

Large Language Models (LLMs) and ECE 445

- Learn how to use LLM to be more productive while learning new skills and knowledge.
- You should be designing the general workflow and use LLM for specific subtasks.
- Understand what LLM does when it generates code or solutions.
- LLMs make mistakes. Do not use this as an excuse.

ChatGPT Ideation Process

I have an idea for a project. I want to develop a custom hardware comprised of a camera connected to a microcontroller and a small display unit. I like the camera to stream data/images to the microcontroller, where I will be running a large language model to recognize various objects and display text messages of the identified objects on the screen. possible microcontrollers I like to use are ESP32 or STM32. is this feasible idea?



Yes, it's feasible — but not in the way you're phrasing "run a large language model on an ESP32/STM32" if you mean an LLM of the ChatGPT/LLaMA class doing vision + language. The right architecture on those MCUs is:

- camera → **lightweight vision model (or classic CV) on-device** → **label(s)** → **render text on small display**
- optionally: **label(s)** → **(phone/edge box/cloud) LLM** for richer natural-language descriptions

On an ESP32/STM32 you can absolutely do **object recognition**, but it's typically **TinyML CNNs / MobileNet-int8 / person-detect**, not a full LLM.

2) "Run an LLM on the microcontroller": ⚠ usually not feasible (in the normal sense)

A true LLM needs:

- **lots of RAM** (often tens to hundreds of MB even when quantized)
- compute that's closer to an application processor/NPU than an MCU
- storage bandwidth and memory architecture MCUs usually lack

On ESP32/STM32 you *can* run tiny "language-ish" models or templated text generation, but it won't behave like an LLM, and it's not the right tool for vision recognition anyway.

Possible Implementation:

A practical architecture that works

Option A: Pure MCU TinyML (best if you insist on ESP32/STM32)

- Camera: OV2640 (common with ESP32-CAM) or similar
- MCU: ESP32 with PSRAM or STM32H7-class board
- Model: quantized classifier/detector (TensorFlow Lite Micro, Edge Impulse, STM32Cube.AI)
- Display: SPI TFT (ILI9341/ST7789) or OLED (SSD1306)
- Output text: map class index → string and render

This gives you: robust demo, fully offline, low power, constrained vocabulary.

how do I connect OV2640 to a MCU? what should I watch for when I design the PCB? I am new to PCB design

OV2640 can be connected to an MCU in two fundamentally different ways. Which one you choose determines whether this is "beginner-friendly" or a fast path to pain.

1) Two connection models for OV2640

A) Parallel DVP (recommended if your MCU supports it)

Quick question (so I can give you a pin-level wiring + PCB block diagram)

Which exact MCU part/board are you leaning toward (ESP32 variant or STM32 family), and are you using an OV2640 module with FFC or an ESP32-CAM style module? I can then outline the exact net groups, recommended layer stack, and a "must-have" schematic block (power + clock + pullups + optional series resistors) tailored to that choice.

How about using ESP32-S3. can this microcontroller handle acquiring an image and running a light CNN? can I also connect a simple display to the microcontroller?

Yes — Espressif Systems's ESP32-S3 is a good fit for **(1) OV2640 image capture + (2) light CNN inference + (3) small display UI**, as long as you design around its real constraints (RAM, bandwidth, and pixel formats).

A workable architecture is:

1. capture a small frame (or downsample/crop)
2. run CNN
3. render just text (or a small overlay) on the display
4. avoid full-screen high-rate TFT updates unless needed

Concrete "feasible" build recipe (that usually works)

- ESP32-S3 module with PSRAM (ideally 8 MB PSRAM)
- OV2640 camera module (FFC or board module)
- SPI TFT (e.g., 240×240 ST7789) or small OLED
- CNN: int8, ~100–500 KB, input 96–160 px, grayscale or RGB565
- Pipeline: capture → resize/crop → normalize → infer → display label

I want to run this system from a 7 V battery. How do I design a power network so that I can power the microcontroller, camera and display?

If your "7 V battery" is a **2-cell Li-ion/LiPo pack** (typical), design for **6.0–8.4 V** at the input (7.4 V nominal, 8.4 V fully charged). The clean way to power an ESP32-S3 + OV2640 module + small display is:

- **Battery (6–8.4 V) → one main buck to 3.3 V** (high efficiency)
- Optional second rail (5 V) only if your display/backlight truly needs it
- **Local filtering and decoupling** at camera and display (this is where most "random resets / corrupted frames" come from)

Below is a practical power-network blueprint you can implement on a first PCB.

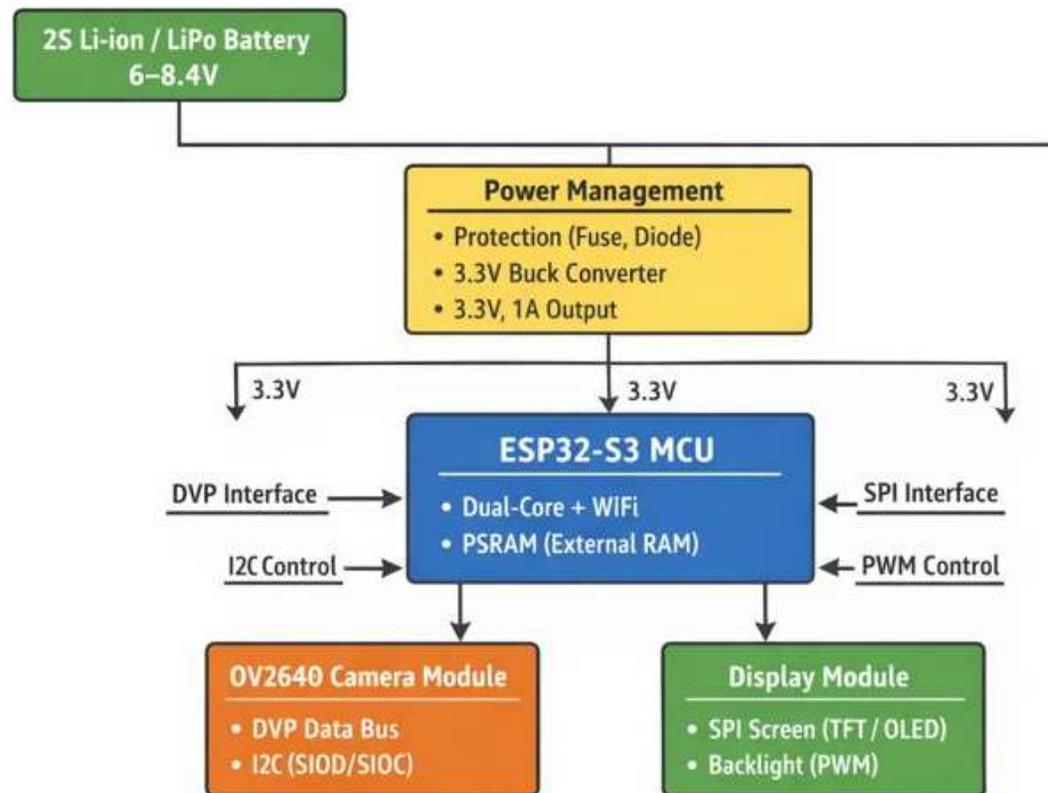
1) Start with a power budget (so the regulators are sized correctly)

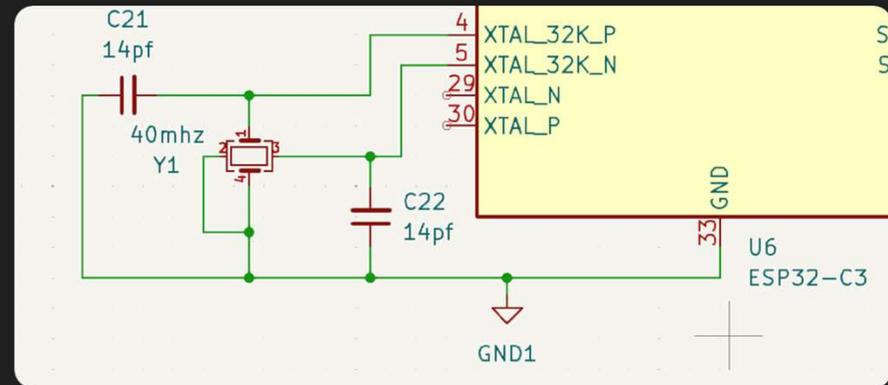
Typical worst-case ballpark (you should design margin into this):

- ESP32-S3 (CPU + Wi-Fi bursts): **200–500 mA** peaks can happen (short spikes)
- OV2640 module: **~60–150 mA** depending on mode
- Small TFT display:
 - logic: **10–30 mA**
 - backlight: **20–150+ mA** (this varies a lot)

Rule of thumb: design your **3.3 V rail for 1 A peak** if you want it to be robust and avoid brownouts.

draw me a block diagram of the entire system (MCU, external RAM, camera, power network and battery) so that I can visualize the entire project





I have designed the schematic for the ESP32-C3. the data sheet is located here: https://documentation.espressif.com/esp32-c3_datasheet_en.pdf. Here is the schematic I have for this MCU and connected it to an oscillator. Are these connection correct?

Your schematic should NOT connect the 40 MHz main crystal to XTAL_32K_P / XTAL_32K_N.

What each pin pair is for (ESP32-C3)

1) XTAL_P / XTAL_N (main crystal, required)

- This is the external main crystal clock input.
- The ESP32-C3 cannot operate without an external main crystal clock.
- Your 40 MHz crystal belongs here.

Espressif Docume...

Writing Code with LLM

- I have an ESP32-C3 zero board. The full documentation is here: https://www.waveshare.com/wiki/ESP32-C3-Zero?srsltid=AfmBOoq_oNjWsJMwKfFI9iauO2g8vhdZNrK-363KvGf02D7K_c9-mkoL . This board has an RGB LED. Look up the pin on the MCU that is connected to the RGB LED and then generate a code that will cycle through all possible color quickly.