II

THE BASICS

4

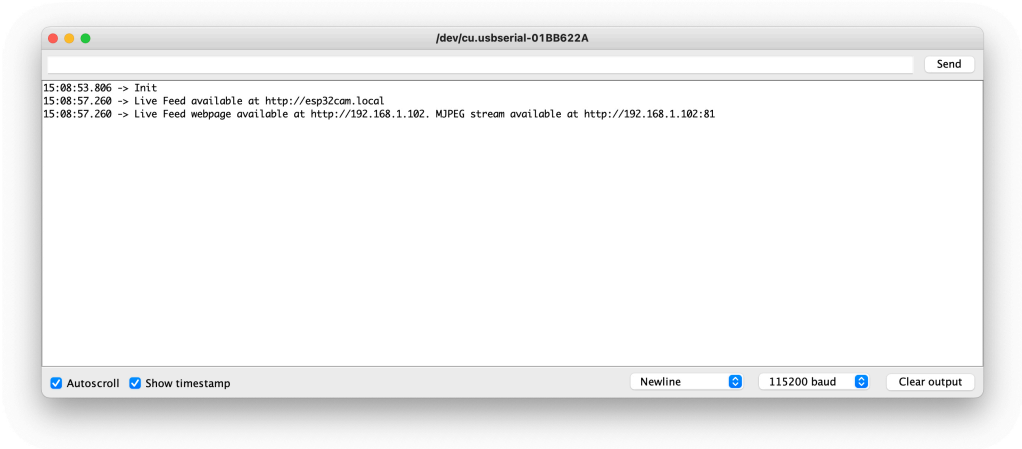
Live Video Streaming

Being able to take pictures is a thing. Viewing a real-time video feed of the camera from your browser is another thing.

This is kind of an "Hello World" project for the Esp32-cam on the web. We'll take on this tradition, but add the "Eloquent" spin to it!

```
#include "esp32cam.h"
#include "esp32cam/http/LiveFeed.h"

// Replace with your WiFi credentials
#define WIFI_SSID "Your SSID"
#define WIFI_PASS "Your password"

// 80 is the port to listen to
// You can change it to whatever you want, 80 is the default
for HTTP
Eloquent::Esp32cam::Cam cam;
Eloquent::Esp32cam::Http::LiveFeed feed(cam, 80);

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

```
    delay(3000);
    Serial.println("Init");

    // see 3_Get_Your_First_Picture for more details
    cam.aithinker();
    cam.highQuality();
    cam.qvga();

    while (!cam.begin())
        Serial.println(cam.getErrorMessage());

    // Connect to WiFi
    // If something goes wrong, print the error message
    while (!cam.connect(WIFI_SSID, WIFI_PASS))
        Serial.println(cam.getErrorMessage());

    // Initialize live feed http server
    // If something goes wrong, print the error message
    while (!feed.begin())
        Serial.println(feed.getErrorMessage());

    // make the camera accessible at http://esp32cam.local
    // (your router must support mDNS)
    if (!cam.viewAt("esp32cam"))
        Serial.println("Cannot create alias, use the IP
        address");
    else
        Serial.println("Live Feed available at
        http://esp32cam.local");

    // display the IP address of the camera
    Serial.println(feed.getWelcomeMessage());
}

void loop() {
}
```



Live Video Streaming welcome message

Connect your PC to the same network of the Esp32-cam and open the browser at the given address. You should see the live video feed of the camera.

```
If you set a resolution higher than QVGA (320 x 240), be sure
your WiFi signal is strong, otherwise the feed will look
laggish.
```

Use this video feed to set the correct position of the camera for your current project or to debug the settings configuration you chose in Chapter 2.

5

Save Photos to SD Card

```
Hardware requirements:
 - AiThinker camera (with SD slot)
 - or External SD card reader
If you miss these requirements, you will still be able to
save pictures to the internal SPIFFS filesystem
```

After the Live Video Streaming sketch, saving pictures to the SD card is the second most asked use case for the Esp32-cam.

This is because the AiThinker model comes equipped with an SD slot, so it's really easy to get started. If you have a different model, you will save pictures to the internal SPIFFS filesystem of the Esp32.

Keep in mind, though, that this filesystem has a little capacity (usually 4 Mbit) and is shared with the program, so you won't be able to store hundreds of images on it.

The following sketch will await for you to input "capture" in the Serial Monitor. Every time you enter that text, it will take a picture and save it to the SD / SPIFFS.

```
// 5_Save_to_SD_MMC

#include <FS.h>
#include <SD_MMC.h>
#include <SPI.h>
#include "esp32cam.h"


Eloquent::Esp32cam::Cam cam;
uint32_t counter = 1;

void setup() {
    Serial.begin(115200);
    delay(3000);
    Serial.println("Init");

    cam.aithinker();
    cam.highQuality();
    cam.vga();

    while (!cam.begin())
        Serial.println(cam.getErrorMessage());

    // Initialize the filesystem
    // If something goes wrong, print an error message
    while (!SD_MMC.begin() || SD_MMC.cardType() == CARD_NONE)
        Serial.println("Cannot init SD Card");

    Serial.println("Enter 'capture' in the Serial Monitor");
}


void loop() {
    if (!Serial.available())
        return;

    if (Serial.readStringUntil('\n') != "capture")
```

```
        return;

    if (!cam.capture()) {
        Serial.println(cam.getErrorMessage());
        return;
    }

    String filename = String("/picture_") + counter + ".jpg";

    if (cam.saveTo(SD_MMC, filename)) {
        Serial.println(filename + " saved to disk");
        counter += 1;
    }
    else {
        Serial.println(cam.getErrorMessage());
    }
}
```

```
// 6_Save_To_SPIFFS

#include <FS.h>
#include <SPIFFS.h>
#include "esp32cam.h"

Eloquent::Esp32cam::Cam cam;
uint32_t counter = 1;

void setup() {
    Serial.begin(115200);
    delay(3000);
    Serial.println("Init");

    cam.aithinker();
    cam.highQuality();
```

```
    cam.vga();

    while (!cam.begin())
        Serial.println(cam.getErrorMessage());

    // Initialize the filesystem
    //If something goes wrong, print an error message
    while (!SPIFFS.begin(true))
        Serial.println("Cannot init SD Card");

    Serial.println("Enter 'capture' in the Serial Monitor");
}

void loop() {
    if (!Serial.available())
        return;

    if (Serial.readStringUntil('\n') != "capture")
        return;

    if (!cam.capture()) {
        Serial.println(cam.getErrorMessage());
        return;
    }

    String filename = String("/picture_") + counter + ".jpg";

    if (cam.saveTo(SPIFFS, filename)) {
        Serial.println(filename + " saved to disk");
        counter += 1;
    }
    else {
        Serial.println(cam.getErrorMessage());
    }
}
```

Every time you capture a picture, it will be saved with an incremental name, like “picture_1.jpg”, “picture_2.jpg” and so on.

Now this is the most basic naming scheme possible, but will fall short as soon as you reboot the board since it will reset the counter and overwrite existing files.

Timestamped pictures

What can be more useful, if you have access to internet, is to name the pictures **based on the current time**, something like "picture_2022-11-11_19:47:00.jpg". This way

1. you don't overwrite existing pictures after reboot
2. you keep track of when the photo was captured

To do so, we need to fetch the current timestamp from an NTP (Network Time Protocol) server. You don't have to mess around, though. The "EloquentEsp32cam" library gives you a comfortable interface, as you can see next.

```
// 7_Save_To_SPIFFS_NTP

#include <FS.h>
#include <SPIFFS.h>
#include "esp32cam.h"
#include "esp32cam/NtpClient.h"

#define WIFI_SSID "Your SSID"
#define WIFI_PASS "Your password"

Eloquent::Esp32cam::Cam cam;
Eloquent::Esp32cam::NtpClient ntp;


void setup() {
```

```
    Serial.begin(115200);
    Serial.println("Init");

    cam.aithinker();
    cam.highQuality();
    cam.vga();

    while (!cam.begin())
        Serial.println(cam.getErrorMessage());

    while (!SPIFFS.begin(true))
        Serial.println("Cannot init SD Card");

    while (!cam.connect(WIFI_SSID, WIFI_PASS))
        Serial.println(cam.getErrorMessage());

    /**
     * Configure NTP server
     * gmt offset is in hours
     * if your timezone doesn't have daylight saving, omit
     the line
     * pool.ntp.org is the default server, so you can omit
     the line
     */
    ntp.gmt(1);
    ntp.daylight();
    ntp.server("pool.ntp.org");

    while (!ntp.begin())
      Serial.println("Cannot get NTP time");

    Serial.println("Enter 'capture' in the Serial Monitor");
}

void loop() {
    if (!Serial.available())
        return;
```

```
    if (Serial.readStringUntil('\n') != "capture")
        return;

    if (!cam.capture()) {
        Serial.println(cam.getErrorMessage());
        return;
    }

    String filename = ntp.getFilename();

    if (cam.saveTo(SPIFFS, filename)) {
        Serial.println(filename + " saved to disk");
    }
    else {
      Serial.print("Cannot save ");
      Serial.print(filename);
      Serial.print(": ");
      Serial.println(cam.getErrorMessage());
    }
}
```

The relevant parts in this new sketch are:

```
/**
 * Include and instantiate NTP client
 */
#include "esp32cam/NtpClient.h"

Eloquent::Esp32cam::NtpClient ntp;
```

```
/**
 * Configure offset, daylight saving (if any) and NTP server
```

```
 address
 */
ntp.gmt(1);
ntp.daylight();
ntp.server("pool.ntp.org");
```

```
/**
 * Generate a new filename in the form
 * /year-month-day_hour-minute-second.jpg
 */
String filename = ntp.getFilename();
```

You can customize the generated filename by passing your preferred formatting. You can find a list of the supported formats at https://www.tutorialspoint.com/c_standard_library/c_function_strftime.htm

As an example, if you want to save pictures named like "/Sunday, November 13 2022 14:41:00.jpg", you can use the following line

```
/**
 * First argument is the prefix
 * It MUST start with "/"
 */
String filename = ntp.getFilename("/", "%A, %B %d %Y
%H:%M:%S");
```

This is just an example: I advise to avoid spaces and punctuation characters in filenames (apart from upper and lower scores)!

6

Pictures Browser in the Browser

Now that we know how to store images on the SD card or the internal SPIFFS filesystem, we want to see them. If you have the SD card, you can remove it from the Esp32-cam and insert into your PC or smartphone to view the files.

But it is inefficient and boring.

It'd be better if we could access the files directly on the board from a browser, without touching the SD card.

The "EloquentEsp32cam" library has a component for this, very similar to the "LiveFeed" component: it's called "ImageBrowser". It works by starting an HTTP server on the Esp32-cam that is accessible from your browser: it displays the list of saved pictures along with control buttons to view, rename and delete those files.

```
#include <FS.h>
#include <SD_MMC.h>
#include <SPI.h>
#include "esp32cam.h"
#include "esp32cam/http/ImageBrowser.h"
```

```
// Replace with your WiFi credentials
#define WIFI_SSID "Your SSID"
#define WIFI_PASS "Your password"

// 80 is the port to listen to
// You can change it to whatever you want, 80 is the default
for HTTP
// If using SPIFFS, replace SD_MMC with SPIFFS
Eloquent::Esp32cam::Cam cam;
Eloquent::Esp32cam::Http::ImageBrowser browser(SD_MMC, 80);

void setup() {
    Serial.begin(115200);
    delay(3000);
    Serial.println("Init");

    cam.m5wide();
    cam.highQuality();
    cam.qvga();

    while (!cam.begin())
        Serial.println(cam.getErrorMessage());

    while (!SD_MMC.begin() || SD_MMC.cardType() == CARD_NONE)
        Serial.println("Cannot init SD Card");

    while (!cam.connect(WIFI_SSID, WIFI_PASS))
        Serial.println(cam.getErrorMessage());

    //Initialize image browser web server
    // If something goes wrong, print the error message
    while (!browser.begin())
        Serial.println(browser.getErrorMessage());

    // set max number of files displayed on a single page
    browser.setMaxNumFilesPerPage(30);
```

```
    cam.viewAt("esp32cam");

    // display the IP address of the server
    Serial.println(browser.getWelcomeMessage());
}


void loop() {
    browser.handle();
}
```

As you can see, the sketch mimics the one from the "Live Video Streaming": you have to instantiate the image browser web server, connect to WiFi and get its IP address (or use the esp32cam.local address, if supported).

Once done, open your web browser and you will see an interface like the following.

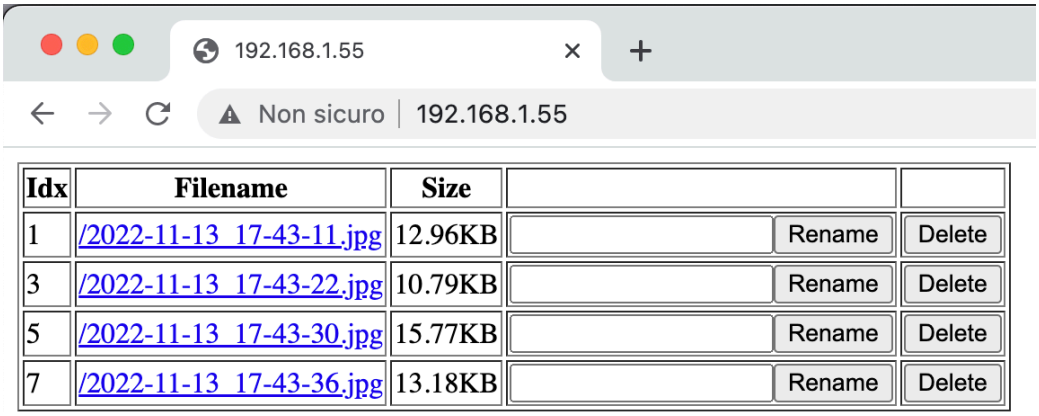


Image Browser example

You have a list of the images found on the filesystem, with the size an a couple actions available:

1. click on the name to see the image
2. enter a new name in the textbox and hit "Rename" to rename the image
3. click on the "Delete" button to delete the image

Now a few notes to be aware.

Note 1. The page may be slow to load

To generate the index table, we need to loop over all the files on the SD card, even if they're not images. **This is slow**. And we cannot do much about it.

Note 2. The files are not sorted

You may think that the files will be sorted by name or creation date. **False**. The files are not sorted in any meaningful way and we cannot do much about it.