

Arduino IDE

- [Conexión con Arduino IDE](#)
- [El primer programa con Arduino IDE: Blink](#)
- [Proyectos](#)
- [Proyectos con Wifi](#)

Conexión con Arduino IDE

El software Arduino IDE lo puedes descargar en <https://www.arduino.cc/>

The screenshot shows the Arduino website's navigation bar with 'SOFTWARE' highlighted. Below it, the 'Arduino Cloud Editor' section is visible. A red arrow points from the 'SOFTWARE' link to the 'Downloads' section. In the 'Downloads' section, the 'Arduino IDE 2.3.4' release is highlighted, along with a table of download options for Windows, Linux, and macOS.

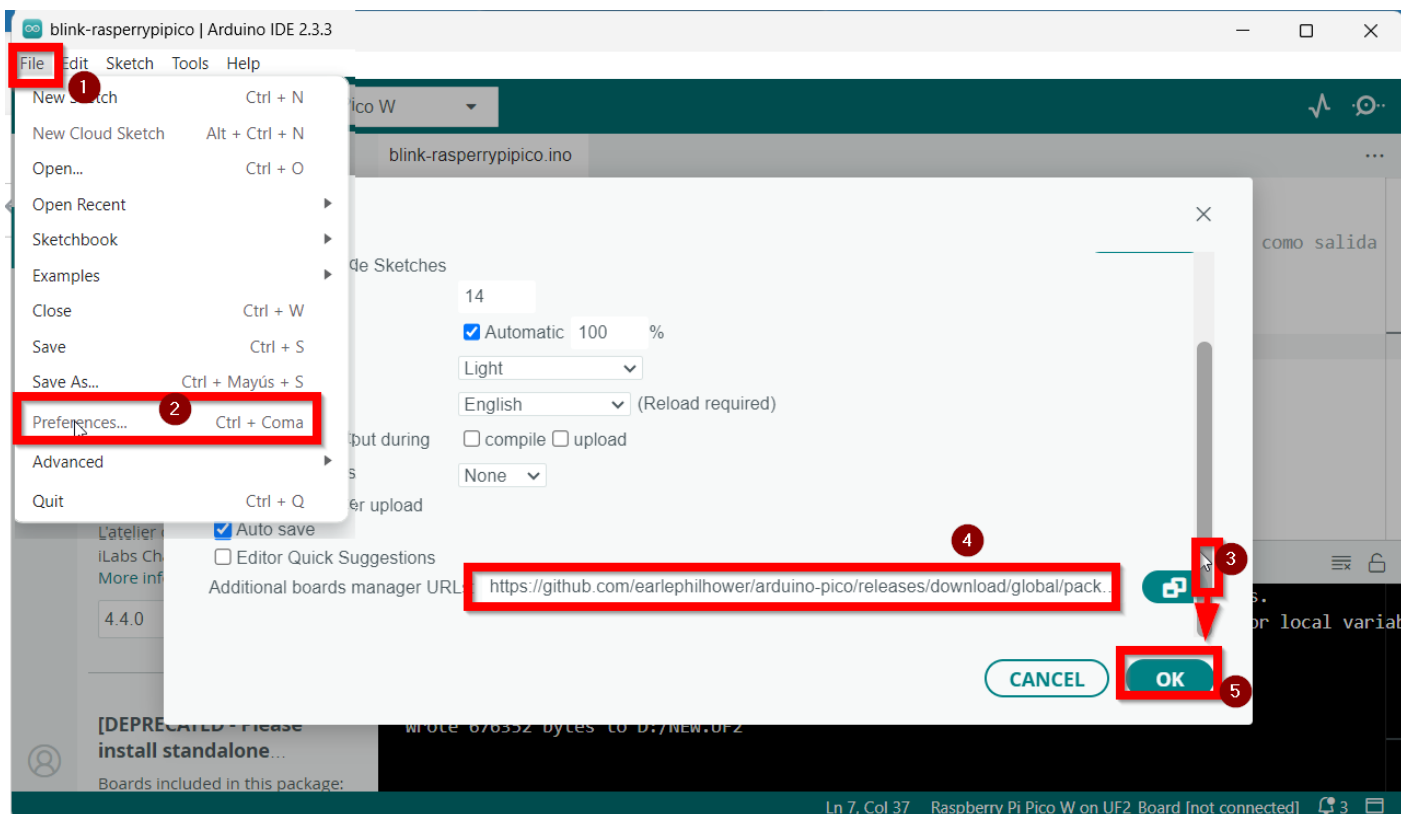
DOWNLOAD OPTIONS	
Windows	Win 10 and newer, 64 bits
Windows	MSI installer
Windows	ZIP file
Linux	Applimage 64 bits (X86-64)
Linux	ZIP file 64 bits (X86-64)
macOS	Intel, 10.15: "Catalina" or newer, 64 bits
macOS	Apple Silicon, 11: "Big Sur" or newer, 64 bits

Una vez instalado vamos a ver cómo podemos programar con el software de Arduino nuestro Picobriks

No sigas las instrucciones de Project Book aquí <https://picobricks.com/pages/projectbook> **NO funcionan.**

Las siguientes instrucciones **sí que funcionan** y son de **Bricogeek** Licencia CC-BY
(hay muchas páginas con las mismas instrucciones pon en un buscador raspberry pico arduino ide)

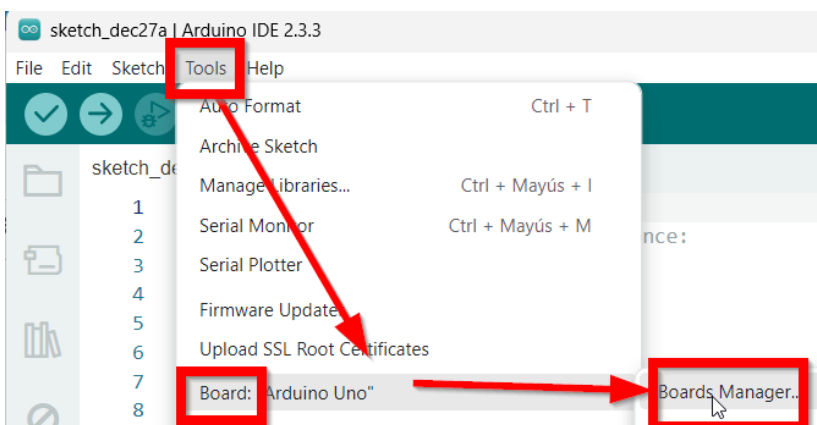
Nos vamos a **Archivo-Preferencias** y le decimos que ponga el siguiente directorio para las librerías :



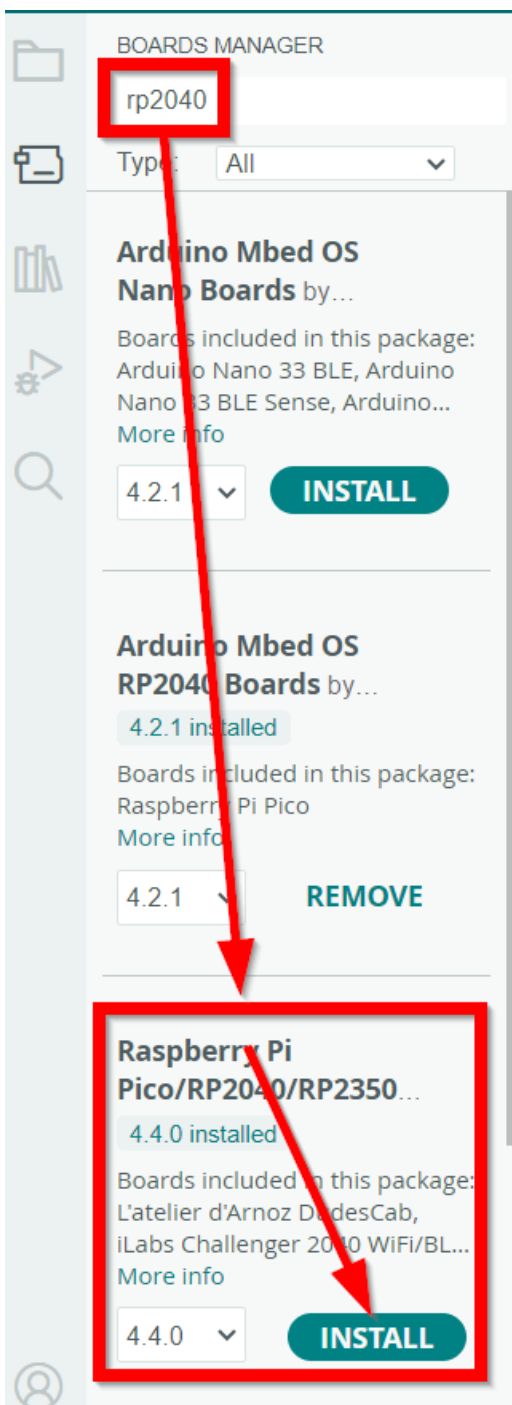
Esta es la URL que pegar para que cargue placas que no vienen por defecto :

`https://github.com/earlephilhower/arduino-pico/releases/download/global/package_rp2040_index.json`

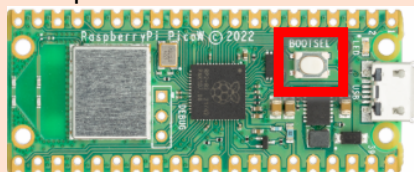
Una vez puesto nos vamos a **Boards manager**..



Y ponemos **RP2040** en el buscador, aparecerá este software, **Raspberry Pi Pico/RP2040** Lo instalamos

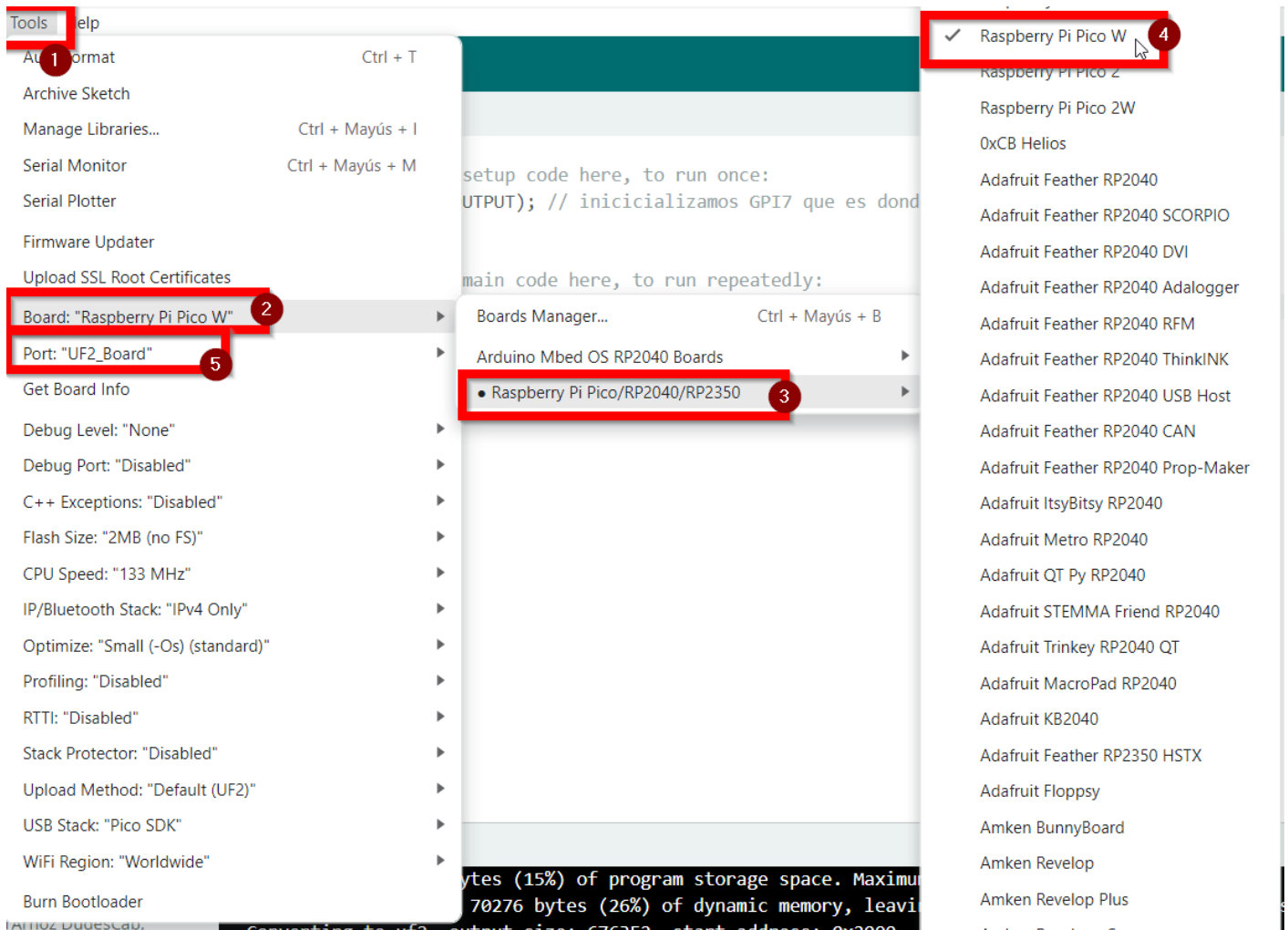
**ATENCIÓN, poner PicoBricks en modo Bootloader**

- 1.-Desconectamos PicoBricks de nuestro ordenador
- 2.- Apretamos el botón **BOOTSEL** **mientras** lo volvemos a conectar al puerto USB

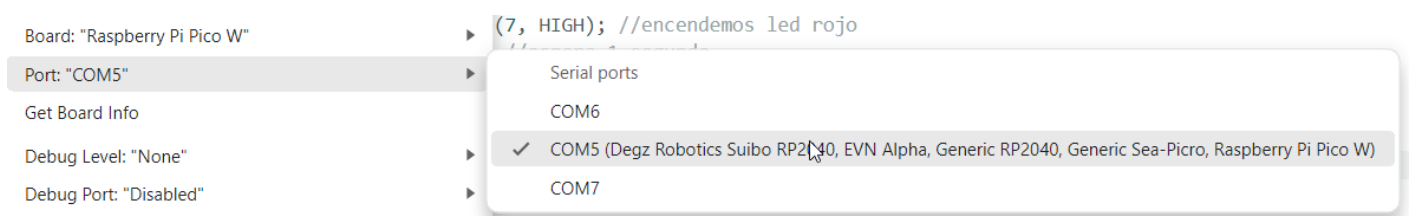


3.- Automáticamente aparecerá una nueva unidad de disco en nuestro ordenador (ya puedes soltar BOOTSEL)

Una vez puesto en modo Bootloader seleccionamos la placa **Raspberry Pi Pico W** (4) y también seleccionamos el puerto (5) **UF2_Board**



También puede salir otro tipo de puerto como este que dice que es el RP2040 Raspberry Pi Pico W



En resumen tiene que salir abajo a la derecha que esta conectado

Ln 1, Col 1 Raspberry Pi Pico on COM3

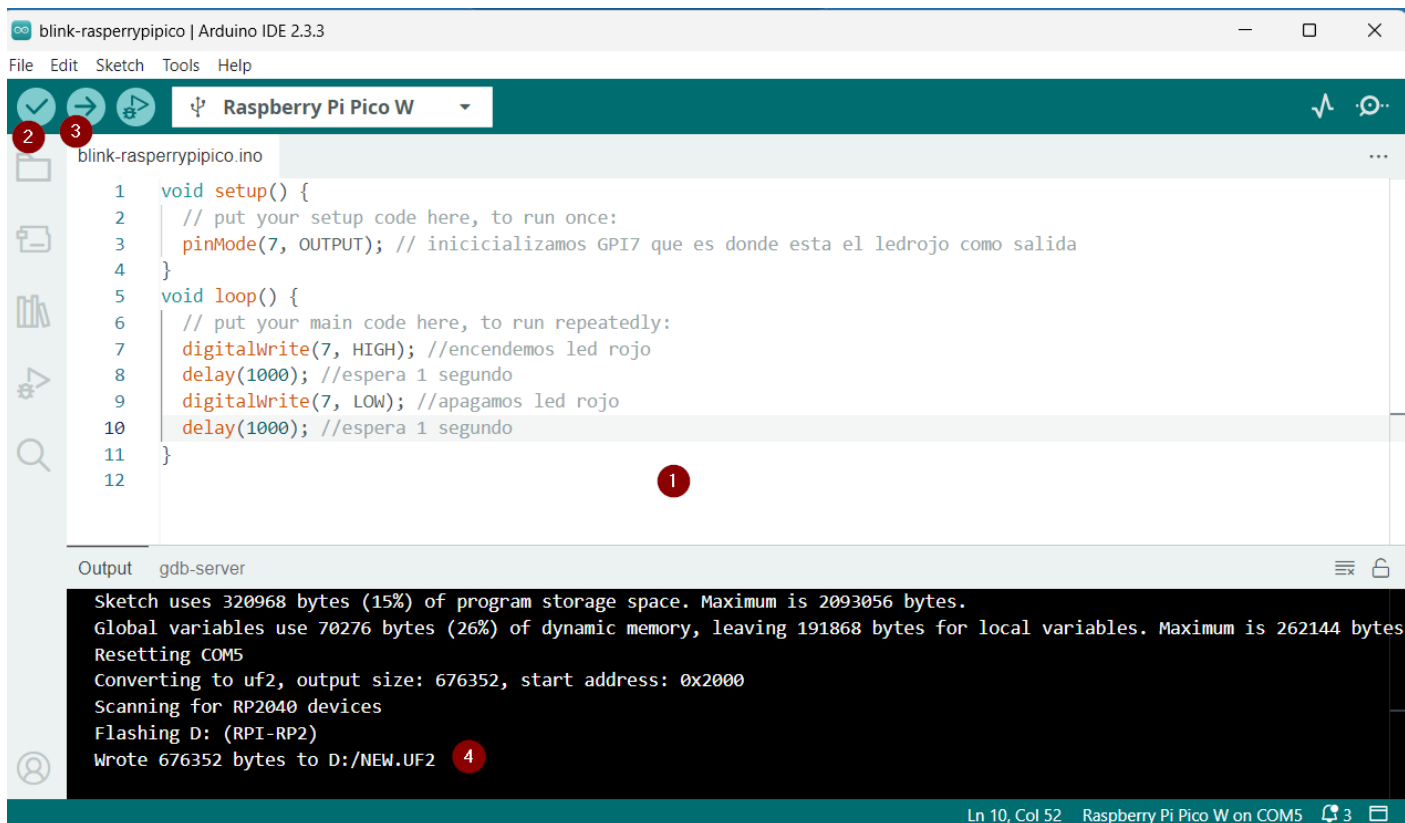


El primer programa con Arduino IDE: Blink

Entramos en Arduino IDE configurado según la página anterior (Board: Raspberry Pi y el puerto COM que corresponda) y pegamos el siguiente código:

```
void setup() {  
    // put your setup code here, to run once:  
    pinMode(7, OUTPUT); // inicializamos GPI7 que es donde esta el ledrojo como salida  
}  
void loop() {  
    // put your main code here, to run repeatedly:  
    digitalWrite(7, HIGH); //encendemos led rojo  
    delay(1000); //espera 1 segundo  
    digitalWrite(7, LOW); //apagamos led rojo  
    delay(1000); //espera 1 segundo  
}
```

Ponemos el código en el área **1**, pulsamos **2** para compilar, y finalmente **3** para que lo suba a PicoBricks, cuando aparezca el mensaje **wrote...** en 4 ya podemos ver que el led rojo empieza a parpadear



The screenshot shows the Arduino IDE 2.3.3 interface. The top menu bar includes File, Edit, Sketch, Tools, and Help. The toolbar has icons for checking, running, and uploading. The board is set to Raspberry Pi Pico W. The sketch file is blink-rasperrypico.ino. The code in the editor is as follows:

```
1 void setup() {  
2   // put your setup code here, to run once:  
3   pinMode(7, OUTPUT); // inicializamos GPI7 que es donde esta el ledrojo como salida  
4 }  
5 void loop() {  
6   // put your main code here, to run repeatedly:  
7   digitalWrite(7, HIGH); //encendemos led rojo  
8   delay(1000); //espera 1 segundo  
9   digitalWrite(7, LOW); //apagamos led rojo  
10  delay(1000); //espera 1 segundo  
11 }  
12
```

The output window at the bottom shows the following messages:

```
Output  gdb-server  
Sketch uses 320968 bytes (15%) of program storage space. Maximum is 2093056 bytes.  
Global variables use 70276 bytes (26%) of dynamic memory, leaving 191868 bytes for local variables. Maximum is 262144 bytes  
Resetting COM5  
Converting to uf2, output size: 676352, start address: 0x2000  
Scanning for RP2040 devices  
Flashing D: (RPI-RP2)  
Wrote 676352 bytes to D:/NEW.UF2
```

Red circles with numbers 1, 2, 3, and 4 are placed on the IDE interface. Circle 1 is on the code editor, circle 2 is on the upload button, circle 3 is on the run button, and circle 4 is on the output window.

Por cierto, el programa **se queda cargado** Pruébalo! desenchúfalo del ordenador, alimenta PicoBricks con un PowerBank o un cargador de móvil con el cable USB y ¡¡ **sigue funcionando !!!**

Proyectos

Los mismos proyectos vistos con PicoBlockly se pueden hacer igual con código.

Repositorio ide picobricks

<https://ide.picobricks.com/examples/examples.ht>

Repositorio Github

En la ruta <https://github.com/Robotistan/PicoBricks/tree/main/Software/Activities> los tienes listos los programas para copiar y pegar

1. - [Blink](#)
2. - [Action-Reaction](#)
3. - [Autonomous Lighting](#)
4. - [Thermometer](#)
5. - [Graphic Monitor](#)
6. - [Dominate the Rhythm](#)
7. - [Show Your Reaction](#)
8. - [My Timer](#)
9. - [Alarm Clock](#)
10. - [Know Your Color](#)
11. [Buzz Wire Game](#)

Libro Projectbook

Los tienes en este libro (en inglés) que lo puedes conseguir aquí

<https://picobricks.com/pages/projectbook>

- PROYECTO BLINK ver pag 25
- PROYECTO ACTION-REACTION ver pag 29
- PROYECTO Autonomous Lighting ver pag 35
- PROYECTO Thermometer ver pag 41
- PROYECTO Graphic Monitor ver pag 46



- PROYECTO Dominate the Rhythm ver pag 55
- PROYECTO Show Your Reaction ver pag 63
- PROYECTO My Timer ver pag 71
- PROYECTO Alarm Clock ver pag 81
- PROYECTO Know Your Color ver pag 90
- PROYECTO Buzz Wire Game ver pag 106

A diferencia de Microblocks, no los explica paso a paso, por lo que es mejor copiar y pegar de los repositorios de Github

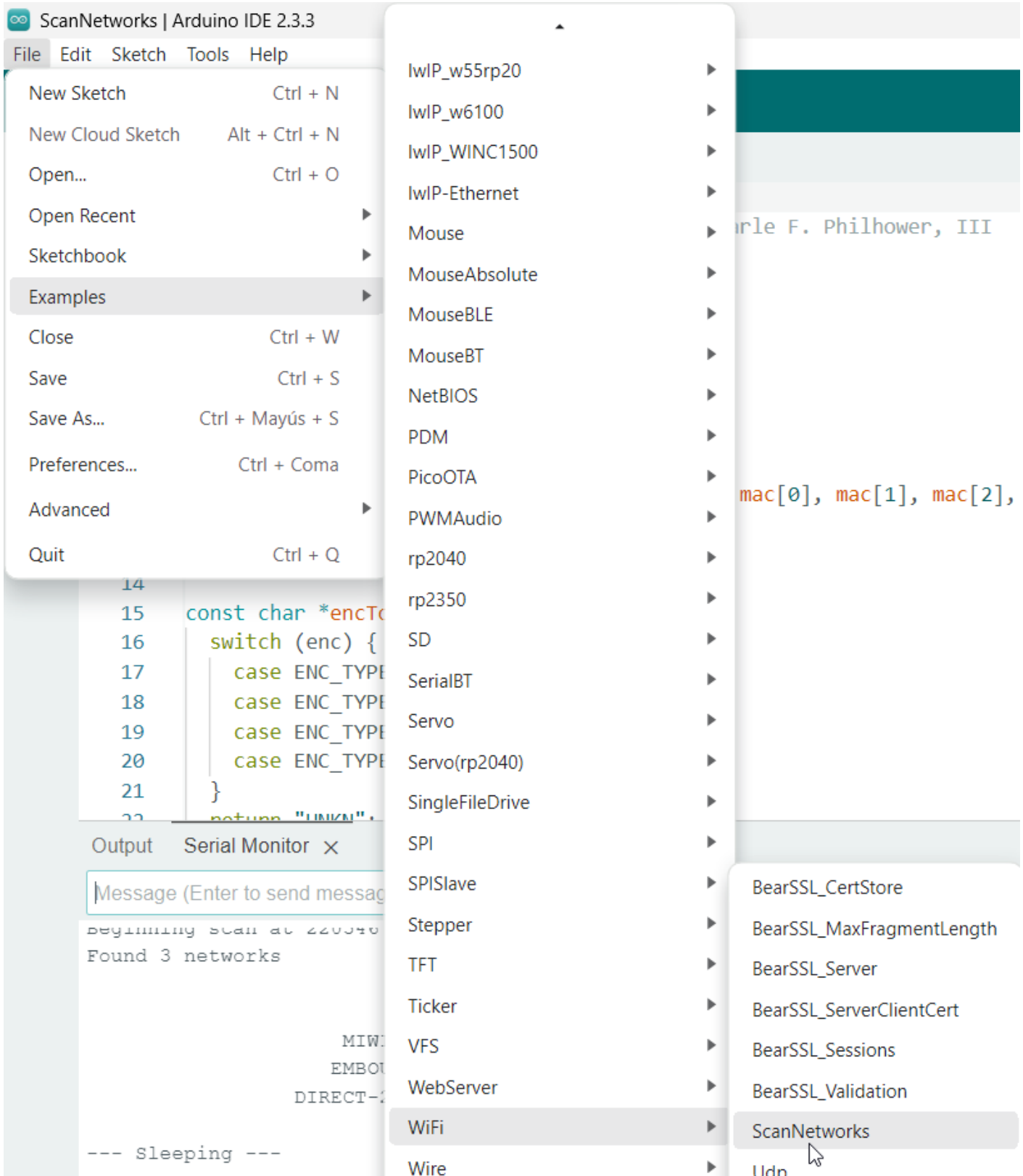
https://drive.google.com/file/d/1PDql_GYyxcz68jqmQAGOLs0YE6SXgPCm/preview

Al no tener licencia CC no los podemos reproducir aquí en este tutorial

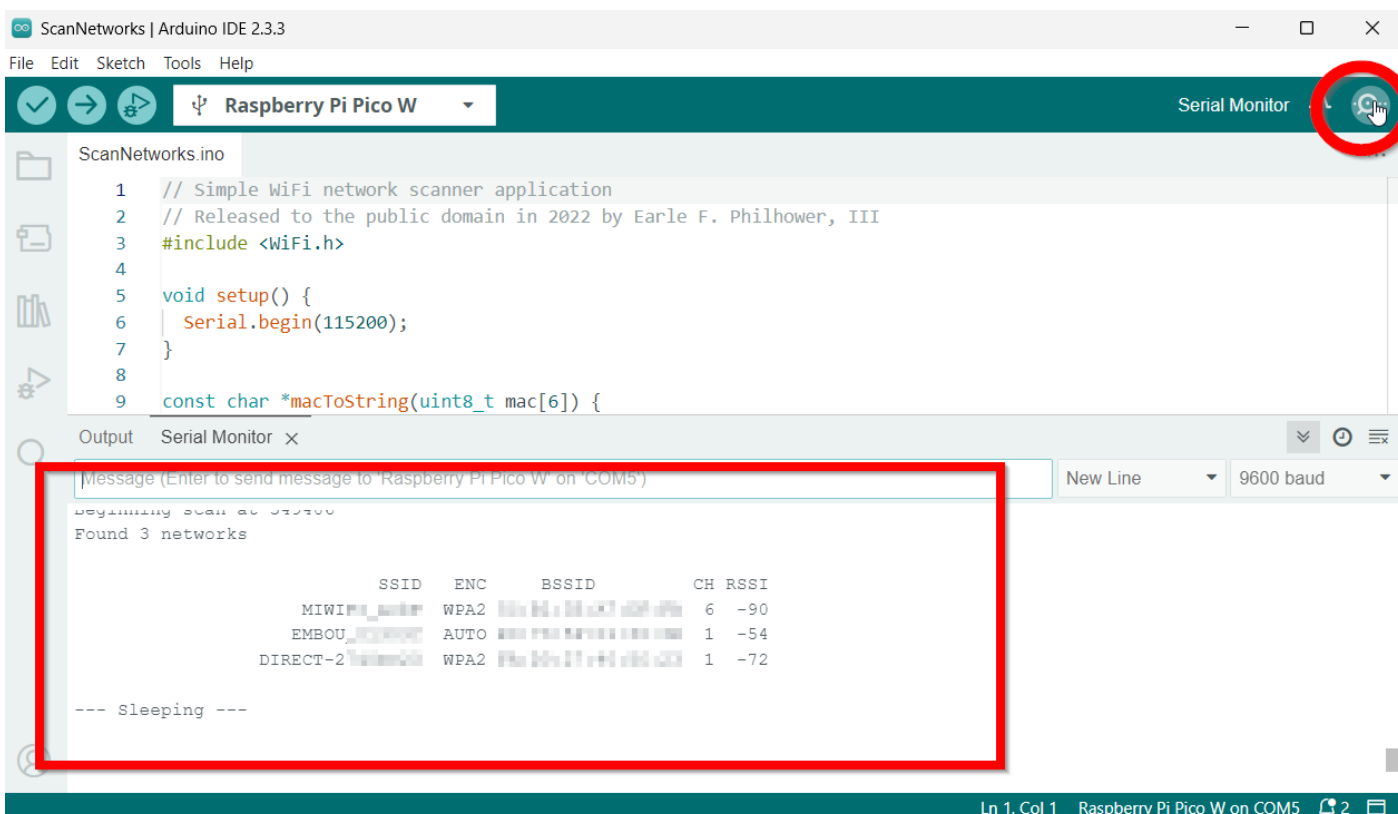
Proyectos con Wifi

ScanNetworks

Podemos ir a Ejemplos y vamos a ejecutar el **ScanNetworks** que no requiere contraseñas de wifi

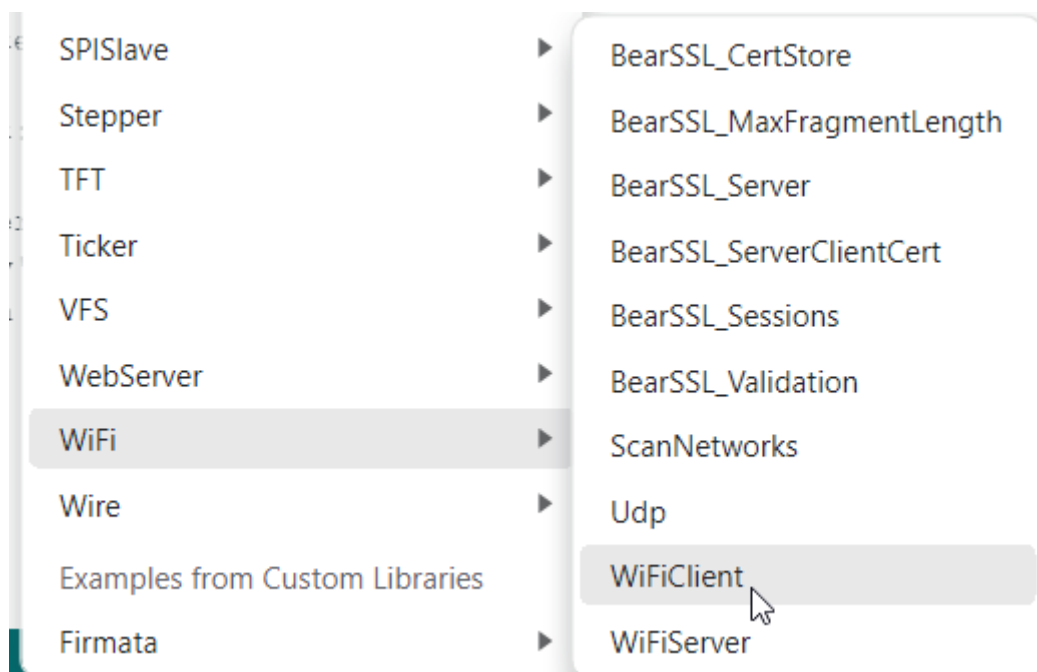


Al ejecutarlo nos sale en la ventana de monitor serie las wifis que encuentra



WifiClient

Si nos vamos a este ejemplo:



Tenemos que ir a las líneas 9 y 10 para poner nuestra wifi y contraseña:

```
#define STASSID "aquituwifi"
#define STAPSK "aqui tu contraseña"
```

Al ejecutarlo llama a esta dirección definida en las líneas 16 y 17

```
const char* host = "djxmmx.net";
const uint16_t port = 17;
```

Y sale **la cita del día** cada 5 minutos en el puerto serie

```
WiFi connected
IP address:
192.168.1.48
connecting to djxmmx.net:17
sending data to server
receiving from remote server
"Thomas wants to get it in quickly, and...Now there's a steal by Bird!
Underneath to DJ, lays it in!
1 second left, what a play by Bird!"
[] - 1987 NBA Eastern Conference Finals

closing connection
connecting to djxmmx.net:17
sending data to server
receiving from remote server
"Oh oh oh oh oh ooh, why'd you have to go oh, away from all, me love, why you leave me, w-why
you leave me?..."
[] - Sean Kingston (Me Love)

closing connection
connecting to djxmmx.net:17
sending data to server
receiving from remote server
"I wanna run away, with you, cuz baby you're my everything..."
[] - Frankie J. (Run Away)
```

closing connection

Encender y apagar un led

De <https://dronebotworkshop.com/picow-arduino/> hemos conseguido este código, donde Raspberry actúa como cliente, pero esta pensado para encender y apagar el led integrado en la Raspberry Pi Pico W, así que le hemos añadido el Led rojo de PicoBrick que esta en GPI7

- En la línea 43 el pin 7 como salida **pinMode(7,OUTPUT);**
- En la línea 97 que encienda el pin 7 también **digitalWrite(7, HIGH);**
- En la línea 102 que apague el pin 7 también **digitalWrite(7, LOW);**
- Acuérdate de poner en las líneas 17 y 18 tu wifi

```
/*  
  Pico W Web Interface Demo  
  picow-web-control-demo.ino  
  Web Interface & WiFi Connection  
  Control the onboard LED with Pico W  
  
  Adapted from ESP32 example by Rui Santos - https://randomnerdtutorials.com  
  
  DroneBot Workshop 2022  
  https://dronebotworkshop.com  
*/  
  
// Load Wi-Fi library  
#include <WiFi.h>  
  
// Replace with your network credentials  
const char* ssid = "pontuwifi";  
const char* password = "pontucontraseña";  
  
// Set web server port number to 80  
WiFiServer server(80);  
  
// Variable to store the HTTP request  
String header;
```

```
// Variable to store onboard LED state
String picoLEDState = "off";

// Current time
unsigned long currentTime = millis();
// Previous time
unsigned long previousTime = 0;
// Define timeout time in milliseconds (example: 2000ms = 2s)
const long timeoutTime = 2000;

void setup() {

    // Start Serial Monitor
    Serial.begin(115200);

    // Initialize the LED as an output
    pinMode(LED_BUILTIN, OUTPUT);
    pinMode(7,OUTPUT);//initialize digital pin 7 as an output

    // Set LED off
    digitalWrite(LED_BUILTIN, LOW);

    // Connect to Wi-Fi network with SSID and password
    WiFi.begin(ssid, password);

    // Display progress on Serial monitor
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }

    // Print local IP address and start web server
    Serial.println("");
    Serial.print("WiFi connected at IP Address ");
    Serial.println(WiFi.localIP());
```



```
// Start Server
server.begin();
}

void loop() {

  WiFiClient client = server.available();  // Listen for incoming clients

  if (client) {                            // If a new client connects,
    currentTime = millis();
    previousTime = currentTime;
    Serial.println("New Client.");          // print a message out in the serial port
    String currentLine = "";               // make a String to hold incoming data from the
client
    while (client.connected() && currentTime - previousTime <= timeoutTime) { // loop while
the client's connected
      currentTime = millis();
      if (client.available()) {             // if there's bytes to read from the client,
        char c = client.read();             // read a byte, then
        Serial.write(c);                   // print it out the serial monitor
        header += c;
        if (c == '\n') {                   // if the byte is a newline character
          // if the current line is blank, you got two newline characters in a row.
          // that's the end of the client HTTP request, so send a response:
          if (currentLine.length() == 0) {
            // HTTP headers always start with a response code (e.g. HTTP/1.1 200 OK)
            // and a content-type so the client knows what's coming, then a blank line:
            client.println("HTTP/1.1 200 OK");
            client.println("Content-type:text/html");
            client.println("Connection: close");
            client.println();

            // Switch the LED on and off
            if (header.indexOf("GET /led/on") >= 0) {
              Serial.println("LED on");
              picoLEDState = "on";
            }
          }
        }
      }
    }
  }
}
```

```

    digitalWrite(LED_BUILTIN, HIGH);
    digitalWrite(7, HIGH);
} else if (header.indexOf("GET /led/off") >= 0) {
    Serial.println("LED off");
    picoLEDState = "off";
    digitalWrite(LED_BUILTIN, LOW);
    digitalWrite(7, LOW);
}

// Display the HTML web page
client.println("<!DOCTYPE html><html>");
client.println("<head><meta name=\"viewport\" content=\"width=device-width,
initial-scale=1\">");
client.println("<link rel=\"icon\" href=\"data:,\>");

// CSS to style the on/off buttons
client.println("<style>html { font-family: Helvetica; display: inline-block;
margin: 0px auto; text-align: center;}");
client.println(".button { background-color: #4CAF50; border: none; color: white;
padding: 16px 40px;");
client.println("text-decoration: none; font-size: 30px; margin: 2px; cursor:
pointer;}");
client.println(".button2 {background-color: #F23A3A;}</style></head>");

// Web Page Heading
client.println("<body><h1>Pico W LED Control</h1>");

// Display current state, and ON/OFF buttons for Onboard LED
client.println("<p>Onboard LED is " + picoLEDState + "</p>");

// Set buttons
if (picoLEDState == "off") {

    //picoLEDState is off, display the ON button
    client.println("<p><a href=\"/led/on\"><button
class=\"button\">ON</button></a></p>");

```

```

    } else {

        //picoLEDState is on, display the OFF button
        client.println("<p><a href=\"/led/off\"><button class=\"button
button2\">OFF</button></a></p>");
    }

    client.println("</body></html>");

    // The HTTP response ends with another blank line
    client.println();
    // Break out of the while loop
    break;
} else { // if you got a newline, then clear currentLine
    currentLine = "";
}
} else if (c != '\r') { // if you got anything else but a carriage return character,
    currentLine += c;    // add it to the end of the currentLine
}
}
}
// Clear the header variable
header = "";
// Close the connection
client.stop();
Serial.println("Client disconnected.");
Serial.println("");
}
}

```

Al ejecutarlo, nos aparece por el puerto serie la IP que se ha conectado:



```

105 // Display the HTML web page
106 client.println("<!DOCTYPE html><html>");
// ... (rest of the code is partially visible)

```

Output Serial Monitor X

Message (Enter to send message to 'Raspberry Pi Pico W' on 'COM5')

```

New Client.
GET /led/off HTTP/1.1
Host: 192.168.1.48
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWe

```

Entramos en un navegador en un ordenador conectado a la misma wifi en la dirección 192.168.1.48 y este es el resultado:

<https://www.youtube.com/embed/haE4GdOd4zo>

¿Y con un servo?

Si tienes un servo puedes conectarlo y también puedes utilizar el código que aparece en De <https://dronebotworkshop.com/picow-arduino/> con las conexiones que indican