ESP32 3D Printing

# ESP32-CAM Auto-Selfie Camera with TFT

Published: July 3, 2020 | 4:47 pm · Updated: July 6, 2020 | 4:06 pm



## A 3D printable camera with a TFT screen and web viewing and deleting of stored photos

For this tutorial I've used an ESP32 -CAM, a 1.8″ TFT screen, an 18650 USB powerbank and a 3D printed case to make a selfie camera that automatically takes a photo when it sees a person's face. The project has a lot of steps but is fairly simple. You can make it version without having a 3D printer.

### Setting up the Arduino IDE

Before uploading the code a few things need to be set up in the Arduino IDE. If this is your first time with the ESP32-CAM in the Arduino IDE you need to set up the ESP32 hardware libraries, learn to connect and test by following this tutorial ESP32-CAM in the Arduino IDE

There's three libraries that need to be installed. The TFT_eSPI can easily be installed from the IDE library manager (Tools > Manage Libraries) by searching for TFT_eSPI. The TFT_eFEX and ESPAsyncWebserver libraries need to be installed by downloading the libraries using the the 'Download ZIP' link and in the IDE installing them with Sketch > Include Library > Add .ZIP Library.

The TFT_eSPI library needs to be configured to work with the ST7735S TFT panel. Copy the contents of the User_Setup.h file into the newly installed library file User_Setup.h file found in Documents > Arduino > libraries > TFT_eSPI. If you find the image quality is poor you can try other xxxxTAB versions. These refer to the colours of the tab on the screen protector but don't match 100%.

If you want to use the countdown animation, the images for this need to be uploaded to the ESP32 memory. To do this follow the instructions to install the data folder uploader here: ESP32 Data Folder Uploader . Remember if you change the partition scheme in the IDE this data will be over-written.

### Uploading the Sketch

Download the ZIP file from the project folder on Github https://github.com/robotzero1/esp32cam-selfiecam and unzip to your Arduino folder (In Windows 10 – Libraries > Documents > Arduino) in a directory named SelfieCam.

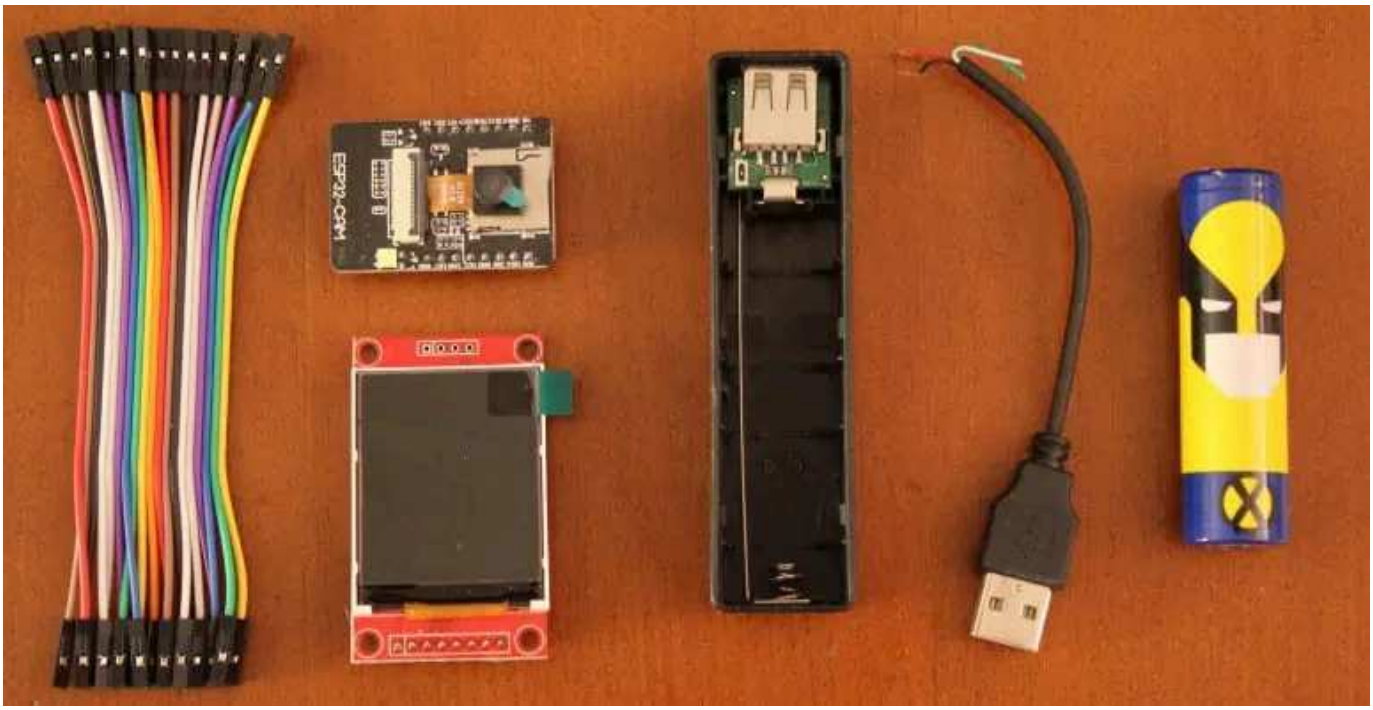
Inside the new directory, open the SelfieCam.ino in the IDE and then use the Tools > ESP32 Sketch Data Upload tool to upload the data directory.

You'll need to reset the ESP32-CAM and then use the following settings in the IDE to upload the Sketch.

Board: ESP32 Dev Module
Upload Speed: 921600
CPU Frequency: 240Mhz
Flash Mode: QIO
Flash Size: 4MB
Partition Scheme: No OTA (2MB APP/2MB SPIFFS)
PSRAM: Enabled

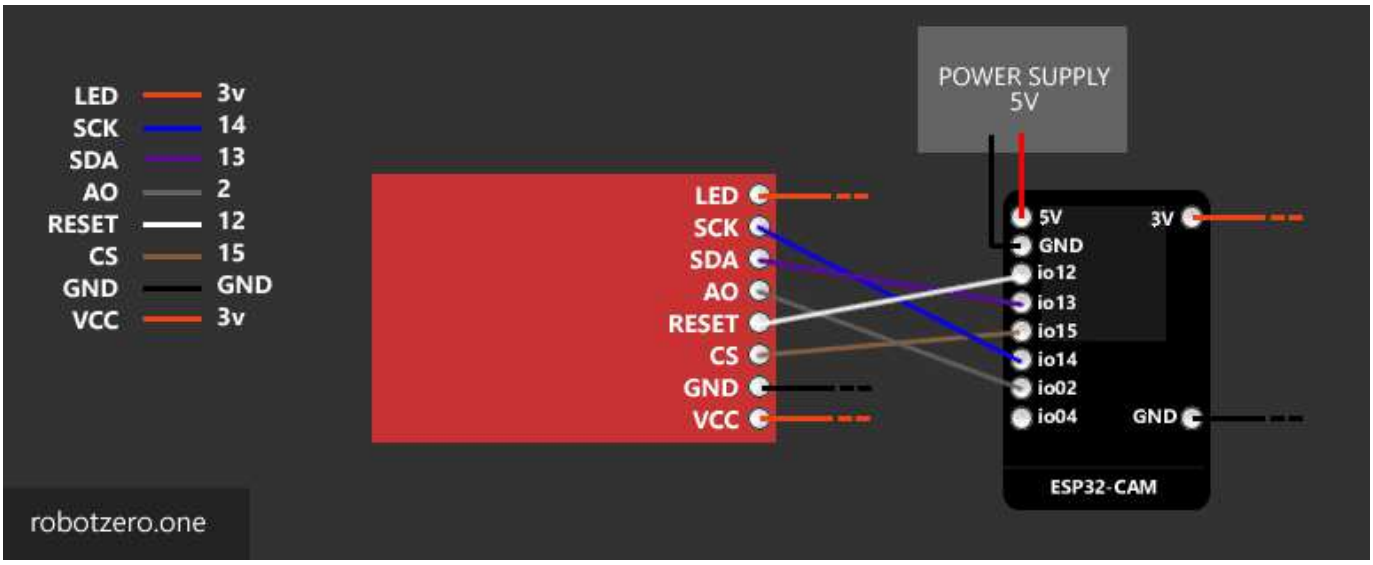
## Components

The project only needs a few components. An ESP32-CAM, a 1.8″ ST7735S TFT screen, 10 male to male dupont cables, a USB powerbank, one 18650 battery and a spare USB cable or terminal block.

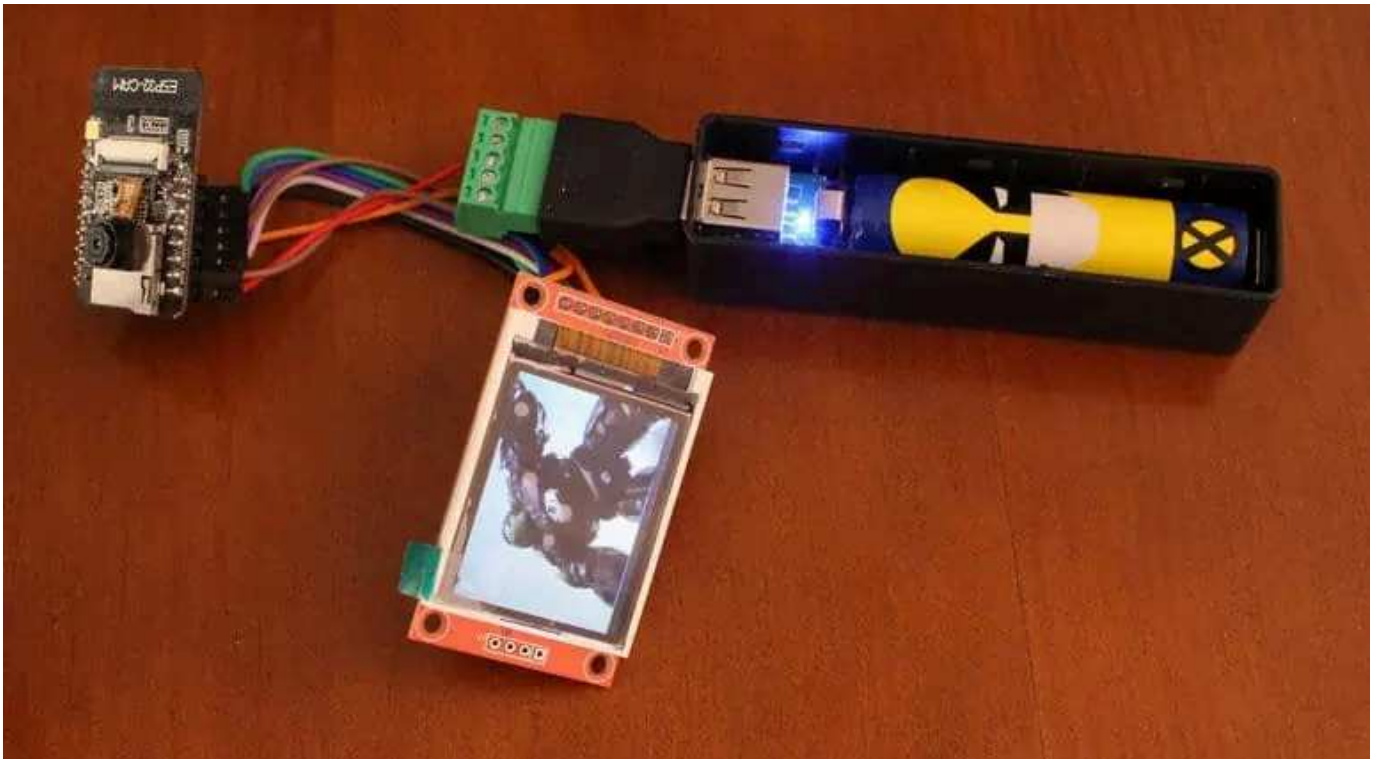


## ESP32 – TFT Wiring Diagram

The project is wired as below. You need to connect two dupont cables to one connector so you can use 3v on the ESP32 to power the LED and VCC pins on the display.



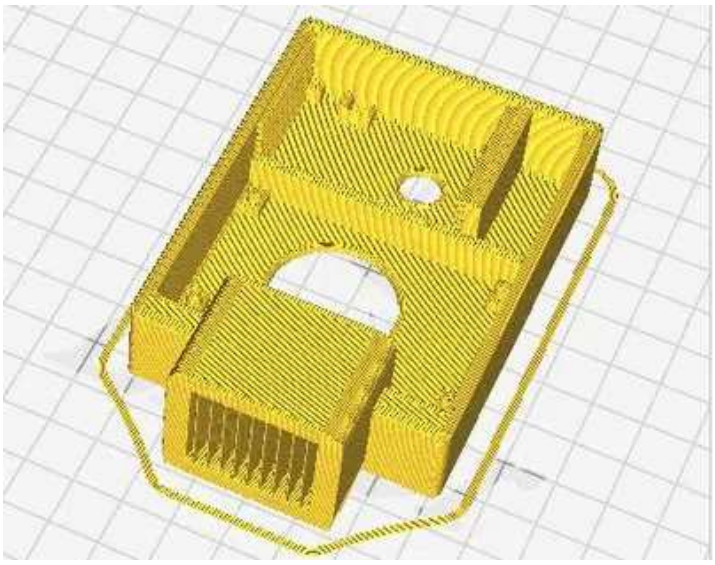

Before adding the components to the 3D model, the project looks like this. I used a USB terminal block (in green) instead of a spare USB cable.

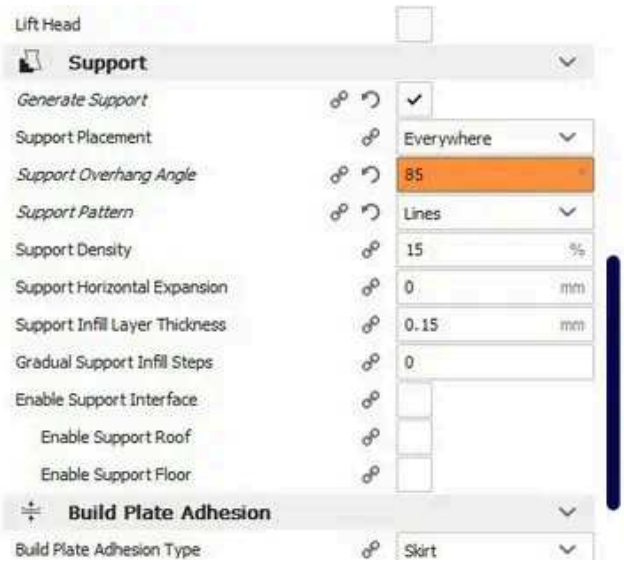


## SelfieCam 3D Printed Model

I created the model in Tinkercad, sliced in Cura and printed on a stock Ender 3.

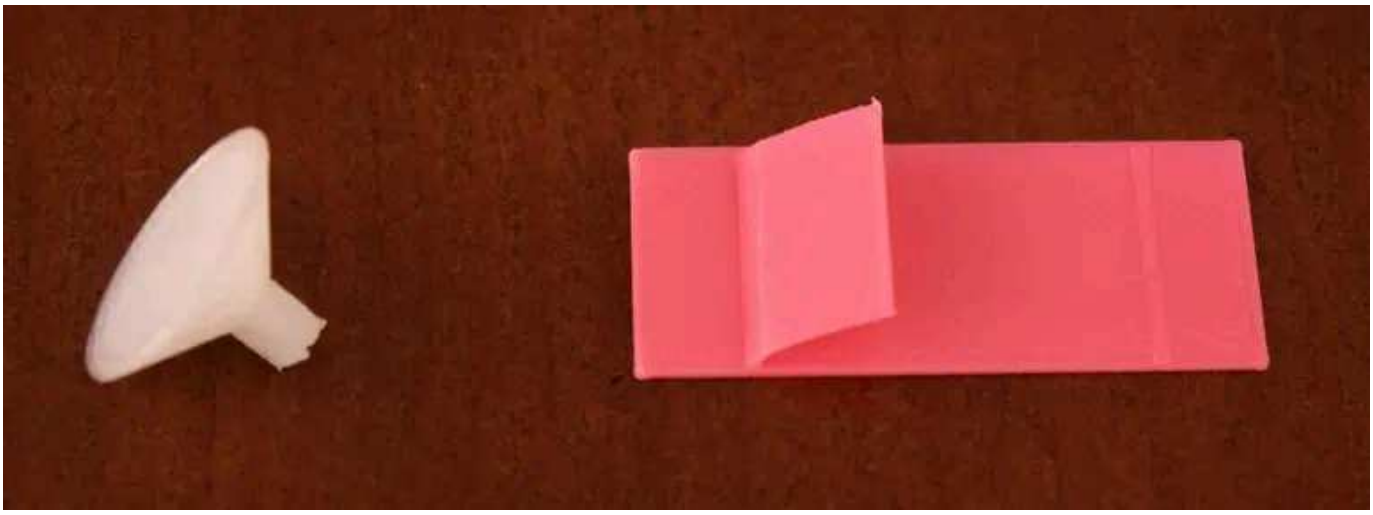
The model needs to be flipped 180° so the front is flat on the print bed. The part where the USB powerbank fits will print better with supports – set the overhang at 85 degrees. I used the standard Ender 3 normal profile.



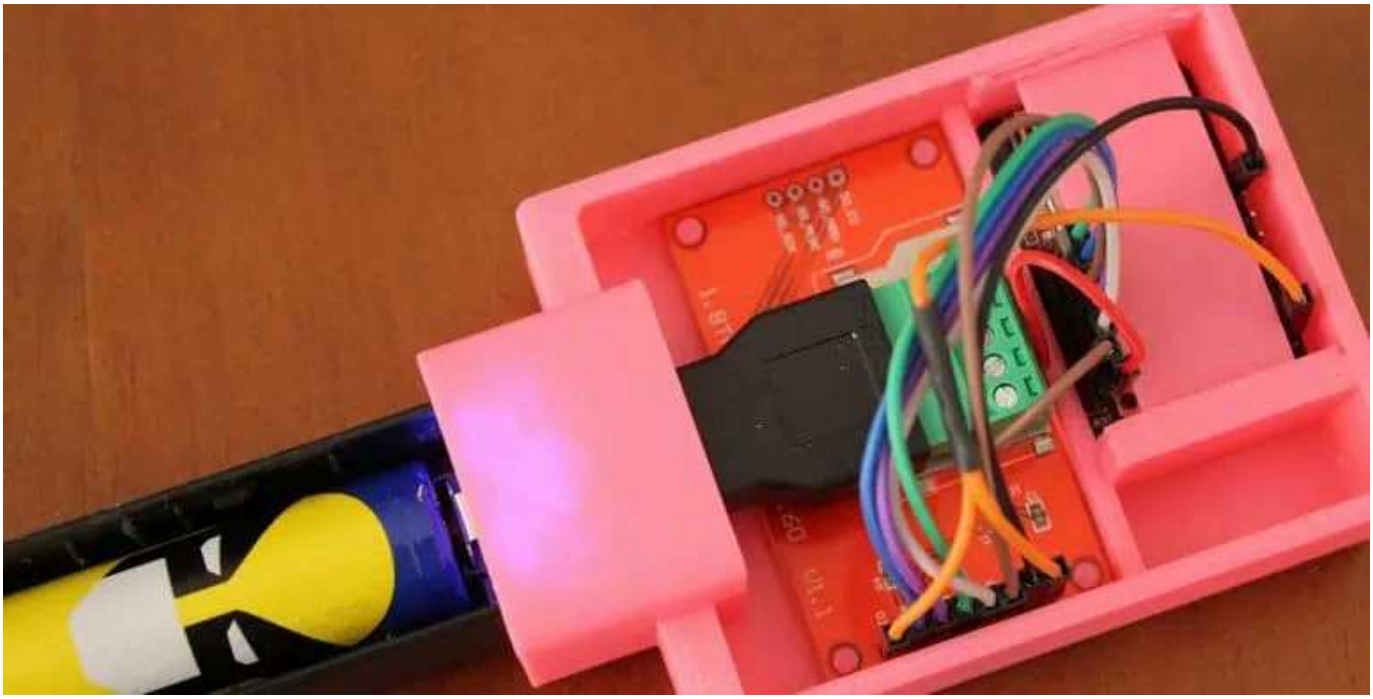



When printed the two sides of the model look like this:


Optionally there are two extra parts to print. A clip that holds the ESP32-CAM in place and a diffuser for the flash. The diffuser should be printed using a transparent filament.



When everything is assembled it should look like this:



## Video Demonstration

Below is a quick video showing the the selfie capture sequence, starting with the face being detected, the flash lighting up, the photo being taken and finally the photo being displayed from the ESP32 SPIFFS storage:

0:00 / 0:26

Full tutorial video on YouTube – https://www.youtube.com/watch?v=j8lVFmjAARA

## Browser to ESP32 Communication

The code uses a mixture of HTTP requests and WebSockets. When the browser first connects to the ESP32 the HTML interface is sent via HTTP with this code:

```
webserver.on("/", HTTP_GET, [](AsyncWebServerRequest * request) {
    Serial.print("Sending interface...");
    AsyncWebServerResponse *response = request->beginResponse_P(200, "text/html",
index_ov2640_html_gz, sizeof(index_ov2640_html_gz));
    response->addHeader("Content-Encoding", "gzip");
    request->send(response);
 });
```

In the browser the interface loads and opens a WebSocket connection to the ESP32. This replies with a list of files in the ESP32 storage – the results of the function below:

```
String filelist_spiffs()
{
  filelist = "";
  fs::File root = SPIFFS.open("/");
  fs::File file = root.openNextFile();
  while (file) {
    String fileName = file.name();
    filelist = filelist + fileName;
    file = root.openNextFile();
  }
  return filelist;
}
```

Back in the browser it processes the list with the code below. addSelfieToScreen() is a function that creates objects in the DOM and fills them to create the visible interface.

```
var filelistFromESP32 = message.data; // list of files from ESP32
var fileIDs = filelistFromESP32.substring(1).split("/"); // remove first / and then split on
subsequent /

fileIDs.forEach(function(item){
        if (item.includes("_t_")){ // thumnail images
                addSelfieToScreen(item);
        }
});
populateImgtags();
```

When all the objects in the interface are created, the populateImgtags() function runs. This uses the fetch() method to request the selfie images from the ESP32. The images are sent from the ESP32 storage to the browser via HTTP with this code:

```
webserver.on("/image", HTTP_GET, [](AsyncWebServerRequest * request) {
   Serial.println("Requesting image from SPIFFS");
   if (request->hasParam("id")) {
     AsyncWebParameter* p = request->getParam("id");
     String imagefile = p->value();
     imagefile = imagefile.substring(4);
     request->send(SPIFFS, "/" + imagefile);
   }
});
```

Every time a new selfie is taken another WebSocket message is sent to the browser using the command ws.textAll((char*)addtobrowser). Again on the browser a new DOM object is created with addSelfieToScreen() and the image is requested with populateImgtag() as above when the interface is first created.

Deleting an image. During the creation of the DOM each image has 'X' added which has an event listener attached with this code:

```
deleteItem.addEventListener("click", function() {
   ws.send("delete:" + selfieID);
});
```

When the 'X' is clicked, a WebSocket request is sent to the ESP32 which then processes this code:

```
String deletefile = incoming.substring(7);
incoming = "";
int fromUnderscore = deletefile.lastIndexOf('_') + 1;
int untilDot = deletefile.lastIndexOf('.');
String fileId = deletefile.substring(fromUnderscore, untilDot);
SPIFFS.remove("/selfie_t_" + fileId + ".jpg");
SPIFFS.remove("/selfie_f_" + fileId + ".jpg");
client->text("removed:" + deletefile);
```

The final line above sends a WebSocket message back. The browser then removes the photo from the interface:

```
function removeSelfieFromScreen(imageid){
      var imageItem = document.getElementById(imageid);
      imageItem.parentElement.remove(); // remove parent div and contents
}
```

---

More tutorials like this – https://robotzero.one/robot-zero-plus/

---

## Resources

Project Code: https://github.com/robotzero1/esp32cam-selfiecam
Editable 3D Model in TinkerCad – https://www.tinkercad.com/things/5fHl1Nb8gHa
Flash and Clip 3D Models – https://www.tinkercad.com/things/bMHwKzBiP4A
STL File for 3D Printer: https://github.com/robotzero1/esp32cam-selfiecam/blob/master/SelfieCam.stl
HTML Thumbnail Grid: https://css-tricks.com/responsive-grid-magazine-layout-in-just-20-lines-of-css/
Async Webserver Docs: https://github.com/me-no-dev/ESPAsyncWebServer
TFT Library: https://github.com/Bodmer/TFT_eSPI
TFT Extras Library: https://github.com/Bodmer/TFT_eFEX
Countdown numbers: https://www.twinkl.es/resource/t-w-32902-numbers-0-31-on-robots