

Deep Sleep ESP32

STARTED

04/04/2025

BY

Michael Pope and Danch

TAGS

hobby

PRINTED

13 Oct 2026

Deep Sleep ESP32 Experiments

Date: 4th April, 2025

Monitor results (MICK)

SHELL

```
$ ssh ava
$ screen
$ cd code/deepsleep/sinatra_server
$ ruby main.rb
```

Check <http://10.1.1.11:4567>

Research

Adruino/ESP32

<https://www.arrow.com/en/research-and-events/articles/esp32-power-consumption-can-be-reduced-with-sleep-modes>

<https://docs.arduino.cc/libraries/arduino-low-power/>

<https://lastminuteengineers.com/esp32-sleep-modes-power-consumption/>

<https://www.instructables.com/ESP32-Deep-Sleep-Tutorial/>

https://docs.espressif.com/projects/esp-idf/en/stable/esp32/api-reference/system/sleep_modes.html

<https://randomnerdtutorials.com/esp32-deep-sleep-arduino-ide-wake-up-sources/>

PICO

<https://www.youtube.com/watch?v=05o5QOZkpwg>

Hardware

Make	Model	Store	\$
ESP32	WROOM-32	Zaitronics	9
Raspberry	PICO	core	8
Raspberry	PICO2	core	15
Buck (LM2596S)		Zaitronics	4
18650 holder DUAL		Zaitronics	3
18650 x 2		core	12
18650 protected x 2		ledtorch	32
Charger		jaycar	34

<https://core-electronics.com.au/sunflower-solar-power-manager-5v.html>

<https://core-electronics.com.au/monocrystalline-solar-panel-5v-1a.html>

(END GAME) : <https://core-electronics.com.au/gravity-lightning-sensor.html>

Power

ESP32

<https://techexplorations.com/guides/esp32/begin/power/>

Test

We want to test the battery life for modem, light, deep sleep and hibernation when waking up every 5 minutes to send one POST request to an external sinatra server.

Setup

Web server that logs time

This will be hosted on the LAN on a computer which is running 24/7.

Setup a sinatra app with sqlite3

One Post route

Log experiment number, and time.

Flask

PYTHON

```
from flask import Flask, request
import os.path
import datetime

app = Flask(__name__)

@app.route('/')
def hello():
    if os.path.isfile("Reading.log"):
        file = open("Reading.log", "r")
        return file
    else:
        return 'No log exists'

@app.route('/query')
def query():

    board = request.args['board']
    test = request.args['test']
    current_time = datetime.datetime.now()

    output = (f"{board}, {test}, {current_time}\n")

    f = open("Reading.log", "a")
    f.write(output)
```

```
f.close()

return output
```

Install

SHELL

```
$ mkdir project
$ cd project
$ python3 -m venv venv
$ source venv/bin/activate
```

Run the server (Old method)

SHELL

```
$ export FLASK_APP=main (name of file running server ie main.py)
$ export FLASK_ENV=development ( old method - use debug instead )
$ flask run -host=0.0.0.0
```

(new method)

SHELL

```
$ flask --app main run --debug --host=0.0.0.0
```

Sinatra

RUBY

```
require 'sinatra/base'

class Main < Sinatra::Base

  set bind: "0.0.0.0"
```

```

get '/' do
  if File.exist?("readings.log")
    File.readlines("readings.log")
  else
    "No readings, log file doesn't exist."
  end
end

get '/log' do
  # Collect params, board=esp32 and test=lightsleep
  #
  # curl "http://127.0.0.1/log?board=esp32&test=lightsleep"
  # curl "http://10.1.1.11:4567/log?board=esp32&test=lightsleep"

  board=params[:board]
  test=params[:test]

  # Define the variable
  output="#{board},#{test},#{Time.now}\n"

  # Write to a file
  File.write("readings.log", output, mode: "a")

  output
end
end

Main.run!

```

Install Ruby and dependencies

SHELL

```

$ curl https://mise.run | sh
$ sudo apt install libyaml-dev libffi-dev
$ mise install ruby@latest
$ bundle

```

Run the server

SHELL

```

$ ssh <pi/server>
$ screen
$ cd code/deepsleep/sinatra_server

```

```
$ ruby main.rb
```

To disconnect Ctrl+a,d

NOTE: To reconnect to screen, type 'screen -R'

Test

SHELL

```
$ curl "http://10.1.1.11:4567/log?board=esp32&test=lightsleep"
```

Mick's server http://10.1.1.11:4567

Pretty up display of data

We want it to look like the following;

Test	Start	End	Duration (m/d/h)
nosleep	dd/mm/yyyy hh:mm	dd/mm/yyyy hh:mm	Mth / days / hours eg: 38hrs
Light sleep			4days, 3hrs
Deep sleep			

RUBY

```
require 'sinatra/base'
require 'debug'

class Main < Sinatra::Base

  set bind: "0.0.0.0"
```

```

def create_output
  output=File.readlines("readings.log")

  # Get the first line from file
  start_test = output.first
  current_test = output.first
  previous_test = output.first
  output_a = []

  output.each_with_index do |line,index|
    current_test = line

    # Check for a change in test or end of file
    if current_test.split(",")[1] != previous_test.split(",")[1] or index ==

      # Record the test into the hash
      output_a << {test: start_test.split(",")[1],
                    start: start_test.split(",")[2] ,
                    end: previous_test.split(",")[2]}

      # Set the new starting line
      start_test = current_test.clone
    end

    # Keep going
    previous_test = current_test.clone
  end

  output_a
end

get '/' do
  if File.exist?("readings.log")
    @output=create_output
    erb :output
  else
    "No readings, log file doesn't exist."
  end
end

get '/log' do
  # Collect params, board=esp32 and test=lightsleep
  #
  # curl "http://127.0.0.1/log?board=esp32&test=lightsleep"
  # curl "http://10.1.1.11:4567/log?board=esp32&test=lightsleep"

  board=params[:board]
  test=params[:test]

  # Define the variable

```

```

output="#{board},#{test},#{Time.now}\n"

# Write to a file
File.write("readings.log", output, mode: "a")

output

end
end

Main.run!

```

Also create a views/output.erb with;

HTML

```

<h1>Results</h1>

<table>
  <th>Test</th>
  <th>Start</th>
  <th>End</th>
  <th>Duration</th>

  <% @output.each do |out| %>
    <tr>
      <td><%= out[:test] %></td>
      <td><%= out[:start] %></td>
      <td><%= out[:end] %></td>
      <td><%= out[:duration] %></td>
    </tr>
  <% end %>
</table>

```

Calculate duration

RUBY

```

# Give us the duration in hours
duration = (Time.new(end_date) - Time.new(start_date)) / 3600

if duration <= 24

```

```

        duration = "#{duration.to_i}hrs"
    elsif duration > 24
        duration = "#{(duration / 24).to_i}d / #{(duration % 24).to_i}hrs"
    end

    # Record the test into the hash
    output_a << {test: start_test.split(",")[1],
                 start: start_date,
                 end: end_date,
                 duration: duration}

```

Setup ESP32

Load code for each test which should do the following

Wake every 5 minutes

Connect to WIFI 2.4Ghz

Send a POST request to our webserver

Include text “experiment #1”

Goes to sleep

Connect 18650 to run ESP32 via Sunflower Solar Power

Monitor how long it lasts

From the webserver we could get stats through the SQL looking at the posts we could calculate how long it’s lasted.

Test: Blink example

C

```

int LED_BUILTIN=2;

// 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);
}

// the loop function runs over and over again forever
void loop() {
    digitalWrite(LED_BUILTIN, HIGH);    // turn the LED on (HIGH is the voltage level)
}

```

```
delay(1000);           // wait for a second
digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making the voltage
delay(1000);           // wait for a second
}
```

Uploading

Load ArduinoIDE

Select ttyUSB

Select Board “ESP32 Dev Module”

NOTE: On Micks I had to hold the BOOT button when uploading

Connection code DRAFT #1: Danch using weather station

This same code will be ran on the ESP32, pico W, pico 2 W, STM

Connecting to Battery

<https://techexplorations.com/guides/esp32/begin/power/>

Two 18650's 2600mAh -> Buck down to 5V -> VIN + GND on the ESP32 WROOM.

Overcharge/Over Discharge protection

[https://www.ledtorchshop.com.au/product/keeppower-18650-protected-rechargeable-2600mah-li-ion-battery-p1826c/?](https://www.ledtorchshop.com.au/product/keeppower-18650-protected-rechargeable-2600mah-li-ion-battery-p1826c/?srsltid=AfmBOooFUK6xXEqkotgG4NWKmxd_VVN2khMAjBUmNW68dRO0EKmQYZQ7)

[srsltid=AfmBOooFUK6xXEqkotgG4NWKmxd_VVN2khMAjBUmNW68dRO0EKmQYZQ7](https://www.ledtorchshop.com.au/product/keeppower-18650-protected-rechargeable-2600mah-li-ion-battery-p1826c/?srsltid=AfmBOooFUK6xXEqkotgG4NWKmxd_VVN2khMAjBUmNW68dRO0EKmQYZQ7)

Test Amps

	Amps (mAh)	Power
Normal	127	5V direct
Wifi usage	158	5V direct
Deep Sleep	11	5V direct
Light Sleep	13	5V direct
No sleep idle	55	

No sleep POST http115

NOTE: In the no sleep it doesn't have to do the setup everytime hence the lower POST reading.

Experiment #1: No Sleep

NOTE: Adjust the wake in this first experiment so that it lasts the least amount of time. So we might want to do every one minute, but not less than a minute.

C

```
#include <WiFi.h>
#include <HTTPClient.h>
#include <string>

using namespace std;

void wifi_connect() {
  WiFi.begin("[SSID]", "[PASSWORD]");

  Serial.println("Connecting");
  while(WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("Connected");
}

void setup() {
  Serial.begin(115200);
  while (!Serial);          /* ? */
  Serial.println("Sleep experiment");
  wifi_connect();
}

void loop() {
  //Check WiFi connection status
  if(WiFi.status()== WL_CONNECTED){
    HTTPClient http;

    // Your Domain name with URL path or IP address with path
    http.begin("http://[SERVERIP]:4567/log?board=esp32&test=nosleep");

    // Send HTTP GET request
```

```

    int httpResponseCode = http.GET();

    if (httpResponseCode==200) {
    Serial.print("Success");
    } else {
    Serial.print("Error code: ");
    Serial.println(httpResponseCode);
    }

    // Free resources
    http.end();

    /* esp_sleep_enable_timer_wakeup(300 * 1000000); //light sleep for 5 mi
    /* esp_light_sleep_start(); */
    }
    else {
        Serial.println("WiFi Disconnected. Attempting to re-connect");
        wifi_connect();
    }

    delay(300000)                /* 5 minutes */
}

```

Experiment #2: Modem Sleep?

Experiment #3: Light Sleep

```

#include <HTTPClient.h>
#include <string>
#include "esp_wifi.h"

using namespace std;

#define uS_TO_S_FACTOR 1000000 /* Conversion factor for micro seconds to seconds */
#define TIME_TO_SLEEP  300      /* Time ESP32 will go to sleep (in seconds) */

void wifi_connect() {

    WiFi.begin("[SSID]", "[PASSWORD]");

    Serial.println("Connecting");
    while(WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("Connected");
}

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

    while (!Serial); /* ? */
    Serial.println("Sleep experiment");
    wifi_connect();
}

int counter = 0;

void loop() {
    //Check WiFi connection status
    if(WiFi.status()== WL_CONNECTED){
        HTTPClient http;

        // Your Domain name with URL path or IP address with path
        http.begin("http://10.1.1.11:4567/log?board=esp32&test=lightsleep");

        // Send HTTP GET request
        int httpResponseCode = http.GET();

        if (httpResponseCode==200) {
            Serial.print("Success");
        } else {
            Serial.print("Error code: ");
            Serial.println(httpResponseCode);
        }
    }
}

```

```

    }

    // Free resources
    http.end();

    // Need to stop wifi before sleeping, then start again.
    esp_wifi_stop();
    esp_sleep_enable_timer_wakeup(TIME_TO_SLEEP * uS_TO_S_FACTOR);
    esp_light_sleep_start();
    esp_wifi_start();
}
else {
    Serial.println("WiFi Disconnected. Attempting to re-connect");
    wifi_connect();
}
//delay(300000);          /* 5 minutes */
}

```

```

#include <HTTPClient.h>
#include <string>
#include "esp_wifi.h"

using namespace std;

#define uS_TO_S_FACTOR 1000000 /* Conversion factor for micro seconds to seconds */
#define TIME_TO_SLEEP  300      /* Time ESP32 will go to sleep (in seconds) */

void wifi_connect() {

    WiFi.begin("[SSID]", "[PASSWORD]");

    Serial.println("Connecting");
    while(WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("Connected");
}

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

    while (!Serial); /* ? */
    Serial.println("Sleep experiment");
    wifi_connect();
}

int counter = 0;

void loop() {
    //Check WiFi connection status
    if(WiFi.status()== WL_CONNECTED){
        HTTPClient http;

        // Your Domain name with URL path or IP address with path
        http.begin("http://10.1.1.11:4567/log?board=esp32&test=lightsleep");

        // Send HTTP GET request
        int httpStatusCode = http.GET();

        if (httpStatusCode==200) {
            Serial.print("Success");
        } else {
            Serial.print("Error code: ");
            Serial.println(httpStatusCode);

```

```

    }

    // Free resources
    http.end();

    // Need to stop wifi before sleeping, then start again.
    esp_wifi_stop();
    esp_sleep_enable_timer_wakeup(TIME_TO_SLEEP * uS_TO_S_FACTOR);
    esp_light_sleep_start();
    esp_wifi_start();
}
else {
    Serial.println("WiFi Disconnected. Attempting to re-connect");
    wifi_connect();
}
//delay(300000);                /* 5 minutes */
}

```

```

#include <HTTPClient.h>
#include <string>
#include "esp_wifi.h"

using namespace std;

#define uS_TO_S_FACTOR 1000000 /* Conversion factor for micro seconds to seconds */
#define TIME_TO_SLEEP  300      /* Time ESP32 will go to sleep (in seconds) */

void wifi_connect() {

    WiFi.begin("[SSID]", "[PASSWORD]");

    Serial.println("Connecting");
    while(WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("Connected");
}

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

    while (!Serial); /* ? */
    Serial.println("Sleep experiment");
    wifi_connect();
}

int counter = 0;

void loop() {
    //Check WiFi connection status
    if(WiFi.status()== WL_CONNECTED){
        HTTPClient http;

        // Your Domain name with URL path or IP address with path
        http.begin("http://10.1.1.11:4567/log?board=esp32&test=lightsleep");

        // Send HTTP GET request
        int httpStatusCode = http.GET();

        if (httpStatusCode==200) {
            Serial.print("Success");
        } else {
            Serial.print("Error code: ");
            Serial.println(httpStatusCode);
        }
    }
}

```

```

    }

    // Free resources
    http.end();

    // Need to stop wifi before sleeping, then start again.
    esp_wifi_stop();
    esp_sleep_enable_timer_wakeup(TIME_TO_SLEEP * uS_TO_S_FACTOR);
    esp_light_sleep_start();
    esp_wifi_start();
}
else {
    Serial.println("WiFi Disconnected. Attempting to re-connect");
    wifi_connect();
}
//delay(300000);                /* 5 minutes */
}

```

```

#include <HTTPClient.h>
#include <string>
#include "esp_wifi.h"

using namespace std;

#define uS_TO_S_FACTOR 1000000 /* Conversion factor for micro seconds to seconds */
#define TIME_TO_SLEEP  300      /* Time ESP32 will go to sleep (in seconds) */

void wifi_connect() {

    WiFi.begin("[SSID]", "[PASSWORD]");

    Serial.println("Connecting");
    while(WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("Connected");
}

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

    while (!Serial); /* ? */
    Serial.println("Sleep experiment");
    wifi_connect();
}

int counter = 0;

void loop() {
    //Check WiFi connection status
    if(WiFi.status()== WL_CONNECTED){
        HTTPClient http;

        // Your Domain name with URL path or IP address with path
        http.begin("http://10.1.1.11:4567/log?board=esp32&test=lightsleep");

        // Send HTTP GET request
        int httpStatusCode = http.GET();

        if (httpStatusCode==200) {
            Serial.print("Success");
        } else {
            Serial.print("Error code: ");
            Serial.println(httpStatusCode);
        }
    }
}

```

```

    }

    // Free resources
    http.end();

    // Need to stop wifi before sleeping, then start again.
    esp_wifi_stop();
    esp_sleep_enable_timer_wakeup(TIME_TO_SLEEP * uS_TO_S_FACTOR);
    esp_light_sleep_start();
    esp_wifi_start();
}
else {
    Serial.println("WiFi Disconnected. Attempting to re-connect");
    wifi_connect();
}
//delay(300000);          /* 5 minutes */
}

```

```

#include <HTTPClient.h>
#include <string>
#include "esp_wifi.h"

using namespace std;

#define uS_TO_S_FACTOR 1000000 /* Conversion factor for micro seconds to seconds */
#define TIME_TO_SLEEP  300      /* Time ESP32 will go to sleep (in seconds) */

void wifi_connect() {

    WiFi.begin("[SSID]", "[PASSWORD]");

    Serial.println("Connecting");
    while(WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("Connected");
}

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

    while (!Serial); /* ? */
    Serial.println("Sleep experiment");
    wifi_connect();
}

int counter = 0;

void loop() {
    //Check WiFi connection status
    if(WiFi.status()== WL_CONNECTED){
        HTTPClient http;

        // Your Domain name with URL path or IP address with path
        http.begin("http://10.1.1.11:4567/log?board=esp32&test=lightsleep");

        // Send HTTP GET request
        int httpStatusCode = http.GET();

        if (httpStatusCode==200) {
            Serial.print("Success");
        } else {
            Serial.print("Error code: ");
            Serial.println(httpStatusCode);
        }
    }
}

```

```

    }

    // Free resources
    http.end();

    // Need to stop wifi before sleeping, then start again.
    esp_wifi_stop();
    esp_sleep_enable_timer_wakeup(TIME_TO_SLEEP * uS_TO_S_FACTOR);
    esp_light_sleep_start();
    esp_wifi_start();
}
else {
    Serial.println("WiFi Disconnected. Attempting to re-connect");
    wifi_connect();
}
//delay(300000);                /* 5 minutes */
}

```

```

#include <HTTPClient.h>
#include <string>
#include "esp_wifi.h"

using namespace std;

#define uS_TO_S_FACTOR 1000000 /* Conversion factor for micro seconds to seconds */
#define TIME_TO_SLEEP  300      /* Time ESP32 will go to sleep (in seconds) */

void wifi_connect() {

    WiFi.begin("[SSID]", "[PASSWORD]");

    Serial.println("Connecting");
    while(WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("Connected");
}

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

    while (!Serial); /* ? */
    Serial.println("Sleep experiment");
    wifi_connect();
}

int counter = 0;

void loop() {
    //Check WiFi connection status
    if(WiFi.status()== WL_CONNECTED){
        HTTPClient http;

        // Your Domain name with URL path or IP address with path
        http.begin("http://10.1.1.11:4567/log?board=esp32&test=lightsleep");

        // Send HTTP GET request
        int httpResponseCode = http.GET();

        if (httpResponseCode==200) {
            Serial.print("Success");
        } else {
            Serial.print("Error code: ");
            Serial.println(httpResponseCode);
        }
    }
}

```

```

    }

    // Free resources
    http.end();

    // Need to stop wifi before sleeping, then start again.
    esp_wifi_stop();
    esp_sleep_enable_timer_wakeup(TIME_TO_SLEEP * uS_TO_S_FACTOR);
    esp_light_sleep_start();
    esp_wifi_start();
}
else {
    Serial.println("WiFi Disconnected. Attempting to re-connect");
    wifi_connect();
}
//delay(300000);                /* 5 minutes */
}

```

```

#include <HTTPClient.h>
#include <string>
#include "esp_wifi.h"

using namespace std;

#define uS_TO_S_FACTOR 1000000 /* Conversion factor for micro seconds to seconds */
#define TIME_TO_SLEEP  300      /* Time ESP32 will go to sleep (in seconds) */

void wifi_connect() {

    WiFi.begin("[SSID]", "[PASSWORD]");

    Serial.println("Connecting");
    while(WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("Connected");
}

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

    while (!Serial); /* ? */
    Serial.println("Sleep experiment");
    wifi_connect();
}

int counter = 0;

void loop() {
    //Check WiFi connection status
    if(WiFi.status()== WL_CONNECTED){
        HTTPClient http;

        // Your Domain name with URL path or IP address with path
        http.begin("http://10.1.1.11:4567/log?board=esp32&test=lightsleep");

        // Send HTTP GET request
        int httpStatusCode = http.GET();

        if (httpStatusCode==200) {
            Serial.print("Success");
        } else {
            Serial.print("Error code: ");
            Serial.println(httpStatusCode);

```

```

    }

    // Free resources
    http.end();

    // Need to stop wifi before sleeping, then start again.
    esp_wifi_stop();
    esp_sleep_enable_timer_wakeup(TIME_TO_SLEEP * uS_TO_S_FACTOR);
    esp_light_sleep_start();
    esp_wifi_start();
}
else {
    Serial.println("WiFi Disconnected. Attempting to re-connect");
    wifi_connect();
}
//delay(300000);          /* 5 minutes */
}

```

```

#include <HTTPClient.h>
#include <string>
#include "esp_wifi.h"

using namespace std;

#define uS_TO_S_FACTOR 1000000 /* Conversion factor for micro seconds to seconds */
#define TIME_TO_SLEEP  300      /* Time ESP32 will go to sleep (in seconds) */

void wifi_connect() {

    WiFi.begin("[SSID]", "[PASSWORD]");

    Serial.println("Connecting");
    while(WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("Connected");
}

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

    while (!Serial); /* ? */
    Serial.println("Sleep experiment");
    wifi_connect();
}

int counter = 0;

void loop() {
    //Check WiFi connection status
    if(WiFi.status()== WL_CONNECTED){
        HTTPClient http;

        // Your Domain name with URL path or IP address with path
        http.begin("http://10.1.1.11:4567/log?board=esp32&test=lightsleep");

        // Send HTTP GET request
        int httpResponseCode = http.GET();

        if (httpResponseCode==200) {
            Serial.print("Success");
        } else {
            Serial.print("Error code: ");
            Serial.println(httpResponseCode);
        }
    }
}

```

```

    }

    // Free resources
    http.end();

    // Need to stop wifi before sleeping, then start again.
    esp_wifi_stop();
    esp_sleep_enable_timer_wakeup(TIME_TO_SLEEP * uS_TO_S_FACTOR);
    esp_light_sleep_start();
    esp_wifi_start();
}
else {
    Serial.println("WiFi Disconnected. Attempting to re-connect");
    wifi_connect();
}
//delay(300000);          /* 5 minutes */
}

```

```

#include <HTTPClient.h>
#include <string>
#include "esp_wifi.h"

using namespace std;

#define uS_TO_S_FACTOR 1000000 /* Conversion factor for micro seconds to seconds */
#define TIME_TO_SLEEP  300      /* Time ESP32 will go to sleep (in seconds) */

void wifi_connect() {

    WiFi.begin("[SSID]", "[PASSWORD]");

    Serial.println("Connecting");
    while(WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("Connected");
}

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

    while (!Serial); /* ? */
    Serial.println("Sleep experiment");
    wifi_connect();
}

int counter = 0;

void loop() {
    //Check WiFi connection status
    if(WiFi.status()== WL_CONNECTED){
        HTTPClient http;

        // Your Domain name with URL path or IP address with path
        http.begin("http://10.1.1.11:4567/log?board=esp32&test=lightsleep");

        // Send HTTP GET request
        int httpStatusCode = http.GET();

        if (httpStatusCode==200) {
            Serial.print("Success");
        } else {
            Serial.print("Error code: ");
            Serial.println(httpStatusCode);
        }
    }
}

```

```

    }

    // Free resources
    http.end();

    // Need to stop wifi before sleeping, then start again.
    esp_wifi_stop();
    esp_sleep_enable_timer_wakeup(TIME_TO_SLEEP * uS_TO_S_FACTOR);
    esp_light_sleep_start();
    esp_wifi_start();
}
else {
    Serial.println("WiFi Disconnected. Attempting to re-connect");
    wifi_connect();
}
//delay(300000);                /* 5 minutes */
}

```

```

#include <HTTPClient.h>
#include <string>
#include "esp_wifi.h"

using namespace std;

#define uS_TO_S_FACTOR 1000000 /* Conversion factor for micro seconds to seconds */
#define TIME_TO_SLEEP  300      /* Time ESP32 will go to sleep (in seconds) */

void wifi_connect() {

    WiFi.begin("[SSID]", "[PASSWORD]");

    Serial.println("Connecting");
    while(WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("Connected");
}

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

    while (!Serial); /* ? */
    Serial.println("Sleep experiment");
    wifi_connect();
}

int counter = 0;

void loop() {
    //Check WiFi connection status
    if(WiFi.status()== WL_CONNECTED){
        HTTPClient http;

        // Your Domain name with URL path or IP address with path
        http.begin("http://10.1.1.11:4567/log?board=esp32&test=lightsleep");

        // Send HTTP GET request
        int httpStatusCode = http.GET();

        if (httpStatusCode==200) {
            Serial.print("Success");
        } else {
            Serial.print("Error code: ");
            Serial.println(httpStatusCode);
        }
    }
}

```

```

    }

    // Free resources
    http.end();

    // Need to stop wifi before sleeping, then start again.
    esp_wifi_stop();
    esp_sleep_enable_timer_wakeup(TIME_TO_SLEEP * uS_TO_S_FACTOR);
    esp_light_sleep_start();
    esp_wifi_start();
}
else {
    Serial.println("WiFi Disconnected. Attempting to re-connect");
    wifi_connect();
}
//delay(300000);                /* 5 minutes */
}

```

```

#include <HTTPClient.h>
#include <string>
#include "esp_wifi.h"

using namespace std;

#define uS_TO_S_FACTOR 1000000 /* Conversion factor for micro seconds to seconds */
#define TIME_TO_SLEEP  300      /* Time ESP32 will go to sleep (in seconds) */

void wifi_connect() {

    WiFi.begin("[SSID]", "[PASSWORD]");

    Serial.println("Connecting");
    while(WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("Connected");
}

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

    while (!Serial); /* ? */
    Serial.println("Sleep experiment");
    wifi_connect();
}

int counter = 0;

void loop() {
    //Check WiFi connection status
    if(WiFi.status()== WL_CONNECTED){
        HTTPClient http;

        // Your Domain name with URL path or IP address with path
        http.begin("http://10.1.1.11:4567/log?board=esp32&test=lightsleep");

        // Send HTTP GET request
        int httpStatusCode = http.GET();

        if (httpStatusCode==200) {
            Serial.print("Success");
        } else {
            Serial.print("Error code: ");
            Serial.println(httpStatusCode);

```

```

    }

    // Free resources
    http.end();

    // Need to stop wifi before sleeping, then start again.
    esp_wifi_stop();
    esp_sleep_enable_timer_wakeup(TIME_TO_SLEEP * uS_TO_S_FACTOR);
    esp_light_sleep_start();
    esp_wifi_start();
}
else {
    Serial.println("WiFi Disconnected. Attempting to re-connect");
    wifi_connect();
}
//delay(300000);                /* 5 minutes */
}

```

```

#include <HTTPClient.h>
#include <string>
#include "esp_wifi.h"

using namespace std;

#define uS_TO_S_FACTOR 1000000 /* Conversion factor for micro seconds to seconds */
#define TIME_TO_SLEEP  300      /* Time ESP32 will go to sleep (in seconds) */

void wifi_connect() {

    WiFi.begin("[SSID]", "[PASSWORD]");

    Serial.println("Connecting");
    while(WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("Connected");
}

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

    while (!Serial); /* ? */
    Serial.println("Sleep experiment");
    wifi_connect();
}

int counter = 0;

void loop() {
    //Check WiFi connection status
    if(WiFi.status()== WL_CONNECTED){
        HTTPClient http;

        // Your Domain name with URL path or IP address with path
        http.begin("http://10.1.1.11:4567/log?board=esp32&test=lightsleep");

        // Send HTTP GET request
        int httpResponseCode = http.GET();

        if (httpResponseCode==200) {
            Serial.print("Success");
        } else {
            Serial.print("Error code: ");
            Serial.println(httpResponseCode);
        }
    }
}

```

```

    }

    // Free resources
    http.end();

    // Need to stop wifi before sleeping, then start again.
    esp_wifi_stop();
    esp_sleep_enable_timer_wakeup(TIME_TO_SLEEP * uS_TO_S_FACTOR);
    esp_light_sleep_start();
    esp_wifi_start();
}
else {
    Serial.println("WiFi Disconnected. Attempting to re-connect");
    wifi_connect();
}
//delay(300000);                /* 5 minutes */
}

```

```

#include <HTTPClient.h>
#include <string>
#include "esp_wifi.h"

using namespace std;

#define uS_TO_S_FACTOR 1000000 /* Conversion factor for micro seconds to seconds */
#define TIME_TO_SLEEP  300      /* Time ESP32 will go to sleep (in seconds) */

void wifi_connect() {

    WiFi.begin("[SSID]", "[PASSWORD]");

    Serial.println("Connecting");
    while(WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("Connected");
}

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

    while (!Serial);          /* ? */
    Serial.println("Sleep experiment");
    wifi_connect();
}

int counter = 0;

void loop() {
    //Check WiFi connection status
    if(WiFi.status()== WL_CONNECTED){
        HTTPClient http;

        // Your Domain name with URL path or IP address with path
        http.begin("http://10.1.1.11:4567/log?board=esp32&test=lightsleep");

        // Send HTTP GET request
        int httpStatusCode = http.GET();

        if (httpStatusCode==200) {
            Serial.print("Success");
        } else {
            Serial.print("Error code: ");
            Serial.println(httpStatusCode);
        }
    }
}

```

```

    }

    // Free resources
    http.end();

    // Need to stop wifi before sleeping, then start again.
    esp_wifi_stop();
    esp_sleep_enable_timer_wakeup(TIME_TO_SLEEP * uS_TO_S_FACTOR);
    esp_light_sleep_start();
    esp_wifi_start();
}
else {
    Serial.println("WiFi Disconnected. Attempting to re-connect");
    wifi_connect();
}
//delay(300000);                /* 5 minutes */
}

```

```

#include <HTTPClient.h>
#include <string>
#include "esp_wifi.h"

using namespace std;

#define uS_TO_S_FACTOR 1000000 /* Conversion factor for micro seconds to seconds */
#define TIME_TO_SLEEP  300      /* Time ESP32 will go to sleep (in seconds) */

void wifi_connect() {

    WiFi.begin("[SSID]", "[PASSWORD]");

    Serial.println("Connecting");
    while(WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("Connected");
}

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

    while (!Serial); /* ? */
    Serial.println("Sleep experiment");
    wifi_connect();
}

int counter = 0;

void loop() {
    //Check WiFi connection status
    if(WiFi.status()== WL_CONNECTED){
        HTTPClient http;

        // Your Domain name with URL path or IP address with path
        http.begin("http://10.1.1.11:4567/log?board=esp32&test=lightsleep");

        // Send HTTP GET request
        int httpStatusCode = http.GET();

        if (httpStatusCode==200) {
            Serial.print("Success");
        } else {
            Serial.print("Error code: ");
            Serial.println(httpStatusCode);
        }
    }
}

```

```

    }

    // Free resources
    http.end();

    // Need to stop wifi before sleeping, then start again.
    esp_wifi_stop();
    esp_sleep_enable_timer_wakeup(TIME_TO_SLEEP * uS_TO_S_FACTOR);
    esp_light_sleep_start();
    esp_wifi_start();
}
else {
    Serial.println("WiFi Disconnected. Attempting to re-connect");
    wifi_connect();
}
//delay(300000);          /* 5 minutes */
}

```

```

#include <HTTPClient.h>
#include <string>
#include "esp_wifi.h"

using namespace std;

#define uS_TO_S_FACTOR 1000000 /* Conversion factor for micro seconds to seconds */
#define TIME_TO_SLEEP  300      /* Time ESP32 will go to sleep (in seconds) */

void wifi_connect() {

    WiFi.begin("[SSID]", "[PASSWORD]");

    Serial.println("Connecting");
    while(WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("Connected");
}

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

    while (!Serial);          /* ? */
    Serial.println("Sleep experiment");
    wifi_connect();
}

int counter = 0;

void loop() {
    //Check WiFi connection status
    if(WiFi.status()== WL_CONNECTED){
        HTTPClient http;

        // Your Domain name with URL path or IP address with path
        http.begin("http://10.1.1.11:4567/log?board=esp32&test=lightsleep");

        // Send HTTP GET request
        int httpStatusCode = http.GET();

        if (httpStatusCode==200) {
            Serial.print("Success");
        } else {
            Serial.print("Error code: ");
            Serial.println(httpStatusCode);

```

```

    }

    // Free resources
    http.end();

    // Need to stop wifi before sleeping, then start again.
    esp_wifi_stop();
    esp_sleep_enable_timer_wakeup(TIME_TO_SLEEP * uS_TO_S_FACTOR);
    esp_light_sleep_start();
    esp_wifi_start();
}
else {
    Serial.println("WiFi Disconnected. Attempting to re-connect");
    wifi_connect();
}
//delay(300000);                /* 5 minutes */
}

```

```

#include <HTTPClient.h>
#include <string>
#include "esp_wifi.h"

using namespace std;

#define uS_TO_S_FACTOR 1000000 /* Conversion factor for micro seconds to seconds */
#define TIME_TO_SLEEP  300      /* Time ESP32 will go to sleep (in seconds) */

void wifi_connect() {

    WiFi.begin("[SSID]", "[PASSWORD]");

    Serial.println("Connecting");
    while(WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("Connected");
}

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

    while (!Serial);          /* ? */
    Serial.println("Sleep experiment");
    wifi_connect();
}

int counter = 0;

void loop() {
    //Check WiFi connection status
    if(WiFi.status()== WL_CONNECTED){
        HTTPClient http;

        // Your Domain name with URL path or IP address with path
        http.begin("http://10.1.1.11:4567/log?board=esp32&test=lightsleep");

        // Send HTTP GET request
        int httpResponseCode = http.GET();

        if (httpResponseCode==200) {
            Serial.print("Success");
        } else {
            Serial.print("Error code: ");
            Serial.println(httpResponseCode);
        }
    }
}

```

```

    }

    // Free resources
    http.end();

    // Need to stop wifi before sleeping, then start again.
    esp_wifi_stop();
    esp_sleep_enable_timer_wakeup(TIME_TO_SLEEP * uS_TO_S_FACTOR);
    esp_light_sleep_start();
    esp_wifi_start();
}
else {
    Serial.println("WiFi Disconnected. Attempting to re-connect");
    wifi_connect();
}
//delay(300000);          /* 5 minutes */
}

```

We need the esp_wifi_stop/start commands otherwise it's not working;

https://www.reddit.com/r/esp32/comments/rncows/light_sleep_and_wifi/?rdt=34754
https://www.reddit.com/r/esp32/comments/rncows/light_sleep_and_wifi/?rdt=34754
https://www.reddit.com/r/esp32/comments/rncows/light_sleep_and_wifi/?rdt=34754
https://www.reddit.com/r/esp32/comments/rncows/light_sleep_and_wifi/?rdt=34754
https://www.reddit.com/r/esp32/comments/rncows/light_sleep_and_wifi/?rdt=34754
https://www.reddit.com/r/esp32/comments/rncows/light_sleep_and_wifi/?rdt=34754
https://www.reddit.com/r/esp32/comments/rncows/light_sleep_and_wifi/?rdt=34754
https://www.reddit.com/r/esp32/comments/rncows/light_sleep_and_wifi/?rdt=34754
https://www.reddit.com/r/esp32/comments/rncows/light_sleep_and_wifi/?rdt=34754
https://www.reddit.com/r/esp32/comments/rncows/light_sleep_and_wifi/?rdt=34754
https://www.reddit.com/r/esp32/comments/rncows/light_sleep_and_wifi/?rdt=34754
https://www.reddit.com/r/esp32/comments/rncows/light_sleep_and_wifi/?rdt=34754
https://www.reddit.com/r/esp32/comments/rncows/light_sleep_and_wifi/?rdt=34754
https://www.reddit.com/r/esp32/comments/rncows/light_sleep_and_wifi/?rdt=34754
https://www.reddit.com/r/esp32/comments/rncows/light_sleep_and_wifi/?rdt=34754
https://www.reddit.com/r/esp32/comments/rncows/light_sleep_and_wifi/?rdt=34754

Experiment #4: Deep Sleep

```
#include <HTTPClient.h>
#include <string>
#include "esp_wifi.h"

using namespace std;

#define uS_TO_S_FACTOR 1000000 /* Conversion factor for micro seconds to seconds */
#define TIME_TO_SLEEP 300 /* Time ESP32 will go to sleep (in seconds) */

void wifi_connect() {

    WiFi.begin("[SSID]", "[PASSWORD]");

    Serial.println("Connecting");
    while(WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("Connected");
}

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

    while (!Serial); /* ? */
    Serial.println("Sleep experiment");
    wifi_connect();
}

int counter = 0;

void loop() {
    //Check WiFi connection status
    if(WiFi.status() == WL_CONNECTED){
        HTTPClient http;

        // Your Domain name with URL path or IP address with path
        http.begin("http://10.1.1.11:4567/log?board=esp32&test=deepsleep");

        // Send HTTP GET request
```

```

int httpResponseCode = http.GET();

if (httpResponseCode==200) {
    Serial.print("Success");
} else {
    Serial.print("Error code: ");
    Serial.println(httpResponseCode);
}

// Free resources
http.end();

// Need to stop wifi before sleeping, then start again.
esp_wifi_stop();
esp_sleep_enable_timer_wakeup(TIME_TO_SLEEP * uS_TO_S_FACTOR);
esp_deep_sleep_start();
esp_wifi_start();

}
else {
    Serial.println("WiFi Disconnected. Attempting to re-connect");
    wifi_connect();
}

//delay(300000);          /* 5 minutes */
}

```

```

#include <HTTPClient.h>
#include <string>
#include "esp_wifi.h"

using namespace std;

#define uS_TO_S_FACTOR 1000000 /* Conversion factor for micro seconds to seconds */
#define TIME_TO_SLEEP 300 /* Time ESP32 will go to sleep (in seconds) */

void wifi_connect() {

  WiFi.begin("[SSID]", "[PASSWORD]");

  Serial.println("Connecting");
  while(WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("Connected");
}

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

  while (!Serial); /* ? */
  Serial.println("Sleep experiment");
  wifi_connect();
}

int counter = 0;

void loop() {
  //Check WiFi connection status
  if(WiFi.status() == WL_CONNECTED){
    HTTPClient http;

    // Your Domain name with URL path or IP address with path
    http.begin("http://10.1.1.11:4567/log?board=esp32&test=deepsleep");

    // Send HTTP GET request
    int httpResponseCode = http.GET();

```

```
if (httpResponseCode==200) {
  Serial.print("Success");
} else {
  Serial.print("Error code: ");
  Serial.println(httpResponseCode);
}

// Free resources
http.end();

// Need to stop wifi before sleeping, then start again.
esp_wifi_stop();
esp_sleep_enable_timer_wakeup(TIME_TO_SLEEP * uS_TO_S_FACTOR);
esp_deep_sleep_start();
esp_wifi_start();

}
else {
  Serial.println("WiFi Disconnected. Attempting to re-connect");
  wifi_connect();
}

//delay(300000);          /* 5 minutes */
}
```

```

#include <HTTPClient.h>
#include <string>
#include "esp_wifi.h"

using namespace std;

#define uS_TO_S_FACTOR 1000000 /* Conversion factor for micro seconds to seconds */
#define TIME_TO_SLEEP 300 /* Time ESP32 will go to sleep (in seconds) */

void wifi_connect() {

  WiFi.begin("[SSID]", "[PASSWORD]");

  Serial.println("Connecting");
  while(WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("Connected");
}

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

  while (!Serial); /* ? */
  Serial.println("Sleep experiment");
  wifi_connect();
}

int counter = 0;

void loop() {
  //Check WiFi connection status
  if(WiFi.status() == WL_CONNECTED){
    HTTPClient http;

    // Your Domain name with URL path or IP address with path
    http.begin("http://10.1.1.11:4567/log?board=esp32&test=deepsleep");

    // Send HTTP GET request
    int httpResponseCode = http.GET();

```

```
if (httpResponseCode==200) {
  Serial.print("Success");
} else {
  Serial.print("Error code: ");
  Serial.println(httpResponseCode);
}

// Free resources
http.end();

// Need to stop wifi before sleeping, then start again.
esp_wifi_stop();
esp_sleep_enable_timer_wakeup(TIME_TO_SLEEP * uS_TO_S_FACTOR);
esp_deep_sleep_start();
esp_wifi_start();

}
else {
  Serial.println("WiFi Disconnected. Attempting to re-connect");
  wifi_connect();
}

//delay(300000);          /* 5 minutes */
}
```

```

#include <HTTPClient.h>
#include <string>
#include "esp_wifi.h"

using namespace std;

#define uS_TO_S_FACTOR 1000000 /* Conversion factor for micro seconds to seconds */
#define TIME_TO_SLEEP 300 /* Time ESP32 will go to sleep (in seconds) */

void wifi_connect() {

  WiFi.begin("[SSID]", "[PASSWORD]");

  Serial.println("Connecting");
  while(WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("Connected");
}

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

  while (!Serial); /* ? */
  Serial.println("Sleep experiment");
  wifi_connect();
}

int counter = 0;

void loop() {
  //Check WiFi connection status
  if(WiFi.status() == WL_CONNECTED){
    HTTPClient http;

    // Your Domain name with URL path or IP address with path
    http.begin("http://10.1.1.11:4567/log?board=esp32&test=deepsleep");

    // Send HTTP GET request
    int httpResponseCode = http.GET();

```

```

if (httpResponseCode==200) {
    Serial.print("Success");
} else {
    Serial.print("Error code: ");
    Serial.println(httpResponseCode);
}

// Free resources
http.end();

// Need to stop wifi before sleeping, then start again.
esp_wifi_stop();
esp_sleep_enable_timer_wakeup(TIME_TO_SLEEP * uS_TO_S_FACTOR);
esp_deep_sleep_start();
esp_wifi_start();

}
else {
    Serial.println("WiFi Disconnected. Attempting to re-connect");
    wifi_connect();
}

//delay(300000);          /* 5 minutes */
}

```

```

#include <HTTPClient.h>
#include <string>
#include "esp_wifi.h"

using namespace std;

#define uS_TO_S_FACTOR 1000000 /* Conversion factor for micro seconds to seconds */
#define TIME_TO_SLEEP 300 /* Time ESP32 will go to sleep (in seconds) */

void wifi_connect() {

  WiFi.begin("[SSID]", "[PASSWORD]");

  Serial.println("Connecting");
  while(WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("Connected");
}

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

  while (!Serial); /* ? */
  Serial.println("Sleep experiment");
  wifi_connect();
}

int counter = 0;

void loop() {
  //Check WiFi connection status
  if(WiFi.status() == WL_CONNECTED){
    HTTPClient http;

    // Your Domain name with URL path or IP address with path
    http.begin("http://10.1.1.11:4567/log?board=esp32&test=deepsleep");

    // Send HTTP GET request
    int httpResponseCode = http.GET();

```

```
if (httpResponseCode==200) {
  Serial.print("Success");
} else {
  Serial.print("Error code: ");
  Serial.println(httpResponseCode);
}

// Free resources
http.end();

// Need to stop wifi before sleeping, then start again.
esp_wifi_stop();
esp_sleep_enable_timer_wakeup(TIME_TO_SLEEP * uS_TO_S_FACTOR);
esp_deep_sleep_start();
esp_wifi_start();

}
else {
  Serial.println("WiFi Disconnected. Attempting to re-connect");
  wifi_connect();
}

//delay(300000);          /* 5 minutes */
}
```

```
#include <HTTPClient.h>
#include <string>
#include "esp_wifi.h"

using namespace std;

#define uS_TO_S_FACTOR 1000000 /* Conversion factor for micro seconds to seconds */
#define TIME_TO_SLEEP  300      /* Time ESP32 will go to sleep (in seconds) */

void wifi_connect() {

    WiFi.begin("[SSID]", "[PASSWORD]");

    Serial.println("Connecting");
    while(WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("Connected");
}

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

    while (!Serial); /* ? */
    Serial.println("Sleep experiment");
    wifi_connect();
}

int counter = 0;

void loop() {
    //Check WiFi connection status
    if(WiFi.status()== WL_CONNECTED){
        HTTPClient http;

        // Your Domain name with URL path or IP address with path
        http.begin("http://10.1.1.11:4567/log?board=esp32&test=deepsleep");

        // Send HTTP GET request
        int httpStatusCode = http.GET();

        if (httpStatusCode==200) {
            Serial.print("Success");
        } else {
            Serial.print("Error code: ");
            Serial.println(httpStatusCode);
        }
    }
}
```

```
}

// Free resources
http.end();

// Need to stop wifi before sleeping, then start again.
esp_wifi_stop();
esp_sleep_enable_timer_wakeup(TIME_TO_SLEEP * uS_TO_S_FACTOR);
esp_deep_sleep_start();
esp_wifi_start();

}
else {
  Serial.println("WiFi Disconnected. Attempting to re-connect");
  wifi_connect();
}

//delay(300000);          /* 5 minutes */
}
```

C

```
#include <HTTPClient.h>
#include <string>
#include "esp_wifi.h"

using namespace std;

#define uS_TO_S_FACTOR 1000000 /* Conversion factor for micro seconds to seconds */
#define TIME_TO_SLEEP  300      /* Time ESP32 will go to sleep (in seconds) */

void wifi_connect() {

    WiFi.begin("[SSID]", "[PASSWORD]");

    Serial.println("Connecting");
    while(WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("Connected");
}

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

    while (!Serial); /* ? */
    Serial.println("Sleep experiment");
    wifi_connect();
}

int counter = 0;

void loop() {
    //Check WiFi connection status
    if(WiFi.status()== WL_CONNECTED){
        HTTPClient http;

        // Your Domain name with URL path or IP address with path
        http.begin("http://10.1.1.11:4567/log?board=esp32&test=deepsleep");

        // Send HTTP GET request
        int httpStatusCode = http.GET();

        if (httpStatusCode==200) {
            Serial.print("Success");
        } else {
            Serial.print("Error code: ");
            Serial.println(httpStatusCode);
        }
    }
}
```

```

    }

    // Free resources
    http.end();

    // Need to stop wifi before sleeping, then start again.
    esp_wifi_stop();
    esp_sleep_enable_timer_wakeup(TIME_TO_SLEEP * uS_TO_S_FACTOR);
    esp_deep_sleep_start();
    esp_wifi_start();

}
else {
    Serial.println("WiFi Disconnected. Attempting to re-connect");
    wifi_connect();
}

//delay(300000);                /* 5 minutes */
}

```

C

```
#include <HTTPClient.h>
#include <string>
#include "esp_wifi.h"

using namespace std;

#define uS_TO_S_FACTOR 1000000 /* Conversion factor for micro seconds to seconds */
#define TIME_TO_SLEEP  300      /* Time ESP32 will go to sleep (in seconds) */

void wifi_connect() {

    WiFi.begin("[SSID]", "[PASSWORD]");

    Serial.println("Connecting");
    while(WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("Connected");
}

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

    while (!Serial); /* ? */
    Serial.println("Sleep experiment");
    wifi_connect();
}

int counter = 0;

void loop() {
    //Check WiFi connection status
    if(WiFi.status()== WL_CONNECTED){
        HTTPClient http;

        // Your Domain name with URL path or IP address with path
        http.begin("http://10.1.1.11:4567/log?board=esp32&test=deepsleep");

        // Send HTTP GET request
        int httpStatusCode = http.GET();

        if (httpStatusCode==200) {
            Serial.print("Success");
        } else {
            Serial.print("Error code: ");
            Serial.println(httpStatusCode);
        }
    }
}
```

```

    }

    // Free resources
    http.end();

    // Need to stop wifi before sleeping, then start again.
    esp_wifi_stop();
    esp_sleep_enable_timer_wakeup(TIME_TO_SLEEP * uS_TO_S_FACTOR);
    esp_deep_sleep_start();
    esp_wifi_start();

}
else {
    Serial.println("WiFi Disconnected. Attempting to re-connect");
    wifi_connect();
}

//delay(300000);                /* 5 minutes */
}

```

C

```
#include <HTTPClient.h>
#include <string>
#include "esp_wifi.h"

using namespace std;

#define uS_TO_S_FACTOR 1000000 /* Conversion factor for micro seconds to seconds */
#define TIME_TO_SLEEP  300      /* Time ESP32 will go to sleep (in seconds) */

void wifi_connect() {

    WiFi.begin("[SSID]", "[PASSWORD]");

    Serial.println("Connecting");
    while(WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("Connected");
}

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

    while (!Serial); /* ? */
    Serial.println("Sleep experiment");
    wifi_connect();
}

int counter = 0;

void loop() {
    //Check WiFi connection status
    if(WiFi.status()== WL_CONNECTED){
        HTTPClient http;

        // Your Domain name with URL path or IP address with path
        http.begin("http://10.1.1.11:4567/log?board=esp32&test=deepsleep");

        // Send HTTP GET request
        int httpResponseCode = http.GET();

        if (httpResponseCode==200) {
            Serial.print("Success");
        } else {
            Serial.print("Error code: ");
            Serial.println(httpResponseCode);
        }
    }
}
```

```
}

// Free resources
http.end();

// Need to stop wifi before sleeping, then start again.
esp_wifi_stop();
esp_sleep_enable_timer_wakeup(TIME_TO_SLEEP * uS_TO_S_FACTOR);
esp_deep_sleep_start();
esp_wifi_start();

}
else {
  Serial.println("WiFi Disconnected. Attempting to re-connect");
  wifi_connect();
}

//delay(300000);          /* 5 minutes */
}
```

C

```
#include <HTTPClient.h>
#include <string>
#include "esp_wifi.h"

using namespace std;

#define uS_TO_S_FACTOR 1000000 /* Conversion factor for micro seconds to seconds */
#define TIME_TO_SLEEP  300      /* Time ESP32 will go to sleep (in seconds) */

void wifi_connect() {

    WiFi.begin("[SSID]", "[PASSWORD]");

    Serial.println("Connecting");
    while(WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("Connected");
}

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

    while (!Serial); /* ? */
    Serial.println("Sleep experiment");
    wifi_connect();
}

int counter = 0;

void loop() {
    //Check WiFi connection status
    if(WiFi.status()== WL_CONNECTED){
        HTTPClient http;

        // Your Domain name with URL path or IP address with path
        http.begin("http://10.1.1.11:4567/log?board=esp32&test=deepsleep");

        // Send HTTP GET request
        int httpStatusCode = http.GET();

        if (httpStatusCode==200) {
            Serial.print("Success");
        } else {
            Serial.print("Error code: ");
            Serial.println(httpStatusCode);
        }
    }
}
```

```
}

// Free resources
http.end();

// Need to stop wifi before sleeping, then start again.
esp_wifi_stop();
esp_sleep_enable_timer_wakeup(TIME_TO_SLEEP * uS_TO_S_FACTOR);
esp_deep_sleep_start();
esp_wifi_start();

}
else {
  Serial.println("WiFi Disconnected. Attempting to re-connect");
  wifi_connect();
}

//delay(300000);                /* 5 minutes */
}
```

C

```
#include <HTTPClient.h>
#include <string>
#include "esp_wifi.h"

using namespace std;

#define uS_TO_S_FACTOR 1000000 /* Conversion factor for micro seconds to seconds */
#define TIME_TO_SLEEP  300      /* Time ESP32 will go to sleep (in seconds) */

void wifi_connect() {

    WiFi.begin("[SSID]", "[PASSWORD]");

    Serial.println("Connecting");
    while(WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("Connected");
}

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

    while (!Serial); /* ? */
    Serial.println("Sleep experiment");
    wifi_connect();
}

int counter = 0;

void loop() {
    //Check WiFi connection status
    if(WiFi.status()== WL_CONNECTED){
        HTTPClient http;

        // Your Domain name with URL path or IP address with path
        http.begin("http://10.1.1.11:4567/log?board=esp32&test=deepsleep");

        // Send HTTP GET request
        int httpResponseCode = http.GET();

        if (httpResponseCode==200) {
            Serial.print("Success");
        } else {
            Serial.print("Error code: ");
            Serial.println(httpResponseCode);
        }
    }
}
```

```
}

// Free resources
http.end();

// Need to stop wifi before sleeping, then start again.
esp_wifi_stop();
esp_sleep_enable_timer_wakeup(TIME_TO_SLEEP * uS_TO_S_FACTOR);
esp_deep_sleep_start();
esp_wifi_start();

}
else {
  Serial.println("WiFi Disconnected. Attempting to re-connect");
  wifi_connect();
}

//delay(300000);          /* 5 minutes */
}
```

C

```
#include <HTTPClient.h>
#include <string>
#include "esp_wifi.h"

using namespace std;

#define uS_TO_S_FACTOR 1000000 /* Conversion factor for micro seconds to seconds */
#define TIME_TO_SLEEP  300      /* Time ESP32 will go to sleep (in seconds) */

void wifi_connect() {

    WiFi.begin("[SSID]", "[PASSWORD]");

    Serial.println("Connecting");
    while(WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("Connected");
}

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

    while (!Serial); /* ? */
    Serial.println("Sleep experiment");
    wifi_connect();
}

int counter = 0;

void loop() {
    //Check WiFi connection status
    if(WiFi.status()== WL_CONNECTED){
        HTTPClient http;

        // Your Domain name with URL path or IP address with path
        http.begin("http://10.1.1.11:4567/log?board=esp32&test=deepsleep");

        // Send HTTP GET request
        int httpResponseCode = http.GET();

        if (httpResponseCode==200) {
            Serial.print("Success");
        } else {
            Serial.print("Error code: ");
            Serial.println(httpResponseCode);
        }
    }
}
```

```

    }

    // Free resources
    http.end();

    // Need to stop wifi before sleeping, then start again.
    esp_wifi_stop();
    esp_sleep_enable_timer_wakeup(TIME_TO_SLEEP * uS_TO_S_FACTOR);
    esp_deep_sleep_start();
    esp_wifi_start();

}
else {
    Serial.println("WiFi Disconnected. Attempting to re-connect");
    wifi_connect();
}

//delay(300000);                /* 5 minutes */
}

```

C

```
#include <HTTPClient.h>
#include <string>
#include "esp_wifi.h"

using namespace std;

#define uS_TO_S_FACTOR 1000000 /* Conversion factor for micro seconds to seconds */
#define TIME_TO_SLEEP  300      /* Time ESP32 will go to sleep (in seconds) */

void wifi_connect() {

    WiFi.begin("[SSID]", "[PASSWORD]");

    Serial.println("Connecting");
    while(WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("Connected");
}

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

    while (!Serial); /* ? */
    Serial.println("Sleep experiment");
    wifi_connect();
}

int counter = 0;

void loop() {
    //Check WiFi connection status
    if(WiFi.status()== WL_CONNECTED){
        HTTPClient http;

        // Your Domain name with URL path or IP address with path
        http.begin("http://10.1.1.11:4567/log?board=esp32&test=deepsleep");

        // Send HTTP GET request
        int httpStatusCode = http.GET();

        if (httpStatusCode==200) {
            Serial.print("Success");
        } else {
            Serial.print("Error code: ");
            Serial.println(httpStatusCode);
        }
    }
}
```

```

    }

    // Free resources
    http.end();

    // Need to stop wifi before sleeping, then start again.
    esp_wifi_stop();
    esp_sleep_enable_timer_wakeup(TIME_TO_SLEEP * uS_TO_S_FACTOR);
    esp_deep_sleep_start();
    esp_wifi_start();

}
else {
    Serial.println("WiFi Disconnected. Attempting to re-connect");
    wifi_connect();
}

//delay(300000);                /* 5 minutes */
}

```

C

```
#include <HTTPClient.h>
#include <string>
#include "esp_wifi.h"

using namespace std;

#define uS_TO_S_FACTOR 1000000 /* Conversion factor for micro seconds to seconds */
#define TIME_TO_SLEEP  300      /* Time ESP32 will go to sleep (in seconds) */

void wifi_connect() {

    WiFi.begin("[SSID]", "[PASSWORD]");

    Serial.println("Connecting");
    while(WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("Connected");
}

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

    while (!Serial); /* ? */
    Serial.println("Sleep experiment");
    wifi_connect();
}

int counter = 0;

void loop() {
    //Check WiFi connection status
    if(WiFi.status()== WL_CONNECTED){
        HTTPClient http;

        // Your Domain name with URL path or IP address with path
        http.begin("http://10.1.1.11:4567/log?board=esp32&test=deepsleep");

        // Send HTTP GET request
        int httpResponseCode = http.GET();

        if (httpResponseCode==200) {
            Serial.print("Success");
        } else {
            Serial.print("Error code: ");
            Serial.println(httpResponseCode);
        }
    }
}
```

```

    }

    // Free resources
    http.end();

    // Need to stop wifi before sleeping, then start again.
    esp_wifi_stop();
    esp_sleep_enable_timer_wakeup(TIME_TO_SLEEP * uS_TO_S_FACTOR);
    esp_deep_sleep_start();
    esp_wifi_start();

}
else {
    Serial.println("WiFi Disconnected. Attempting to re-connect");
    wifi_connect();
}

//delay(300000);                /* 5 minutes */
}

```

C

```
#include <HTTPClient.h>
#include <string>
#include "esp_wifi.h"

using namespace std;

#define uS_TO_S_FACTOR 1000000 /* Conversion factor for micro seconds to seconds */
#define TIME_TO_SLEEP  300      /* Time ESP32 will go to sleep (in seconds) */

void wifi_connect() {

    WiFi.begin("[SSID]", "[PASSWORD]");

    Serial.println("Connecting");
    while(WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("Connected");
}

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

    while (!Serial); /* ? */
    Serial.println("Sleep experiment");
    wifi_connect();
}

int counter = 0;

void loop() {
    //Check WiFi connection status
    if(WiFi.status()== WL_CONNECTED){
        HTTPClient http;

        // Your Domain name with URL path or IP address with path
        http.begin("http://10.1.1.11:4567/log?board=esp32&test=deepsleep");

        // Send HTTP GET request
        int httpStatusCode = http.GET();

        if (httpStatusCode==200) {
            Serial.print("Success");
        } else {
            Serial.print("Error code: ");
            Serial.println(httpStatusCode);
        }
    }
}
```

```

    }

    // Free resources
    http.end();

    // Need to stop wifi before sleeping, then start again.
    esp_wifi_stop();
    esp_sleep_enable_timer_wakeup(TIME_TO_SLEEP * uS_TO_S_FACTOR);
    esp_deep_sleep_start();
    esp_wifi_start();

}
else {
    Serial.println("WiFi Disconnected. Attempting to re-connect");
    wifi_connect();
}

//delay(300000);                /* 5 minutes */
}

```

C

```
#include <HTTPClient.h>
#include <string>
#include "esp_wifi.h"

using namespace std;

#define uS_TO_S_FACTOR 1000000 /* Conversion factor for micro seconds to seconds */
#define TIME_TO_SLEEP  300      /* Time ESP32 will go to sleep (in seconds) */

void wifi_connect() {

    WiFi.begin("[SSID]", "[PASSWORD]");

    Serial.println("Connecting");
    while(WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("Connected");
}

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

    while (!Serial); /* ? */
    Serial.println("Sleep experiment");
    wifi_connect();
}

int counter = 0;

void loop() {
    //Check WiFi connection status
    if(WiFi.status()== WL_CONNECTED){
        HTTPClient http;

        // Your Domain name with URL path or IP address with path
        http.begin("http://10.1.1.11:4567/log?board=esp32&test=deepsleep");

        // Send HTTP GET request
        int httpStatusCode = http.GET();

        if (httpStatusCode==200) {
            Serial.print("Success");
        } else {
            Serial.print("Error code: ");
            Serial.println(httpStatusCode);
        }
    }
}
```

```

    }

    // Free resources
    http.end();

    // Need to stop wifi before sleeping, then start again.
    esp_wifi_stop();
    esp_sleep_enable_timer_wakeup(TIME_TO_SLEEP * uS_TO_S_FACTOR);
    esp_deep_sleep_start();
    esp_wifi_start();

}
else {
    Serial.println("WiFi Disconnected. Attempting to re-connect");
    wifi_connect();
}

//delay(300000);                /* 5 minutes */
}

```

From lightsleep to deepsleep the only line I changed is this;From lightsleep to deepsleep the only line I changed is this;From lightsleep to deepsleep the only line I changed is this;From lightsleep to deepsleep the only line I changed is this;From lightsleep to deepsleep the only line I changed is this;From lightsleep to deepsleep the only line I changed is this;From lightsleep to deepsleep the only line I changed is this;From lightsleep to deepsleep the only line I changed is this;From lightsleep to deepsleep the only line I changed is this;

c

```
esp_deep_sleep_start();
```

Experiment #5: Hibernation

Results

Board Experiment Complexity x/5WIFI PostTime

esp32No sleep 0 5min 38hrs

esp32Modem sleep

esp32lightsleep

esp32deepsleep

esp32hibernation

pico No sleep

pico Modem sleep

pico lightsleep

pico deepsleep

pico hibernation

pico2 No sleep

pico2 Modem sleep

pico2 lightsleep

pico2 deepsleep

pico2 hibernation

Board Experiment Complexity x/5WIFI PostTime

esp32No sleep 0 5min 38hrs

esp32Modem sleep

esp32lightsleep

esp32deepsleep

esp32hibernation

pico No sleep

pico Modem sleep

pico lightsleep

pico deepsleep

pico hibernation

pico2 No sleep

pico2 Modem sleep

pico2 lightsleep

pico2 deepsleep

pico2 hibernation

Board Experiment Complexity x/5WIFI PostTime

esp32No sleep 0 5min 38hrs

esp32Modem sleep

esp32lightsleep

esp32deepsleep

esp32hibernation

pico No sleep

pico Modem sleep

pico lightsleep

pico deepsleep

pico hibernation

pico2 No sleep

pico2 Modem sleep

pico2 lightsleep

pico2 deepsleep

pico2 hibernation

Board Experiment Complexity x/5WIFI PostTime

esp32No sleep 0 5min 38hrs

esp32Modem sleep

esp32lightsleep

esp32deepsleep

esp32hibernation

pico No sleep

pico Modem sleep

pico lightsleep

pico deepsleep

pico hibernation

pico2 No sleep

pico2 Modem sleep

pico2 lightsleep

pico2 deepsleep

pico2 hibernation

Board Experiment Complexity x/5WIFI PostTime

esp32No sleep	0	5min	38hrs
---------------	---	------	-------

esp32Modem sleep

esp32lightsleep

esp32deepsleep

esp32hibernation

pico No sleep

pico Modem sleep

pico lightsleep

pico deepsleep

pico hibernation

pico2 No sleep

pico2 Modem sleep

pico2 lightsleep

pico2 deepsleep

pico2 hibernation

Board Experiment Complexity x/5WIFI PostTime

esp32No sleep	0	5min	38hrs
---------------	---	------	-------

esp32Modem sleep

esp32lightsleep

esp32deepsleep

esp32hibernation

pico No sleep

pico Modem sleep

pico lightsleep

pico deepsleep

pico hibernation

pico2 No sleep

pico2 Modem sleep

pico2 lightsleep

pico2 deepsleep

pico2 hibernation

Board Experiment Complexity x/5WIFI PostTime

esp32No sleep	0	5min	38hrs
---------------	---	------	-------

esp32Modem sleep

esp32lightsleep

esp32deepsleep

esp32hibernation

pico No sleep

pico Modem sleep

pico lightsleep

pico deepsleep

pico hibernation

pico2 No sleep

pico2 Modem sleep

pico2 lightsleep

pico2 deepsleep

pico2 hibernation

Board Experiment Complexity x/5WIFI PostTime

esp32No sleep 0 5min 38hrs

esp32Modem sleep

esp32lightsleep

esp32deepsleep

esp32hibernation

pico No sleep

pico Modem sleep

pico lightsleep

pico deepsleep

pico hibernation

pico2 No sleep

pico2 Modem sleep

pico2 lightsleep

pico2 deepsleep

pico2 hibernation

Board Experiment Complexity x/5WIFI PostTime

esp32No sleep 0 5min 38hrs

esp32Modem sleep

esp32lightsleep

esp32deepsleep

esp32hibernation

pico No sleep

pico Modem sleep

pico lightsleep

pico deepsleep

pico hibernation

pico2 No sleep

pico2 Modem sleep

pico2 lightsleep

pico2 deepsleep

pico2 hibernation

Board Experiment Complexity x/5WIFI PostTime

esp32No sleep	0	5min	38hrs
---------------	---	------	-------

esp32Modem sleep

esp32lightsleep

esp32deepsleep

esp32hibernation

pico No sleep

pico Modem sleep

pico lightsleep

pico deepsleep

pico hibernation

pico2 No sleep

pico2 Modem sleep

pico2 lightsleep

pico2 deepsleep

pico2 hibernation

Board Experiment Complexity x/5WIFI PostTime

esp32No sleep	0	5min	38hrs
---------------	---	------	-------

esp32Modem sleep

esp32lightsleep

esp32deepsleep

esp32hibernation

pico No sleep

pico Modem sleep

pico lightsleep

pico deepsleep

pico hibernation

pico2 No sleep

pico2 Modem sleep

pico2 lightsleep

pico2 deepsleep

pico2 hibernation

Board Experiment Complexity x/5WIFI PostTime

esp32No sleep	0	5min	38hrs
---------------	---	------	-------

esp32Modem sleep

esp32lightsleep

esp32deepsleep

esp32hibernation

pico No sleep

pico Modem sleep

pico lightsleep

pico deepsleep

pico hibernation

pico2 No sleep

pico2 Modem sleep

pico2 lightsleep

pico2 deepsleep

pico2 hibernation

Board	Experiment	Complexity	x/5	WIFI	PostTime
-------	------------	------------	-----	------	----------

esp32	No sleep	0		5min	38hrs
-------	----------	---	--	------	-------

esp32	Modem sleep				
-------	-------------	--	--	--	--

esp32	lightsleep				
-------	------------	--	--	--	--

esp32	deepsleep				
-------	-----------	--	--	--	--

esp32	hibernation				
-------	-------------	--	--	--	--

pico	No sleep				
------	----------	--	--	--	--

pico	Modem sleep				
------	-------------	--	--	--	--

pico	lightsleep				
------	------------	--	--	--	--

pico	deepsleep				
------	-----------	--	--	--	--

pico	hibernation				
------	-------------	--	--	--	--

pico2	No sleep				
-------	----------	--	--	--	--

pico2	Modem sleep				
-------	-------------	--	--	--	--

pico2	lightsleep				
-------	------------	--	--	--	--

pico2	deepsleep				
-------	-----------	--	--	--	--

pico2	hibernation				
-------	-------------	--	--	--	--

Board	Experiment	Complexity	x/5	WIFI	PostTime
-------	------------	------------	-----	------	----------

esp32	No sleep	0		5min	38hrs
-------	----------	---	--	------	-------

esp32	Modem sleep				
-------	-------------	--	--	--	--

esp32	lightsleep				
-------	------------	--	--	--	--

esp32	deepsleep				
-------	-----------	--	--	--	--

esp32	hibernation				
-------	-------------	--	--	--	--

pico	No sleep				
------	----------	--	--	--	--

pico	Modem sleep				
------	-------------	--	--	--	--

pico	lightsleep				
------	------------	--	--	--	--

pico	deepsleep				
------	-----------	--	--	--	--

pico	hibernation				
------	-------------	--	--	--	--

pico2 No sleep

pico2 Modem sleep

pico2 lightsleep

pico2 deepsleep

pico2 hibernation

Board Experiment Complexity x/5WIFI PostTime

esp32No sleep	0	5min	38hrs
---------------	---	------	-------

esp32Modem sleep

esp32lightsleep

esp32deepsleep

esp32hibernation

pico No sleep

pico Modem sleep

pico lightsleep

pico deepsleep

pico hibernation

pico2 No sleep

pico2 Modem sleep

pico2 lightsleep

pico2 deepsleep

pico2 hibernation

Board Experiment Complexity x/5WIFI PostTime

esp32No sleep	0	5min	38hrs
---------------	---	------	-------

esp32Modem sleep

esp32lightsleep

esp32deepsleep

esp32hibernation

pico No sleep

pico Modem sleep

pico lightsleep

pico deepsleep

pico hibernation

pico2 No sleep

pico2 Modem sleep

pico2 lightsleep

pico2 deepsleep

pico2 hibernation

Later (with Solar)Later (with Solar)Later (with Solar)



"Batteries" by en:User:Brianiac licensed under PUBLIC DOMAIN.