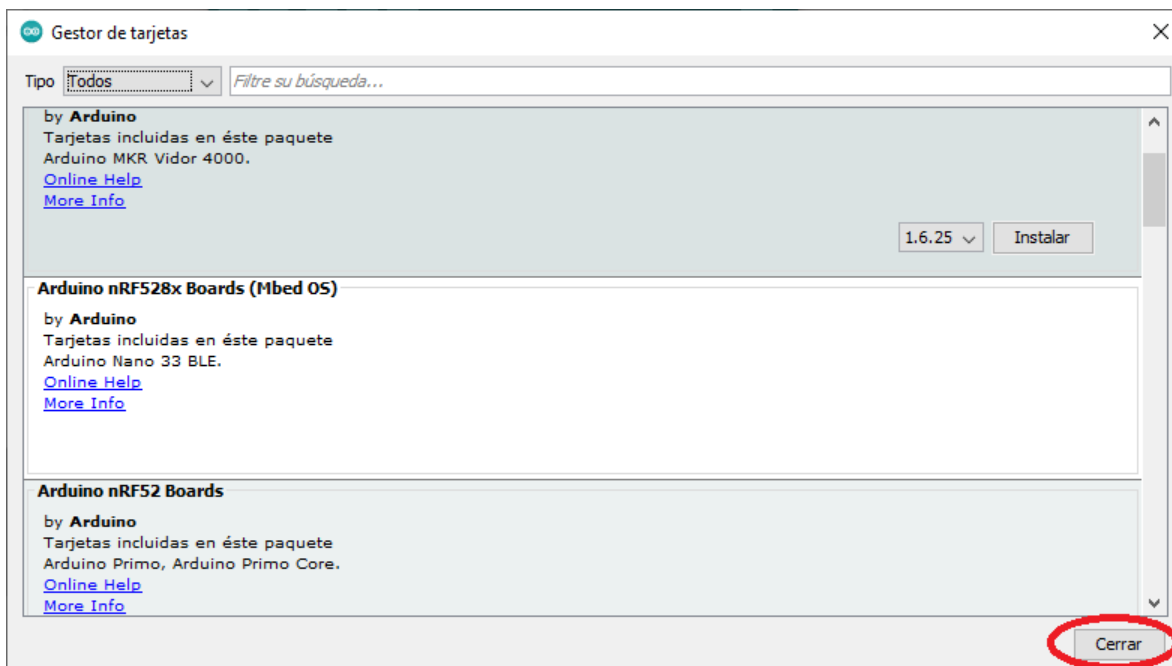
Seleccione la versión 1.0.4 y presione el botón “Instalar” e iniciará la instalación de la versión seleccionada:



Espere a que la barra de progreso termine:



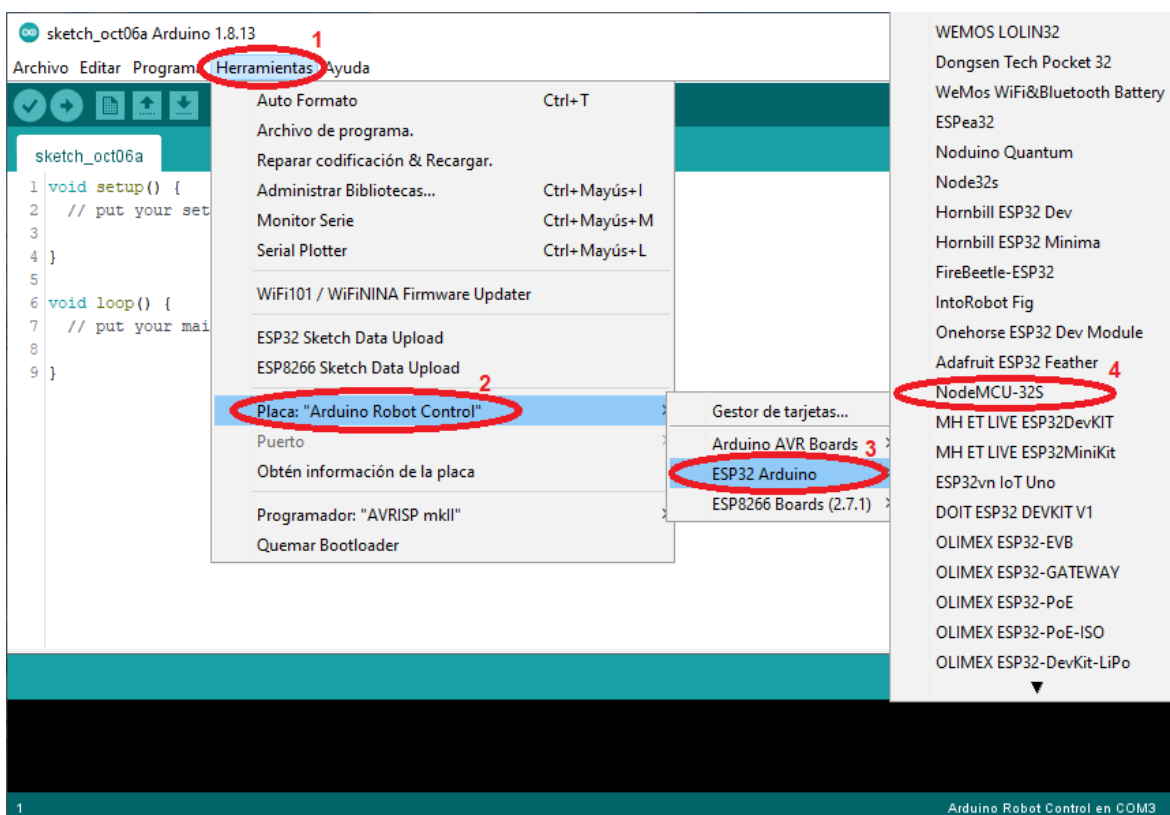
Cuando se termine de actualizar presione el botón de cerrar.



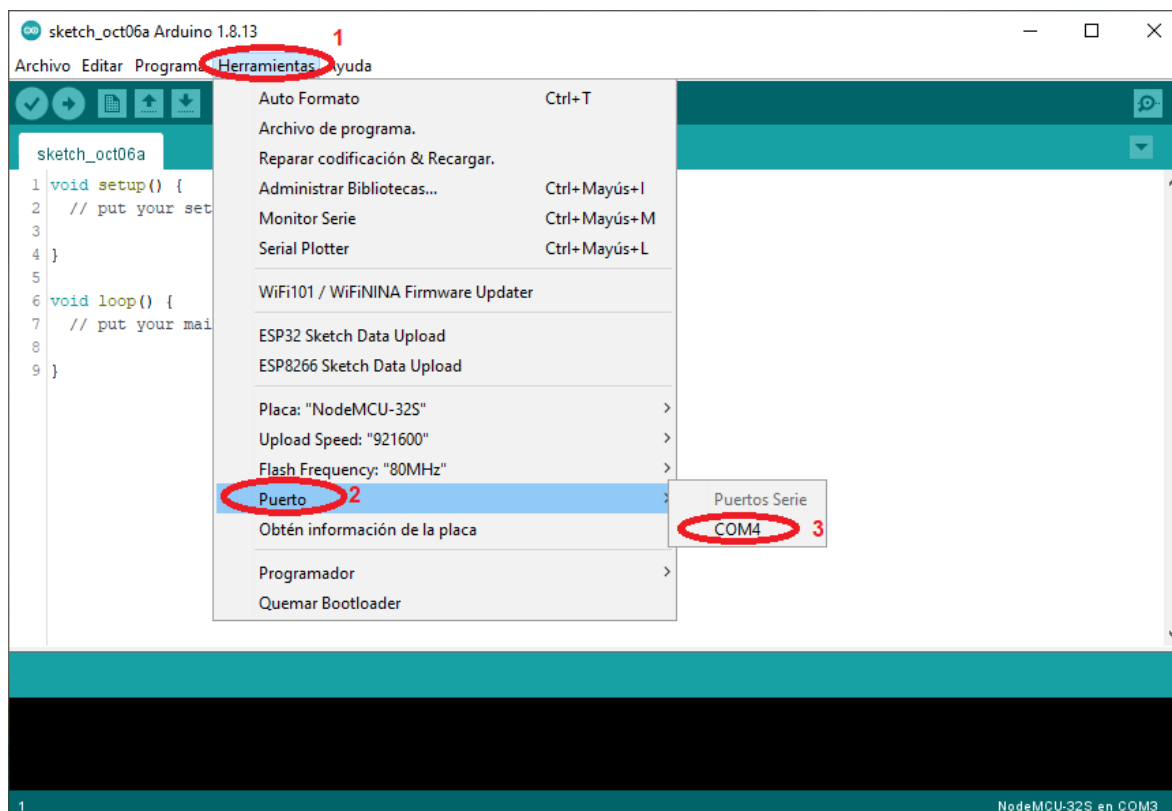
1.3 Ejecutando el ejemplo “blink”

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 siguiente 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:



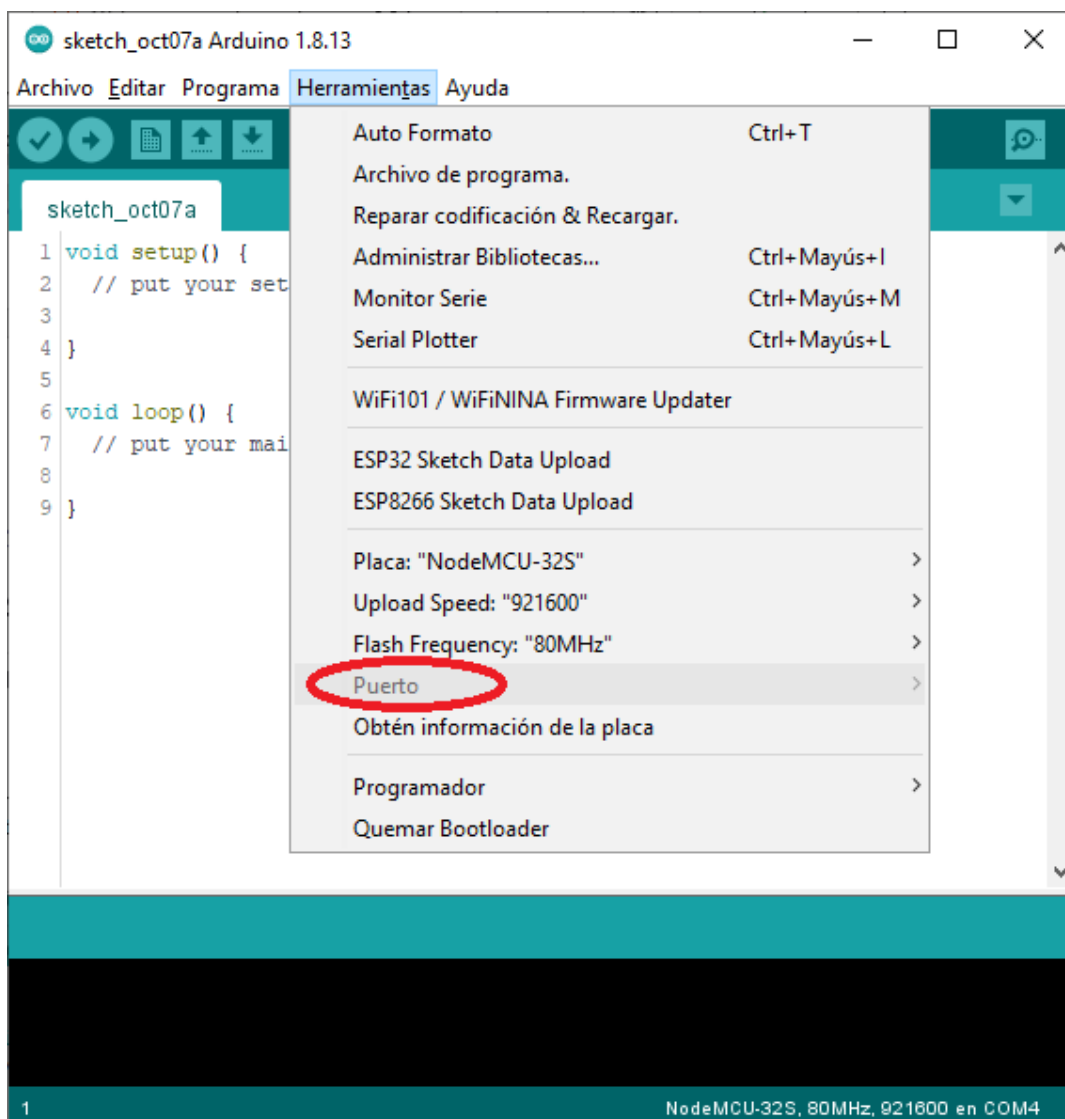
IMPORTANTE:

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 INSTALARA 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).

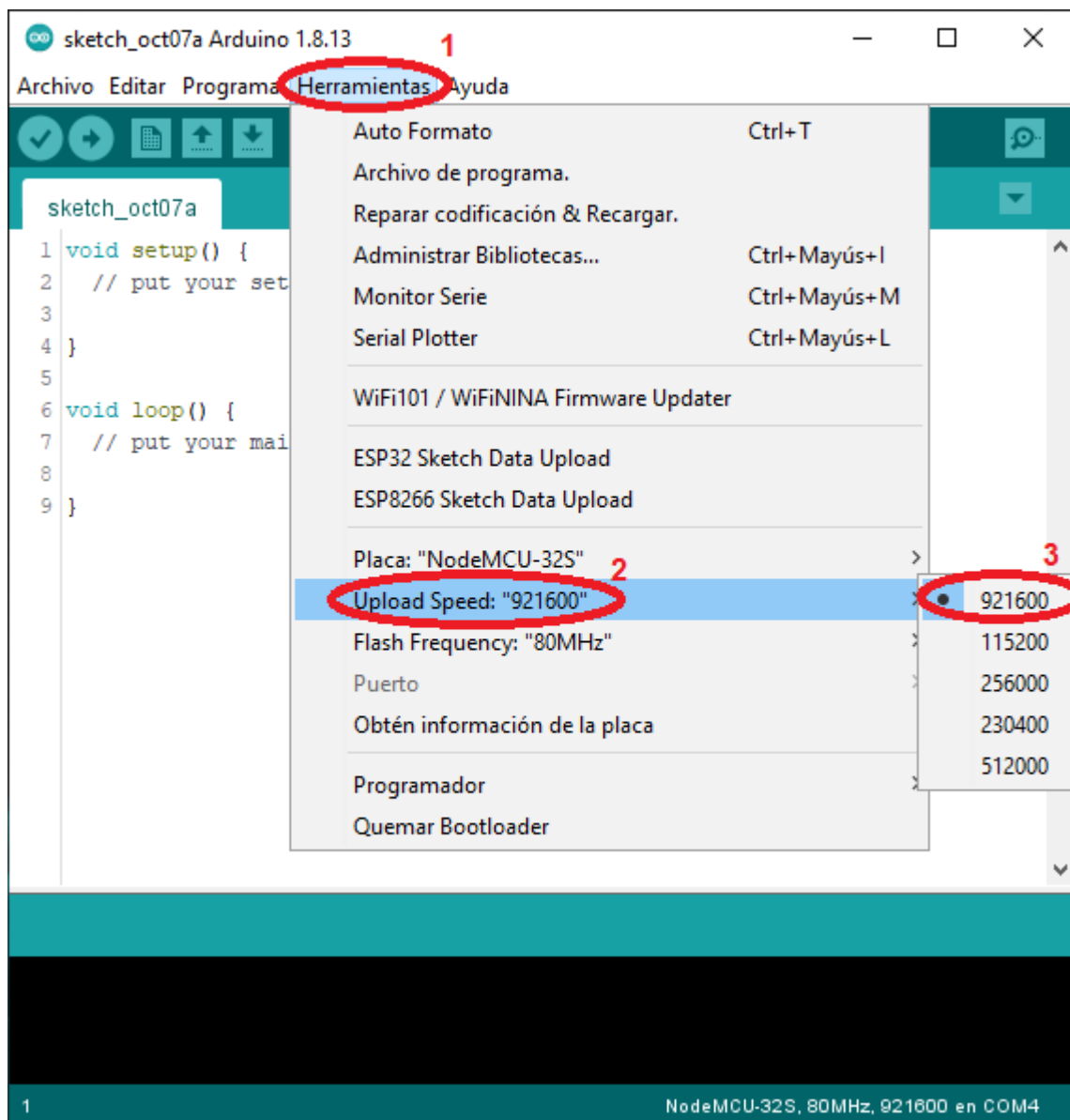
Si el puerto se muestra en gris como se indica en la siguiente figura:



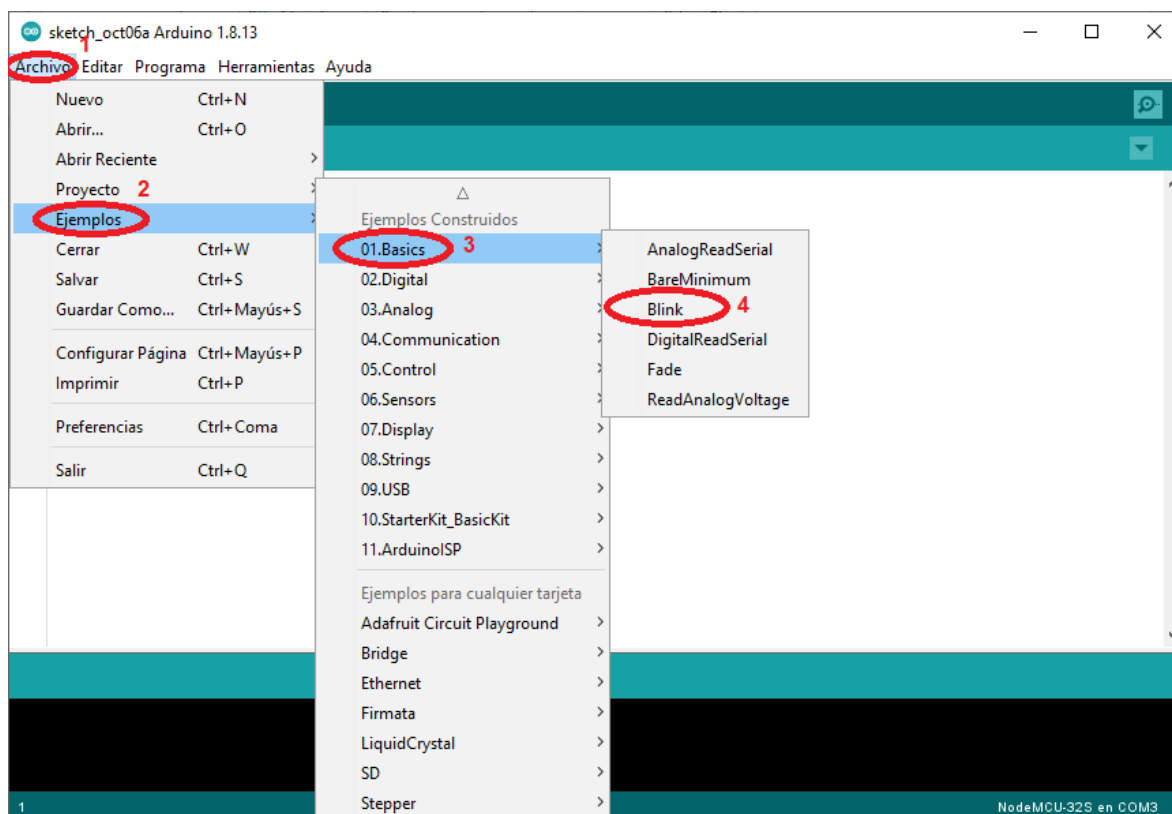
... indica que el módulo ESP32 no fue reconocido. Esto puede deberse a que:



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.

Seguidamente, 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:

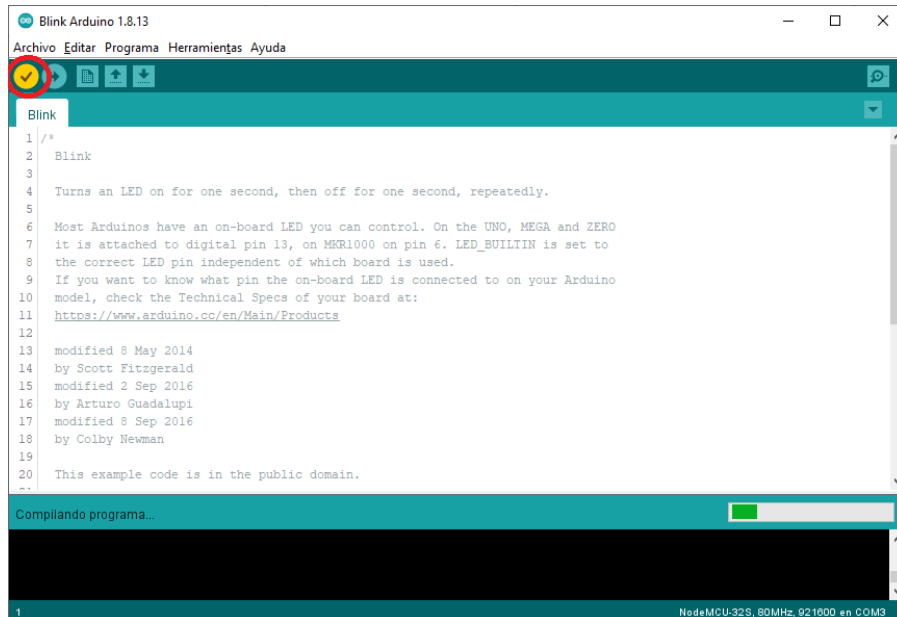


Posteriormente, se requiere escoger el ejemplo siguiente:

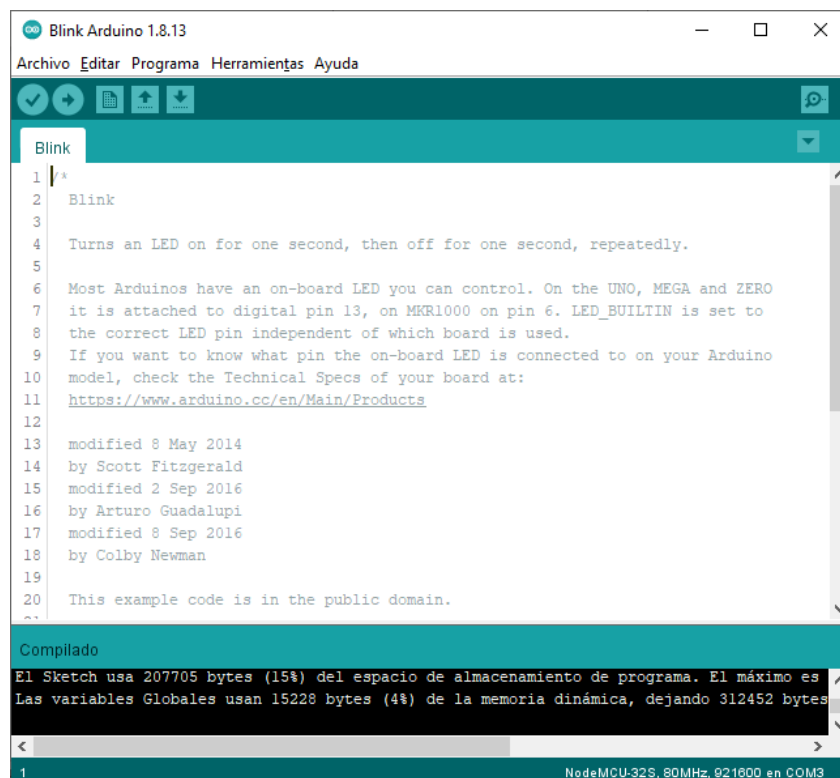


Con el botón  se verifica el ejemplo y posteriormente con el botón  se verifica y descarga el código a la tarjeta de desarrollo.

Para verificar un programa presione el botón  y nos indicará si hay errores:



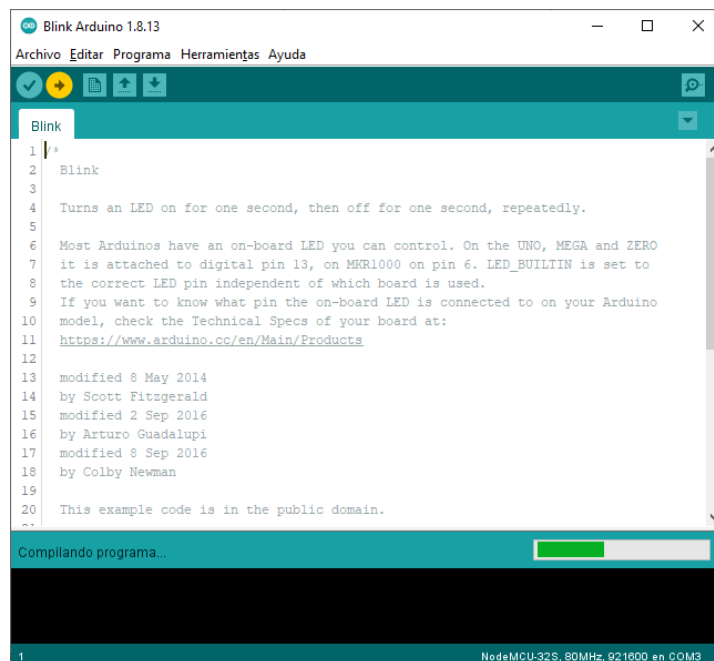
... espere que la barra de progreso termine y nos deberá de mostrar lo siguiente:



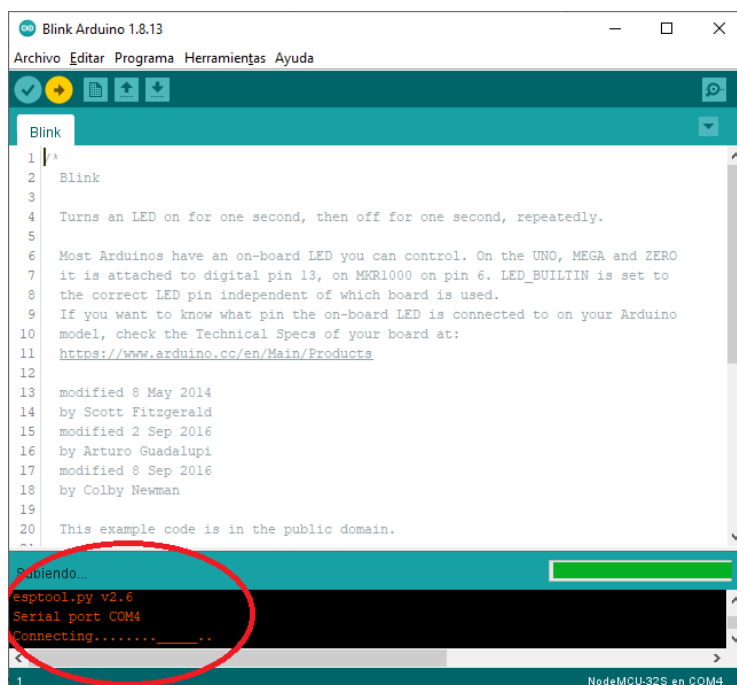
Para verificar y descargar el programa en su tarjeta de desarrollo presione el botón



y se mostrará lo siguiente:



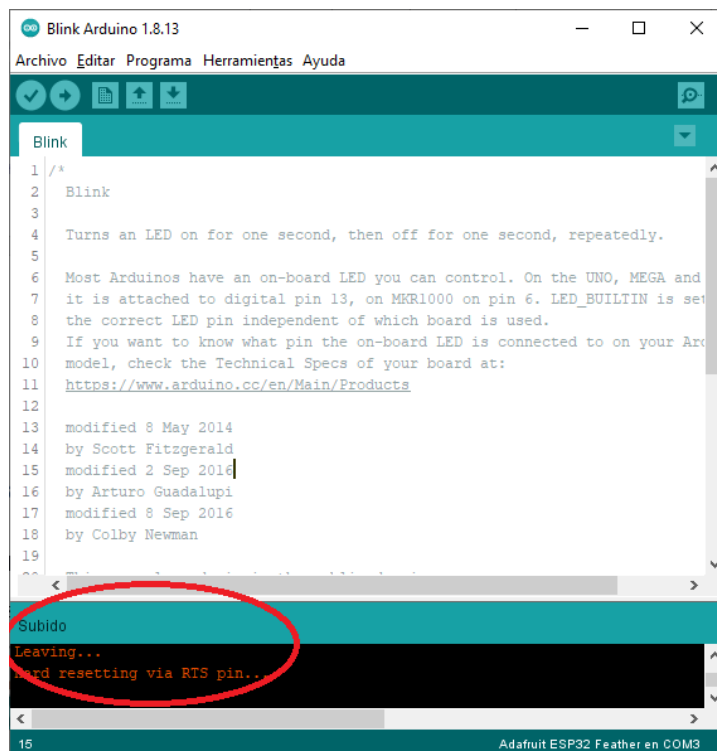
... espere a que la barra de progreso se llene y cuando se muestre lo siguiente:



Presione el botón IO0 como se muestra a continuación:



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:

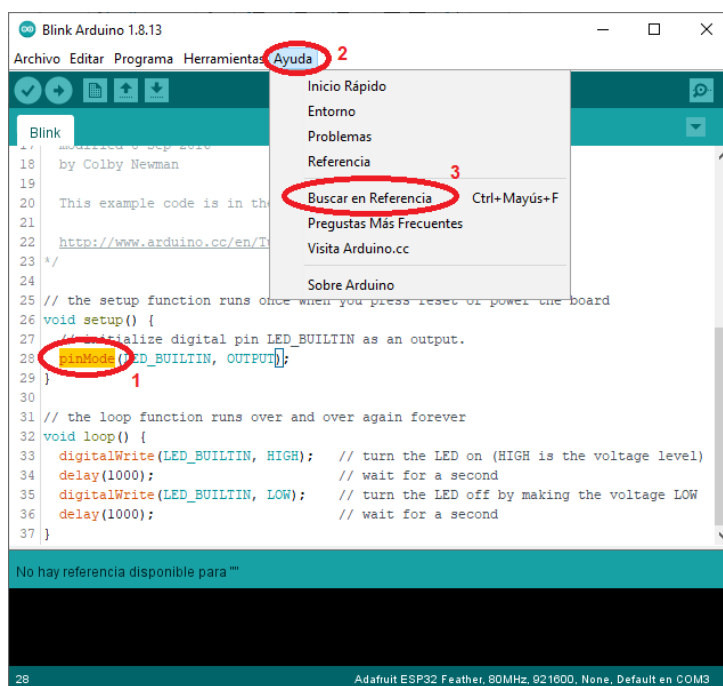


1.4 Usando la ayuda del IDE del Arduino

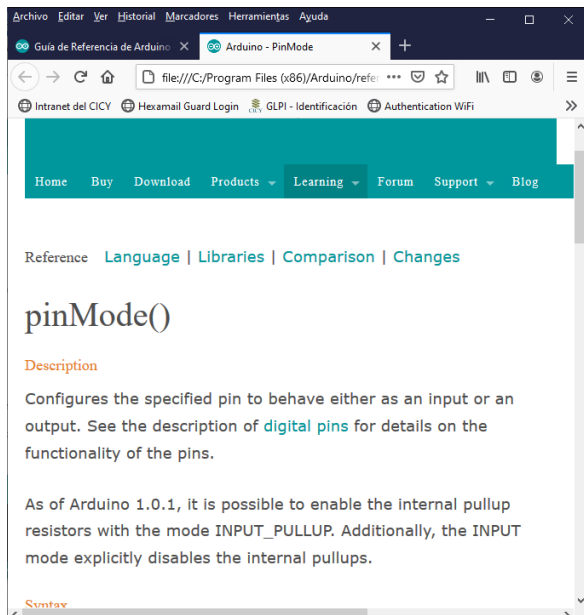
Objetivo específico: Aprenderá a usar la ayuda que ofrece el IDE del Arduino y usar el traductor que incorporan los navegadores Mozilla Firefox y Chrome para traducir la ayuda del inglés al español.

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

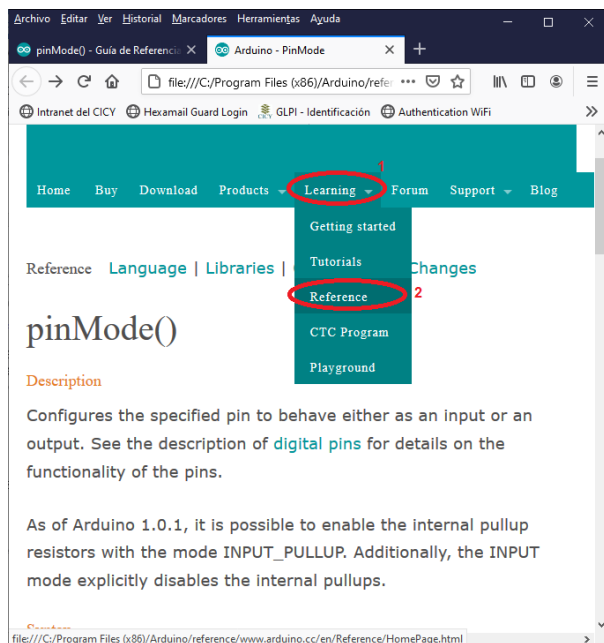
Para este punto se requiere tener abierto el ejemplo "Blink.ino" (ver punto anterior), primero seleccione la sentencia que desea conocer, dé un click en la pestaña ayuda y dé un click en "Buscar en referencia", así como se muestra:



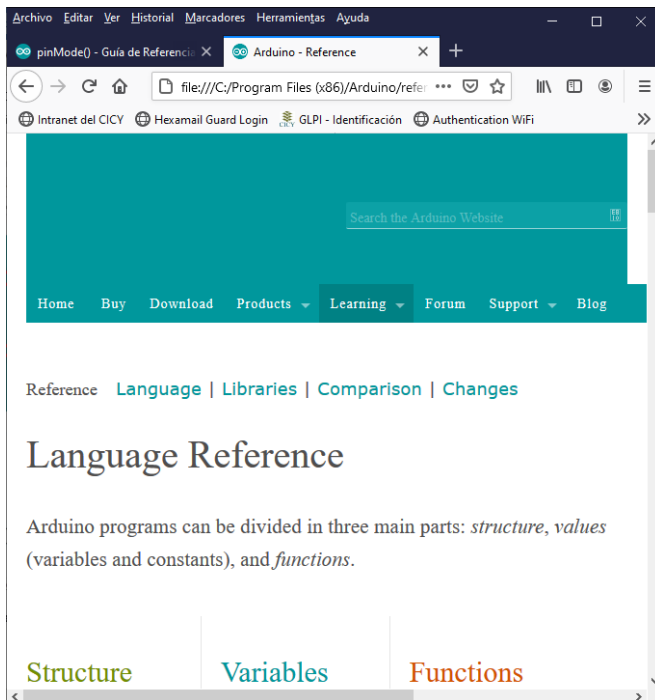
Se abrirá el navegador de internet y mostrará lo siguiente:



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:

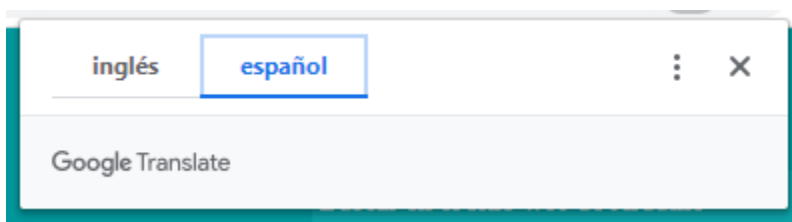


Si dispone de conectividad de internet, la ayuda la encontrará en español:

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

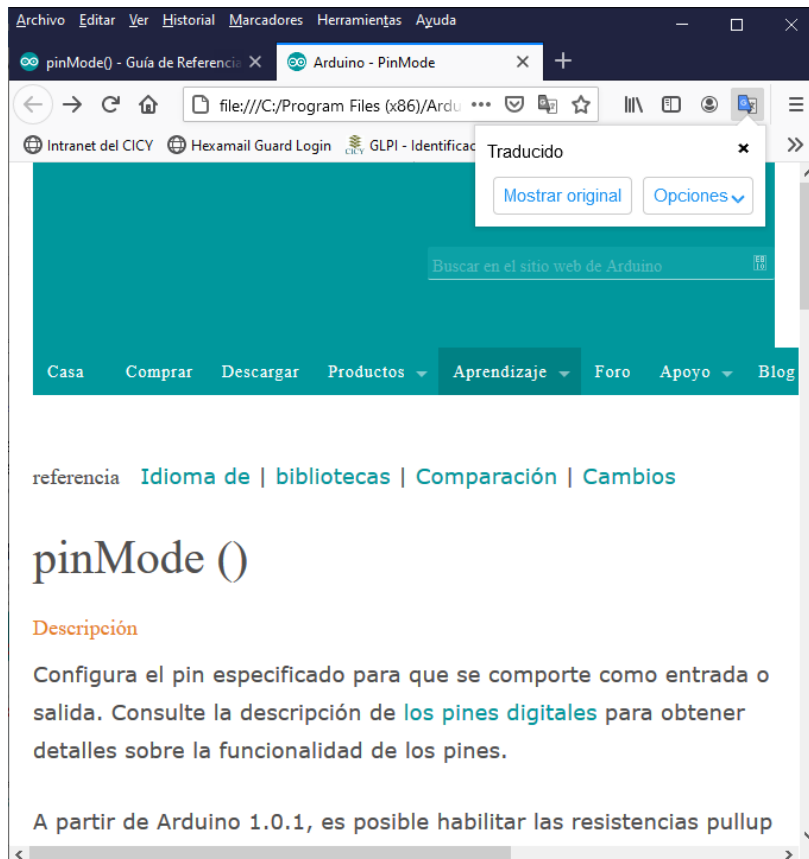
... pero la descripción de las funciones estará en inglés. ☹️

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
  retardo ( 1000 ) ;              // espera una segunda
  escritura digital ( ledPin , LOW) ; // establece el apagado del LED
  retardo de ( 1000 ) ;          // espera un segundo
}
```

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

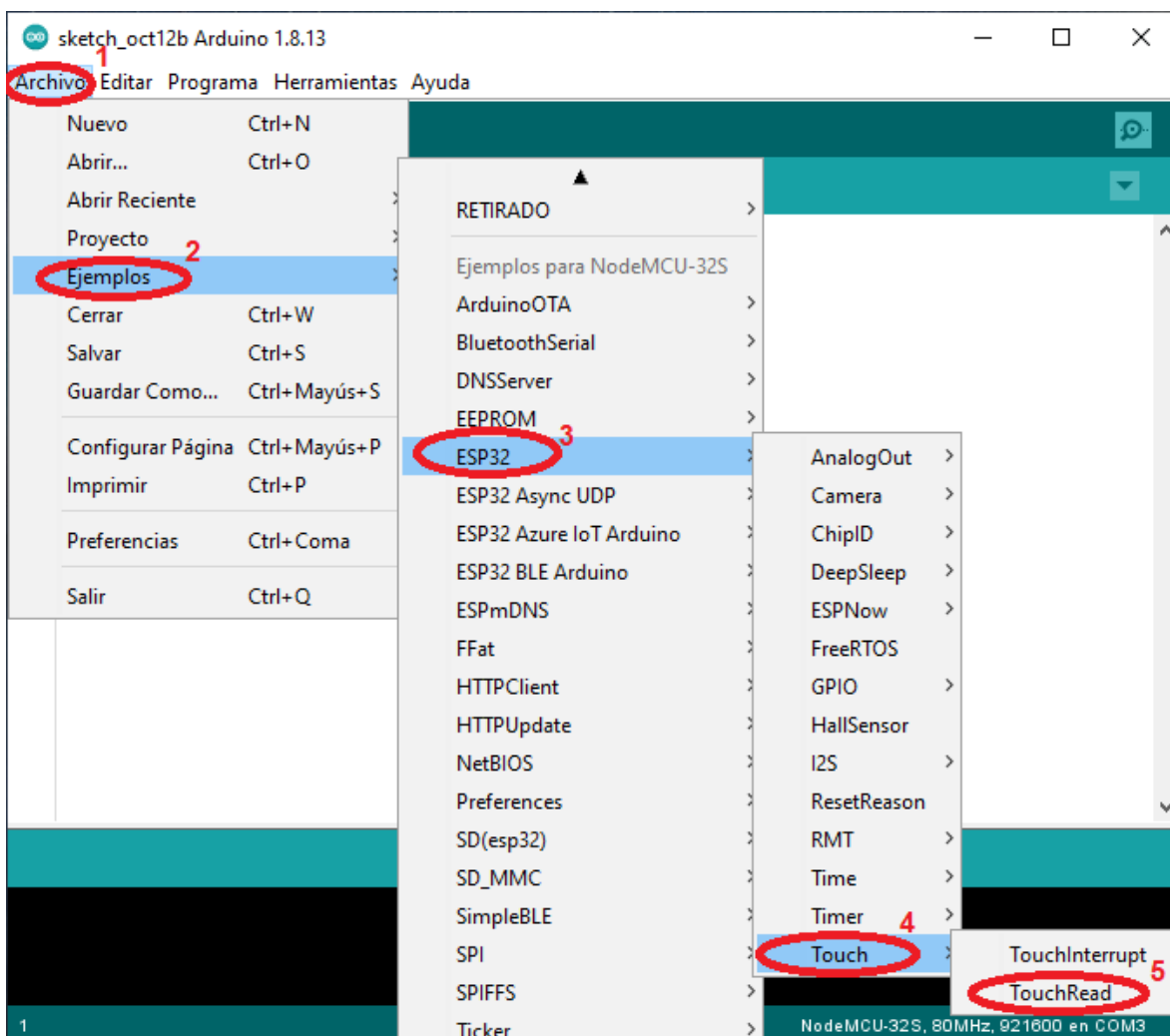
```
// la función setup() se ejecuta una vez al reiniciar o encender la placa
void setup() {
  // inicializa el pin digital llamado LED_BUILTIN como salida.
  pinMode(LED_BUILTIN, OUTPUT);
}


// 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
voltaje)
  delay(1000);                        // espera un segundo
  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

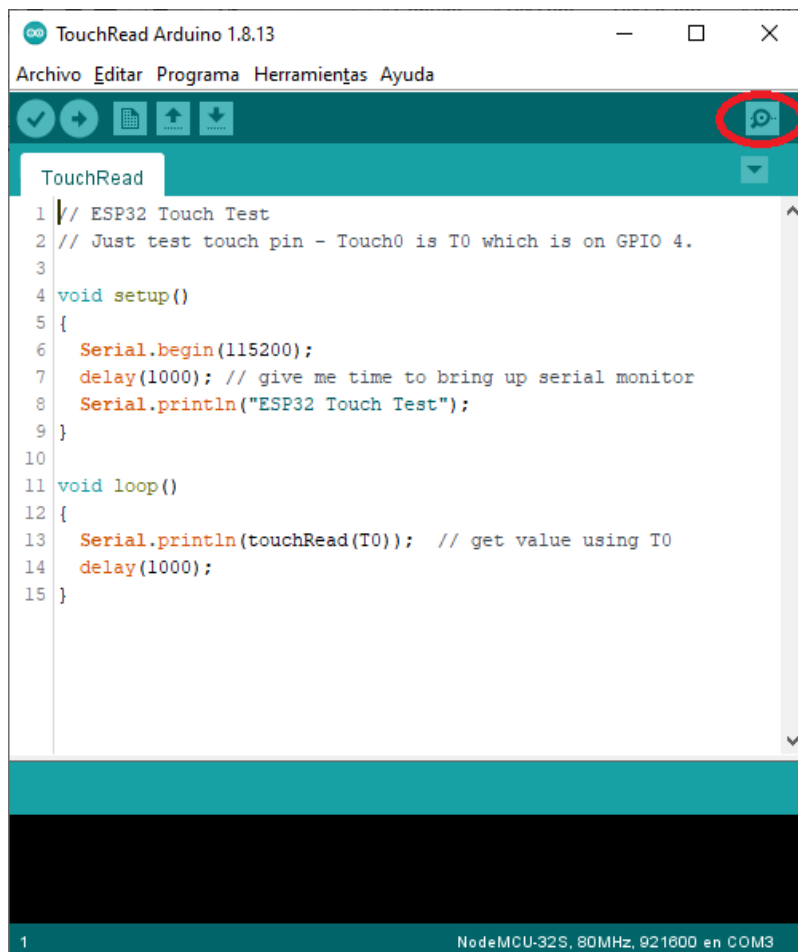
Objetivo específico: Utilizará el monitor del puerto serie para mostrar datos de variables.

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

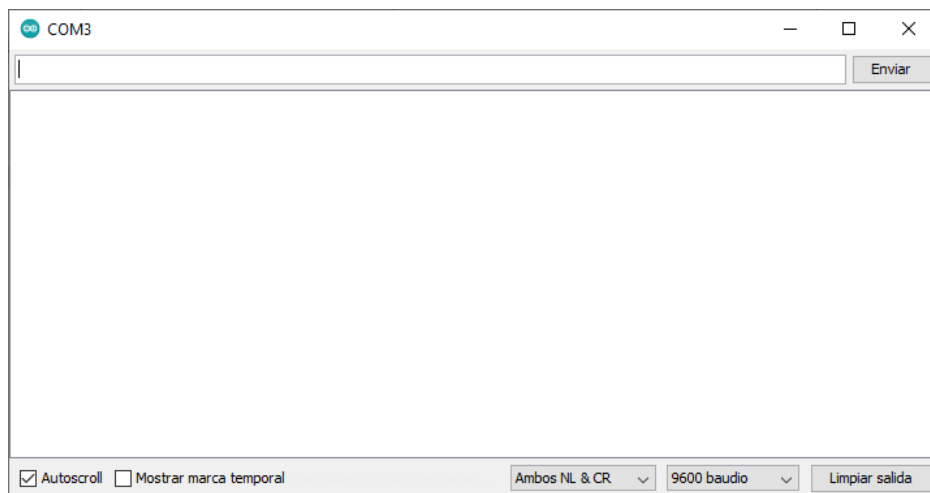


No olvide que: Con el botón  se verifica y descarga el código a la tarjeta de desarrollo.

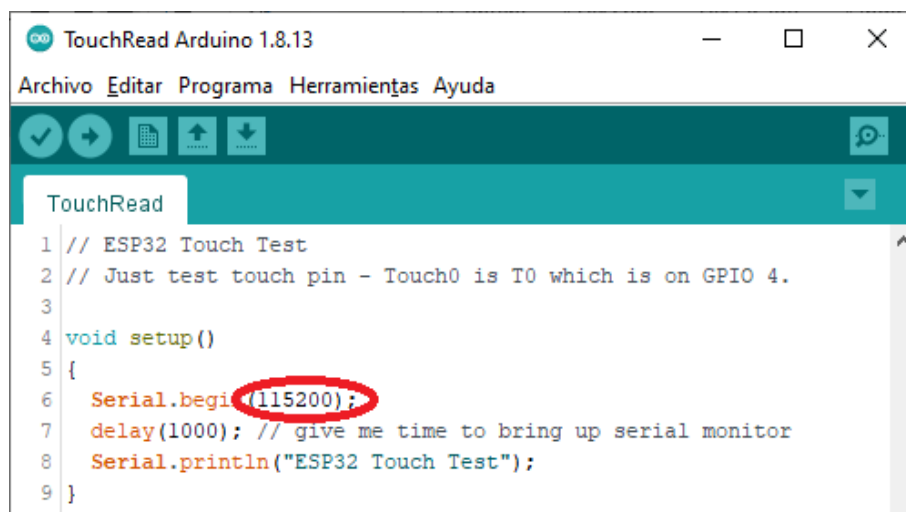
A continuación, ubíquese en la parte superior derecha del IDE del Arduino y dé un click al botón del mouse, así como se muestra a continuación:



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



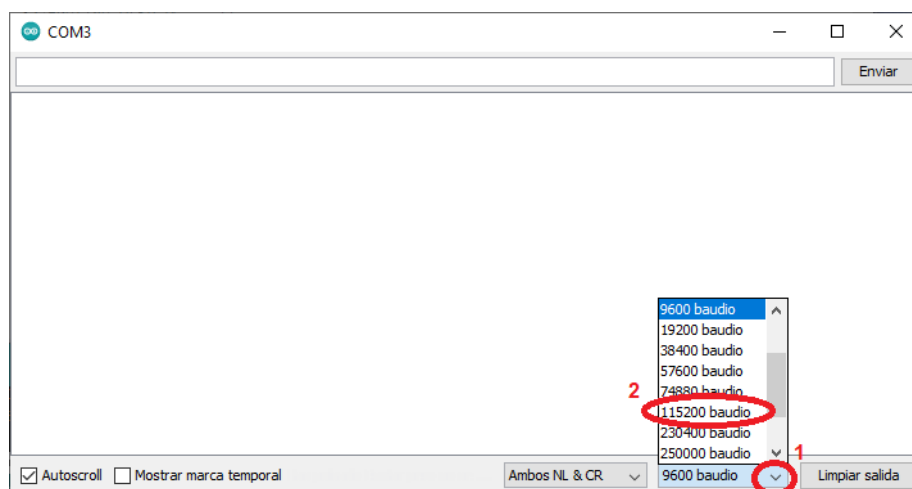
Regrese a la ventana del IDE del Arduino, observe y memorice el valor que se declara en la línea 6 del código:



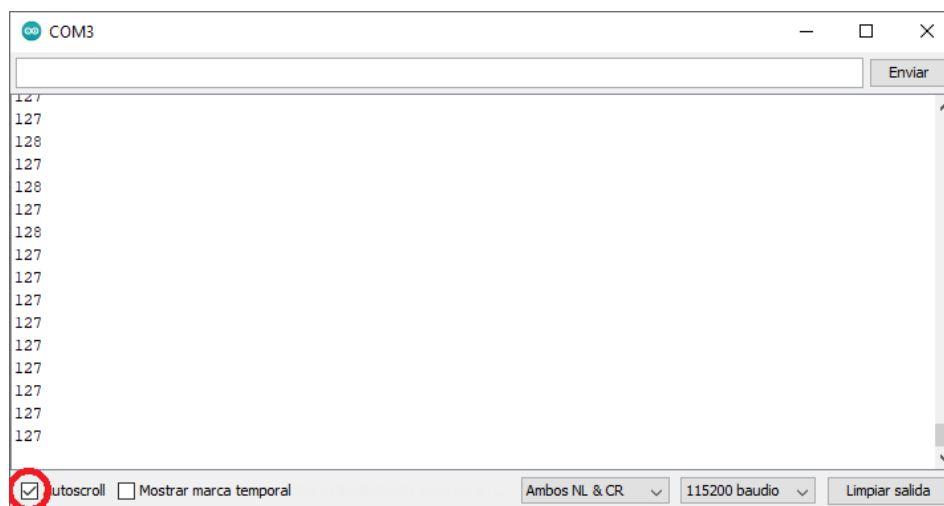
```

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 }
  
```

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:



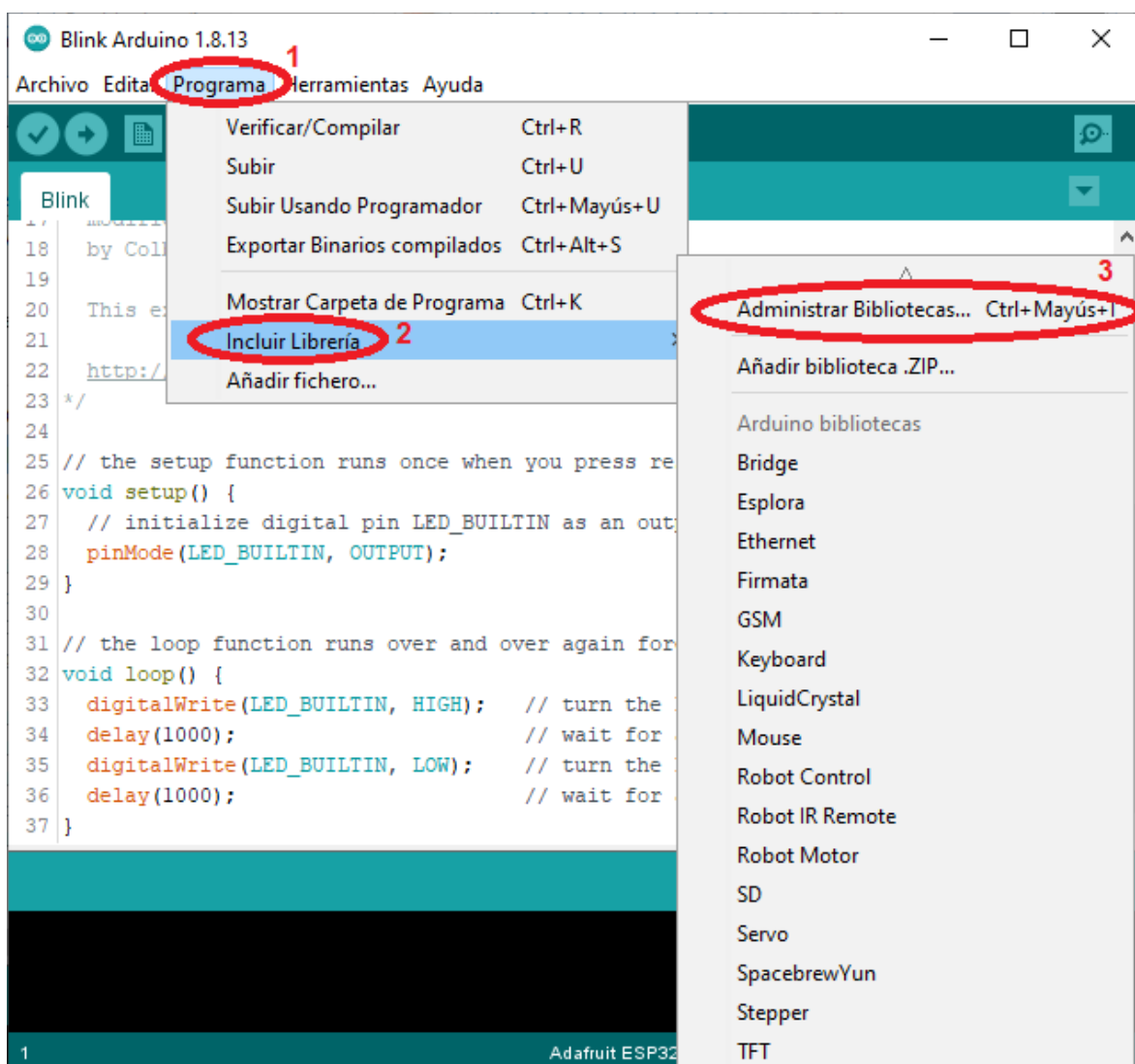
Asegúrese de haber activado el checkbox con la etiqueta “autoscroll”, esto con el objetivo de visualizar el último valor enviado por la tarjeta de desarrollo.

1.6 Agregando bibliotecas al IDE del Arduino del catálogo de drivers

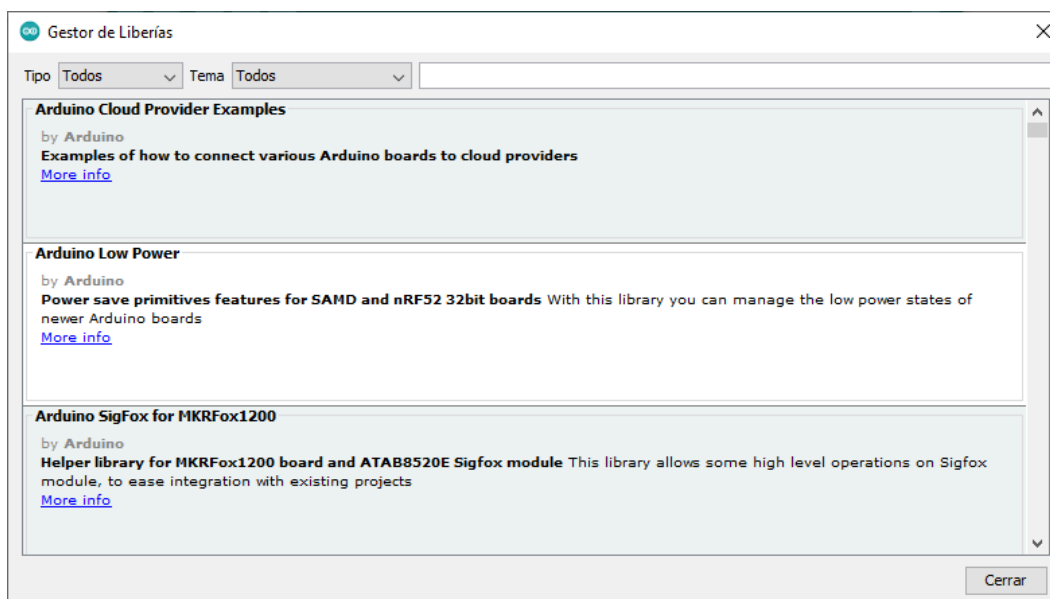
Objetivo específico: Aprenderá la manera de instalar las bibliotecas para el hardware que pretenda utilizar en su proyecto, en este caso: pantalla OLED Mod SSD1306, la biblioteca gfxlibrary (con sus dependencias), el sensor DS18B20 y el sensor DHT22 (AM2301).

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

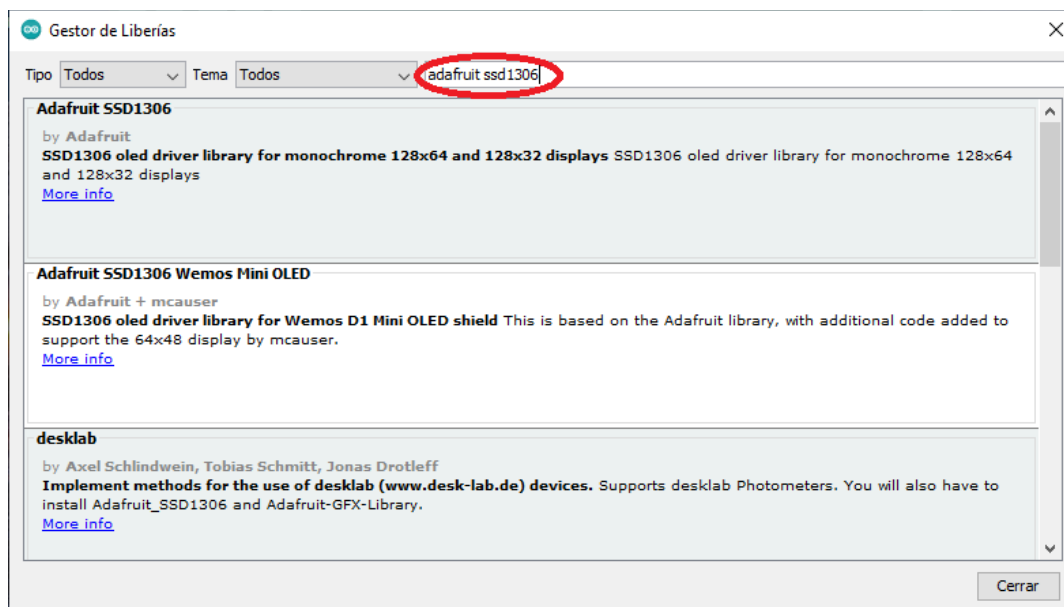
Se inicia presionando de manera secuencial las opciones que se muestran a continuación:



Y se abrirá una ventana como se muestra a continuación:

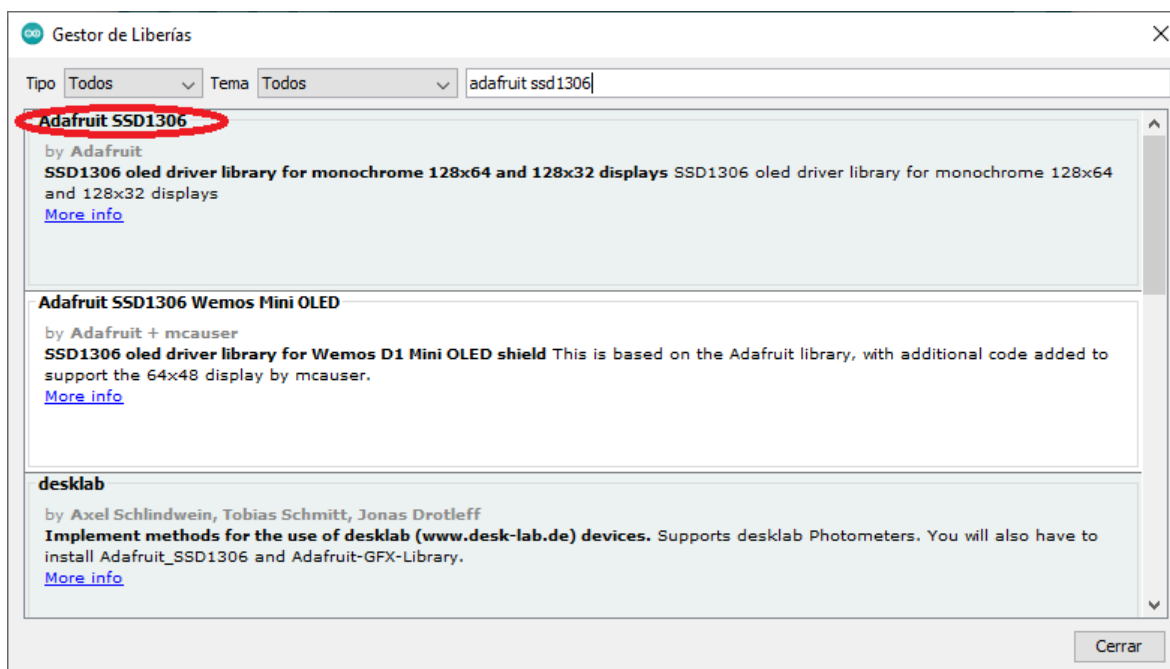


Ubique el campo de texto con la etiqueta “Tema” y escriba el nombre de la biblioteca que usted desea, en este caso escriba “adafruit ssd1306”:

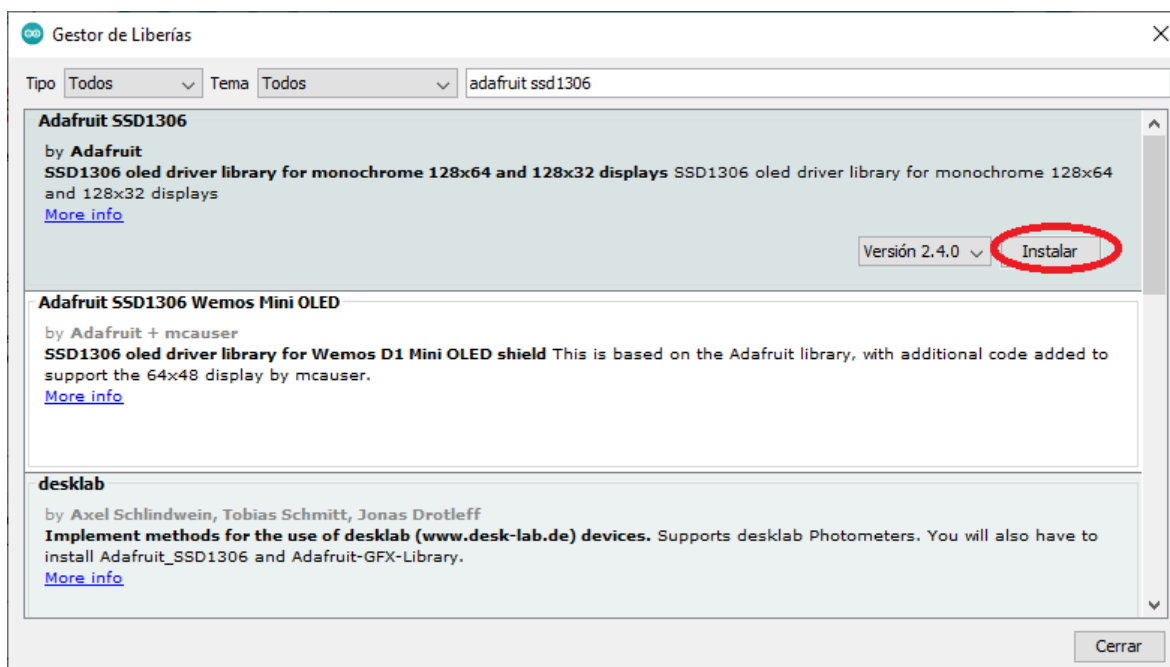


El gestor de bibliotecas automáticamente lo buscará en la base de datos y posteriormente mostrará las versiones disponibles, en este caso la biblioteca

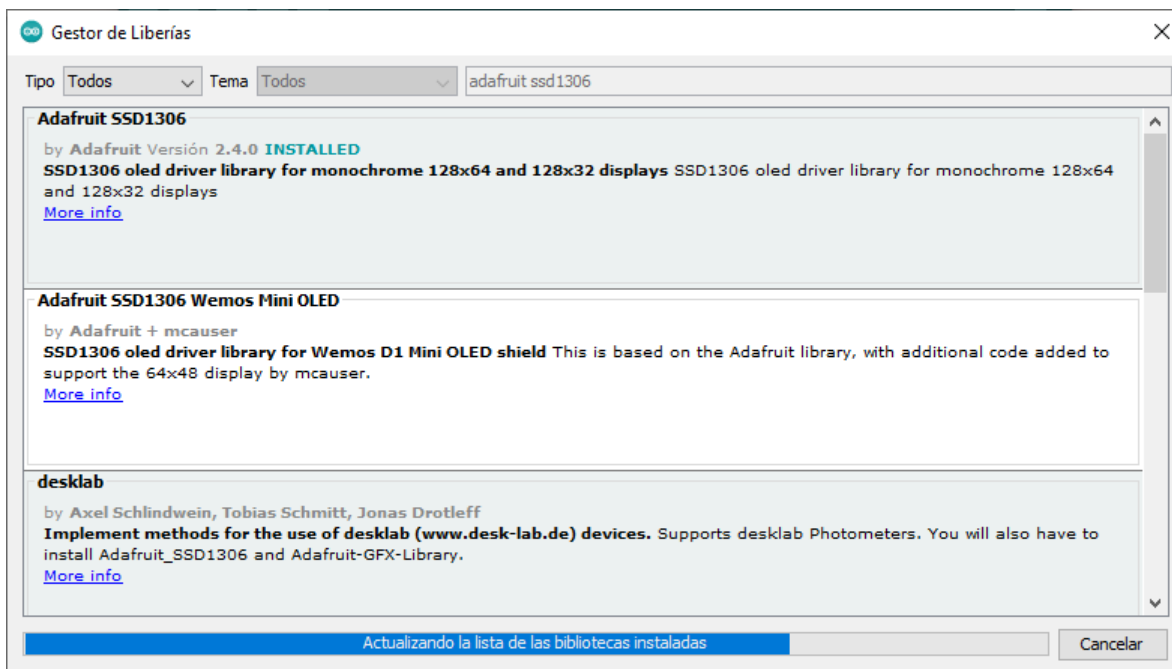
deseada, será la que se muestra primero:



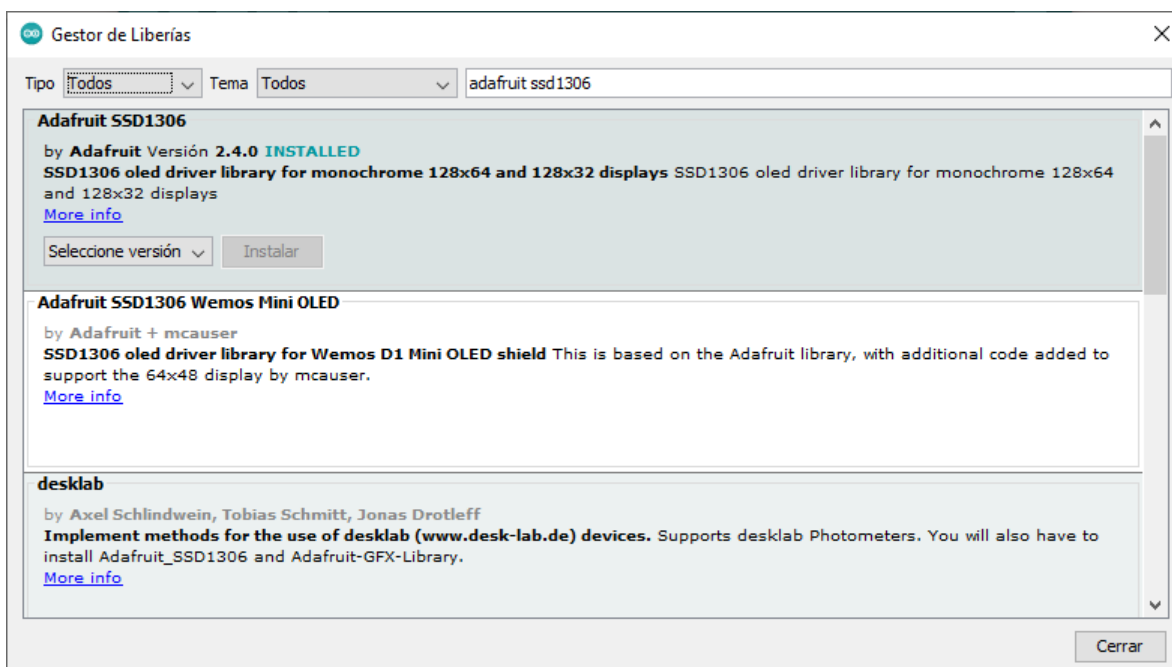
Ponga el cursor del mouse en el campo con el texto Adafruit SSD1306, se mostrarán la versión y un botón llamado “instalar”, presione el botón “Instalar”.



Espere a que la barra de progreso se complete:

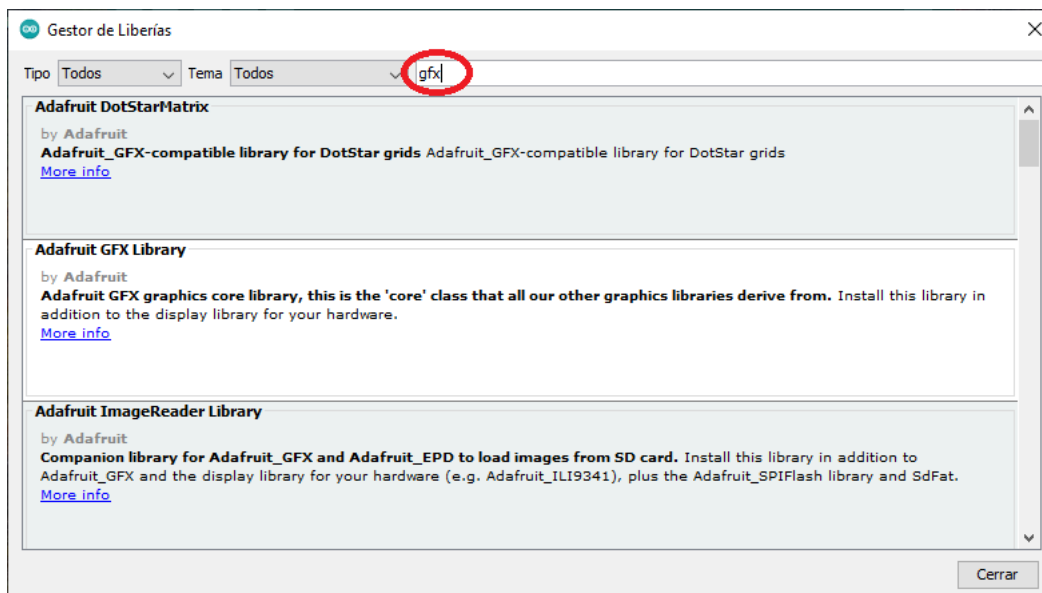


Cuando en la esquina inferior derecha se muestre el botón “cerrar”, indicará que la biblioteca ha sido instalada, sin embargo, aún no presione el botón:

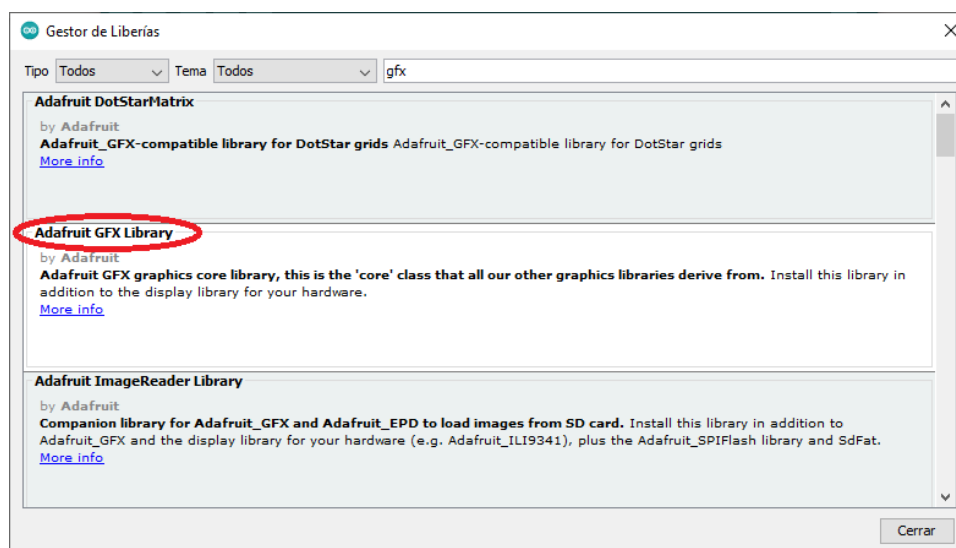


Esta biblioteca depende de otra llamada GFX Library y se instala de la misma

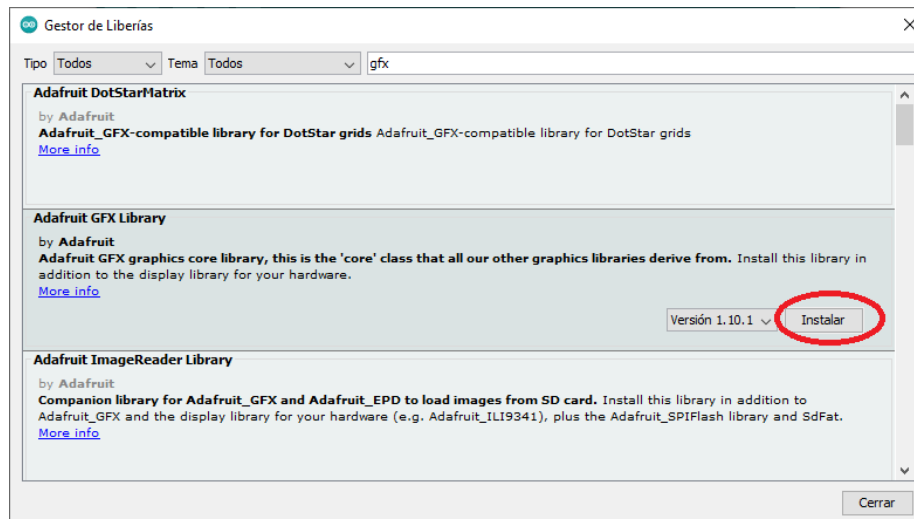
manera, borre lo anterior y teclee “gfx” en el campo de tema, así como se muestra:



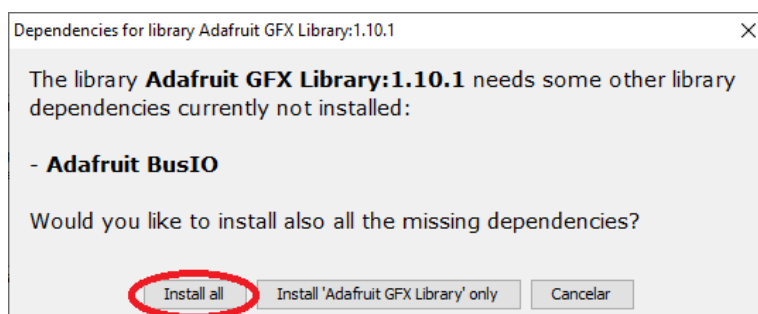
... la biblioteca deseada se muestra inmediatamente:



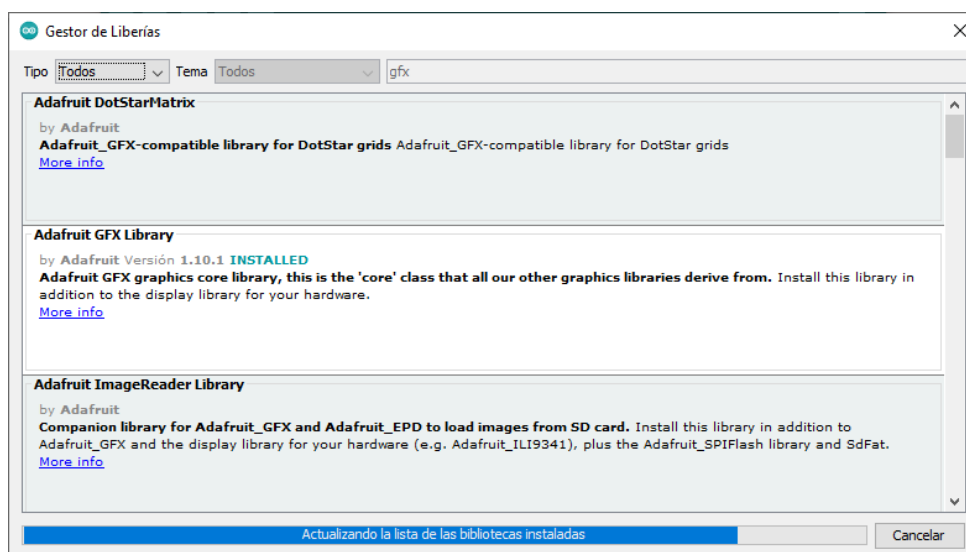
Ponga el mouse sobre el campo mostrado en la figura anterior, el cual tiene el texto “Adafruit GFX Library” y se mostrarán la versión y el botón “instalar”:



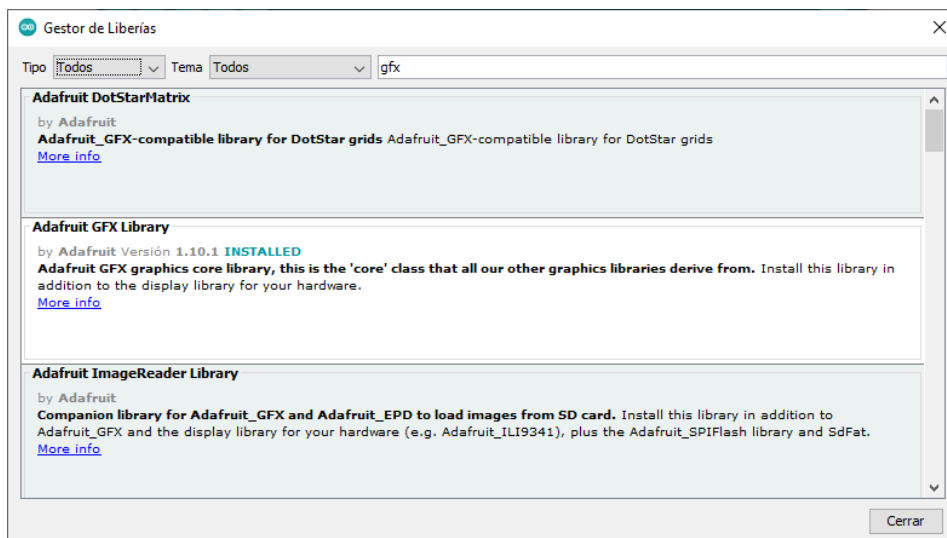
Al presionar instalar, le mostrará la siguiente ventana:



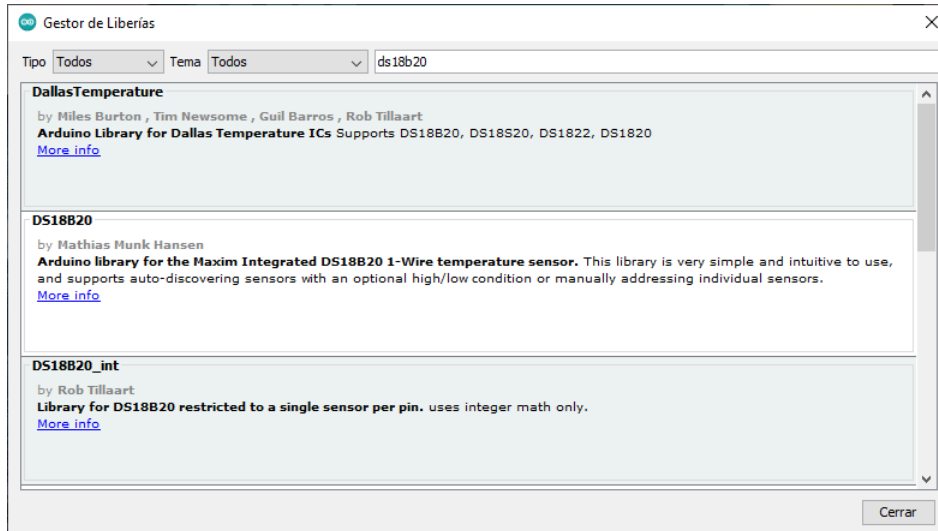
... presión el botón "Install all" y espere a que la instalación finalice:



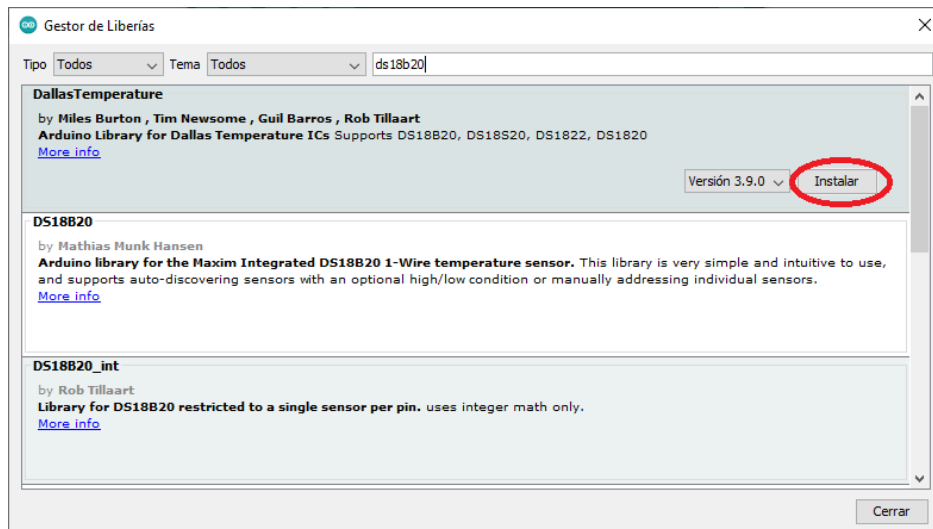
Cuando la instalación finalice, aún no presione el botón cerrar:



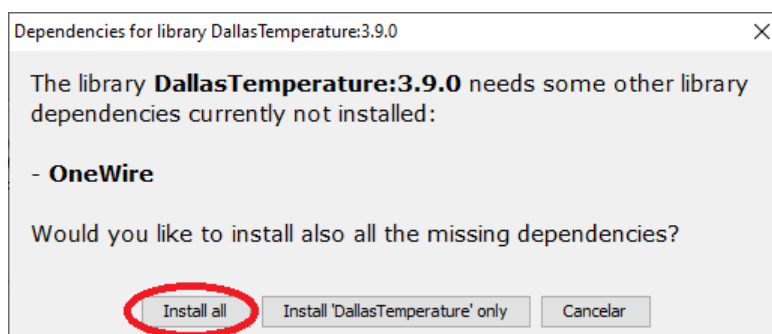
Aún queda pendiente la instalación de la biblioteca DS18B20 que se utilizará en el proyecto, así que, borre lo anterior y escriba “ds18b20” en el campo marcado como “Tema”, así como se muestra:



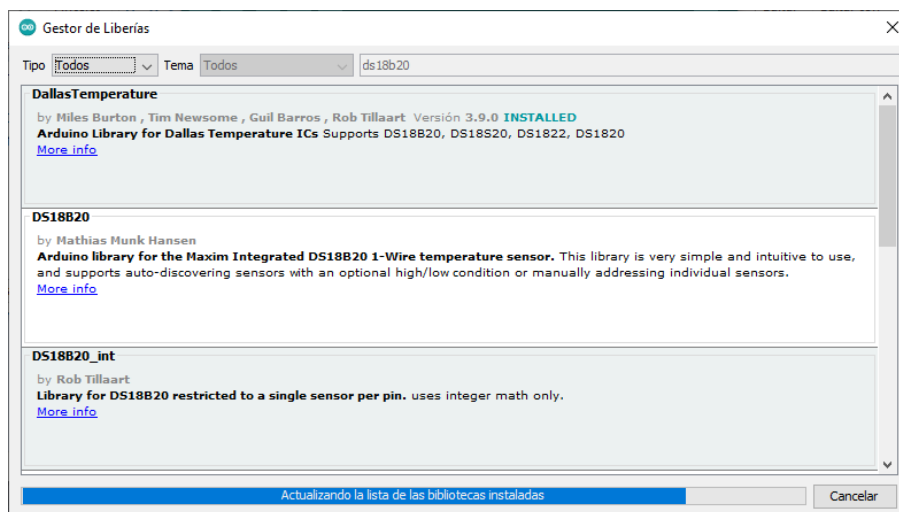
Ponga el cursor de mouse en el área del campo llamado “Dallas temperature” y se activarán los campos con la versión y el botón de instalar:



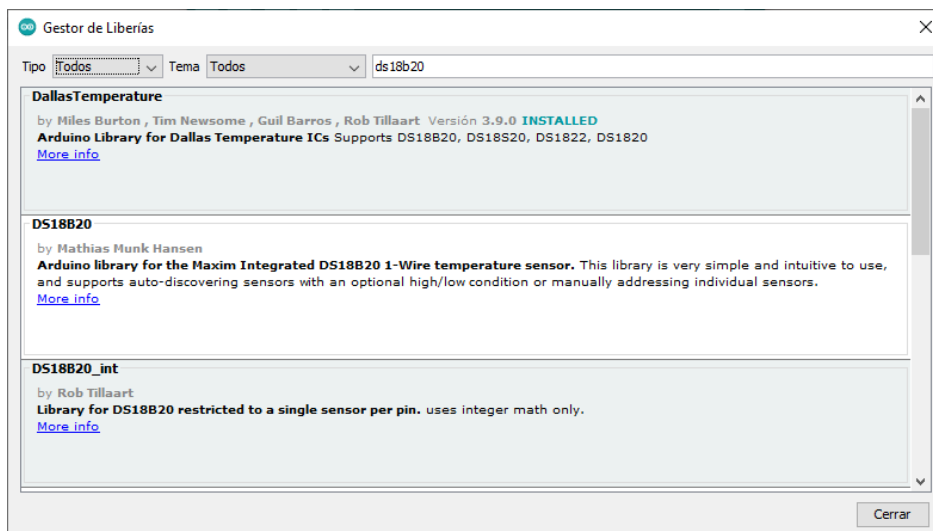
Presione el botón con el texto “instalar” y saldrá una ventana como se muestra a continuación:



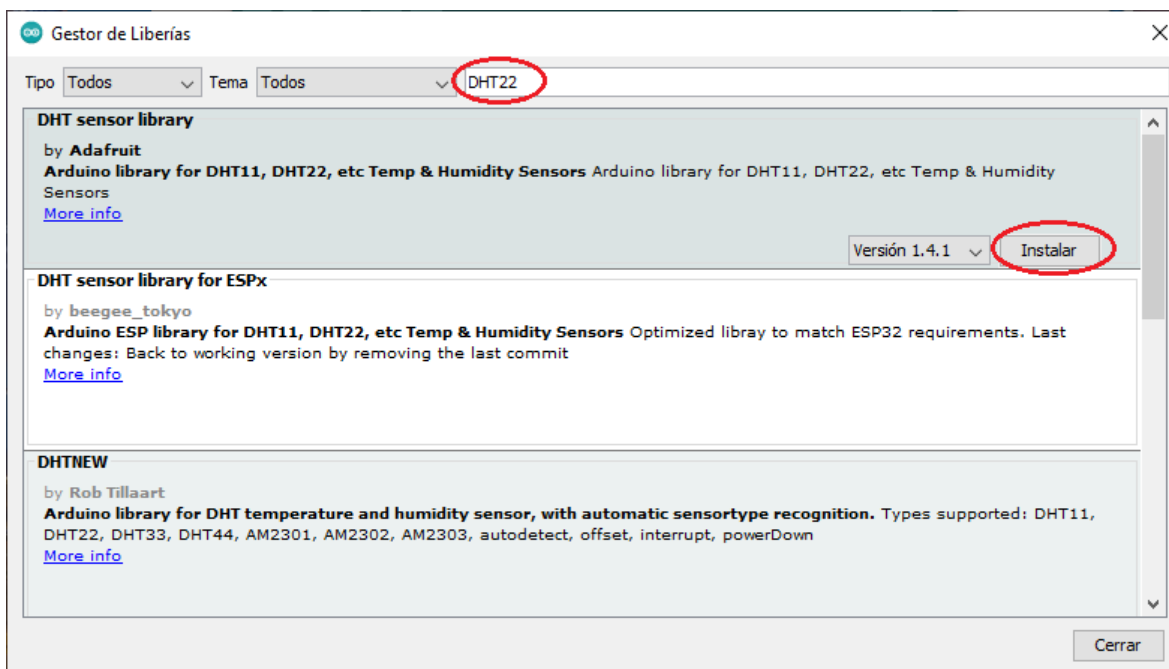
... presione el botón “Install all” y saldrá la siguiente ventana:



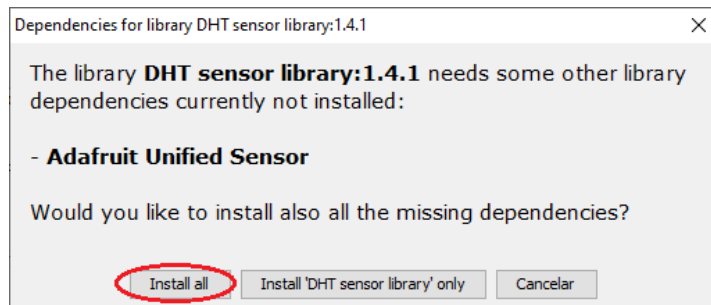
... espere a que la instalación termine y se mostrará una ventana como ésta:



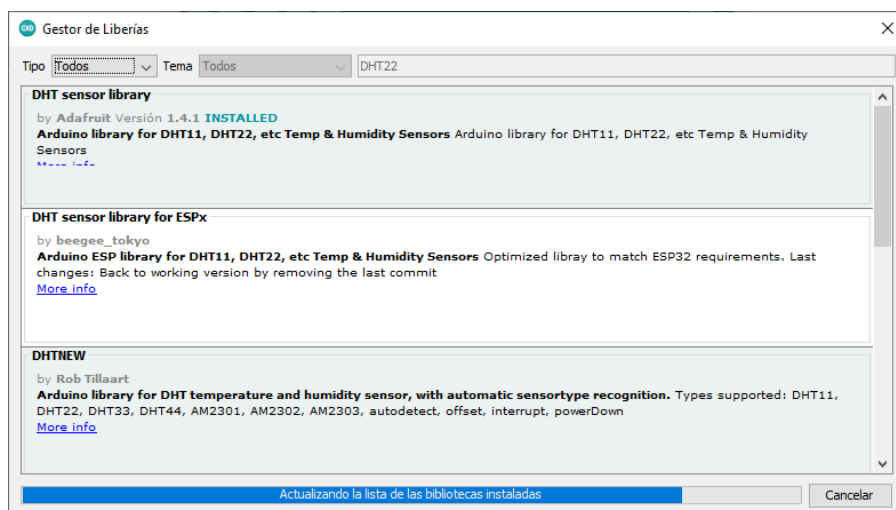
Es relevante mencionar que existe otro sensor, el cual, es ampliamente utilizado y de igual manera, se instalará su biblioteca correspondiente. El sensor mencionado anteriormente es el DHT22 (AM2301) y se inicia la instalación de la manera siguiente: borre lo escrito anteriormente y teclee DHT22 en el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



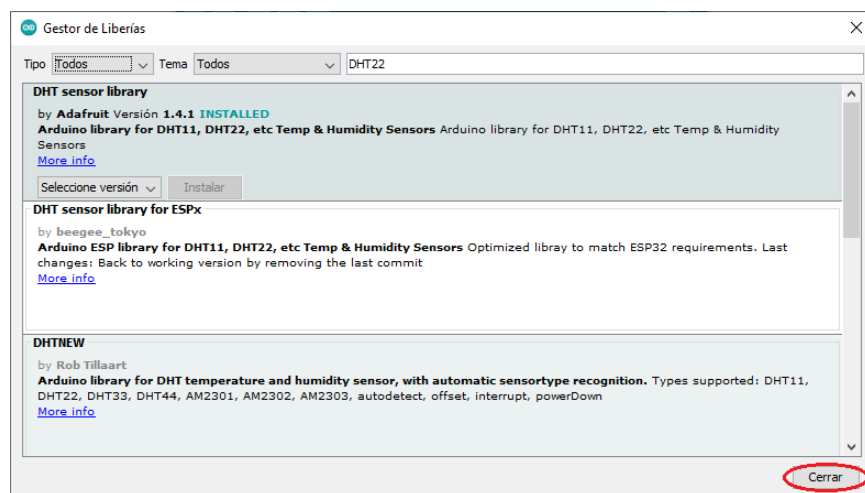
... e inmediatamente se mostrará en primer lugar la biblioteca del sensor DHT, seleccione la última versión y presione el botón “instalar” y a continuación le saldrá una ventana como se muestra:



... 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.

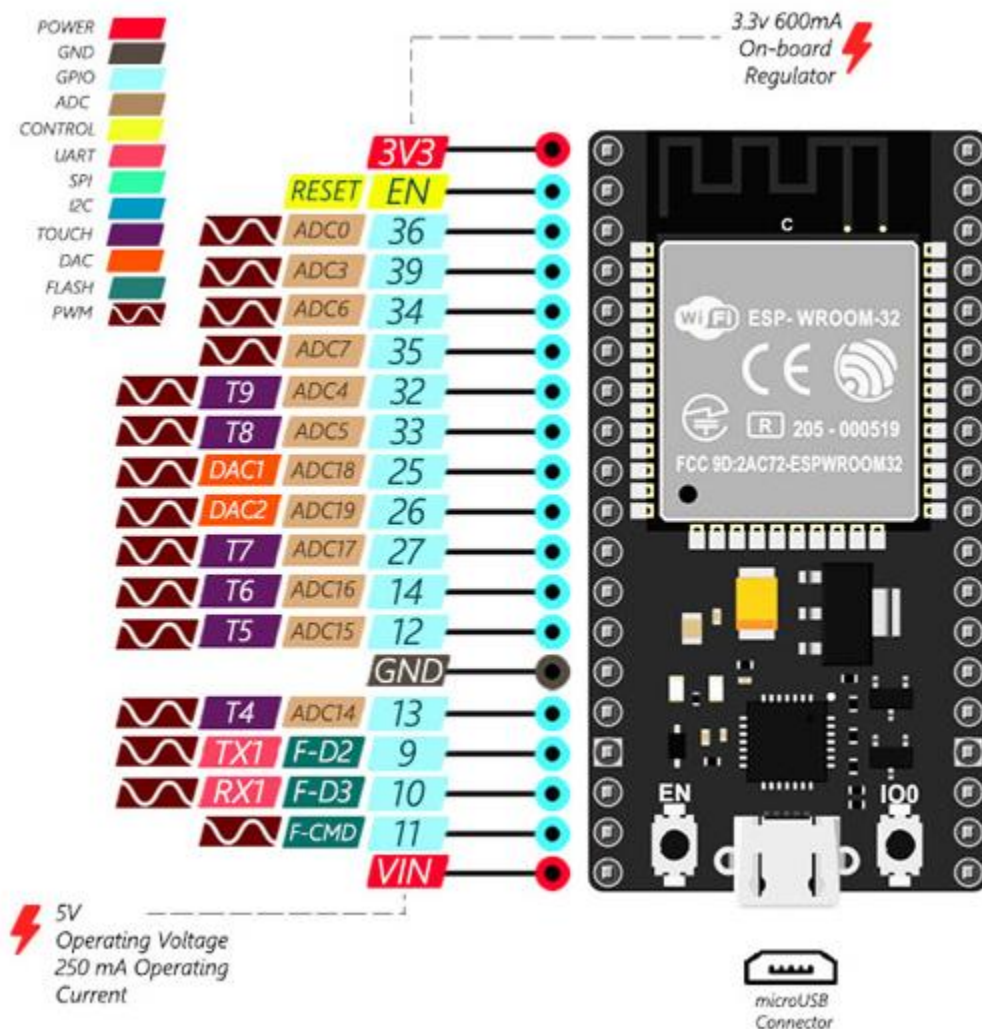
El módulo NodeMCU 32S tiene como núcleo un chip ESP32 y sólo requiere un voltaje de 3.3V. Dicho chip se compone de dos núcleos y pueden ser controlados individualmente. Se puede programar con el IDE del Arduino para la fácil implementación de dispositivos IoT. Adicionalmente, el módulo NodeMCU 32S contiene 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:

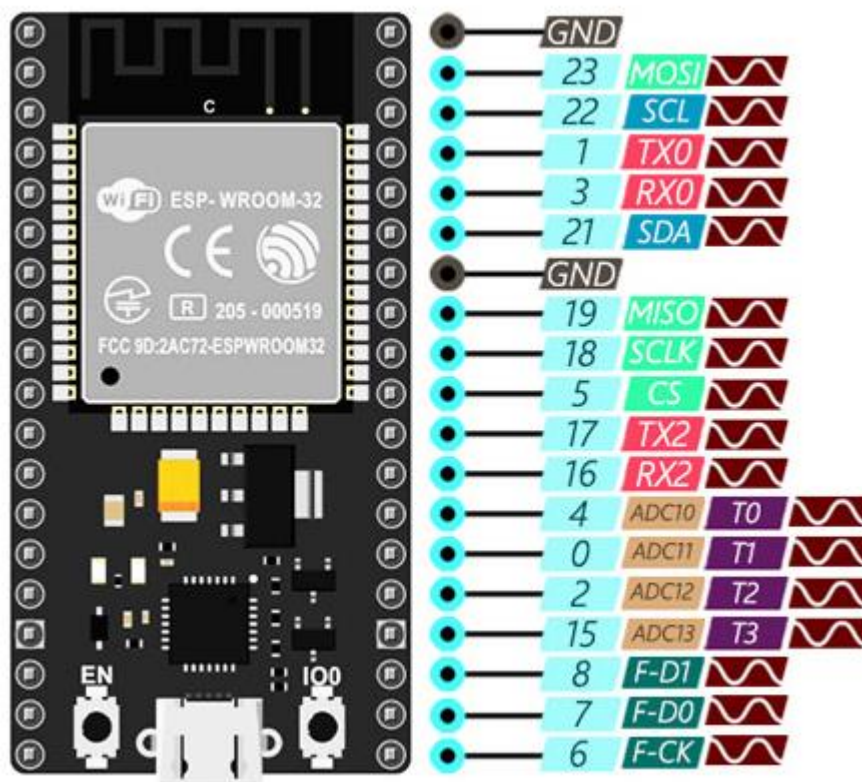


GPIO0 se encuentra conectado al pushbutton (ubicado en la esquina inferior derecha) marcado con el texto IO0.

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

Los pines indicados con el cuadro color azul claro () 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 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.

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

19	OK	OK	VSPI MISO
21	OK	OK	
22	OK	OK	
23	OK	OK	VSPI MOSI
25	OK	OK	
26	OK	OK	
27	OK	OK	
32	OK	OK	
33	OK	OK	
34	OK		Sólo INPUT
35	OK		Sólo INPUT
36	OK		Sólo INPUT
39	OK		Sólo INPUT

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

Consideración importante

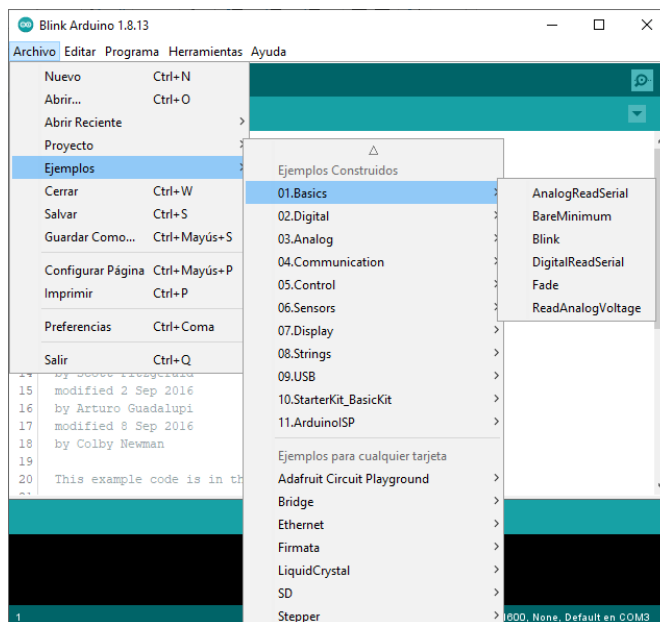
La estructura de programación del Arduino es muy sencilla, se debe considerar lo siguiente, el método `setup()` solamente se ejecuta una vez y sirve para configurar los periféricos:

```
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:
}
```

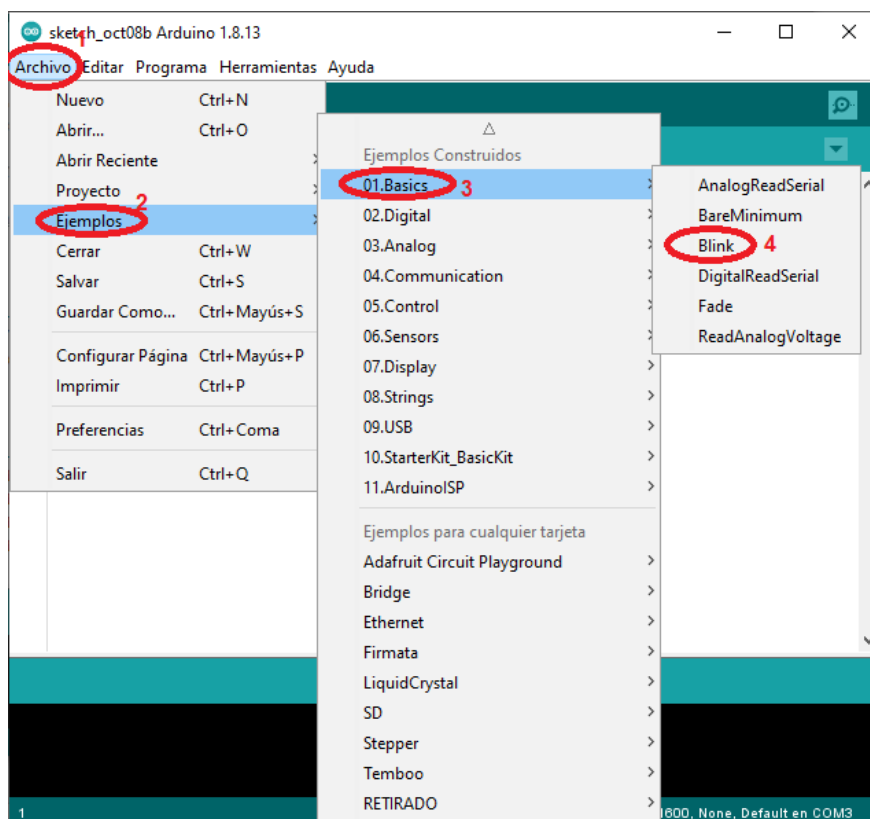
Todos los ejemplos que a continuación se mencionan se encuentran en la sección de ejemplos:



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
    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:

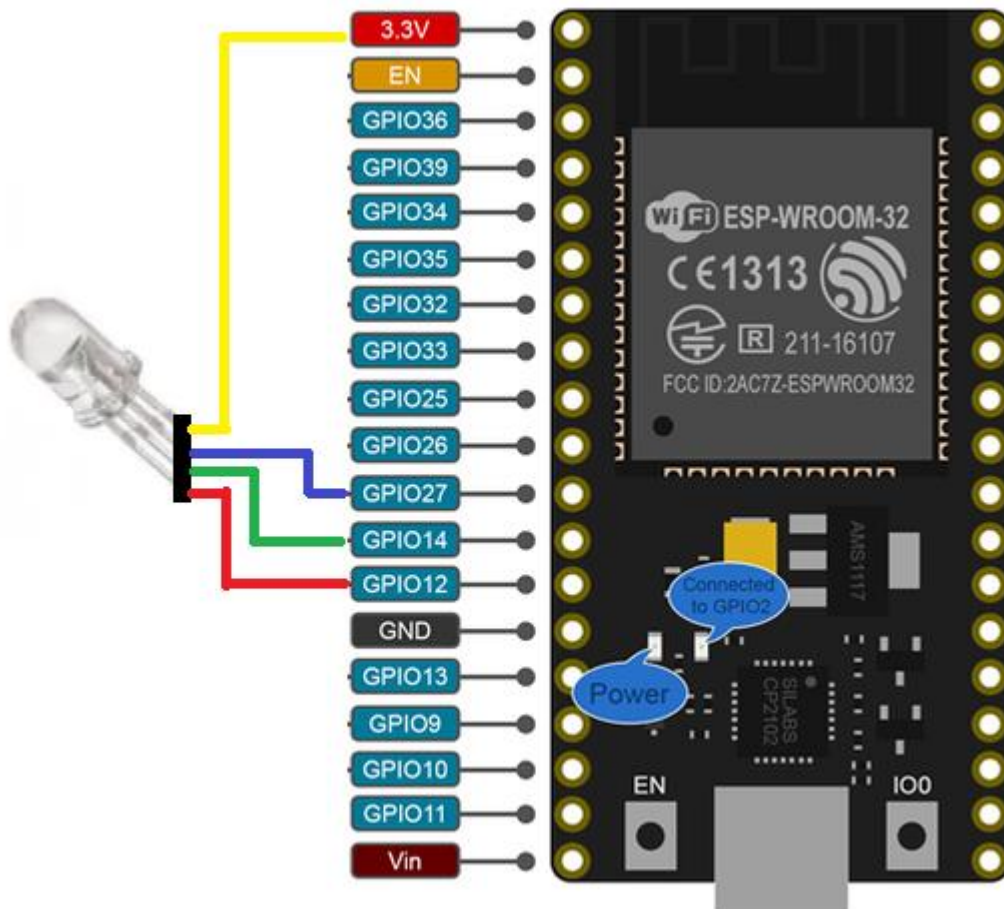
```
int LEDR = 2;
```

```
// 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)
    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:



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 con el botón  se verifica y descarga el código a la tarjeta de desarrollo...

... Tampoco olvide que cada vez que se desea transferir el código al Módulo ESP32 es necesario presionar y mantener presionado el botón "IO0" y presionar momentáneamente el botón de "reset" ubicados en los costados del conector microUSB del módulo NodeMCU.

¿Qué sucede si quiero 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 esta 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 esta 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.

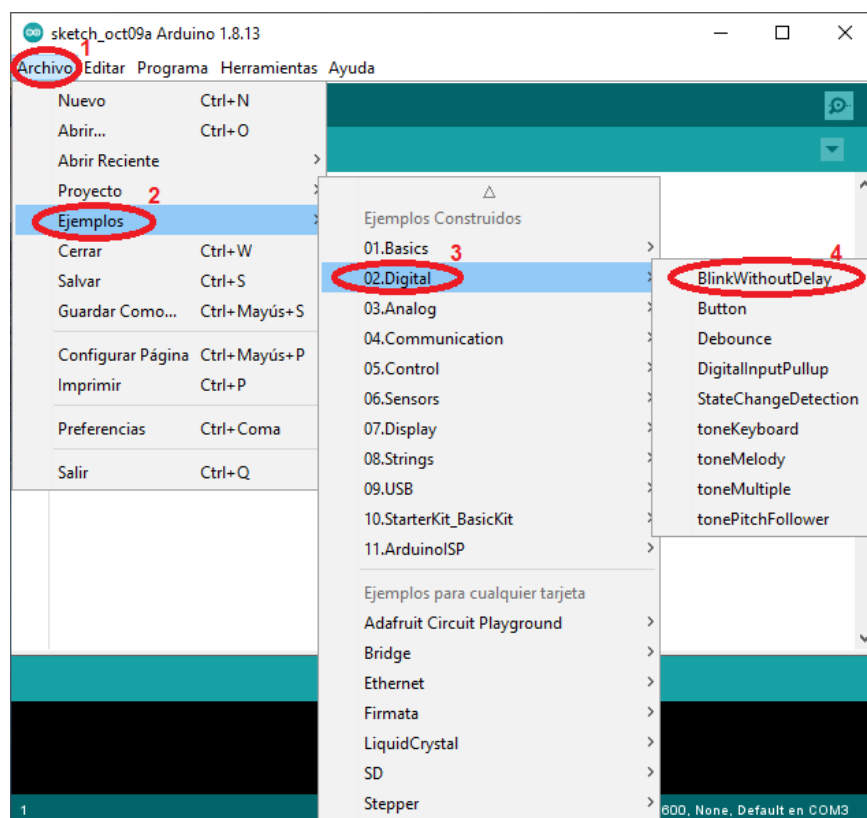
Considere que `delay(1000)`; el valor de 1000 corresponde a 1 segundo.

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:



... **OJO**, no olvide que con el botón  se verifica y descarga el código a la tarjeta de desarrollo...

... 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
```



Centro de Investigación Científica de Yucatán A. C. Departamento de Instrumentación

www.cicy.mx

```
// 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 ejecutará 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`.



Centro de Investigación Científica de Yucatán A. C.
Departamento de Instrumentación
www.cicy.mx

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



Centro de Investigación Científica de Yucatán A. C.

Departamento de Instrumentación

www.cicy.mx

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



Centro de Investigación Científica de Yucatán A. C.
Departamento de Instrumentación

www.cicy.mx

```
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”.

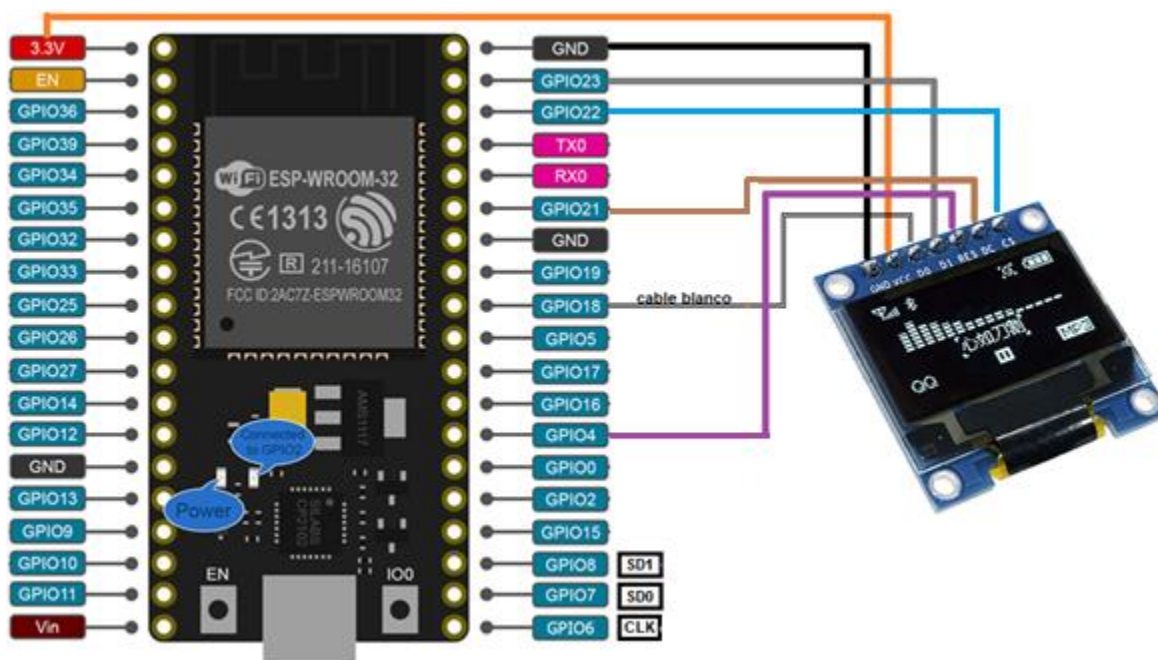
Actividades:

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

2.2.3 Usando la pantalla OLED SSD1306

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

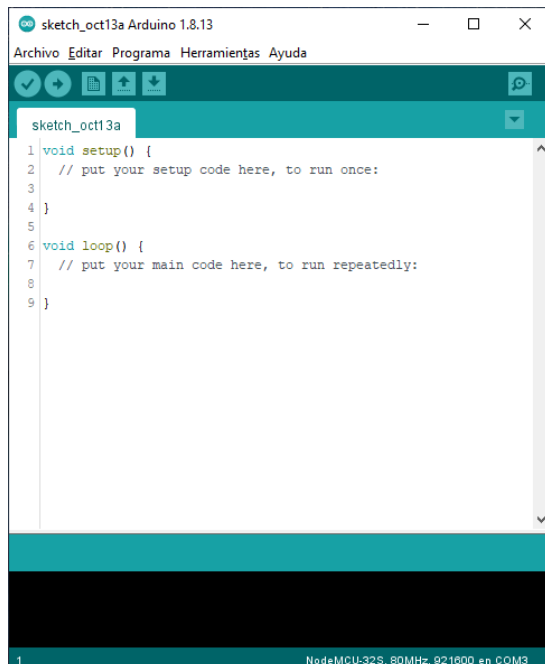
Para iniciar con este punto, se requiere a ver unas conexiones a la pantalla OLED y al módulo NodeMCU así como se muestra en la pantalla:



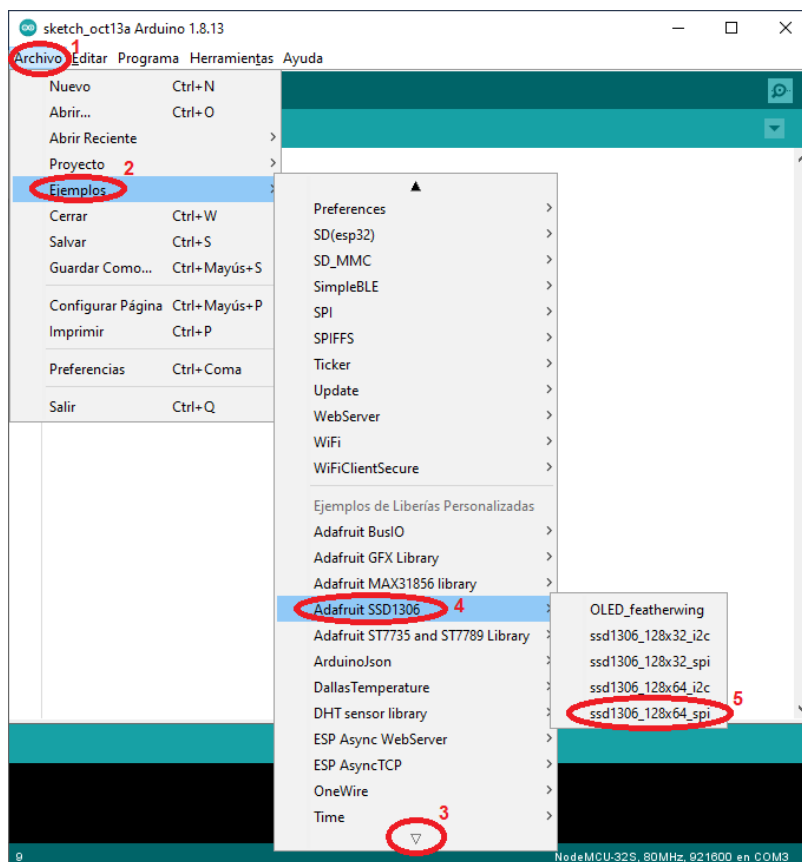
IMPORTANTE: Si el conexionado de la pantalla no se hace correctamente, puede dañarse irremediablemente. **Las conexiones que más debe cuidar son las de GND y VCC** (las dos primeras contando de izquierda a derecha) que corresponden a los cables **negro y naranja**.

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:

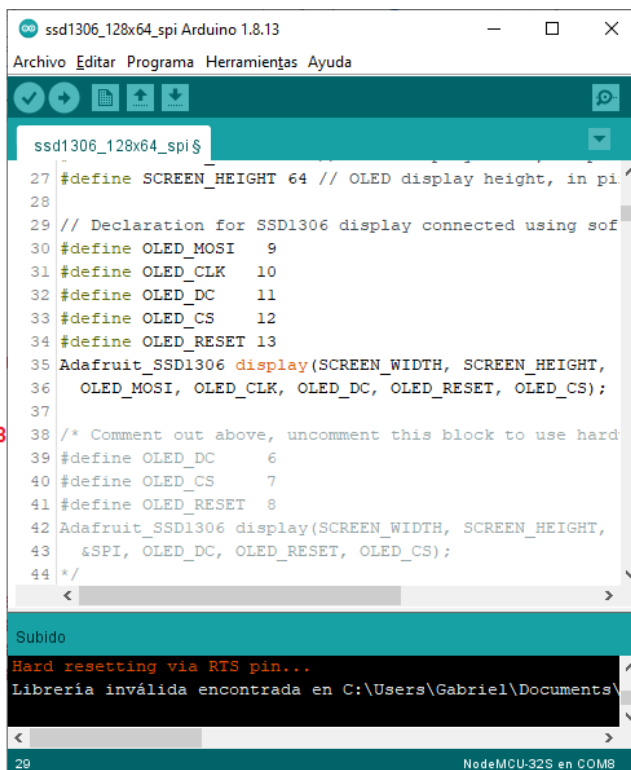


Proceda a abrir el ejemplo siguiente la siguiente secuencia:

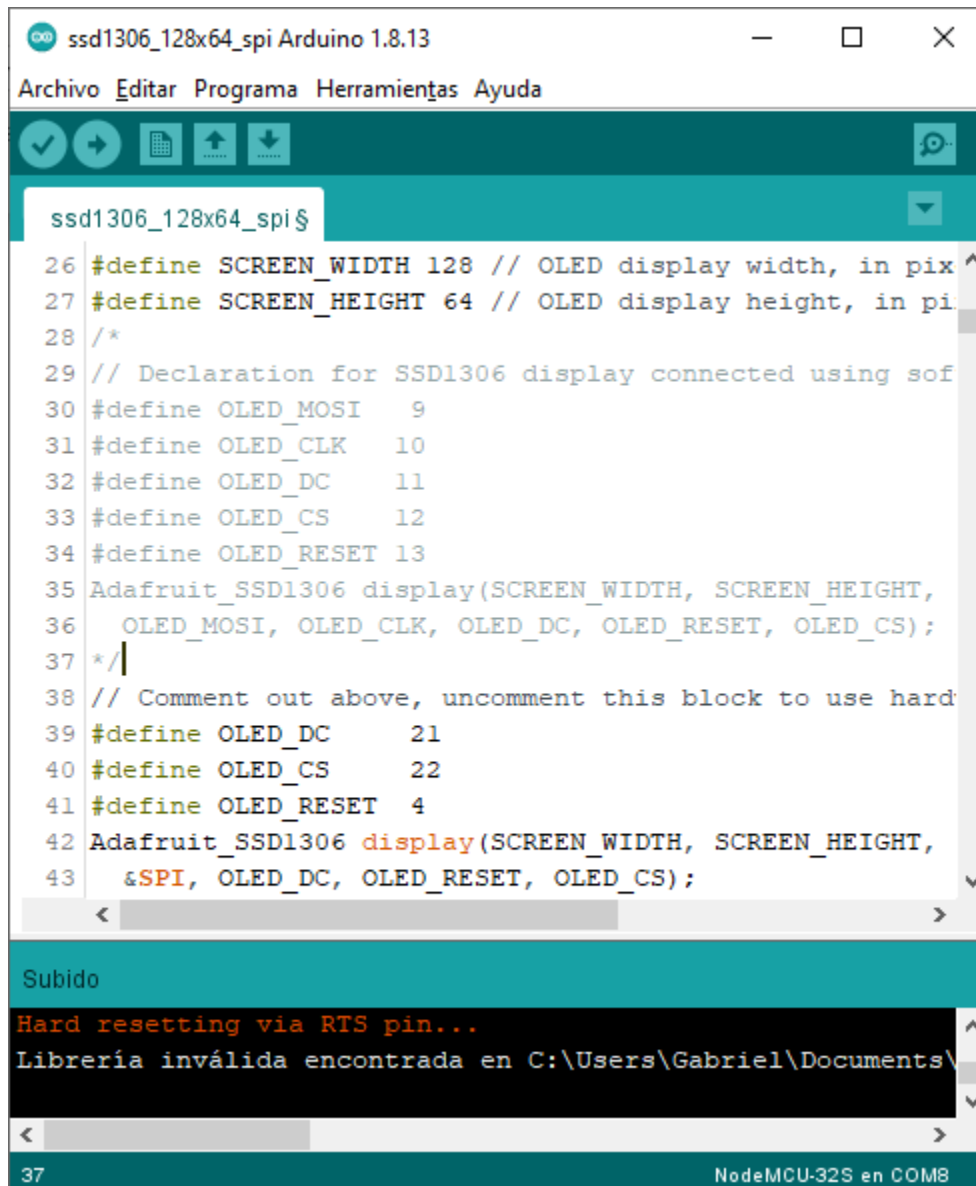


Antes de ejecutar el ejemplo, se requiere hacer unas modificaciones al código, así como se muestra a continuación:

1. Agregue /* en la línea 28
2. Agregue */ en la línea 37
3. Elimine el /* y cambiela por // en la línea 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 línea 44



El código debe quedar, como se muestra a continuación:



```

ssd1306_128x64_spi Arduino 1.8.13
Archivo Editar Programa Herramientas Ayuda

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);
  
```


Subido

Hard resetting via RTS pin...

Librería inválida encontrada en C:\Users\Gabriel\Documents\

37 NodeMCU-32S en COM8

Compile y descargue a su tarjeta de desarrollo.

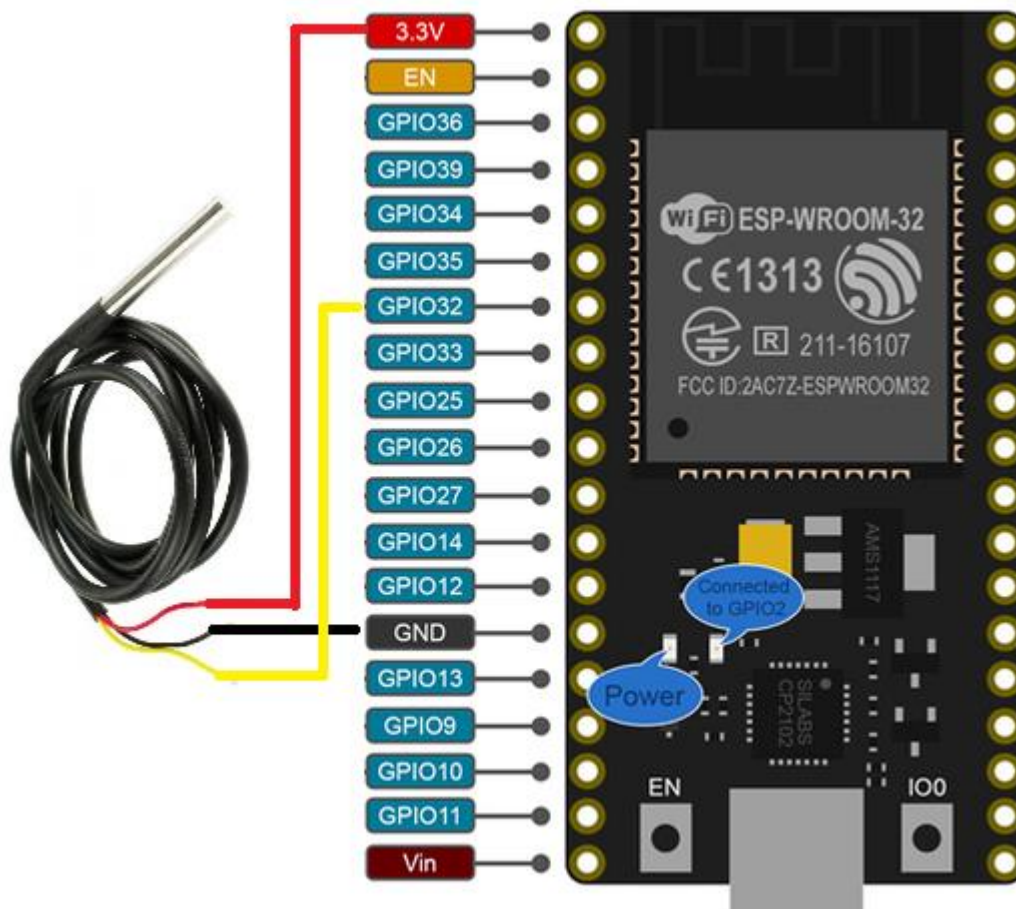
No olvide que: Con el botón  se verifica y descarga el código a la tarjeta de desarrollo.

¿Qué observa en la pantalla?

2.2.4 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.

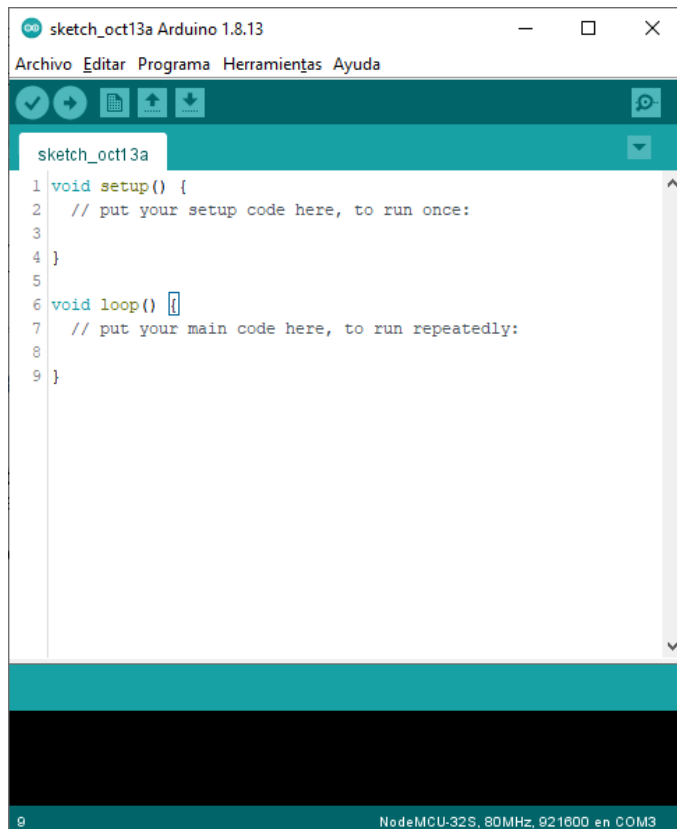
Antes de iniciar, se requiere del conexionado del sensor como se muestra en la figura:



IMPORTANTE: 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.

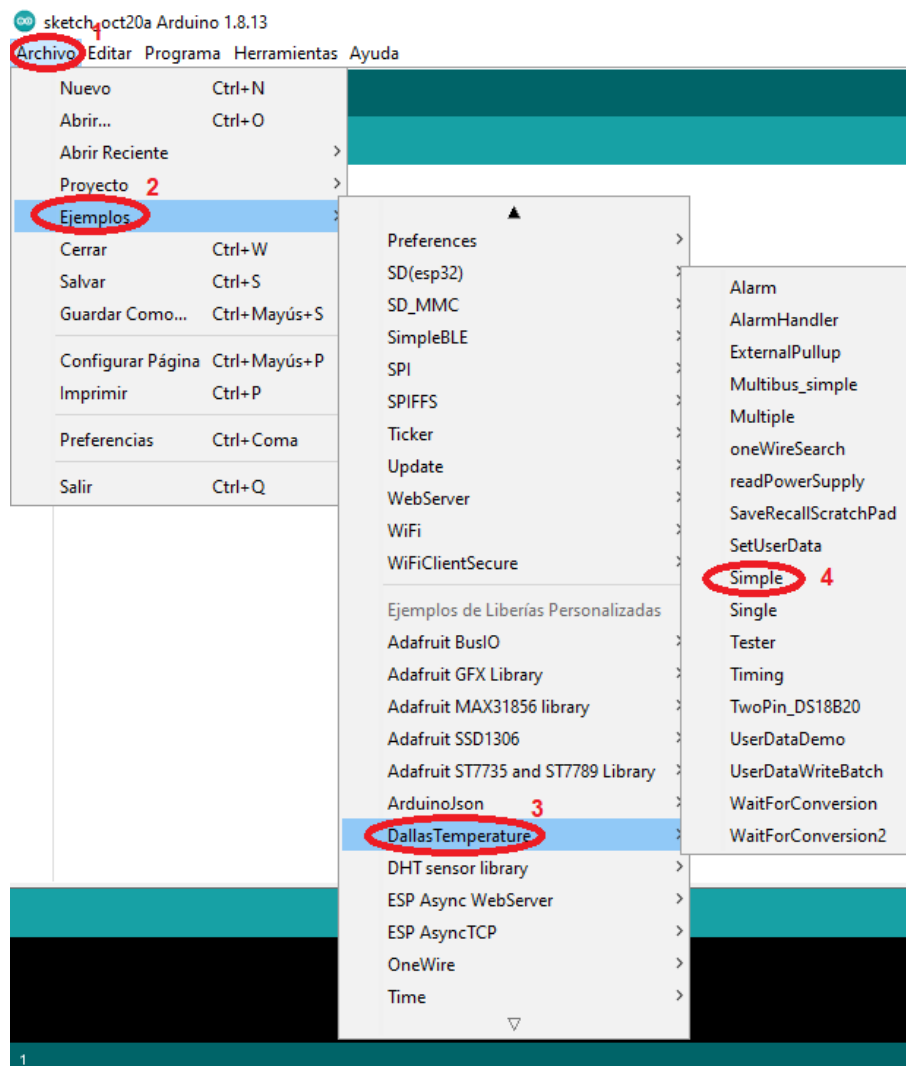
Seguidamente, se requiere abrir el IDE del Arduino como sigue:

A screenshot of the Arduino IDE software interface. The window title is "sketch_oct13a Arduino 1.8.13". The menu bar includes "Archivo", "Editar", "Programa", "Herramientas", and "Ayuda". The toolbar contains icons for opening files, saving, compiling, uploading, and monitoring. The main text area shows the following code:

```
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 COM3".

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 con el botón  se verifica y descarga el código a la tarjeta de desarrollo. Tampoco olvide poner en modo bootloader el NodeMCU 32s

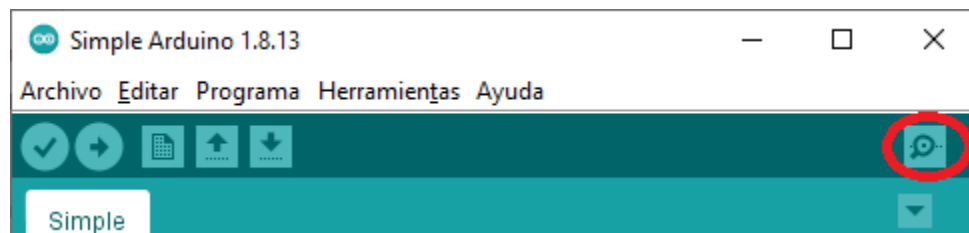
presionando y mantener presionado el botón IO0 mientras presiona momentáneamente el botón de RESET, ambos ubicados en los costados del conector microUSB de su tarjeta de desarrollo.

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:

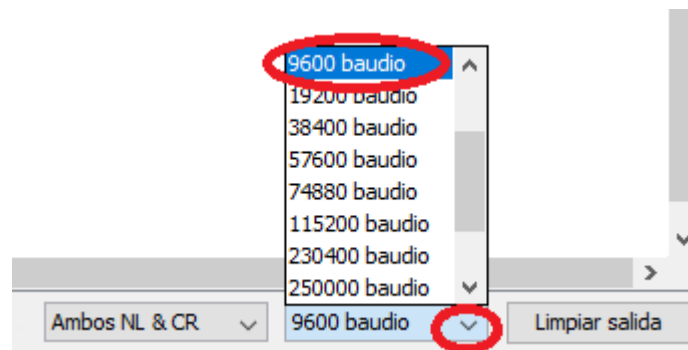
Observe y memorice el valor señalado a continuación:

```
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:

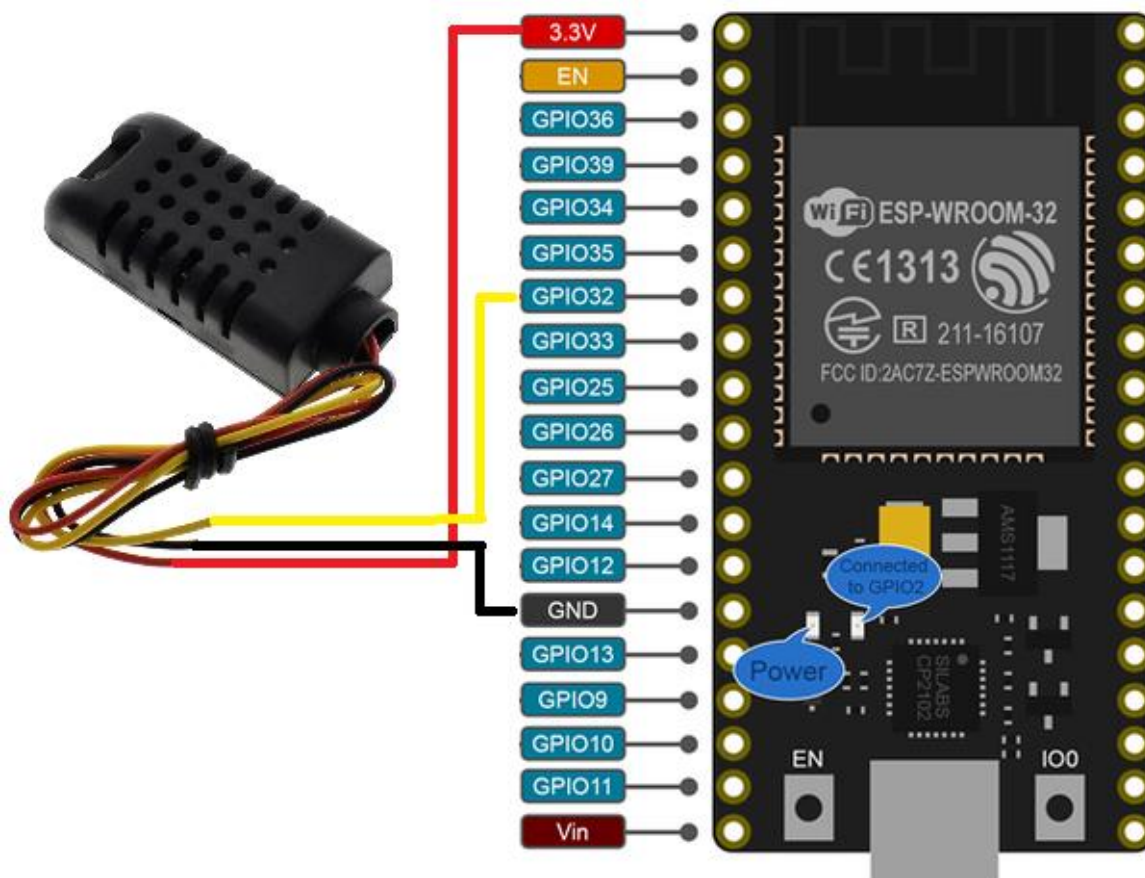
¿Qué datos se aprecian?

¿Puede ubicar el dato de la temperatura?

2.2.5 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.

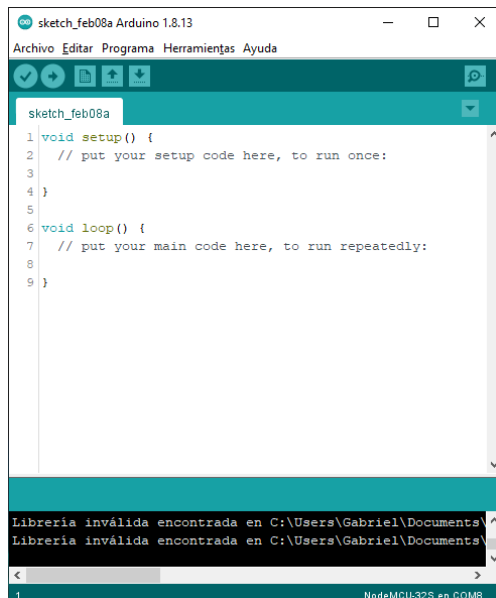
Antes de iniciar, se requiere del conexionado del sensor como se muestra en la figura:



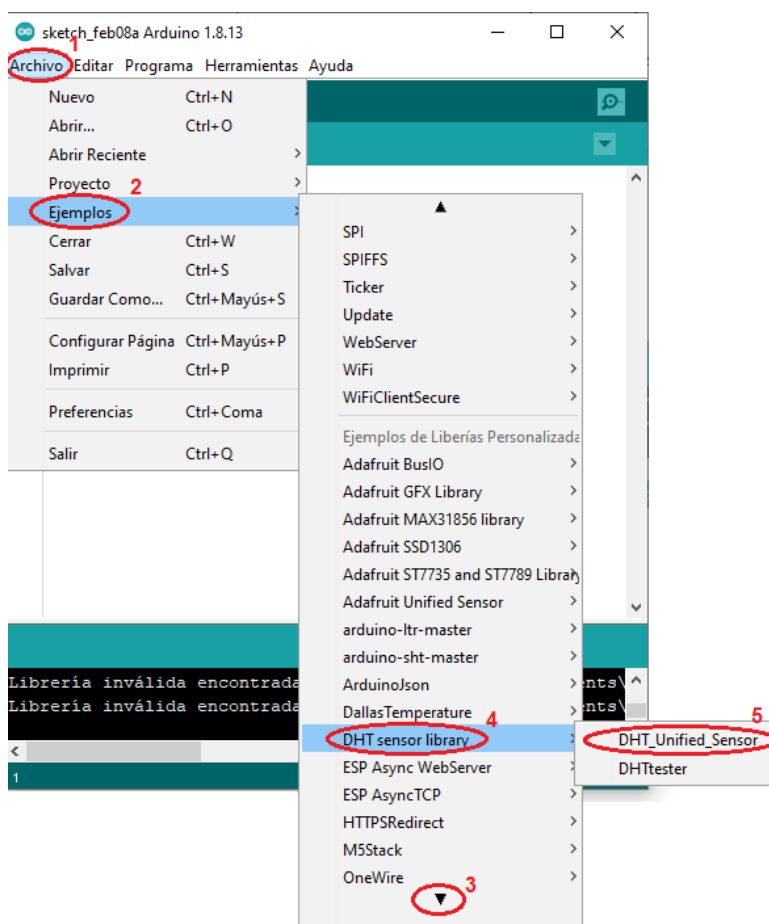
IMPORTANTE: El dispositivo puede dañarse si es conectado de manera incorrecta. Los pines críticos son el GND y 3.3V (cable Negro y rojo)

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

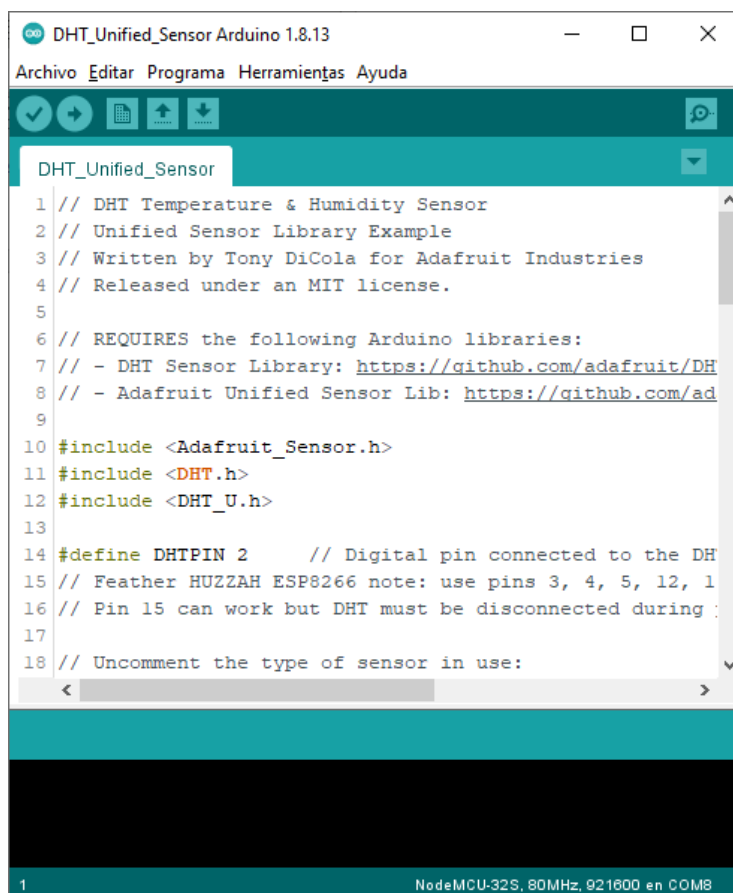
Seguidamente, se requiere abrir el IDE del Arduino como sigue:



Presione en la secuencia indicada los siguientes menús:



... y se abrirá el ejemplo siguiente:




```
DHT_Unified_Sensor
1 // DHT Temperature & Humidity Sensor
2 // Unified Sensor Library Example
3 // Written by Tony DiCola for Adafruit Industries
4 // Released under an MIT license.
5
6 // REQUIRES the following Arduino libraries:
7 // - DHT Sensor Library: https://github.com/adafruit/DHT
8 // - Adafruit Unified Sensor Lib: https://github.com/ad
9
10 #include <Adafruit_Sensor.h>
11 #include <DHT.h>
12 #include <DHT_U.h>
13
14 #define DHTPIN 2 // Digital pin connected to the DH
15 // Feather HUZZAH ESP8266 note: use pins 3, 4, 5, 12, 1
16 // Pin 15 can work but DHT must be disconnected during :
17
18 // Uncomment the type of sensor in use:
```

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.

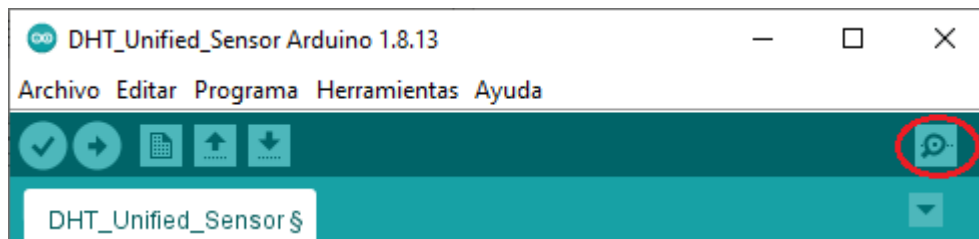
OJO, no olvide que con el botón  se verifica y descarga el código a la tarjeta de desarrollo. Tampoco olvide poner en modo bootloader el NodeMCU 32s presionando y mantener presionado el botón IO0 mientras presiona momentáneamente el botón de RESET, ambos ubicados en los costados del conector microUSB de su tarjeta de desarrollo.

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:

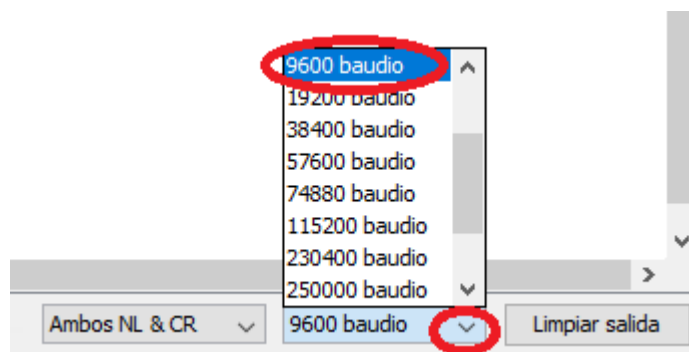
Observe y memorice el valor señalado a continuación:

```
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?

2.2.6 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. 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:

a) *La coordenada en el eje 'X'* donde empieza la esquina superior izquierda del rectángulo, recuerde que la pantalla es de 128 x 64 pixeles, o sea, x puede tener un valor máximo de 127 y el eje y puede tener hasta 63

b) *La coordenada en el eje 'Y'* donde empieza la esquina superior izquierda del rectángulo.

c) *El ancho del rectángulo*, éste 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)

d) *La altura del rectángulo*, esta 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.