# Project Overview

This project implements a firmware application for the ESP32 Dev Kit that downloads a file via HTTPS from a public URL and writes it to SPIFFS (SPI Flash File System) with a target speed of at least 400KBps. Implemented error handling (storage limits and timeouts) features.

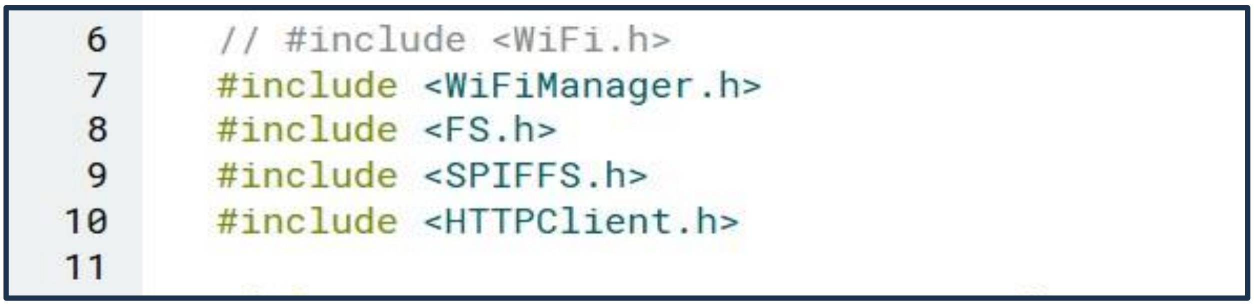
## Features and Functionality

- **Dynamic configuration**: Wi-Fi configuration via WiFiManager library or can hard coded.
- **SPIFFS operations**
  - Write
  - Read
  - Delete
  - Append
  - Format SPIFFS
- **Download file**
  - Buffer allocation for faster reading and writing to file
  - Speed optimized to be at least 400kbps (download + write)
  - Logs download time and speed of files
- **Error handling**
  - Automatically connect to Wi-Fi, if failed opens access point to configure new credentials if needed.
  - **Time out** : Sets a timeout to prevent hanging of https requests
  - **Storage limits** : Checks the SPIFFS storage before downloading or writing to file.

## Hardware Requirements

- ESP32
- Internet connection

## Code implementation



```
// #include <WiFi.h>
#include <WiFiManager.h>
#include <FS.h>
#include <SPIFFS.h>
#include <HTTPClient.h>

```

Include the necessary header files

```
#define BUFFER_SIZE  2048      // adjust as needed

WiFiManager wifimanager;

String filename;

```

Define the buffer size for download of file

Create an object for the class WiFiManager

Declare a global string for filename

```
bool isNum(String input);
void PrintCommands();
void ListFiles();
void WriteFile();
void ReadFile();
void AppendFile();
void DeleteFile();
void FormatSPIFFS();
void DownloadFile();

```

Function Declarations

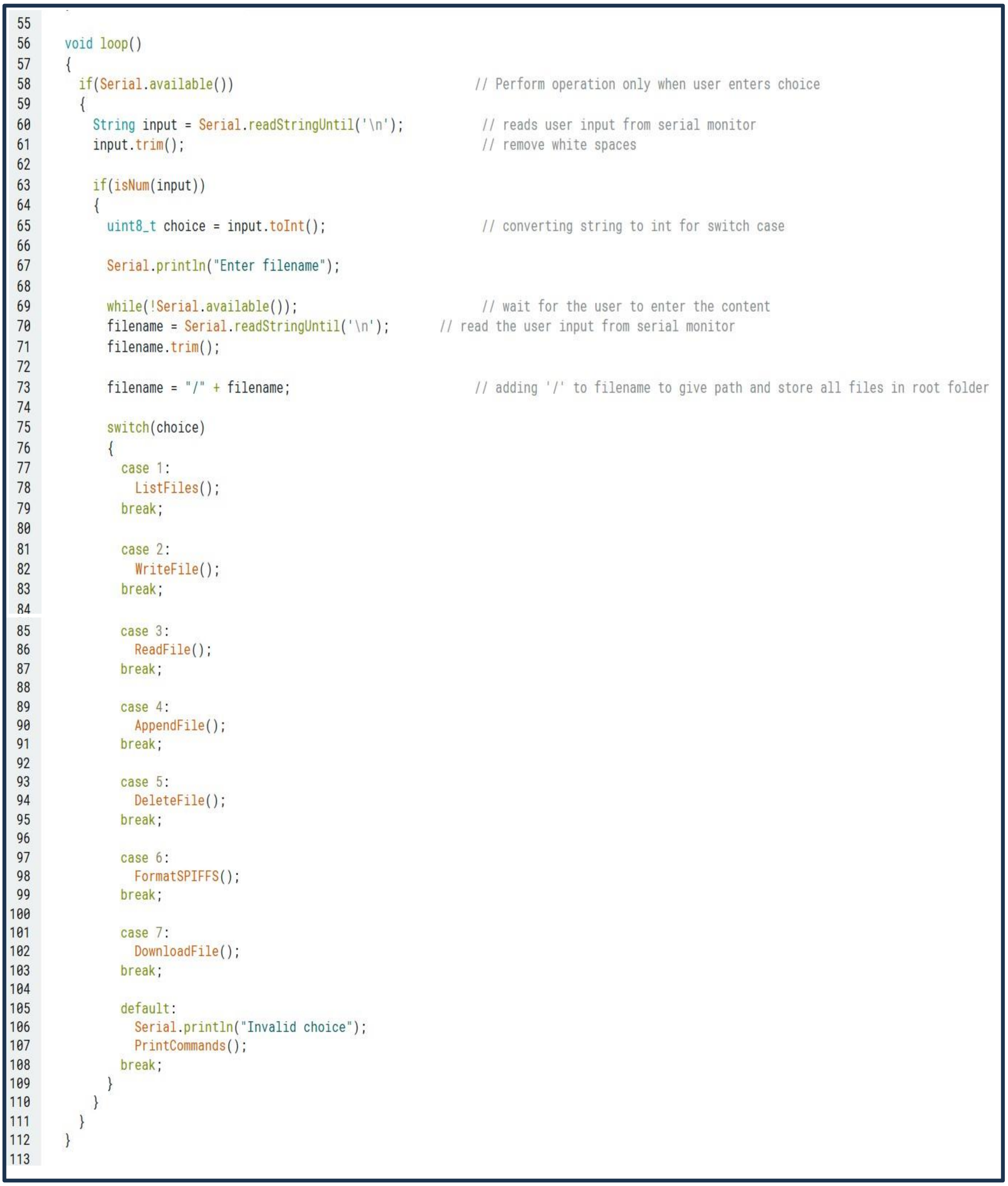
```

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

  // for dynamic congiuring of wifi ssid and password
  wifimanager.autoconnect("ESP32_WiFi_Config");

  // // for hard coding the wifi ssid and password
  // WiFi.mode(WIFI_STA);
  // WiFi.begin("LENOVO laptop", "08!0g5Y7");           // start wifi connecion
  // Serial.println("Connecting...");

  // while(WiFi.status() != WL_CONNECTED)                // wait till the wifi is connected
  // {
  //   Serial.print(".");
  //   delay(500);
  // }

  if(SPIFFS.begin(true))                                // Mounting SPIFFS
    Serial.println("SPIFFS Mounted successfully");
  else
  {
    Serial.println("SPIFFS Mount Failed");
    return;
  }
  PrintCommands();
}

```

Runs once.

Initialize the serial communication

The function wifimanager.autoConnect("ESP32_WiFi_Config"); connects to saved wifi network, if unable to connect it creates an access point with hotspot named ESP32_WiFi_Config.

SPIFFS.begin(true) mounts the SPIFFS and if fails to mount it formats the SPIFFS and then mounts the new file system.



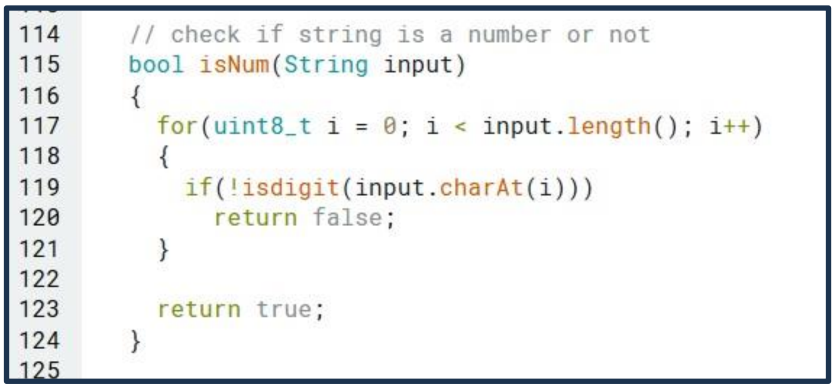
```
void loop()
{
  if(Serial.available())                               // Perform operation only when user enters choice
  {
    String input = Serial.readStringUntil('\n');        // reads user input from serial monitor
    input.trim();                                       // remove white spaces

    if(isNum(input))
    {
      uint8_t choice = input.toInt();                   // converting string to int for switch case

      Serial.println("Enter filename");

      while(!Serial.available());                       // wait for the user to enter the content
      filename = Serial.readStringUntil('\n');       // read the user input from serial monitor
      filename.trim();

      filename = "/" + filename;                       // adding '/' to filename to give path and store all files in root folder

      switch(choice)
      {
        case 1:
          ListFiles();
        break;

        case 2:
          WriteFile();
        break;

        case 3:
          ReadFile();
        break;

        case 4:
          AppendFile();
        break;

        case 5:
          DeleteFile();
        break;

        case 6:
          FormatSPIFFS();
        break;

        case 7:
          DownloadFile();
        break;

        default:
          Serial.println("Invalid choice");
          PrintCommands();
        break;
      }
    }
  }
}
```

Checks if there is any input from the user

Store the input in a string and remove the trailing and leading white spaces

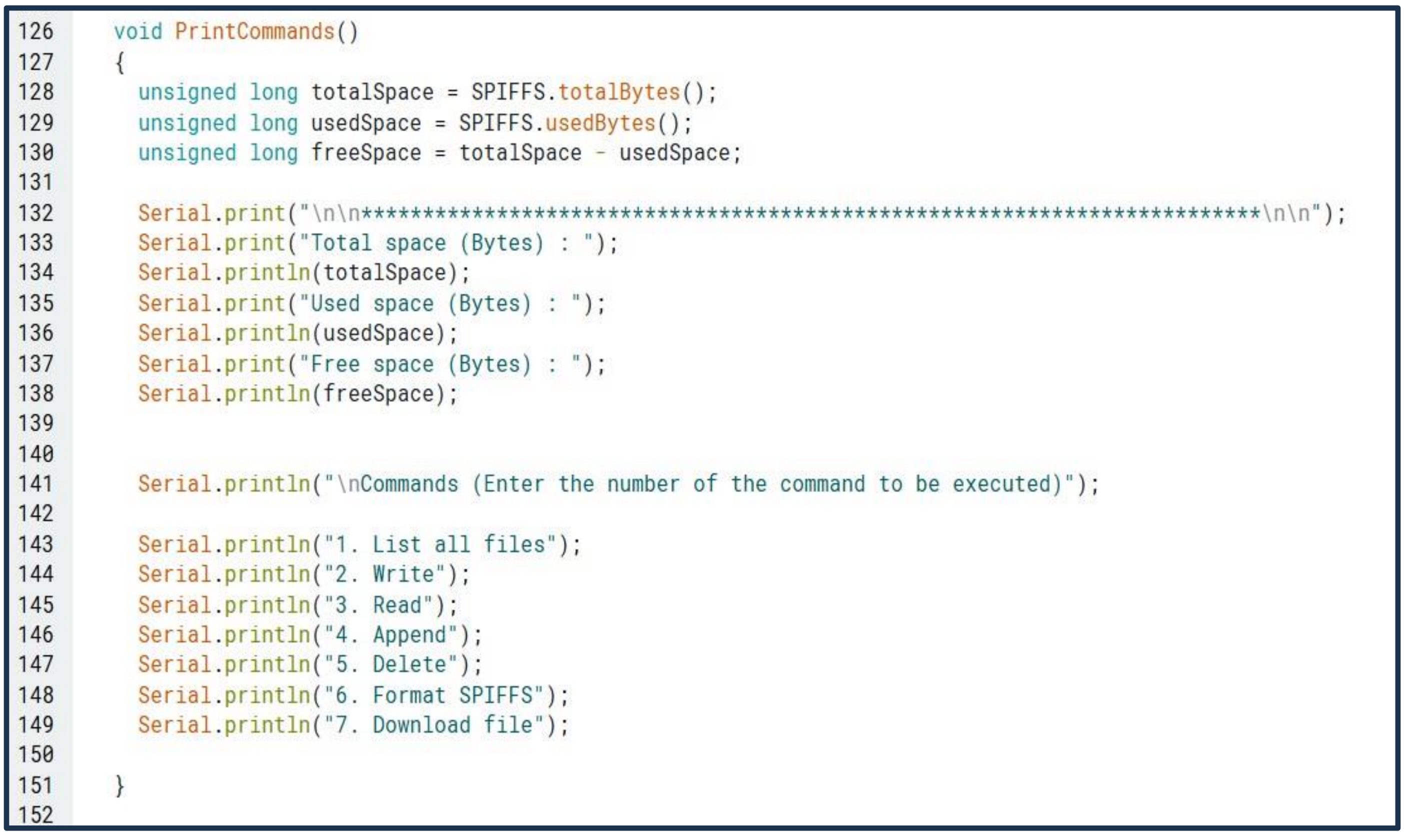
Checks if the input is a number and then converts the string into integer.

Wait for the user to enter the filename and add a ‘/’ to the filename for storing all the files in root directory based on the choice of the user the operation is performed



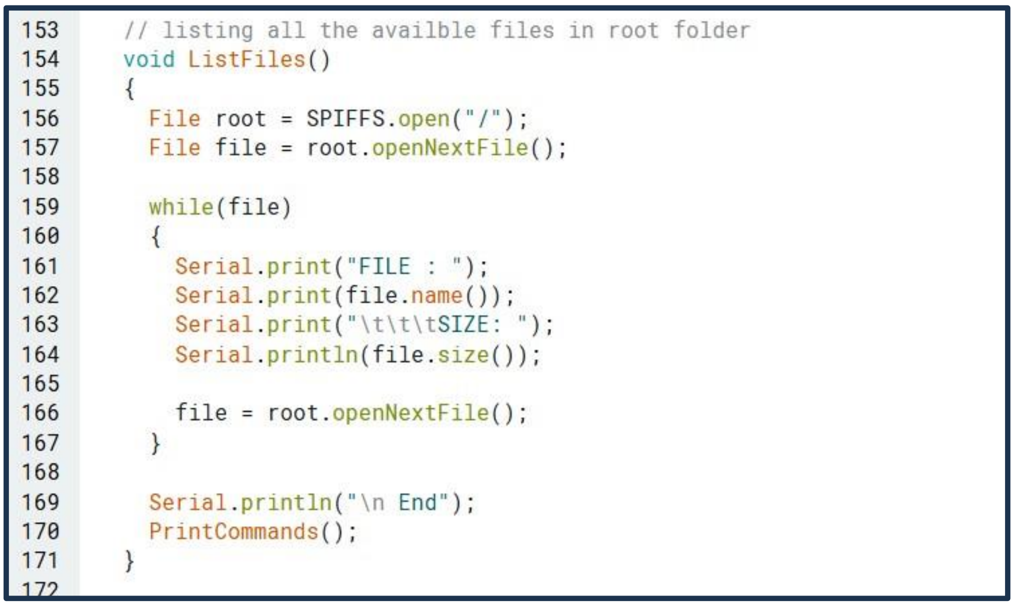
```
  // check if string is a number or not
  bool isNum(String input)
  {
    for(uint8_t i = 0; i < input.length(); i++)
    {
      if(!isdigit(input.charAt(i)))
        return false;
    }

    return true;
  }
```

This function checks the user input is a number or not, If yes it returns true, if not it returns false



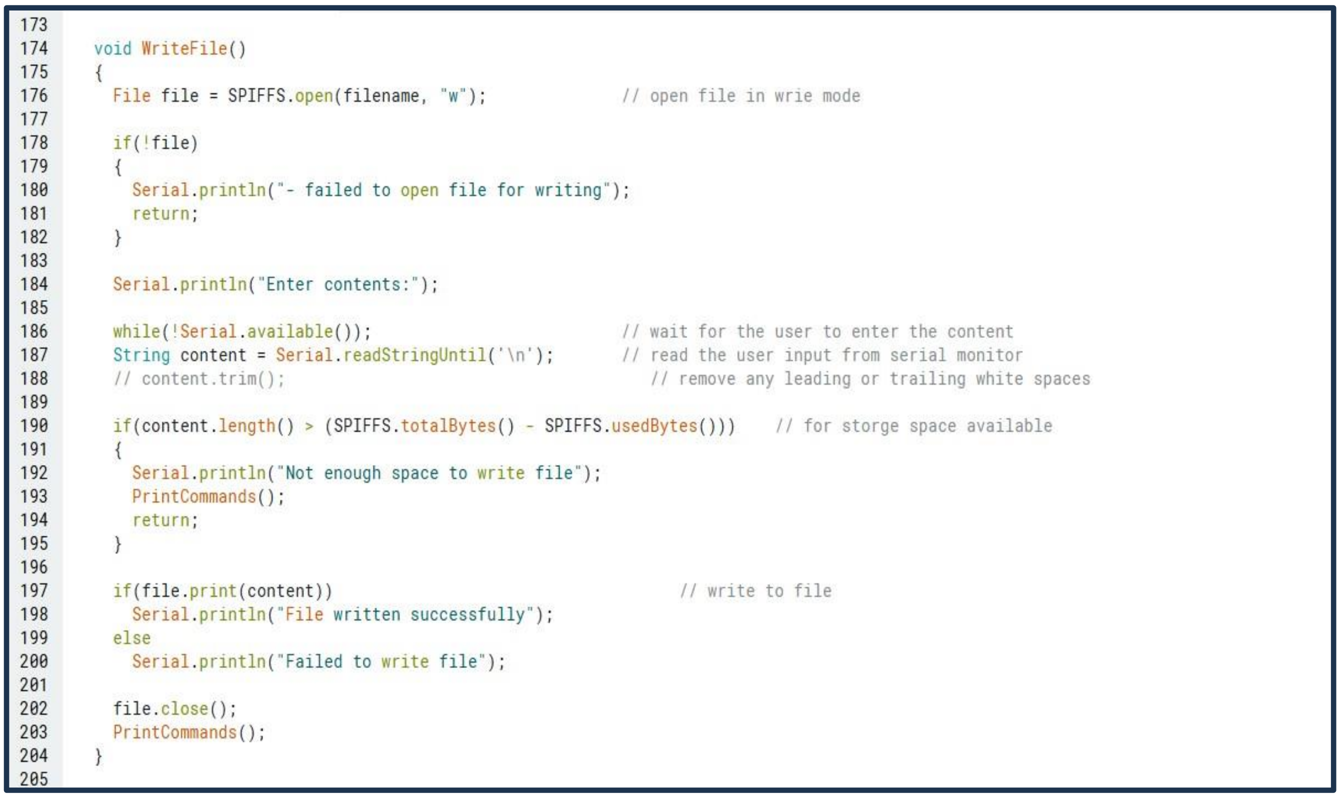
```
void PrintCommands()
{
  unsigned long totalSpace = SPIFFS.totalBytes();
  unsigned long usedSpace = SPIFFS.usedBytes();
  unsigned long freeSpace = totalSpace - usedSpace;

  Serial.print("\n\n**************************************************************************\n\n");
  Serial.print("Total space (Bytes) : ");
  Serial.println(totalSpace);
  Serial.print("Used space (Bytes) : ");
  Serial.println(usedSpace);
  Serial.print("Free space (Bytes) : ");
  Serial.println(freeSpace);


  Serial.println("\nCommands (Enter the number of the command to be executed)");

  Serial.println("1. List all files");
  Serial.println("2. Write");
  Serial.println("3. Read");
  Serial.println("4. Append");
  Serial.println("5. Delete");
  Serial.println("6. Format SPIFFS");
  Serial.println("7. Download file");

}
```

Prints the commands and available space in SPIFFS on the serial monitor



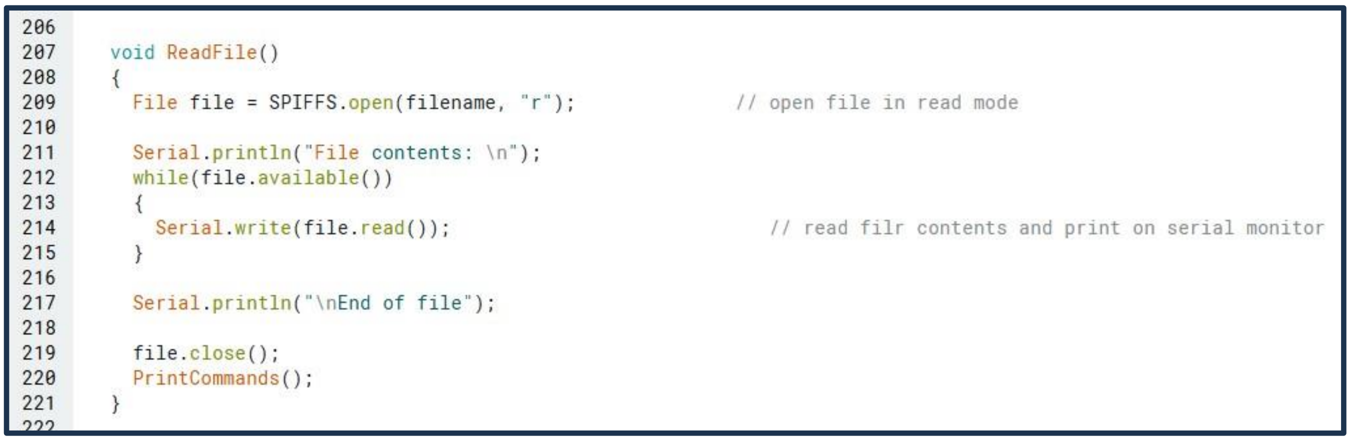
```
// listing all the availble files in root folder
void ListFiles()
{
  File root = SPIFFS.open("/");
  File file = root.openNextFile();

  while(file)
  {
    Serial.print("FILE : ");
    Serial.print(file.name());
    Serial.print("\t\t\tSIZE: ");
    Serial.println(file.size());

    file = root.openNextFile();
  }

  Serial.println("\n End");
  PrintCommands();
}
```

Lists all the files in the root directory



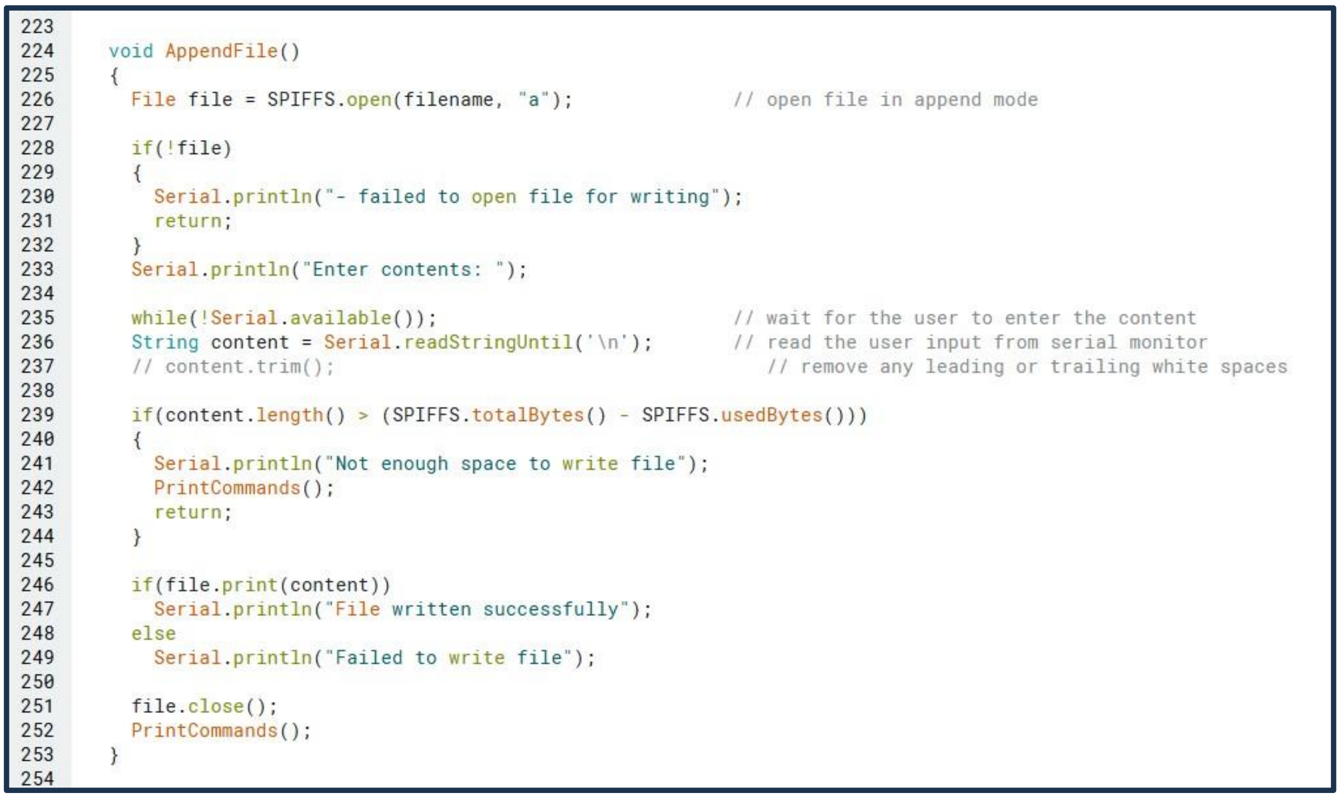
```

void WriteFile()
{
  File file = SPIFFS.open(filename, "w");              // open file in wrie mode

  if(!file)
  {
    Serial.println("- failed to open file for writing");
    return;
  }

  Serial.println("Enter contents:");

  while(!Serial.available());                          // wait for the user to enter the content
  String content = Serial.readStringUntil('\n');       // read the user input from serial monitor
  // content.trim();                                      // remove any leading or trailing white spaces

  if(content.length() > (SPIFFS.totalBytes() - SPIFFS.usedBytes()))    // for storge space available
  {
    Serial.println("Not enough space to write file");
    PrintCommands();
    return;
  }

  if(file.print(content))                                  // write to file
    Serial.println("File written successfully");
  else
    Serial.println("Failed to write file");

  file.close();
  PrintCommands();
}
```

Opens a file in write mode (creates it if not found). Takes input from user in the serial monitor, checks the size available in SPJFFS, if available writes the contents into the file, if not available returns without writing.



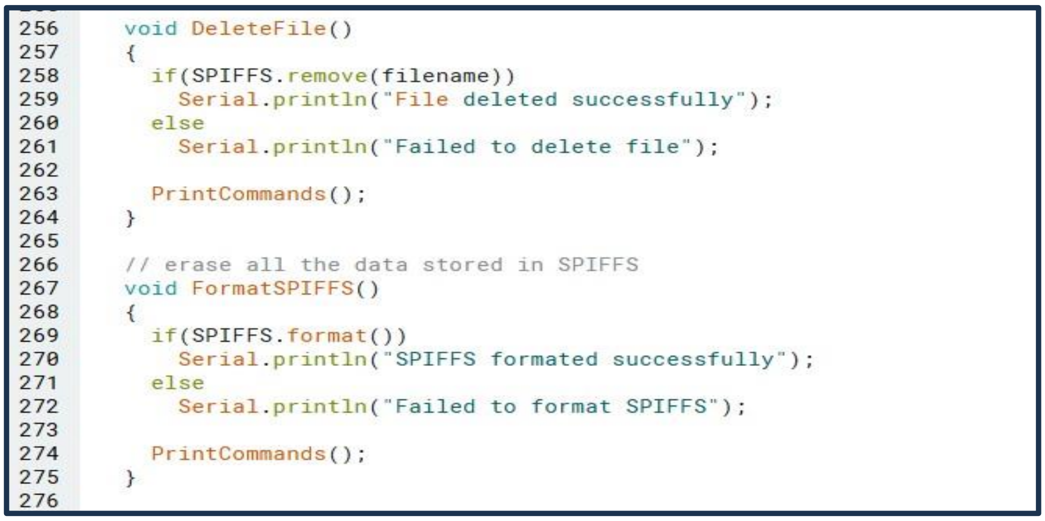
```

void ReadFile()
{
  File file = SPIFFS.open(filename, "r");              // open file in read mode

  Serial.println("File contents: \n");
  while(file.available())
  {
    Serial.write(file.read());                           // read filr contents and print on serial monitor
  }

  Serial.println("\nEnd of file");

  file.close();
  PrintCommands();
}
```

Opens the file in read mode and prints the contents of the file in the serial monitor



```

void AppendFile()
{
  File file = SPIFFS.open(filename, "a");             // open file in append mode

  if(!file)
  {
    Serial.println("- failed to open file for writing");
    return;
  }
  Serial.println("Enter contents: ");

  while(!Serial.available());                         // wait for the user to enter the content
  String content = Serial.readStringUntil('\n');      // read the user input from serial monitor
  // content.trim();                                     // remove any leading or trailing white spaces

  if(content.length() > (SPIFFS.totalBytes() - SPIFFS.usedBytes()))
  {
    Serial.println("Not enough space to write file");
    PrintCommands();
    return;
  }

  if(file.print(content))
    Serial.println("File written successfully");
  else
    Serial.println("Failed to write file");

  file.close();
  PrintCommands();
}

```

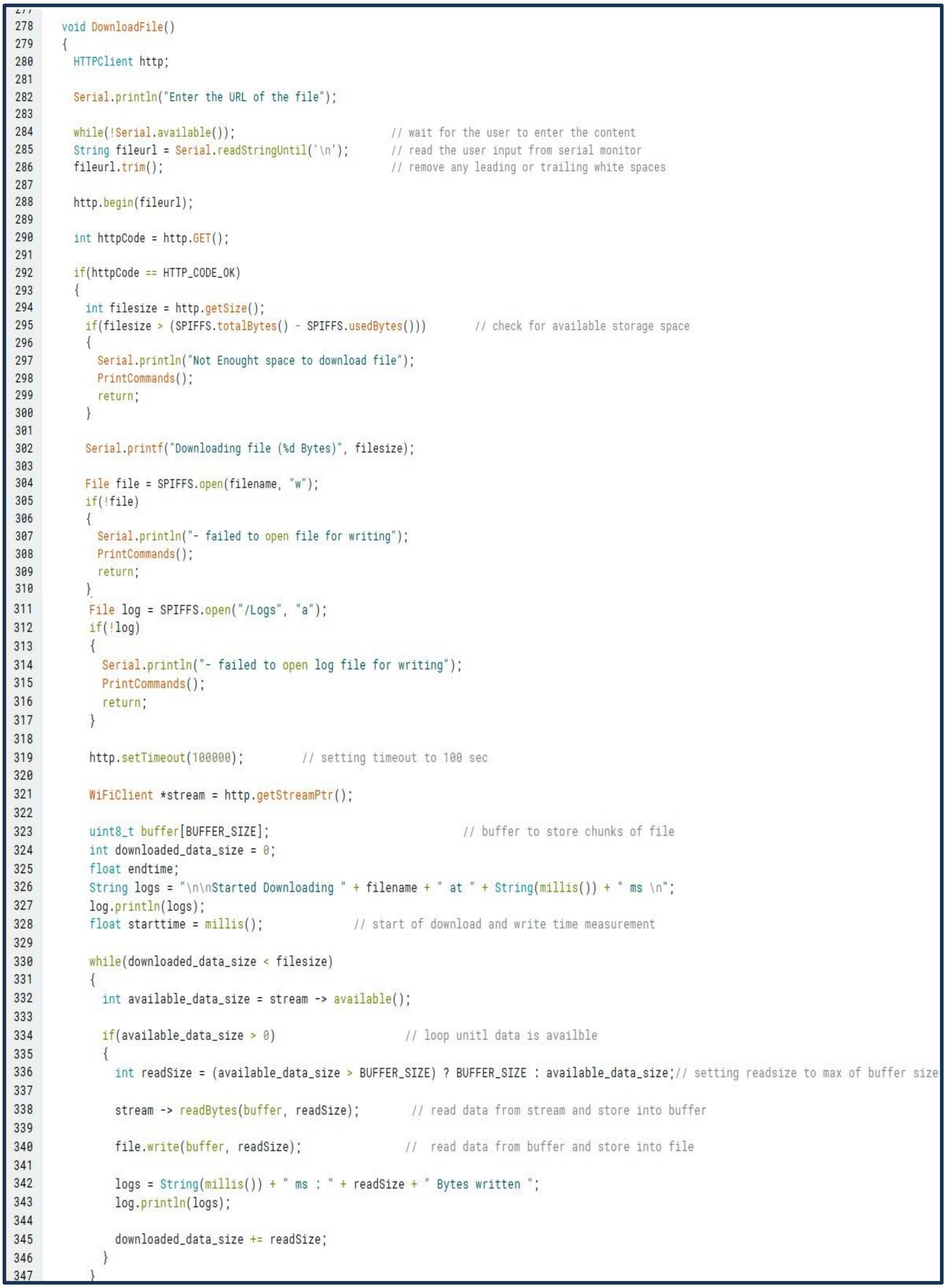
Opens a file in append mode (creates it if not found). Takes input from user in the serial monitor, checks the size available in SPJFFS, if available writes the contents into the file, if not available returns without writing.



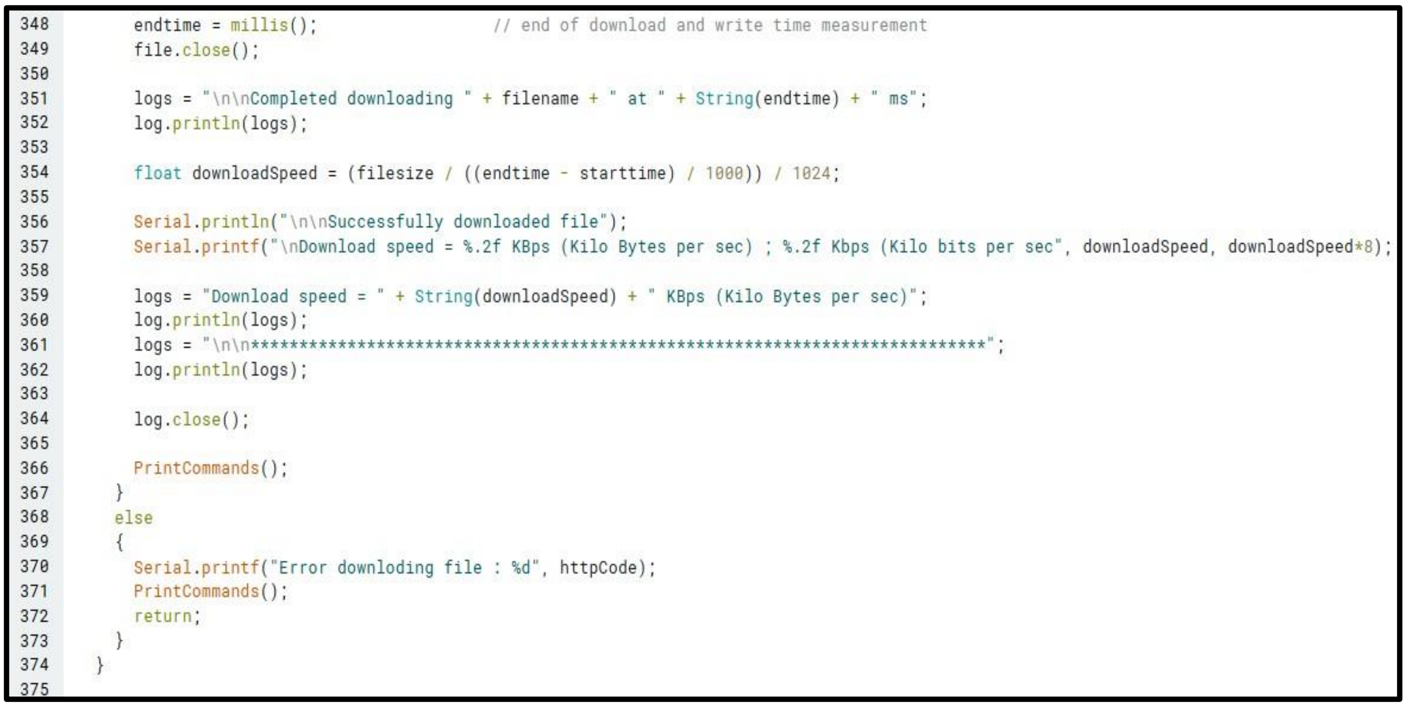
```
void DeleteFile()
{
  if(SPIFFS.remove(filename))
    Serial.println("File deleted successfully");
  else
    Serial.println("Failed to delete file");

  PrintCommands();
}

// erase all the data stored in SPIFFS
void FormatSPIFFS()
{
  if(SPIFFS.format())
    Serial.println("SPIFFS formated successfully");
  else
    Serial.println("Failed to format SPIFFS");

  PrintCommands();
}

```

Deletes a particular file whose name is entered by the user

FormatSPIFFS() erases all the data in file system and it like new completely blank.



```
void DownloadFile()
{
  HTTPClient http;

  Serial.println("Enter the URL of the file");

  while(!Serial.available());                           // wait for the user to enter the content
  String fileurl = Serial.readStringUntil('\n');        // read the user input from serial monitor
  fileurl.trim();                                       // remove any leading or trailing white spaces

  http.begin(fileurl);

  int httpCode = http.GET();

  if(httpCode == HTTP_CODE_OK)
  {
    int filesize = http.getSize();
    if(filesize > (SPIFFS.totalBytes() - SPIFFS.usedBytes()))        // check for available storage space
    {
      Serial.println("Not Enought space to download file");
      PrintCommands();
      return;
    }

    Serial.printf("Downloading file (%d Bytes)", filesize);

    File file = SPIFFS.open(filename, "w");
    if(!file)
    {
      Serial.println("- failed to open file for writing");
      PrintCommands();
      return;
    }
    File log = SPIFFS.open("/Logs", "a");
    if(!log)
    {
      Serial.println("- failed to open log file for writing");
      PrintCommands();
      return;
    }

    http.setTimeout(100000);         // setting timeout to 100 sec

    WiFiClient *stream = http.getStreamPtr();

    uint8_t buffer[BUFFER_SIZE];                                 // buffer to store chunks of file
    int downloaded_data_size = 0;
    float endtime;
    String logs = "\n\nStarted Downloading " + filename + " at " + String(millis()) + " ms \n";
    log.println(logs);
    float starttime = millis();              // start of download and write time measurement

    while(downloaded_data_size < filesize)
    {
      int available_data_size = stream -> available();

      if(available_data_size > 0)                      // loop unitl data is availble
      {
        int readSize = (available_data_size > BUFFER_SIZE) ? BUFFER_SIZE : available_data_size;// setting readsize to max of buffer size

        stream -> readBytes(buffer, readSize);        // read data from stream and store into buffer

        file.write(buffer, readSize);                // read data from buffer and store into file

        logs = String(millis()) + " ms : " + readSize + " Bytes written ";
        log.println(logs);

        downloaded_data_size += readSize;
      }
    }
```



```
        endtime = millis();                  // end of download and write time measurement
        file.close();

        logs = "\n\nCompleted downloading " + filename + " at " + String(endtime) + " ms";
        log.println(logs);

        float downloadSpeed = (filesize / ((endtime - starttime) / 1000)) / 1024;

        Serial.println("\n\nSuccessfully downloaded file");
        Serial.printf("\nDownload speed = %.2f KBps (Kilo Bytes per sec) ; %.2f Kbps (Kilo bits per sec", downloadSpeed, downloadSpeed*8);

        logs = "Download speed = " + String(downloadSpeed) + " KBps (Kilo Bytes per sec)";
        log.println(logs);
        logs = "\n\n****************************************************************************";
        log.println(logs);

        log.close();

        PrintCommands();
      }
      else
      {
        Serial.printf("Error downloding file : %d", httpCode);
        PrintCommands();
        return;
      }
    }
```

Create  an object for HTTPClient.

Waits for the user to enter the public URL from which the file is to be downloaded.

Connects to public URL and makes a GET request to download the file, if the http response is 200 it gets the size of the file and checks the space available in SPIFFS.

Creates a file with the name specified by the user and also a log file to log the download data and download speed.

Set a timeout of 100sec.

Define a pointer of WiFiClient to keep track of available data. Create a buffer to read and write data to file.

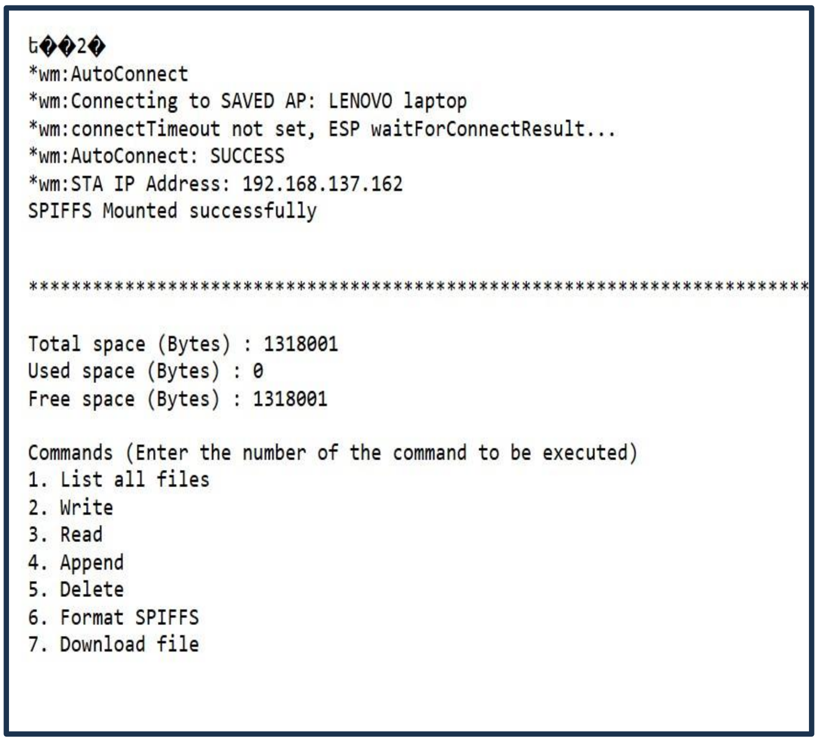
Read the data and write it to file and also log the data into Log file and at the end the download speed Is printed on the serial monitor as well as logged in LOGS file.

## Flashing

1. Connect the ESP32 to PC
2. Open Arduino IDE and setup the IDE for ESP board.
3. Click Upload in the upper left corner and wait for the code to get uploaded.

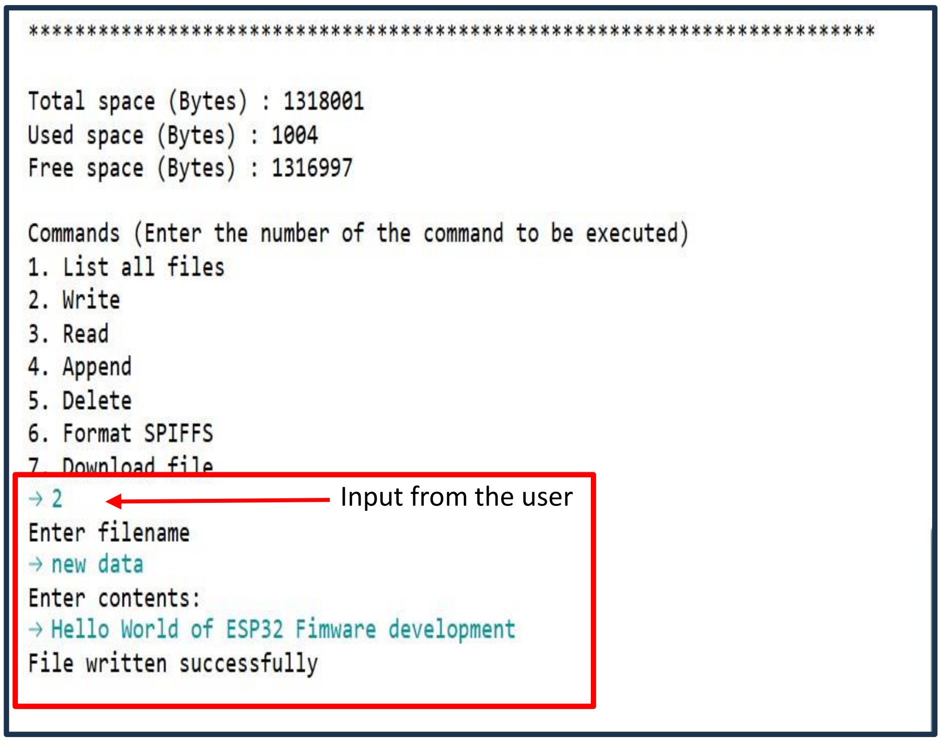
## Output

Click on the serial monitor in the upper right corner. Push the EN button on the ESP32 and you should see a similar output on the serial monitor.



```
t��2�
*wm:AutoConnect
*wm:Connecting to SAVED AP: LENOVO laptop
*wm:connectTimeout not set, ESP waitForConnectResult...
*wm:AutoConnect: SUCCESS
*wm:STA IP Address: 192.168.137.162
SPIFFS Mounted successfully


*************************************************************************

Total space (Bytes) : 1318001
Used space (Bytes) : 0
Free space (Bytes) : 1318001

Commands (Enter the number of the command to be executed)
1. List all files
2. Write
3. Read
4. Append
5. Delete
6. Format SPIFFS
7. Download file
```

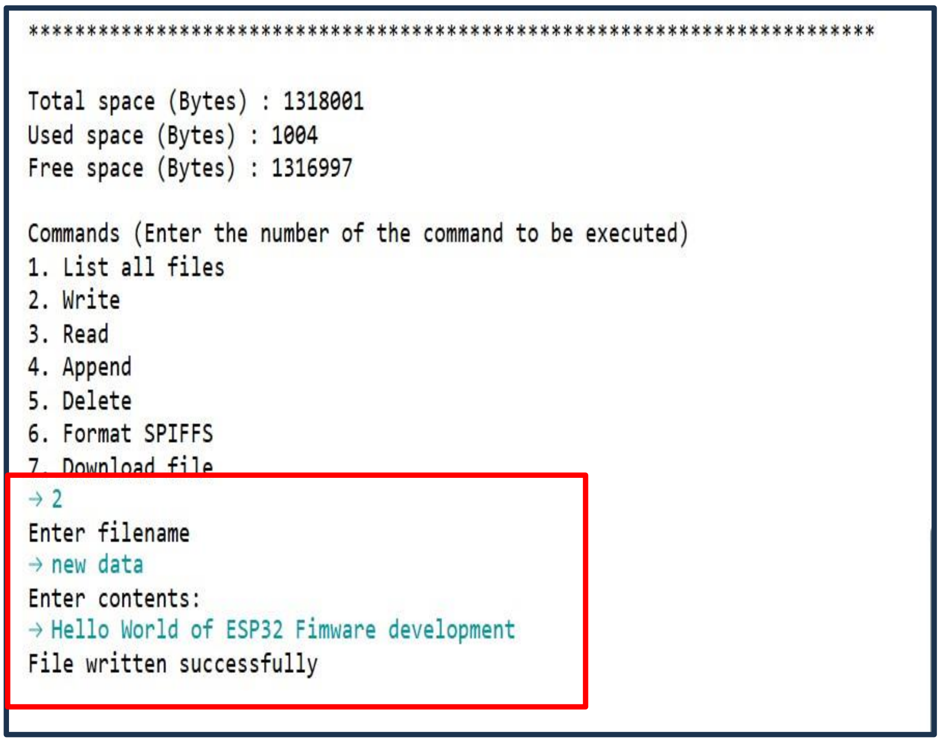
Enter the number of the command to be executed. Enter the file name and follow the onscreen instructions if any.



```
*******************************************************************************


Total space (Bytes) : 1318001
Used space (Bytes) : 1004
Free space (Bytes) : 1316997

Commands (Enter the number of the command to be executed)
1. List all files
2. Write
3. Read
4. Append
5. Delete
6. Format SPIFFS
7. Download file
→ 2
Enter filename
→ new data
Enter contents:
→ Hello World of ESP32 Fimware development
File written successfully
```

**Performing operations:**

**1. Wrtie**



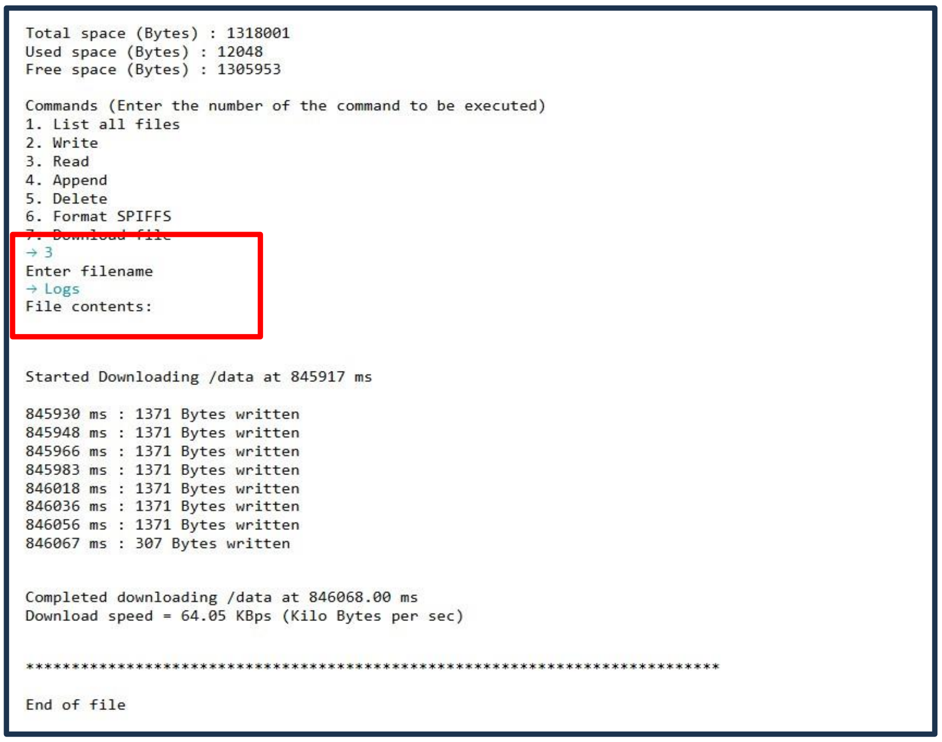
```
*************************************************************************

Total space (Bytes) : 1318001
Used space (Bytes) : 1004
Free space (Bytes) : 1316997

Commands (Enter the number of the command to be executed)
1. List all files
2. Write
3. Read
4. Append
5. Delete
6. Format SPIFFS
7. Download file
→ 2
Enter filename
→ new data
Enter contents:
→ Hello World of ESP32 Fimware development
File written successfully
```

Enter the number of the command ‘2’ in the serial monitor.

Enter the filename you want to write/create.

Enter the contents you want to write to a file, and press enter.

**2.Read**

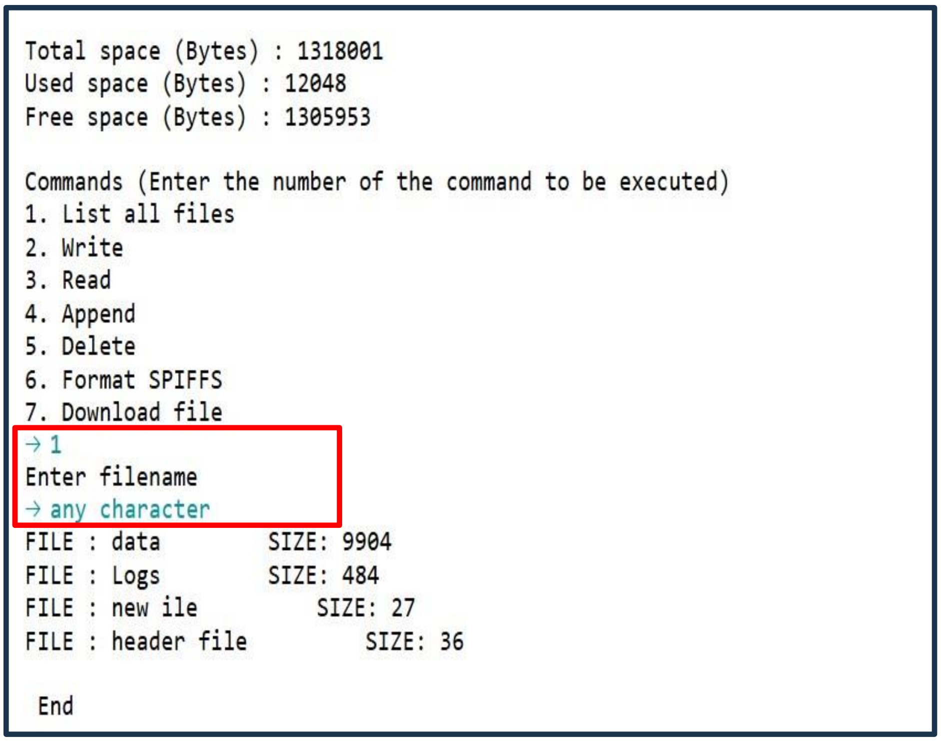


```
Total space (Bytes) : 1318001
Used space (Bytes) : 12048
Free space (Bytes) : 1305953

Commands (Enter the number of the command to be executed)
1. List all files
2. Write
3. Read
4. Append
5. Delete
6. Format SPIFFS
7. Download file
→ 3
Enter filename
→ Logs
File contents:


Started Downloading /data at 845917 ms

845930 ms : 1371 Bytes written
845948 ms : 1371 Bytes written
845966 ms : 1371 Bytes written
845983 ms : 1371 Bytes written
846018 ms : 1371 Bytes written
846036 ms : 1371 Bytes written
846056 ms : 1371 Bytes written
846067 ms : 307 Bytes written


Completed downloading /data at 846068.00 ms
Download speed = 64.05 KBps (Kilo Bytes per sec)

*****************************************************************************

End of file
```

Enter the command number ‘3’ in the serial monitor.

Enter the filename you want to read, the contents of the file will be printed onto the serial monitor.

**3.List Files**

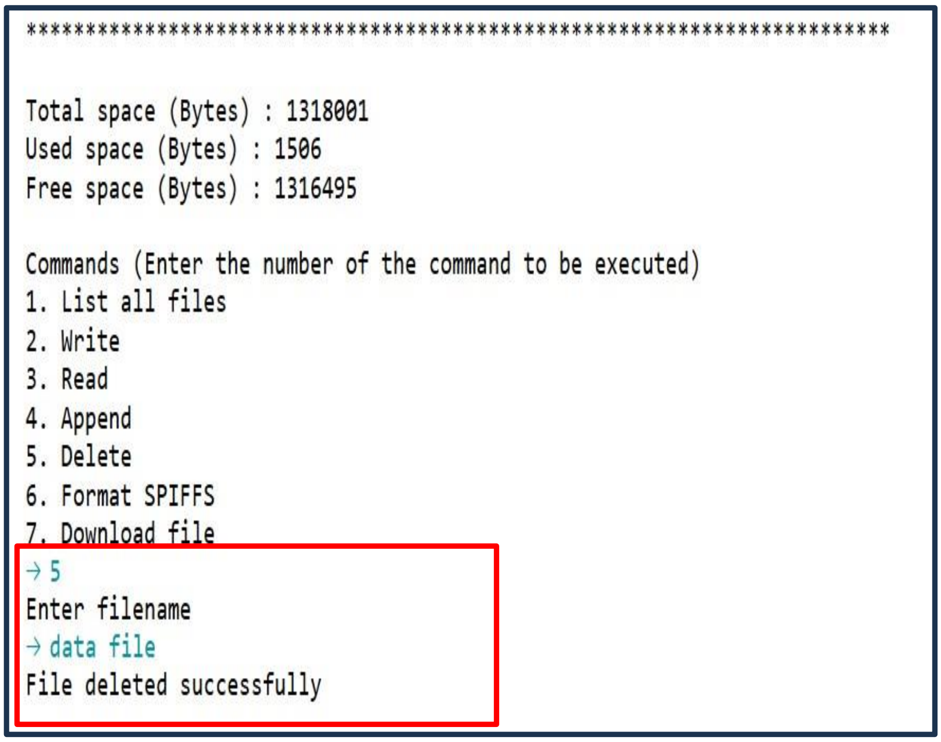


```
Total space (Bytes) : 1318001
Used space (Bytes) : 12048
Free space (Bytes) : 1305953

Commands (Enter the number of the command to be executed)
1. List all files
2. Write
3. Read
4. Append
5. Delete
6. Format SPIFFS
7. Download file
→ 1
Enter filename
→ any character
FILE : data         SIZE: 9904
FILE : Logs         SIZE: 484
FILE : new ile          SIZE: 27
FILE : header file          SIZE: 36

 End
```

Enter the number of the command ‘1’.

Enter any character and all the files with their size in bytes will be listed on the serial monitor.

**4.Delete File**

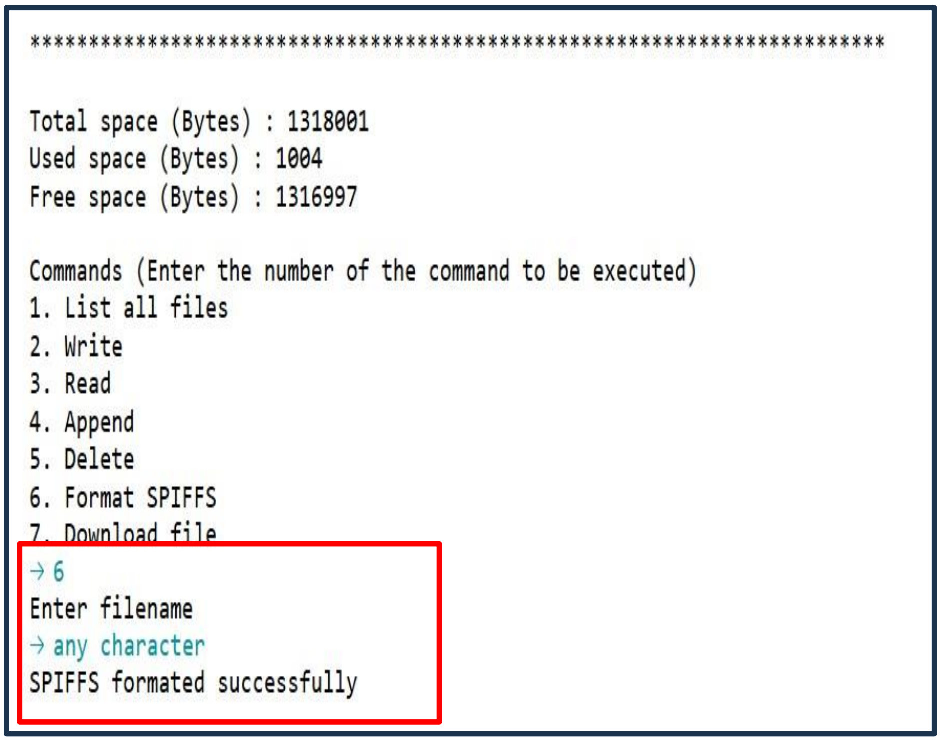


```
************************************************************************

Total space (Bytes) : 1318001
Used space (Bytes) : 1506
Free space (Bytes) : 1316495

Commands (Enter the number of the command to be executed)
1. List all files
2. Write
3. Read
4. Append
5. Delete
6. Format SPIFFS
7. Download file
→ 5
Enter filename
→ data file
File deleted successfully
```

Enter the number of the command ‘5’.

Enter the name of the file to deleted.

**5.Format SPIFFS**

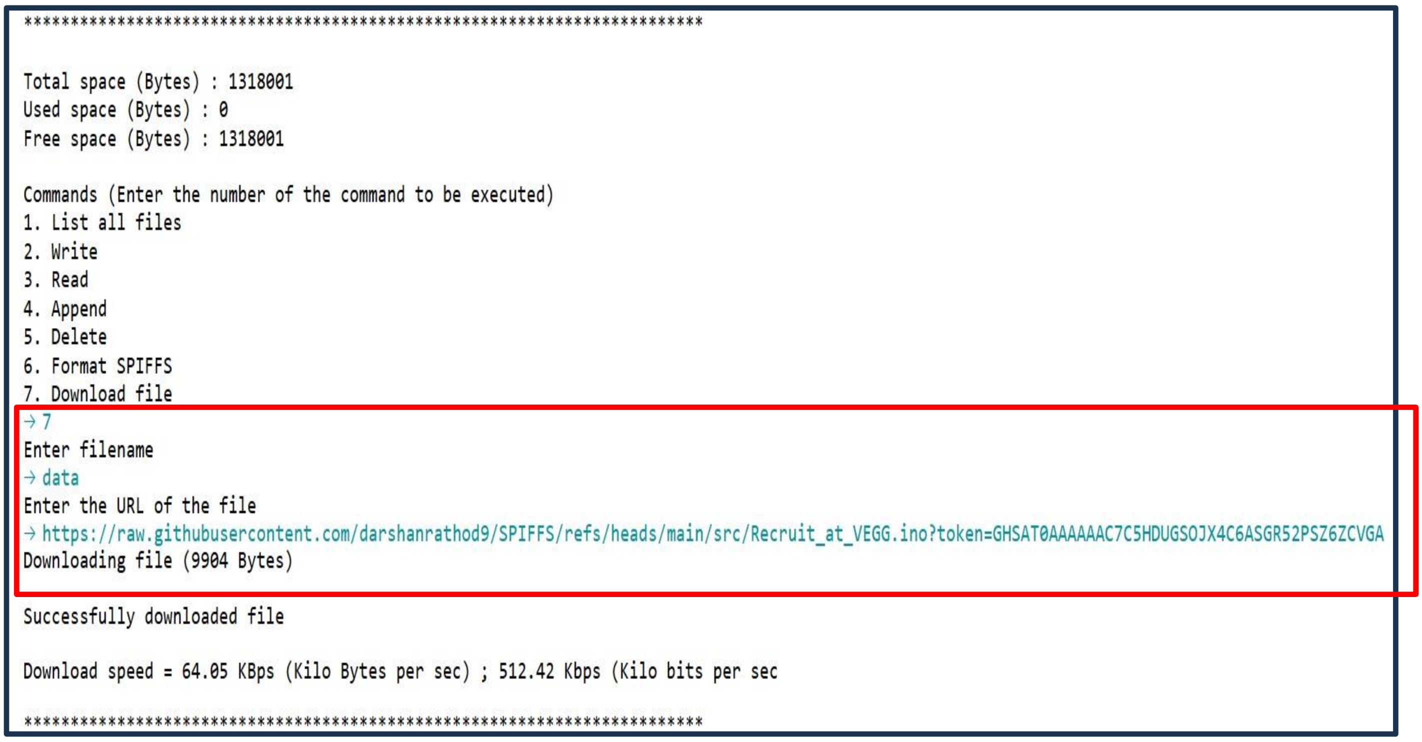


```
*************************************************************************

Total space (Bytes) : 1318001
Used space (Bytes) : 1004
Free space (Bytes) : 1316997

Commands (Enter the number of the command to be executed)
1. List all files
2. Write
3. Read
4. Append
5. Delete
6. Format SPIFFS
7. Download file
→ 6
Enter filename
→ any character
SPIFFS formated successfully
```

Enter the number of the command ‘6’.

Enter any character and the SPIFFS will be formatted

**6.Download File**



```
*************************************************************************

Total space (Bytes) : 1318001
Used space (Bytes) : 0
Free space (Bytes) : 1318001

Commands (Enter the number of the command to be executed)
1. List all files
2. Write
3. Read
4. Append
5. Delete
6. Format SPIFFS
7. Download file
→ 7
Enter filename
→ data
Enter the URL of the file
→ https://raw.githubusercontent.com/darshanrathod9/SPIFFS/refs/heads/main/src/Recruit_at_VEGG.ino?token=GHSAT0AAAAAAC7C5HDUGSOJX4C6ASGR52PSZ6ZCVGA
Downloading file (9904 Bytes)

Successfully downloaded file

Download speed = 64.05 KBps (Kilo Bytes per sec) ; 512.42 Kbps (Kilo bits per sec

*************************************************************************
```

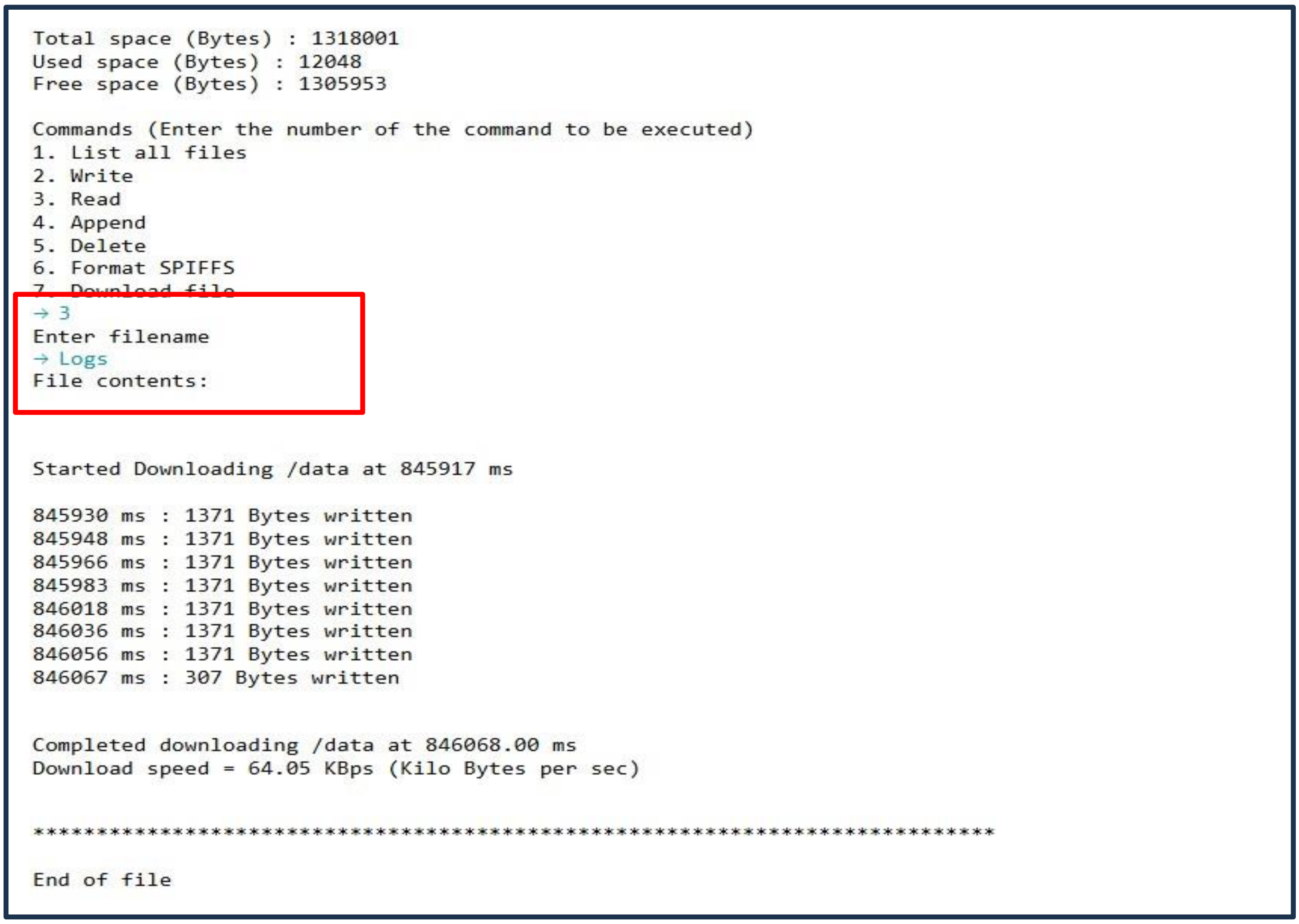
Enter the number of the command ‘7’.

Enter the name of the file in which the data will be written

Enter the public URL from where the file will be downloaded.

After downloading the file the download speed will be printed on serial monitor and logged in Logs file.

## 7.Read logs



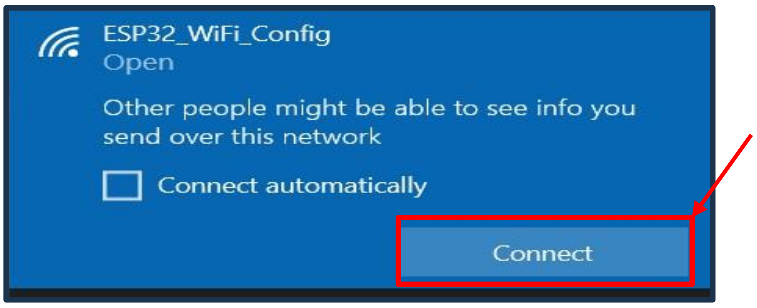
```
Total space (Bytes) : 1318001
Used space (Bytes) : 12048
Free space (Bytes) : 1305953

Commands (Enter the number of the command to be executed)
1. List all files
2. Write
3. Read
4. Append
5. Delete
6. Format SPIFFS
7. Download file
→ 3
Enter filename
→ Logs
File contents:


Started Downloading /data at 845917 ms

845930 ms : 1371 Bytes written
845948 ms : 1371 Bytes written
845966 ms : 1371 Bytes written
845983 ms : 1371 Bytes written
846018 ms : 1371 Bytes written
846036 ms : 1371 Bytes written
846056 ms : 1371 Bytes written
846067 ms : 307 Bytes written


Completed downloading /data at 846068.00 ms
Download speed = 64.05 KBps (Kilo Bytes per sec)


****************************************************************************

End of file
```

Enter the command number ‘3’ in the serial monitor.

Enter “Logs” in serial monitor, the contents of the file will be printed onto the serial monitor.

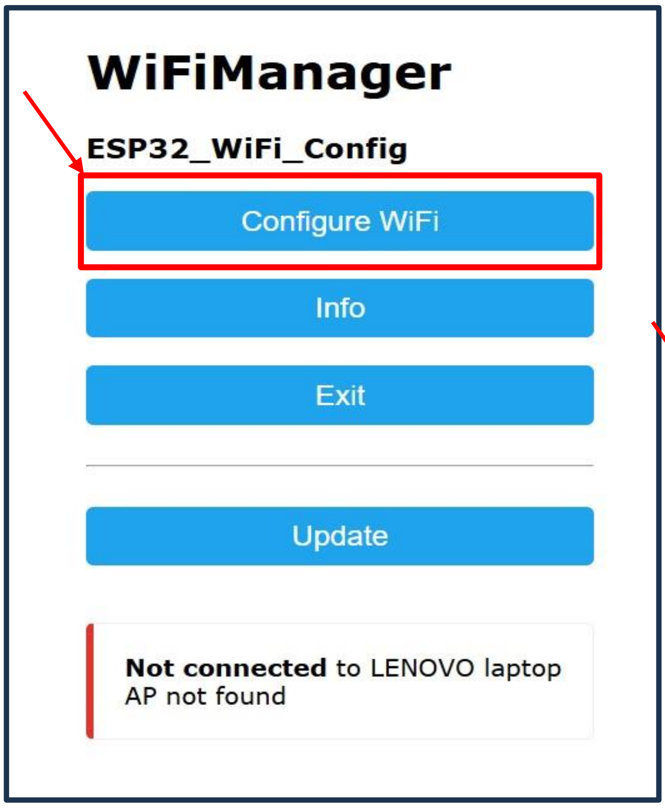
## WiFi Configuration via WiFi manager library

Connect to the hotspot named “ESP32_WiFi_Config”



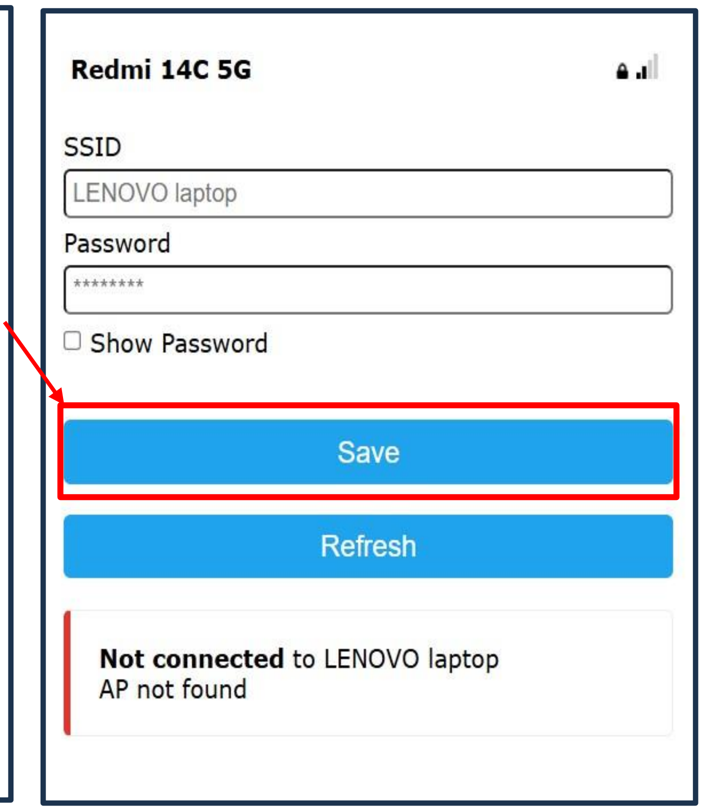
ESP32_WiFi_Config
Open
Other people might be able to see info you send over this network

The wifi config window will open automatically, if it does not open enter “192.168.4.1” in the browser.



WiFiManager
ESP32_WiFi_Config

**Not connected** to LENOVO laptop
AP not found



Redmi 14C 5G
SSID: LENOVO laptop
Password: ********

**Not connected** to LENOVO laptop
AP not found

Click on “Configure WiFi” and then enter the wifi credentials and save it.