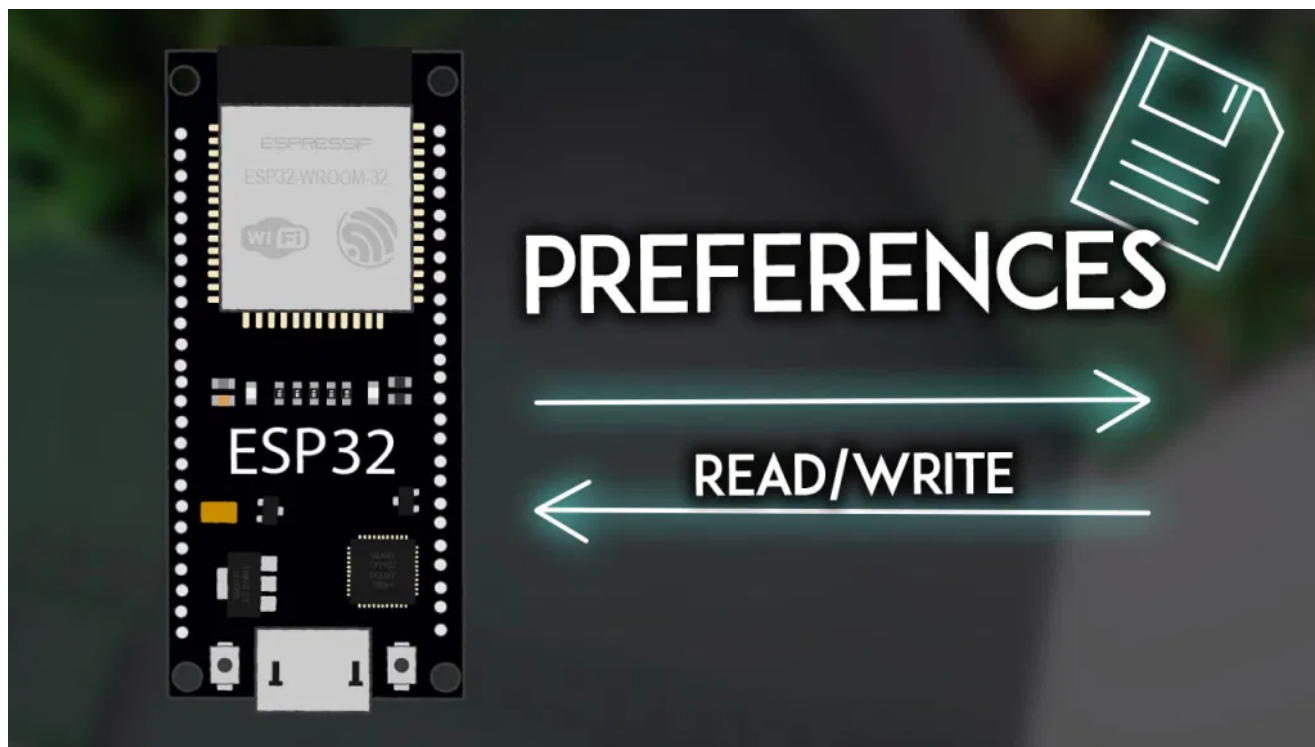


# ESP32 Save Data Permanently using Preferences Library

This guide shows how to save data permanently on the ESP32 flash memory using the `Preferences.h` library. The data held in the flash memory persists across resets or power failures. Using the `Preferences.h` library is useful to save data like network credentials, API keys, threshold values, or even the last state of a GPIO. You'll learn how to save and read data from flash memory.



In this tutorial, we'll cover the following topics:

- [Save \*key:value\* pairs;](#)
- [Read a \*key\* value;](#)
- [Example 1: Save \*key:value\* pairs;](#)
- [Example 2: ESP32 – Save/Read Network Credentials using the Preferences.h Library;](#)
- [Example 3: ESP32 – Remember Last GPIO State After RESET;](#)

In a [previous tutorial](#), we recommended using the EEPROM library to save data on flash memory. However, the EEPROM library is deprecated in favor of the `Preferences.h` library. This library is “installed” automatically when you install the ESP32 boards in your Arduino IDE.

The `Preferences.h` library is preferably used to store variable values through key:value pairs. Saving data permanently can be important to:

- remember the last state of a variable;
- save settings;
- save how many times an appliance was activated;
- or any other data type you need to save permanently.

If, instead of variables, you need to save files on the ESP32, we recommend using the filesystem (SPIFFS) instead. To learn how to save files in the ESP32 filesystem, you can read one of the following tutorials:

- [Install ESP32 Filesystem Uploader in Arduino IDE](#)
- [ESP32 with VS Code and PlatformIO: Upload Files to Filesystem \(SPIFFS\)](#)

## Save Data Using Preferences.h Library

The data saved using preferences is structured like this:

```
namespace {  
  key:value  
}
```

You can save different keys on the same namespace, for example:

```
namespace {  
  key1: value1  
  key2: value2  
}
```

In a practical example, this configuration could be used to save your network credentials:

```
credentials {  
  ssid: "your_ssid"  
  pass: "your_pass"  
}
```

In the preceding example, `credentials` is the namespace, and `ssid` and `pass` are the keys.

You can also have multiple namespaces with the same key (but each key with its value):

```
namespace1{  
  key:value1  
}  
namespace2{  
  key:value2  
}
```

When using the `Preferences.h` library, you should define the data type you want to save. Later, if you want to read that data, you must know the saved data type. In other words, the data type of writing and reading should be the same.

You can save the following data types using `Preferences.h`: `char`, `Uchar`, `short`, `Ushort`, `int`, `Uint`, `long`, `Ulong`, `long64`, `Ulong64`, `float`, `double`, `bool`, `string` and `bytes`.

For more information, you can access the [Preferences.cpp file here](#).

## Preferences.h Library Useful Functions

To use the `Preferences.h` library to store data, first you need to include it in

---

```
#include <Preferences.h>
```

Then, you must initiate an instance of the Preferences library. You can call it `preferences` , for example:

```
Preferences preferences;
```

After this, you can use the following methods to handle data using the `Preferences.h` library.

## Start Preferences

The `begin()` method opens a “storage space” with a defined namespace. The `false` argument means that we’ll use it in read/write mode. Use `true` to open or create the namespace in read-only mode.

```
preferences.begin("my-app", false);
```

In this case, the namespace name is `my-app` . **Namespace name is limited to 15 characters.**

## Clear Preferences

Use `clear()` to clear all preferences under the opened namespace (it doesn’t delete the namespace):

```
preferences.clear();
```

## Remove Key

Remove a key from the opened namespace:

```
preferences.remove(key);
```

## Close Preferences

Use the `end()` method to close the preferences under the opened namespace:

```
preferences.end();
```

## Put a Key Value (Save a value)

You should use different methods depending on the variable type you want to save.

<b>Char</b>	<code>putChar(const char* key, int8_t value)</code>
<b>Unsigned Char</b>	<code>putUChar(const char* key, int8_t value)</code>
<b>Short</b>	<code>putShort(const char* key, int16_t value)</code>
<b>Unsigned Short</b>	<code>putUShort(const char* key, uint16_t value)</code>
<b>Int</b>	<code>putInt(const char* key, int32_t value)</code>
<b>Unsigned Int</b>	<code>putUInt(const char* key, uint32_t value)</code>
<b>Long</b>	<code>putLong(const char* key, int32_t value)</code>
<b>Unsigned Long</b>	<code>putULong(const char* key, uint32_t value)</code>
<b>Long64</b>	<code>putLong64(const char* key, int64_t value)</code>
<b>Unsigned Long64</b>	<code>putULong64(const char* key, uint64_t value)</code>
<b>Float</b>	<code>putFloat(const char* key, const float_t value)</code>
<b>Double</b>	<code>putDouble(const char* key, const double_t value)</code>

<b>String</b>	<code>putString(const char* key, const String value)</code>
<b>Bytes</b>	<code>putBytes(const char* key, const void* value, size_t len)</code>

## Get a Key Value (Read Value)

Similarly, you should use different methods depending on the variable type you want to get.

<b>Char</b>	<code>getChar(const char* key, const int8_t defaultValue)</code>
<b>Unsigned Char</b>	<code>getUChar(const char* key, const uint8_t defaultValue)</code>
<b>Short</b>	<code>getShort(const char* key, const int16_t defaultValue)</code>
<b>Unsigned Short</b>	<code>getUShort(const char* key, const uint16_t defaultValue)</code>
<b>Int</b>	<code>getInt(const char* key, const int32_t defaultValue)</code>
<b>Unsigned Int</b>	<code>getUInt(const char* key, const uint32_t defaultValue)</code>
<b>Long</b>	<code>getLong(const char* key, const int32_t defaultValue)</code>
<b>Unsigned Long</b>	<code>getULong(const char* key, const uint32_t defaultValue)</code>
<b>Long64</b>	<code>getLong64(const char* key, const int64_t defaultValue)</code>
<b>...</b>	<b>...</b>

<b>Float</b>	<code>getFloat(const char* key, const float_t defaultValue)</code>
<b>Double</b>	<code>getDouble(const char* key, const double_t defaultValue)</code>
<b>Bool</b>	<code>getBool(const char* key, const bool defaultValue)</code>
<b>String</b>	<code>getString(const char* key, const String defaultValue)</code>
<b>String</b>	<code>getString(const char* key, char* value, const size_t maxlen)</code>
<b>Bytes</b>	<code>getBytes(const char* key, void * buf, size_t maxlen)</code>

## Remove a Namespace

In the Arduino implementation of Preferences, there is no method of completely removing a namespace. As a result, over the course of several projects, the ESP32 non-volatile storage (nvs) Preferences partition may become full. To completely erase and reformat the NVS memory used by Preferences, create a sketch that contains:

```
#include <nvs_flash.h>

void setup() {
  nvs_flash_erase(); // erase the NVS partition and...
  nvs_flash_init(); // initialize the NVS partition.
  while(true);
}

void loop() {
  1
```

You should download a new sketch to your board immediately after running the above, or it will reformat the NVS partition every time it is powered up.

## Preferences.h – Save key:value Pairs

For a simple example on how to save and get data using `Preferences.h`, in your Arduino IDE, go to **File > Examples > Preferences > StartCounter**.

```
// Open Preferences with my-app namespace. Each application module,
// has to use a namespace name to prevent key name collisions. We w
// RW-mode (second parameter has to be false).
// Note: Namespace name is limited to 15 chars.
preferences.begin("my-app", false);

// Remove all preferences under the opened namespace
//preferences.clear();

// Or remove the counter key only

//preferences.remove("counter");

// Get the counter value, if the key does not exist, return a default
// Note: Key name is limited to 15 chars.
unsigned int counter = preferences.getUInt("counter", 0);

// Increase counter by 1
counter++;

// Print the counter to Serial Monitor
Serial.printf("Current counter value: %u\n", counter);

// Store the counter to the Preferences
preferences.putUInt("counter", counter);

// Close the Preferences
preferences.end();
```



This example increases a variable called `counter` between resets. This illustrates that the ESP32 “remembers” the value even after a reset.

Upload the previous sketch to your ESP32 board. Open the Serial Monitor at a baud rate of 115200 and press the on-board RST button. You should see the `counter` variable increasing between resets.



```
entry 0x400806ac
Current counter value: 3
Restarting in 10 seconds...
ets Jun  8 2016 00:22:57

rst:0xc (SW_CPU_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x3fff0018,len:4
load:0x3fff001c,len:1044
load:0x40078000,len:8896
load:0x40080400,len:5816
entry 0x400806ac
Current counter value: 4
Restarting in 10 seconds...
```

## How the Code Works

This example uses the functions we’ve seen in the previous sections.

First, include the `Preferences.h` library.

```
#include <Preferences.h>
```

Then, create an instance of the library called `preferences`.

---

In the `setup()` , initialize the Serial Monitor at a baud rate of 115200.

```
Serial.begin(115200);
```

Create a “storage space” in the flash memory called `my-app` in read/write mode. You can give it any other name.

```
preferences.begin("my-app", false);
```

Get the value of the `counter` key saved on preferences. If it doesn't find any value, it returns `0` by default (which happens when this code runs for the first time).

```
unsigned int counter = preferences.getUInt("counter", 0);
```

The `counter` variable is increased one unit every time the ESP runs:

```
counter++;
```

Print the value of the `counter` variable:

```
Serial.printf("Current counter value: %u\n", counter);
```

Store the new value on the “counter” key:

```
preferences.putUInt("counter", counter);
```

Close the Preferences.

Finally, restart the ESP32 board:

```
ESP.restart();
```

---

## ESP32 – Save/Read Network Credentials using the Preferences.h Library

The `Preferences.h` library is many times used to save your network credentials permanently on the flash memory. This way, you don't have to hard code the credentials in every sketch that involves connecting the ESP32 to the internet.

In this section, we'll show you two simple sketches that might be useful in your projects:

- [Save Network Credentials using Preferences.h](#)
- [Connect to Wi-Fi with Network Credentials Saved on Preferences](#)

To learn more about ESP32 Wi-Fi related functions, read the following article:

- [ESP32 Useful Wi-Fi Library Functions \(Arduino IDE\)](#)

## Save Network Credentials using Preferences.h

The following sketch saves your network credentials permanently on the ESP32 flash memory using `Preferences.h`.

```
The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.  
*/
```

```
#include <Preferences.h>
```

```
Preferences preferences;
```

```
const char password = REPLACE_WITH_YOUR_PASSWORD ,

void setup() {
  Serial.begin(115200);
  Serial.println();

  preferences.begin("credentials", false);
  preferences.putString("ssid", ssid);
  preferences.putString("password", password);

  Serial.println("Network Credentials Saved using Preferences");

  preferences.end();
}

void loop() {

}
```

[View raw code](#)

Don't forget to insert your network credentials in the following variables:

```
const char* ssid = "REPLACE_WITH_YOUR_SSID";
const char* password = "REPLACE_WITH_YOUR_PASSWORD";
```

## How the Code Works

Let's take a quick look at the relevant parts of the code for this example.

In the `setup()` , create a new storage space on the flash memory with the `credentials` namespace.

```
preferences.begin("credentials", false);
```

---

Then, create a key called `ssid` that saves your SSID value (`ssid` variable) – use the `putString()` method.

```
preferences.putString("ssid", ssid);
```

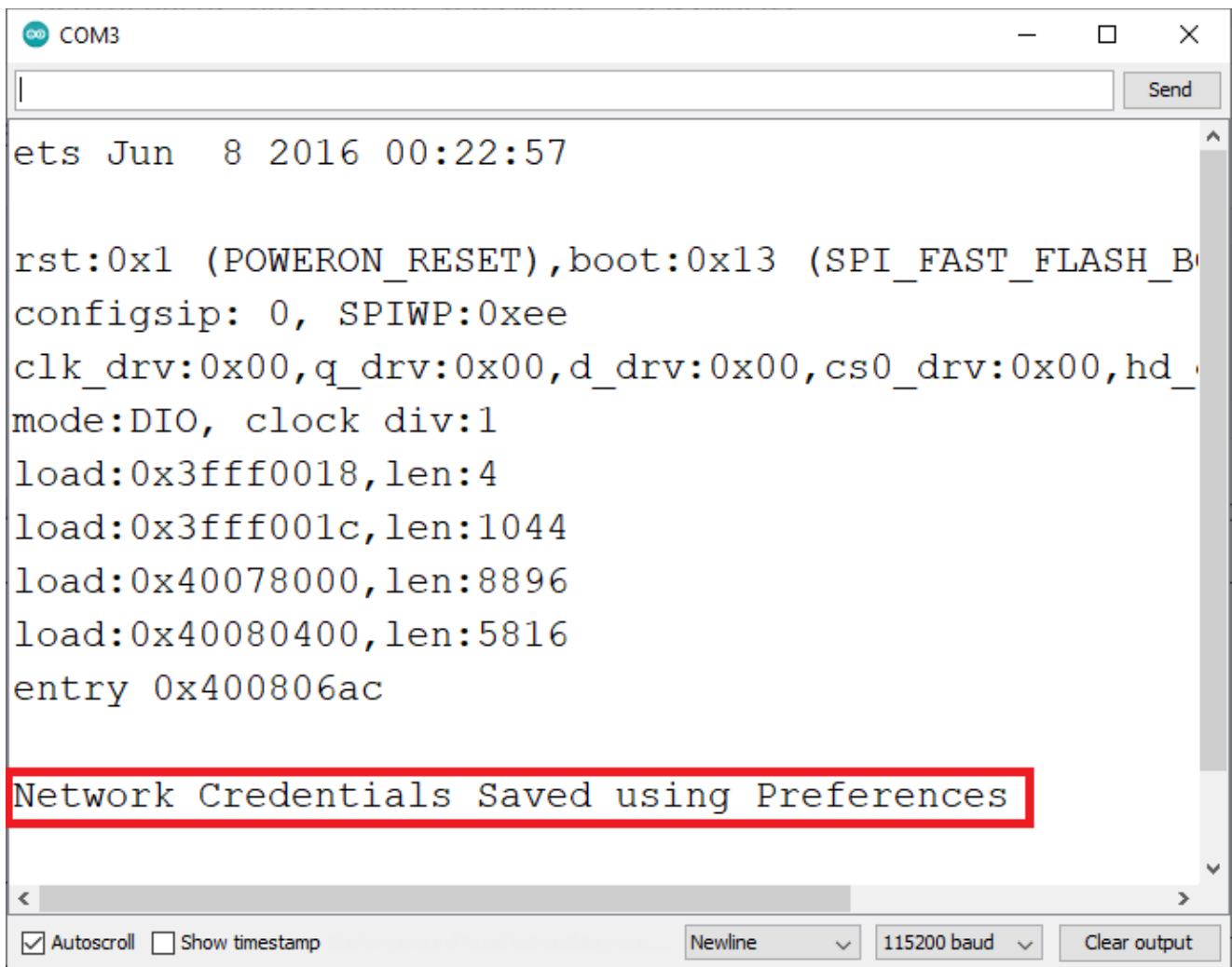
Add another key called `password` to save the password value (`password` variable):

```
preferences.putString("password", password);
```

So, your data is structured in this way:

```
credentials{  
  ssid: your_ssid  
  password: your_password  
}
```

Upload the code to your board and this is what you should get on the Serial Monitor:



```
ets Jun  8 2016 00:22:57

rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_
mode:DIO, clock div:1
load:0x3fff0018,len:4
load:0x3fff001c,len:1044
load:0x40078000,len:8896
load:0x40080400,len:5816
entry 0x400806ac

Network Credentials Saved using Preferences
```

In the following example, we'll show you how to read the network credentials from preferences and use them to connect the ESP32 to your network.

## Connect to Wi-Fi with Network Credentials Saved on Preferences

The following sketch gets the network credentials' values and connects to your network using those credentials.

```
/*
  Rui Santos
  Complete project details at https://RandomNerdTutorials.com/esp32-s3-wifi-credentials/

  Permission is hereby granted, free of charge, to any person obtaining
  a copy of this software and associated documentation files.
```

```
*/
```

```
#include <Preferences.h>
```

```
#include "WiFi.h"
```

```
Preferences preferences;
```

```
String ssid;
```

```
String password;
```

```
void setup() {
```

```
    Serial.begin(115200);
```

```
    Serial.println();
```

```
    preferences.begin("credentials", false);
```

```
    ssid = preferences.getString("ssid", "");
```

[View raw code](#)

## How the Code Works

Let's take a quick look at the relevant parts of the code for this example.

Open the `credentials` namespace:

```
preferences.begin("credentials", false);
```

Get the SSID and password values using the `getString()` method. You need to use the key name that you used to save the variables, in this case, `ssid` and `password` keys:

```
ssid = preferences.getString("ssid", "");
```

```
password = preferences.getString("password", "");
```

As a second argument to the `getString()` function, we passed an empty String. This is the returned value in case there aren't `ssid` or `password` keys saved on preferences.

If that's the case, we print a message indicating that there aren't any saved values:

```
if (ssid == "" || password == ""){  
    Serial.println("No values saved for ssid or password");  
}
```

Otherwise, we connect to Wi-Fi using the SSID and password saved on preferences.

```
else {  
    // Connect to Wi-Fi  
    WiFi.mode(WIFI_STA);  
    WiFi.begin(ssid.c_str(), password.c_str());  
    Serial.print("Connecting to WiFi ..");  
    while (WiFi.status() != WL_CONNECTED) {  
        Serial.print('.');  
        delay(1000);  
    }  
    Serial.println(WiFi.localIP());  
}
```

Upload this code to your board after the previous one (to ensure that you have the credentials saved). If everything goes as expected, this is what you should get on your Serial Monitor.



```
COM3
|
Send

rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
config: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_
mode:DIO, clock div:1
load:0x3fff0018,len:4
load:0x3fff001c,len:1044
load:0x40078000,len:8896
load:0x40080400,len:5816
entry 0x400806ac

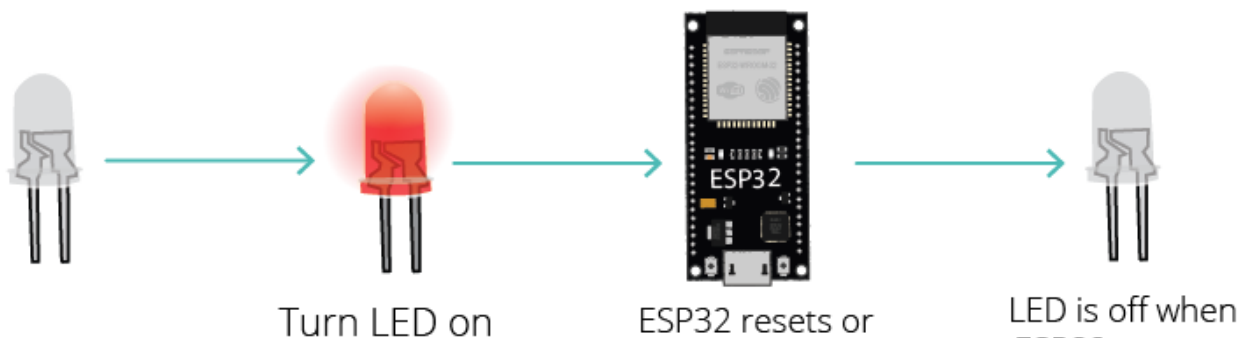
Connecting to WiFi ...192.168.1.114
```

☒ Autoscroll ☐ Show timestamp Newline 115200 baud Clear output

## Remember Last GPIO State After RESET

Another application of the `Preferences.h` library is to save the last state of an output. For example, imagine the following scenario:

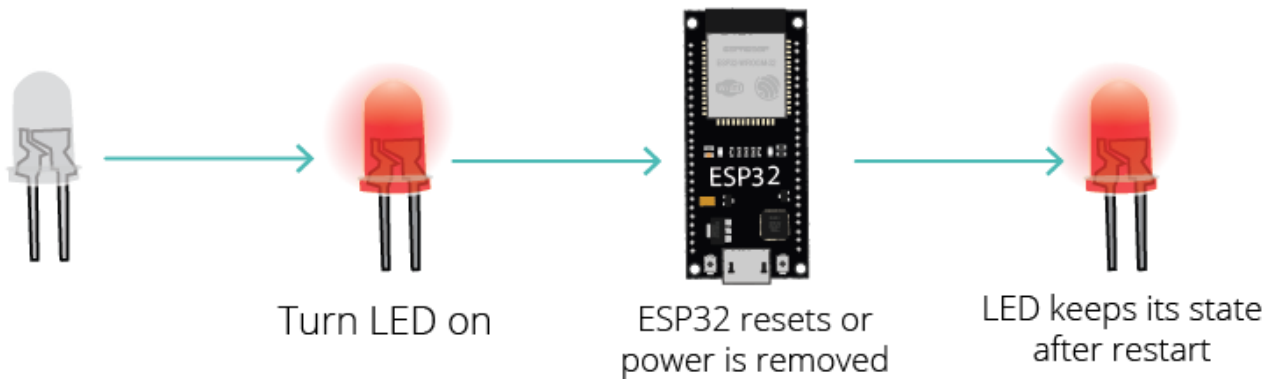
1. You're controlling an output with the ESP32;
2. You set your output to turn on;
3. The ESP32 suddenly loses power;
4. When the power comes back on, the output stays off – because it didn't keep its last state.



You don't want this to happen. You want the ESP32 to remember what was happening before losing power and return to the last state.

To solve this problem, you can save the lamp's state in the flash memory. Then, you need to add a condition at the beginning of your sketch to check the last lamp state and turn the lamp on or off accordingly.

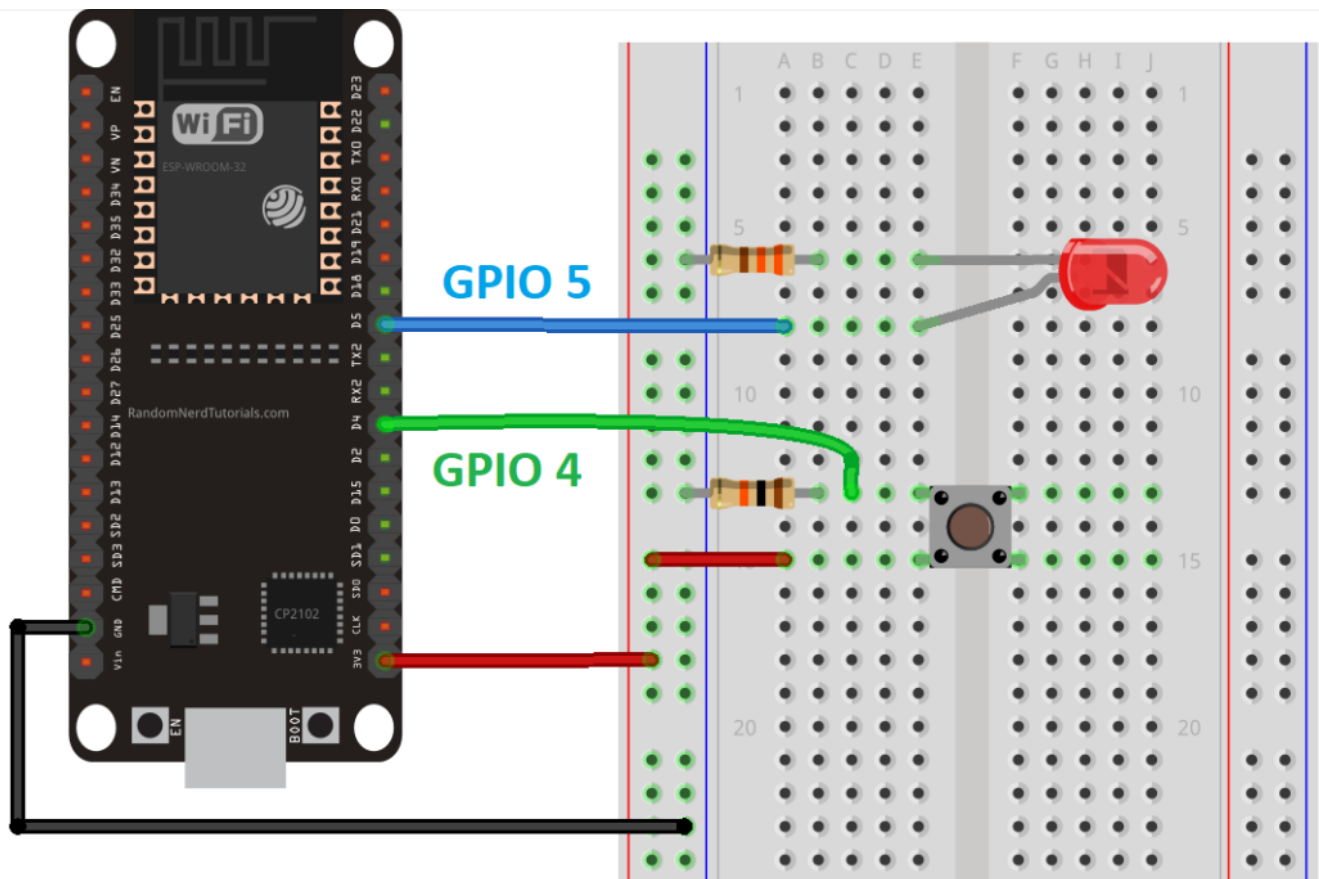
The following figure shows what we're going to do:



We'll show you an example using an LED and a pushbutton. The pushbutton controls the LED state. The LED keeps its state between resets. This means that if the LED is lit when you remove power, it will be lit when it gets powered again.

## Schematic Diagram

Wire a pushbutton and an LED to the ESP32 as shown in the following schematic diagram.



Recommended reading: ESP32 Pinout Reference: Which GPIO pins should you use?

## Code

This is a debounce code that changes the LED state every time you press the pushbutton. But there's something special about this code – it remembers the last LED state, even after resetting or removing power from the ESP32. This is possible because we save the led state on Preferences whenever it changes.

```
/*
```

```
Rui Santos
```

```
Complete project details at https://RandomNerdTutorials.com/esp32-s
```

```
Permission is hereby granted, free of charge, to any person obtaining
of this software and associated documentation files.
```

```
The above copyright notice and this permission notice shall be included
in all copies or substantial portions of the Software.
```

```
#include <Preferences.h>
```

```
Preferences preferences;
```

```
const int buttonPin = 4;
```

```
const int ledPin = 5;
```

```
bool ledState;
```

```
bool buttonState;
```

```
int lastButtonState = LOW;
```

```
unsigned long lastDebounceTime = 0; // the last time the output pin was
```

```
unsigned long debounceDelay = 50; // the debounce time; increase if the
```

```
void setup() {
```

```
    Serial.begin(115200);
```

[View raw code](#)

## How the Code Works

Let's take a quick look at the relevant parts of code for this example.

In the `setup()`, start by creating a section in the flash memory to save the GPIO state. In this example, we've called it `gpio`.

```
preferences.begin("gpio", false);
```

Get the GPIO state saved on Preferences on the `state` key. It is a boolean variable, so use the `getBool()` function. If there isn't any `state` key yet (which happens when the ESP32 first runs), return `false` (the LED will be off).

```
ledState = preferences.getBool("state", false);
```

```
Serial.printf("LED state before reset: %d \n", ledState);  
// set the LED to the last stored state  
digitalWrite(ledPin, ledState);
```

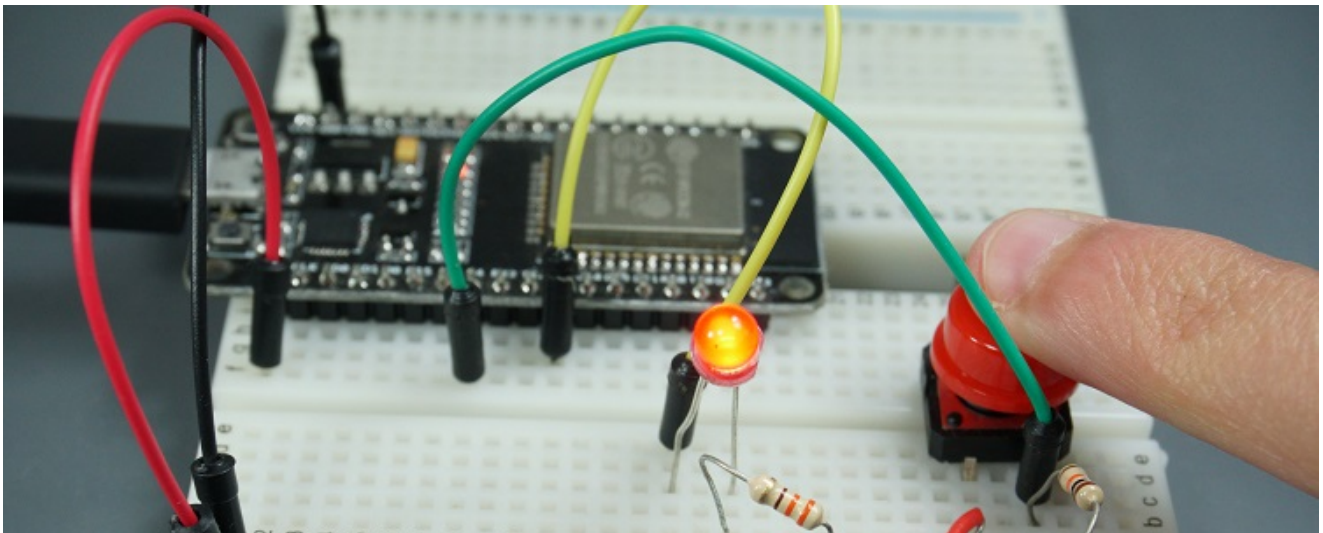
Finally, in the `loop()` update the `state` key on Preferences whenever there's a change.

```
// save the LED state in flash memory  
preferences.putBool("state", ledState);  
  
Serial.printf("State saved: %d \n", ledState);
```

## Demonstration

Upload the code to your board and wire the circuit. Open the Serial Monitor at a baud rate of 115200 and press the on-board RST button.

Press the pushbutton to change the LED state and then remove power or press the RST button.



When the ESP32 restarts, it will read the last state saved on Preferences and set

---

there's a change on the GPIO state.

## Wrapping Up

In this tutorial, you've learned how to save data permanently on the ESP32 flash memory using the Preferences.h library. This library is handy to save key:value pairs. Data held on the flash memory remains there even after resetting the ESP32 or removing power.

If you need to store bigger amounts of data or files, you should use the ESP32 filesystem (SPIFFS) or a microSD card instead:

- [Install ESP32 Filesystem Uploader in Arduino IDE](#)
- [ESP32 with VS Code and PlatformIO: Upload Files to Filesystem \(SPIFFS\)](#)
- [ESP32 Data Logging Temperature to MicroSD Card](#)

We hope you've found this tutorial useful.

- [Learn ESP32 with Arduino IDE](#)
- [Build Web Servers with ESP32 and ESP8266](#)
- [More ESP32 Projects and Tutorials...](#)



PCB Fabrication & Assembly

**ONLY \$5 for 10 PCBs**

✓ 24-hour Build Time    ✓ Quality Guaranteed

✓ Most Soldermask Colors:



[Order now](#)



### [eBook] Build Web Servers with ESP32 and ESP8266 (2nd Edition)

Build Web Server projects with the ESP32 and ESP8266 boards to control outputs and monitor sensors remotely. Learn HTML, CSS, JavaScript and client-server communication protocols [DOWNLOAD »](#)

## Recommended Resources

ESP8266, Arduino, and Node-RED.

[Home Automation using ESP8266 eBook and video course »](#) Build IoT and home automation projects.

[Arduino Step-by-Step Projects »](#) Build 25 Arduino projects with our course, even with no prior experience!

## What to Read Next...



[ESP8266 NodeMCU Async Web Server – Control Outputs with Arduino IDE \(ESPAsyncWebServer library\)](#)

[ESP8266 Troubleshooting Guide](#)

[ESP-NOW Two-Way Communication Between ESP32 Boards](#)

**Enjoyed this project? Stay updated by subscribing our**

☰ Menu



## 81 thoughts on “ESP32 Save Data Permanently using Preferences Library”

**Steve Platt**

March 2, 2021 at 7:04 pm

Hi Rui and all,

First, good post! A clear explanation of Preferences and how to use them (especially when preserving WiFi credentials, ESP-NOW node ID information, and so on).

I used Preferences in a large ESP32 project a while back. It is useful if you have a small-ish number of data items that need to be stored.

However, I had around a half-dozen or so larger structures, each with a dozen or more fields. I maintained the half-dozen or so keys, using putBytes/getBytes to save/restore data.

As the project grew, this became cumbersome. New versions of the project had additional data, and I could not use Preferences to maintain stored-data integrity as I moved to new versions (with the larger data blobs). I eventually switched over to SPIFFS and ArduinoJSON. This way, newer code could autodetect missing data, supply defaults, and then save the updated JSON data for later runs.

For simpler projects, Preferences works well. For larger projects, RandomNerd readers may wish to look into SPIFFS/JSON as a more flexible alternative.

-Steve

[Reply](#)

**Sara Santos**

March 2, 2021 at 11:01 pm

Hi Steve.

Thanks for your comment.

That's right. Preferences is good to save a small number of values.

If you need to save huge amounts of data, it is preferable to create a file in SPIFFS and use JSON.

Thanks for following our work.

Regards,

Sara

[Reply](#)

**P Ban**

February 28, 2022 at 9:49 am

how much should be the size of the data for storing user preferences?  
How many variables can be stored? How many times can I store (write) the data using preference?

[Reply](#)

**jj**

March 2, 2021 at 8:31 pm

SPIFFS is deprecated. You should be promoting LittleFS now.

**Sara Santos**

March 2, 2021 at 11:05 pm

I don't think SPIFFS is deprecated for the ESP32.

It is deprecated for the ESP8266.

Regards,

Sara

[Reply](#)

**Martin Lyon**

March 2, 2021 at 8:34 pm

With EEPROM, we know how much space was allotted. I think it was 500 bytes or 1K. Then, we specified how memory memory we needed. That gave confidence we would not overrun the space and screw up something else, like our program. Is there any such protection with Preferences? What if we save 1000 variables (exaggeration, of course)?

[Reply](#)

**Michael**

March 2, 2021 at 9:41 pm

Is this available through MicroPython? It looks very useful but I can't find it in MP docs.

**Sara Santos**

March 2, 2021 at 11:03 pm

Hi.

I don't think so.

To save data permanently, you can create a file on the ESP32 flash to store data.

See this: <https://forum.micropython.org/viewtopic.php?t=1620>

Regards,

Sara

[Reply](#)

**Carl Hage**

March 3, 2021 at 12:16 am

Nice tutorial. I hadn't heard of Preferences.h. It is a wrapper on the Espressif NVS API, which has a few more features. The NVS documentation has a good explanation of how this works:

[https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/storage/nvs\\_flash.html](https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/storage/nvs_flash.html)

NVS is optimized to store a set of key:value pairs grouped by namespace, and written into flash by appending in 4096 byte pages. When a new value is written the prior value is marked deleted, and new value appended. This way it doesn't wear out the flash. A RAM hash table helps search for entries (128-640 bytes of RAM per 4096 byte flash page). It's mainly intended for short values.

The main difference between Preferences.h and the NVS api is an

I also just learned about the Arduino error logging, `log_e().log_v()`, used by `Preferences.cpp`. Rui & Sara maybe that would make a nice additional tutorial to explain how logging works and how to control the level selected at compile time. It seems nontrivial, since `stdout` can be redefined and selecting log levels is slightly different in PlatformIO than basic Arduino IDE.

Note you can still use JSON with the NVS, storing the JSON as a string or blob. I haven't figured out the SPIFFS format, but depending on the JSON size, partition sizes, and write frequency, NVS could be better. [Note you can also select the partition in the `begin()`.] NVS writes 32 byte chunks vs 256 byte pages with SPIFFS. Both wear level across 4096 byte blocks in the partition.

[Reply](#)

**Alfons**

May 4, 2021 at 7:51 pm

Muy buena aportación Carl Hage !!!!

Lo tendré en cuenta, sobre todo en el tema de la encriptación para todo el tipo de credenciales en donde veo que NVS mejora a Preferences.

Para casos simples de variables de estado creo que Preferences es más simple de utilizar ¿no?

[Reply](#)

**Marcel**

March 21, 2022 at 3:13 pm

Thanks for the extra info Carl!

I invite you to take a look at an article I wrote a while back about the new logging API at <https://thingpulse.com/esp32-logging/>.

[Reply](#)

**james**

March 3, 2021 at 1:44 am

That is awesome I've been looking at how to save files to SD card .

[Reply](#)

**Sara Santos**

March 3, 2021 at 10:54 am

Hi James.

We'll post a tutorial about SD card soon.

Regards,

Sara

[Reply](#)

**Peter**

March 3, 2021 at 12:15 pm

As always, a very good guide  
Many Thanks

[Reply](#)

**lubond**

March 3, 2021 at 5:36 pm

Hi.

As usual another nice tutorial was born. Thank you Sara & Rui for your efforts and for the space you provide here for all of us.

I like key-value access way on this library. I like the automatic variable type recognition, background eeprom addressing. All it's methods seem simple and useful. But what more about very frequent writes?

Someone may be concerned about the life and durability of the flash. There are very frequent writes to "eeprom" required also in my projects. Maybe I don't understand well and maybe I am not alone, so let me have a few questions.

Carl, you said "This way it doesn't wear out the flash." But how is it possible, when there is a commit after each write?

Carl, did you mean that the commit will be done only and only on the "4096 byte flash page" (meaning namespace?) and not on the whole partition from the namespace comes? If so, would it be enough to separate commit (remove commit from every write method) and execute it separately? I am not very sure how Preferences lib (nvm.h) shares flash partition.



<https://tinkerman.cat/post/EEPROM-rotation-for-esp8266-and-esp32/>

Unfortunately this nice library as it is, despite that it shares same library (nvs.h), it does not allow key-value access. Also it does not support automatic variable type recognition, does not support background addressing in flash space and so on. Please correct me if I am wrong ... On the other side this library inherits all well known methods from famous standard library EEPROM.h. And very important for me is, that there the commit method may be called once after all data have been prepared previously (written in buffer). So this way does not wear out the flash, because each additional commit reduces it's life.

In this sense back to Preferences library.

Is it possible to consider each open namespace in Preference as a "partition" on which and only which a commit will be performed? If so, memory wear out can be solved by using multiple namespaces and by switching (rotating) between them.

I am not sure if it works this way. Can anyone answer that, please?

[Reply](#)

**Xylopyrographer**

March 5, 2021 at 6:13 pm

Short discussion on wear levelling: <https://esp32.com/viewtopic.php?t=3380>

If that is an issue, probably best to use the FAT or SPIFF implementation. In the Arduino implementation, a commit is performed with each ".put".

All the nitty-gritty on NVS implementation:

[https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/storage/nvs\\_flash.html](https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/storage/nvs_flash.html)

[Reply](#)

---

**Ed**

March 4, 2021 at 10:45 am

Good to know. I was not even aware of ths library. didnt see it in between the 'examples'

[Reply](#)

**McKwen2**

March 4, 2021 at 9:11 pm

Hi Rui and Sara,

Thank you very much for this tutorial. It will solve my long standing ESP8266 problem. Now, I have to switch to ESP32.

[Reply](#)

**Xylopyrographer**

March 4, 2021 at 11:31 pm

[UPDATE]

Please check comment: <https://randomnerdtutorials.com/esp32-save-data-permanently-preferences/#comment-734357>

[Reply](#)

**Sara Santos**

March 6, 2021 at 10:54 am

Hi.

Thank you so much for sharing your knowledge.

I've updated the tutorial with some of your tips.

Thanks for taking the time to do this.

Regards,

Sara

[Reply](#)

**Xylopyrographer**

April 3, 2021 at 4:35 pm

Hi Sara. Didn't realize that markdown is supported by default when posting. Is there a way to edit a post so I can clean up the formatting?  
Thanks.

[Reply](#)

**Mat**

July 4, 2021 at 1:15 pm

Hey, quick note to say I just read through your long comment and found it very useful. Basically most of the way through reading the article I was thinking "yeh this is great but it's a bit simple – I'll definitely need to have some way of writing initial factory default values to memory, which also means I will need to create a bool flag

couple of hours writing that comment – it definitely helps me as I'm writing my first firmware for an ESP32 and trying to figure these things out. Cheers!

[Reply](#)

### **Xylopyrographer**

July 5, 2021 at 5:16 pm

Mat, glad it was of use. Was also writing my first Arduino sketch at the time and figured this might help others fill a few gaps I found in the docs. All the best.

### **Xylopyrographer**

March 28, 2022 at 5:42 pm

**Sara:** I wrote a lengthy [comment] (<https://randomnerdtutorials.com/esp32-save-data-permanently-preferences/#comment-566327>) a while back to the [ESP32 Preferences] (<https://randomnerdtutorials.com/esp32-save-data-permanently-preferences/>) tutorial. Since then I've found that some of the information in that post is quite incorrect — mostly in the way I described how keys within a namespace are created.

I've now corrected all this and have had published in the official [arduino-esp32 project documentation](#) the [Preferences API] (<https://docs.espressif.com/projects/arduino-esp32/en/latest/api/preferences.html>) and a [Tutorial] (<https://docs.espressif.com/projects/arduino-esp32/en/latest/tutorials/preferences.html>) to match.

If it doesn't cause too much trouble, could you flag my original post on your site with a disclaimer (or delete it completely in lieu of this note) and point to the official documents instead.

Thanks again for a great site and all the work you put into it.

[Reply](#)

**Sara Santos**

March 28, 2022 at 10:33 pm

Hi.  
Thanks for sharing your documentation.  
I've updated your previous comment.  
Regards,  
Sara

[Reply](#)

**Paul**

March 5, 2021 at 1:23 pm

I great tutorial for the ESP32. However, because of it's very small size, I use Node Mini ESP8266 quite often. Does Preferences work on ESP8266?

Thx, and keep up the great work,  
P

[Reply](#)

**Sara Santos**

March 6, 2021 at 12:09 am

Hi Paul.

Unfortunately, I don't think it is compatible with the ESP8266.

Regards,

Sara

[Reply](#)

**smMahmoodi**

January 30, 2022 at 4:15 pm

For those who are stock with ESP8266 (like me!) I have found a library which passed the simple read and write test on my board. Here is the URL:

<https://github.com/maarten-pennings/Nvm>

You need to add the extracted zip file to Ardiono IDE's "libraries" directory.

[Reply](#)

**Ralph McCleery**

March 9, 2021 at 5:40 am

How would you use this to store an array of values? It will be 30 x int's of

[Reply](#)

**Steve Platt**

March 9, 2021 at 10:53 am

If you have `int myArr[30]`, you can use `myPrefs.putBytes("tag", myArr, 30*sizeof(int))`.

I usually use a typedef for the array, and just use `sizeof(theType)` instead of the `30*sizeof(int)`.

[Reply](#)

**Ralph McCleery**

March 9, 2021 at 9:37 pm

Thank you Steve,  
I've got a bit of learning / testing to do to see if I can get this to work.  
Very much appreciate your guidance.

I used tinkercad to figure out how to do a "run-once" when the first usage sample is calculated which then populates the averaging array thereafter we step along the array each write and update.

This does create a new question though, if after the first full set of array usage values and have a complete 30 day average and want to store that as a historical value would it be better to do all the storage in "SPIFFS" or do the averaging in "Preferences" and the logging in "SPIFFS"?

Cheers

**Steve Platt**

March 13, 2021 at 6:06 pm

Hi Ralph,

I don't think you can use SPIFFS and Preferences at the same time. I haven't examined the driver code, but I imagine they read/write raw(ish) flash blocks. (Same with FATFS vs SPIFFS, both use the raw flash space.)

That's why I moved to SPIFFS from Preferences for the near term. I may move to FAT at some point, but at least the API is the same!

Does your node have WiFi access? If so, you can connect, grab the date from an NTP server, and use this to construct the archival filenames under SPIFFS or FAT (at least as far as filename length restrictions). If you've done this, you can also set up the real time clock on the ESP as well, should you need to timestamp the datapoints in the data file.

-Steve

[Reply](#)

**Ralph McCleery**

May 20, 2021 at 12:38 am

Sorry Steve, I missed your reply for some reason and I've been sidetracked with other things so my progress stopped for a while. Back to it now though..... After some investigating I found out why no reply, my bad. Should be OK now.



As you have probably already gathered I'm just a hobbyist and am self learning so sometimes things take me a while to understand.

Thanks for the compatibility warning.

Yes I already have NTP running on a weather prediction node that has been running for some time now to debug the modifications I did to the original sketch. The goal is to pass outdoor temp/humidity/water tank level data sent from remote remote node. It's the remote node where I want to save the array. The remote node will be run taking advantage of "Deep Sleep" power saving and I was thinking of getting it running as a stand-alone "Blynk" instance first to aid development hence the need to save the array. I've already got the "Blynk" side running just not with the array and "Deep Sleep".

Right or wrong I find it easier to work things out by building in modules and then when each works reliably I combine them.

Cheers  
Ralph

**M. Rizki Setiawan**

March 29, 2021 at 8:51 am

Hii,  
is it possible to replace "state" in below line with var?  
I try to make recursive routine to get some key  
ex: "state"+ String(i),  
I try but still fail

ledState = preferences.getBool("state", false);

[Reply](#)

**Sara Santos**

March 29, 2021 at 10:26 am

Hi.

I'm sorry, but I didn't understand your question.

Regards,

Sara

[Reply](#)

**M. Rizki Setiawan**

March 29, 2021 at 1:03 pm

I am sorry Sara,

I want to read the bool value from key in namespace.

```
bool a;  
for (int x = 1; x < 4 ;x++){  
a = preferences.getBool("Var_A" + String(x), false);  
Serial.Println(a);  
}
```

the will give the read value

Var\_A1

Var\_A2

Var\_A3

Var\_A4

[Reply](#)

**M. Rizki Setiawan**

Hi Sara,

I already found the solution, thanks for the good tutorial

```
for (int x = 1; x < DO_n+1;x++){  
a = preferences.getBool(("AM_S_CH"+String(x)).c_str(), false);  
b = preferences.getBool(("AM_C_CH"+String(x)).c_str(), false);  
c =  
preferences.getBool(("SCH_"+String(preferences.getInt(("SCH_CH  
"+String(x)+"_D"+String(c_day)).c_str()))).c_str(), false);  
d = preferences.getBool(("DI1_En_CH"+String(x)).c_str(),  
false);  
e = preferences.getBool(("DI2_En_CH"+String(x)).c_str(),  
false);  
f = preferences.getBool(("DI3_En_CH"+String(x)).c_str(),  
false);
```

[Reply](#)

**Xylopyrographer**

March 30, 2021 at 4:51 pm

Hi. Might be a couple of things to check. Outside of this snippet, I'll assume the namespace was opened with a `preferences.begin("namespace", true);` statement? Also assuming this namespace has keys created elsewhere that match the descriptions in your statements `a =` thought to `f =`?

When using the `preferences.getXXX("keyname", default);` form, `default` is returned if `"keyname"` does not exist in the currently opened namespace. If you're sure `"keyname"` exists, use the `preferences.getXXX("keyname");` form.

Also, keynames are limited to 13 characters maximum. The

There is a pretty detailed explanation in this comment above:  
<https://randomnerdtutorials.com/esp32-save-data-permanently-preferences/#comment-566327>

**Sara Santos**

March 30, 2021 at 5:41 pm

Great!

**M. Rizki Setiawan**

April 2, 2021 at 7:32 am

Dear Xylopyrographer,  
the preferences.begin(“namespace”, true); already define above ,  
I just copy some off code,

thanks for the explanation,

by the way,

- is there any way to list/print the available namespace and keys ?
- and is there any keys limit number we can create?

[Reply](#)

**Xylopyrographer**

April 3, 2021 at 4:33 pm

It looks like there is a way to list what is in a namespace. That facility isn't supported in the ESP32 Arduino Preferences implementation but is available using the ESP32 function calls directly. See:

[https://docs.espressif.com/projects/esp-idf/en/release-v4.1/api-reference/storage/nvs\\_flash.html](https://docs.espressif.com/projects/esp-idf/en/release-v4.1/api-reference/storage/nvs_flash.html)

and look near the bottom of that page for the `nvs_entry_find()` method.

If I recall the default NVS partition is made up of 4 pages of 4096 bytes each. So that makes about 16k of NVS storage available. This is the limit, not really the number of entries. From that same link, there are the methods that can be used to determine the amount of NVS available.

In your sketch,

```
#include nvs.h
#include nvs_flash.h
```

and you'll be able to access these methods.

[Reply](#)

**M. Rizki Setiawan**

April 5, 2021 at 4:29 pm

I try to write 1440 keys with bool type  
but only 500 can be write, then the nvs become hang, cannot write  
anything, should be format again.

with i = 1440, is there any other method to store so much keys, >?

```
for (int x = 0;x < i;x++){
  Serial.print("Address : ");
  Serial.print(x);
```

```
Serial.print(" yang ditulis : ");  
Serial.println(j);  
preferences.putBool(("SCH"+b+"_"+String(g+x)).c_str(),j);  
}  
preferences.end();//Close the Preferences.  
Serial.println("jadwal berhasil diupdate");}
```

[Reply](#)

### Xylopyrographer

April 5, 2021 at 10:11 pm

I'd suggest taking a look at this line:

```
preferences.putBool(("SCH"+b+"_"+String(g+x)).c_str(),j);
```

A namespace key can be a maximum of 15 characters. Is there a chance this could cause that limit to be exceeded? The `.putX();` functions will fail if passed a key name longer than 15 characters. (The fail is graceful—writing an entry to the log. The function call returns 0. No value is written.)

Are all the key names created before the sketch tries to write to them?

If that isn't an issue, would suggest writing a test sketch that creates a namespace and then creates the 1440 key names, and then populates them but in a simple format like:

```
"A" + String(i)
```

so that it's certain that the 15 character limit won't be exceeded.

You could also:

test the return value after doing a `preferences.putBool()`. That set of

test the length of the key name to ensure its length is no greater than 15 characters before attempting to do a put.

Bool's are written to NVS as uint8\_t data type. You could use that to add a bit of math on the space required for your values.

Hope that helps.

[Reply](#)

**Eduardo**

April 10, 2021 at 10:36 pm

When should I use preferences.end()? There were examples that this command was used and in other examples it was not used.

[Reply](#)

**Sara Santos**

April 12, 2021 at 9:20 am

Hi Eduardo.

It is mandatory to call preferences.end(), if you want to open preferences on a different namespace than the previously open namespace.

Regards,

Sara

[Reply](#)

**Lee**

May 5, 2021 at 9:02 pm

I hope I can still use the EEPROM library, I only need to save an array of byte data, this seems far more complex?? (new ESP32 used from Arduino Nano)

[Reply](#)

**Sara Santos**

May 6, 2021 at 1:06 pm

Hi.

Yes, I think you can still use it.

Here is the tutorial for EEPROM: <https://randomnerdtutorials.com/esp32-flash-memory/>

Regards,

Sara

[Reply](#)

**Lee**

May 7, 2021 at 8:48 pm

Thanks..... I've decided I can simply put these bytes in the EEPROM that's already on board the DS3231 in the circuit... just Wire.h needed, nice and easy.... 😊

[Reply](#)



**Lee**

June 2, 2021 at 3:31 am

I gave this a good go last night, could not get it to work.... it compiled fine, but nothing was ever saved. It was only a single byte value I needed saved – lucky that old EEPROM library still works a treat, 5 minutes later all was how I wanted it....

[Reply](#)

**Mark Griffo**

June 28, 2021 at 7:08 pm

Wow just wow great tutorial. i am trying to get the following to work but to no avail. I'm new to Arduino esp32. i am trying to save the state of 7 GPIOs every time `getOutputStates()` is called

I am having problems getting the following line to work and not sure how to get it going

```
putInt(String(outputGPIOs[i]), int32_t String(digitalRead(outputGPIOs[i])));
```

```
String getOutputStates() {  
  JSONVar myArray;  
  for (int i = 0; i < NUM_OUTPUTS; i++) {  
    myArray["gpios"][i]["output"] = String(outputGPIOs[i]);
```

```
  
    myArray["gpios"][i]["state"] =  
      String(digitalRead(outputGPIOs[i]));  
    Serial.println(String(outputGPIOs[i]) + "=" +  
      String(digitalRead(outputGPIOs[i])));
```

```
//preferences.putBool("state", String(digitalRead(outputGPIOs[i])));  
  
}
```

i get the following error:

exit status 1

invalid conversion from 'int' to 'const char\*' [-fpermissive]

Thank you

[Reply](#)

**Mark Griffo**

June 28, 2021 at 7:10 pm

i noticed i cut off preferences. on a couple of lines:

```
String getOutputStates() {  
  JSONVar myArray;  
  for (int i = 0; i < NUM_OUTPUTS; i++) {  
    myArray["gprios"][i]["output"] = String(outputGPIOs[i]);  
  
    myArray["gprios"][i]["state"] = String(digitalRead(outputGPIOs[i]));  
    Serial.println(String(outputGPIOs[i]) + "=" +  
      String(digitalRead(outputGPIOs[i])));  
  
    preferences.putInt(int(outputGPIOs[i]), int32_t  
      (digitalRead(outputGPIOs[i])));  
  
    //preferences.putBool("state", String(digitalRead(outputGPIOs[i])));  
  
  }
```

[Reply](#)

**Ralph McCleery**

June 29, 2021 at 9:51 pm

Hi Mark,

I had the same struggles composing the complex statements. I wanted to build a 30 day averaging array and store an incrementing index value, last sensor read value and the array[30]. The reason to work this way is the use of “Deep Sleep” mode.

Being a self learning hacker (no formal training) and tinkerer I did a fair bit of research and realized it would be far better to use “SPIFFS” with an abstract layer helper library and stay away from the use of “JSONVar” which can lead to memory leakage issues (or so I read)

I created an example that uses “daily” random values simulating 22500lts as a full water tank and each reading reduces by what would be a “daily use” and is put into the array[10] to get the average use value. Obviously the array size can be anything you want. Example sketch here <https://pastebin.com/jPG5cXBk>

Note: Use the “SPIFFS” format sketch on this site to start over and I discovered even with this you’ll get an incorrect first value so I used the “clear()” class to get round that. You can use “esptool.py” and do an “erase\_flash” which will zero the flash but you must do a “SPIFFS format” before loading a new sketch or it will error.

Hope this helps

Cheers

Ralph

[Reply](#)

**Mark Griffo**

Ralph,

Thanks for the pointers. i think i have figured out. the “key” in preferences has to be char can not be any other type.

I am using SPIFFS for the web stuff to turn on and off the relays manually and set the schedule on and off times for each relay using an RTC.

I’m not concerned about memory leakage as the eps will be reset every night at midnight. now that i have the preferences working.

Thanks  
Mark

[Reply](#)

**Iuri Schmoeller**

August 3, 2021 at 1:13 am

Could this library work on a sensor DataLogger? Saving the data given from sensor in the flash memory...

[Reply](#)

**Sara Santos**

August 3, 2021 at 4:56 pm

Hi.

Yes, it would work.

But I recommend using a microSD card instead

<https://randomnerdtutorials.com/esp32-microsd-card-arduino/>

Regards,  
Sara

[Reply](#)

**Troy Dixon**

August 16, 2021 at 5:46 pm

Hi, Im looking for some guidance on getting the preferences library to store the mqtt server for the arduino PubSubClient. I can store wifi ssid, password, mqtt username and password, but I cant figure out how to get the server name stored. Tried as a string value and char value in the preferences library. Anyone out there have a suggestion? i an getting the name of the server (likely the IP address) from a text box on a nextion display then hoping to use the settings to connect to mqtt server. Ive tried converting the string value to a char, etc.. c\_str() just cant seem to figure out how to get it to work!

in my setup i try to get the server name like so:

(THIS IS JUST THE LASTEST TEST – THIS CODE HAS BEEN WORKED OVER MANY TIMES TRYING TO USE CHAR RATHER THAN STRING, ETC..)

```
String mqttserver="192.168.1.45";  
String mqttuser= "admin";  
String mqttpass = "123456";
```

```
PubSubClient client(mqttserver, 1883, callback, wifiClient); //THIS FAILS  
//PubSubClient client("192.168.0.45", 1883, callback, wifiClient); //THIS  
WORKS FINE
```

```
String Wifissid="Mango-2.4";
```

```

setup() {
  preferences.begin("myapp", false);

  preferences.putString("mqttserver", mqttserver);
  preferences.putString("mqttuser", mqttuser);
  preferences.putString("mqttpass", mqttpass);
  preferences.putString("wifissid", Wifissid);
  preferences.putString("wifipass", Wifipass);

  WiFi.disconnect();
  String p_ssid = preferences.getString("wifissid", "");
  String p_pass = preferences.getString("wifipass", "");

  WiFi.begin(p_ssid.c_str(), p_pass.c_str());

  Serial.print("Connecting to WiFi ..");

  uint32_t moment=millis();
  while ((WiFi.status() != WL_CONNECTED) && (millis()-moment < 8000)) { //
    make sure to do a timeout in
    // case of incorrect details to prevent eternal loop
    Serial.print(".");
    delay(100); // or yield(); (delay() includes a yield();)
  }

  if (WiFi.status() == WL_CONNECTED) {
    Serial.println('Connected');
    Serial.println(WiFi.localIP());
    Serial.println(mqttserver);
    Serial.println(mqttuser.c_str());
    Serial.println(mqttpass.c_str());
    //PubSubClient client(*mqttserver, 1883, callback, wifiClient);
  }

  //TRYING TO OVERRIDE THE INITIAL DEFINITION HERE
  //const char *mqserver = preferences.getString("mqttserver", "").c_str();
  // client.setServer("192.168.0.45",1883);

```

```
}
```

[Reply](#)

**Sara Santos**

August 18, 2021 at 2:10 pm

Hi.

Can you better describe what is exactly the error that you get?

Regards,

Sara

[Reply](#)

**Jesio**

September 9, 2021 at 10:22 pm

ola, estou tentando atualizar uma chave no preferences pelo webserver. Porem nao consigo fazer o putString dentro do server.on, quando faço a chamada via GET.

```
//- ATIVA/DESATIVA (MR1)
server.on("/ativaMR1", HTTP_GET, [](AsyncWebServerRequest *request){
request->send(200, "text/html");
jso.putString("teste", jesio);
Serial.println();
Serial.println("SOLICITAÇÃO OK: ATIVAR (MR1)");
Serial.println();
});
```

Podem me ajudar?

att

[Reply](#)

**macca-nz**

September 9, 2021 at 11:47 pm

Jesio,

You need to read up on the difference between HTTP\_GET and HTTP\_POST

From your HTML you send data to to be processed by the ESP with a HTTP\_POST command and if you want to bring data into your HTML you use the HTTP\_GET.

Might be worth having a look at an example like this

<https://randomnerdtutorials.com/esp32-http-get-post-arduino/#http-post>

Cheers

[Reply](#)

**Sara Santos**

September 11, 2021 at 9:02 pm

Hi.

You have to run that command outside of the server.on function.

Instead, you should have a flag variable that will change its value when it receives that request.

Then, in the loop(), check that the value of that variable, and if that's the case, call the putString function.

I hope this helps



Regards,  
Sara

[Reply](#)

**Andreas**

October 29, 2021 at 10:02 am

Hello,

I have a problem with preferences.

I am making an esp ap, with an web interface to change some settings.

One of the settings is to change the wifi channel.

To change the channel I have a dropdown on the interface which changes

the index of the url (like: 192.168.4.1/Channel/1)(which works). Then I am

getting the index and comparing them in a function and if it's a special

index the preferences get changed with a value

(like: if (header.indexOf("GET /Channel/1") >= 0) {

preferences.putInt("channel", 1); }) (this works either). But if the value is 10

and over it just takes the 1 and saves it.

The "channel" preferences is an integer.

I hope this should be enough and it doesn't sound too confusing.

best regards

Andreas

[Reply](#)

**Sara Santos**

October 29, 2021 at 1:15 pm

Hi.

The issue that you're facing is normal because the `indexOf` locates a string within another string. So, it will find the "1" if you have "10" "11" "1111" and so on.

So, the best way to do this is to get the number after the last "/".

First, locate the last "/" position—you can use the `lastIndexOf()` function:

<https://www.arduino.cc/reference/en/language/variables/data-types/string/functions/lastindexof/>

```
int lastSlash = header.lastIndexOf("/");
```

Then, calculate the string size:

```
int stringLen = header.length()
```

And then cut the string to get the number:

```
channel = header.substring(lastSlash+1, stringLen);
```

I hope this helps.

Regards,

Sara

[Reply](#)

**Andreas**

November 2, 2021 at 9:07 am

Heii,

I've tried what you said but I don't really understand how this works or should work.

I used `indexOf` before even with numbers higher than 9 and it worked.

I don't want to get the number after the last slash, I just want to check if the header index matches what I have in the if.

Example:

```
if (header.indexOf("GET /ChannelSet/1") >= 0)
{
  preferences.putInt("channel", 1);
}
```

```
preferences.putInt("channel", 2);  
}
```

...

```
else if (header.indexOf("GET /ChannelSet/13") >= 0)  
{  
preferences.putInt("channel", 13);  
}
```

(I know I could make this somehow shorter but it just has to work, memory and speed is not important, and I'm kinda lazy)

I used the same for signal strength with numbers like 19, 17 and it works the only difference is the type. I save it as a string (preferences.putString("outputStateStr", "18")) instead of an integer.

Maybe I don't understand how the indexOf works, but I don't know what I'm doing wrong or right.

Sry if this is annoying.

Regards,  
Andreas

[Reply](#)

**Eric**

November 24, 2021 at 1:58 am

Thank you for this great article coming from PIC with eeprom, I have one question regarding the "preferences.end" statement. For me it is not clear when to use that statement. Even with the help of the posting Xylopyrographer did on March 4, I am not sure when to use it.

Thank you for your time,

Eric

## Xylopyrographer

November 26, 2021 at 12:02 am

Hi Eric. Best answer is, every time your app finishes writing or reading data using the Preferences library, it should use a “preferences.end” to safely close the NVS and to release the preferences object. This ensures the NVS is in a good state and it also releases system memory back to the pool. So sequence would be preferences.begin → put or get your data → preferences.end

[Reply](#)

## Giambattista

January 31, 2022 at 6:50 am

Hi, I would ti save data received over BLE ti memory e then send this data over WiFi, how can I do It?

[Reply](#)

## Stefan

January 31, 2022 at 7:54 am

I want to comment on a – in my opinion very bad habit regarding naming objects –

I think it's a really bad habit to name an object with a similar name than the library-filename.

Sure the library's filename starts with a capital letter "P" and the object's name starts with lowercase "p" in my opinion this difference is too small to demonstrate which names are fixed and which names are user-definable  
In my opinion the name of the object should reflect its nature  
I would prefer  
Preferences myPrefObj;

Where "my" indicates it IS user-definable  
"Pref" gives the hint what it is  
and "Obj" that it is an "object" that has functions which can be accessed by writing "objectName.functionName"  
"objectName.functionName"  
the objects name dot the functions name

My naming-convention differs from the usual but  
every advanced user and of course every expert will understand my naming instantly. The difference is that BEGINNERS will understand MORE.

If advanced users and experts understand it immediately and BEGINNERS understand more what arguments are left against such a naming-convention?  
NONE! It is just a bad habit nobody things about.

[Reply](#)

**Sara Santos**

January 31, 2022 at 10:42 pm

Hi.

Yes. You are right.

I used the convention on the Preferences library example.

Regards,

Sara

**Furkan**

February 12, 2022 at 3:28 pm

Hello, I will transfer files from a client to ESP32. After the file is transferred, I want to print it to the sd card, how can I do it?

[Reply](#)

**Brom**

February 27, 2022 at 11:48 pm

First get your data over to the ESP32.

Then in Arduino IDE, checkout Examples->SD(esp32)->SDTest.

Use

```
n = SDFile.read(byte *buffer, BUFSIZE);
```

to read,

```
n = SDFile.write(byte *buffer, BUFSIZE);
```

to write, and

```
SDFile.seek(a)
```

to set the read/write address in the file.

[Reply](#)

**Frank Mentiplay**

March 23, 2022 at 11:00 pm

Hello

Great article. I am using a NodeMCU-32S board and the preference lab doesn't work for me.

The counter doesn't increase when the board resets. If the problem is with my board could you recommend an ESP32 dev board that you would guarantee me that I could read/write to the ESP32 EEPROM on that board.

Thanks

[Reply](#)

**Sara Santos**

March 25, 2022 at 6:34 pm

Hi.

Do you get any errors?

Do you have another ESP32 board to experiment with?

Regards.

Sara

[Reply](#)

**RJ**

April 13, 2022 at 8:56 pm

If running battery power only, does using preferences have a significant impact on power consumption? Ideally I would like to save some key parameters in preferences for if/when the battery dies, but not at the expense of consuming more power in the process so that the battery

[Reply](#)

**Gerstl Reinhold**

April 24, 2022 at 8:38 am

Hello

Great article.

I had read on web “However, the good news is that the EEPROM.write() on the ESP32 has the same properties of update. It only writes to EEPROM if we want to write something different.

Now using the preferences insted of EEPROM library do the preferences.putUInt the same thing? For example, if the new value is equal to the value stored in ESP32 flash memory, then the preferences.putint counts as a write cycle or not.

Thanks and sorry for my bad english

[Reply](#)

**Gerstl Reinhold**

April 24, 2022 at 9:28 am

Hi Sara

Great article

I read in one of yuor article “Contrary to the Arduino, the ESP32 doesn’t have an EEPROM.update() function.

however, the good news is that the EEPROM.write() on the ESP32 has the same properties of update. It only writes to EEPROM if we want to write something different.”



stored value. So no write cycle ?

Thank

Reinhold

[Reply](#)

**Dhaval Shukla**

May 5, 2022 at 12:48 pm

Hey, how can I iterate the namespaces? I would like to utilise code in a way that creates separate namespaces like np1, np2, np3 and so on maybe using a loop?

I would like to create separate namespaces in order to store and manage these namespaces later on....

Here's the code I have tried. Feel free to let me know if there is any other way to do it:

```
#include<Preferences.h>
```

```
Preferences ok;
```

```
int i = 0;
```

```
void setup() {
```

```
Serial.begin(115200);
```

```
}
```

```
void loop() {
```

```
while (i < 289) {
```

```
char testarr[] = "test 1";
```

```
Serial.println(testarr);
```

```
ok.begin(testarr, false);
testarr[5] = int(i);
Serial.println(testarr);
ok.putString("rtc", "09/09/2022 09:09:09");
ok.end();

}
while (i < 289) {
char testarr[] = "test 1";
Serial.println(testarr);

ok.begin(testarr, false);
testarr[5] = int(i);
Serial.println(testarr);
String eh = ok.getString("rtc");
Serial.println(eh);
ok.end();

}
}
```

Feel free to correct me as well.

[Reply](#)

**Eric**

June 12, 2022 at 1:43 am

It appears that Preferences does not work on the ESP32-C3 development boards (I'm using an Adafruit ESP32-C3 QT-Py). the StartCounter example does not work – the countr does not increment.

ESP-ROM:esp32c3-api1-20210207

Current counter value: 1  
Restarting in 10 seconds...  
ESP-ROM:esp32c3-api1-20210207  
Build:Feb 7 2021  
rst:0x3 (RTC\_SW\_SYS\_RST),boot:0xd (SPI\_FAST\_FLASH\_BOOT)  
Saved PC:0x403816f2  
SPIWP:0xee  
mode:DOUT, clock div:1  
load:0x3fcd6100,len:0x38c  
load:0x403ce000,len:0x6ac  
load:0x403d0000,len:0x2464  
SHA-256 comparison failed:  
Calculated:  
ae4b6389c51bca37025d68f98f30f6e4b83b375cef091401f27fa9596bdb9d  
df  
Expected: ff  
Attempting to boot anyway...  
entry 0x403ce000

Current counter value: 1  
Restarting in 10 seconds...  
ESP-ROM:esp32c3-api1-20210207  
Build:Feb 7 2021  
rst:0x3 (RTC\_SW\_SYS\_RST),boot:0xd (SPI\_FAST\_FLASH\_BOOT)  
Saved PC:0x403816f2  
SPIWP:0xee  
mode:DOUT, clock div:1  
load:0x3fcd6100,len:0x38c  
load:0x403ce000,len:0x6ac  
load:0x403d0000,len:0x2464  
SHA-256 comparison failed:  
Calculated:  
ae4b6389c51bca37025d68f98f30f6e4b83b375cef091401f27fa9596bdb9d  
df  
Expected: ff  
Attempting to boot anyway...  
entry 0x403ce000

Current counter value: 1  
Restarting in 10 seconds...

Does any know what the issue is with boards based on these chips? Id there an alternative storage method/library I can use in the meantime?

[Reply](#)

**Steve**

June 28, 2022 at 2:04 pm

I made a comment yesterday – not yet displayed, thank goodness, about string lengths. Please ignore it. My 'long' strings of 21 characters were being handled fine except in the printf statements I'd used for debugging. Reading that keys were limited in length, I jumped to the wrong conclusion.

[Reply](#)

**Sara Santos**

June 30, 2022 at 2:25 pm

Ok.  
It's deleted.  
Regards,  
Sara

[Reply](#)

(Writing through a translator) Hi Sarah, I found a way to store an array of any type in memory, the way is strange but I hope you understand.

A little explanation if we do this

```
Serial.println("qwerty"+1);
```

Then we'll see werty the first letter will disappear and I used it like this, please write what you think.

```
#include <Preferences.h>
```

```
Preferences preferences;
```

```
int MyArray[3];
```

```
void setup() {
```

```
  Serial.begin(9600);
```

```
  preferences.begin("MyArray", false);
```

```
  for (int i = 0; i <= 2; i++) {
```

```
    MyArray[i] = preferences.getInt("123MyArray[]" + i, 0);
```

```
  }
```

```
  MyArray[0]++;
```

```
  MyArray[1]++;
```

```
  MyArray[2]++;
```

```
  Serial.print("MyArray[0] "); Serial.println(MyArray[0]);
```

```
  Serial.print("MyArray[1] "); Serial.println(MyArray[1]);
```

```
  Serial.print("MyArray[2] "); Serial.println(MyArray[2]);
```

```
  for (int i = 0; i <= 2; i++) {
```

```
    preferences.putInt("123MyArray[]" + i, MyArray[i]);
```

```
  }
```

```
  preferences.end();
```

```
  Serial.println("Restarting in 10 seconds...");
```

```
  delay(10000);
```

```
  ESP.restart();
```

```
void loop() {
```

```
}
```

[Reply](#)

**Sara Santos**

October 7, 2022 at 1:07 pm

Thanks for sharing this snippet.

Regards,

Sara

[Reply](#)

## Leave a Comment

Name \*

Email \*

Website

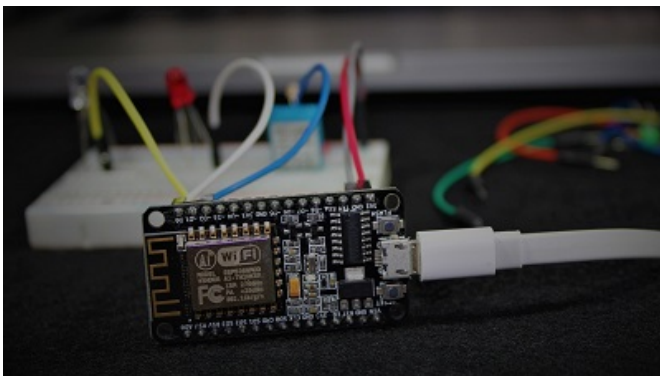
---

☐ Notify me of new posts by email.

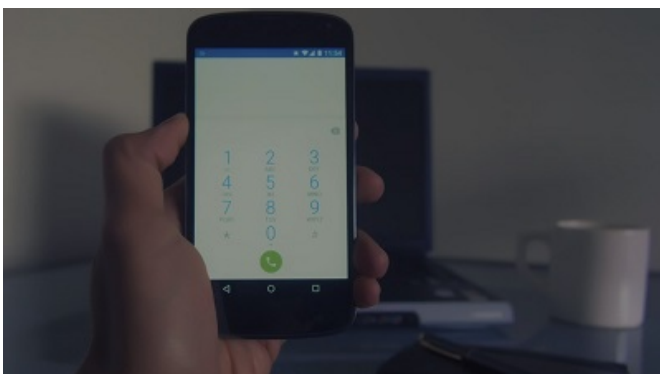
Post Comment



[Visit Maker Advisor – Tools and Gear for makers, hobbyists and DIYers »](#)



[Home Automation using ESP8266 eBook and video course »](#) Build IoT and home automation projects.



[Build Web Servers with ESP32 and](#)

---

[About](#)   [Support](#)   [Terms and Conditions](#)   [Privacy Policy](#)   [Refunds](#)   [Complaints' Book](#)

[MakerAdvisor.com](#)   [Join the Lab](#)

Copyright © 2013-2023 · [RandomNerdTutorials.com](#) · All Rights Reserved