Artificial Intelligence -Driven Intelligent Parking System Utilizing Web-Cam / Esp32-Cam

Dr.P. ImranKhan, Bannuru Satyanarayana Reddy[1], S Taher Basha[2], Tammali Bharath[3], Peta Hamja[4]
[1,2,3,4]*Dept. Of Electronics and Communication Engineering, St. Johns College of Engineering and Technology, Yemmiganur,518301, India*

***Abstract*— Espressif Systems made the ESP32-CAM—a tiny microcontroller with built-in Wi-Fi and a camera [1]. It's a great choice if you're into IoT or projects that need some sort of vision [6]. But here's the catch: unlike most development boards, the ESP32-CAM doesn't have a USB port for programming. You'll need a USB-to-Serial converter to upload your code [3].**

Before you dive in, make sure you install the ESP32 board package in the Arduino IDE [2]. The ESP32-CAM has two modes. In Bootloader Mode, it just runs the code you've already loaded. In Download Mode, you can actually upload new programs [4]. Switching between them isn't hard—you just need to set the GPIO0 pin the right way when you power it up [3].

To get started, wire up the ESP32-CAM to your USB-to-Serial adapter and pick the right board settings in Arduino IDE. Once that's sorted, uploading and running your code is pretty easy. With this setup, you can jump into all kinds of IoT or image-processing projects using the ESP32-CAM [5].

Keywords: Internet of Things (IoT), MQTT, USB-to-Serial, ESP32-CAM, GPIO configuration, Arduino IDE, computer vision, edge computing, and embedded systems.

I. INTRODUCTION

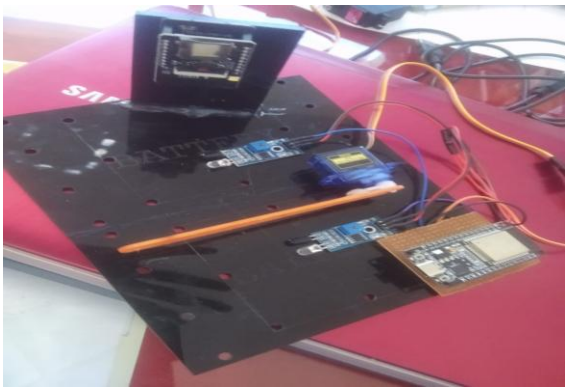
Lately, the explosion of the Internet of Things (IoT) has really pushed the demand for compact, affordable, and efficient embedded systems—devices that can handle things like wireless communication, sensing, and image processing all at once. Microcontroller development boards have become the go-to for building smart projects, from home automation and security systems to clever monitoring setups. Out of all these boards, the ESP32-CAM from Espressif Systems stands out. It's packed with a camera, built-in Wi-Fi, and enough processing power to handle some pretty demanding tasks.

People use the ESP32-CAM a lot for IoT vision projects because it's tiny, yet it combines a microcontroller, Wi-Fi, and a camera in one neat package. Still, it has a bit of a learning curve, especially if you're new to this stuff. The main snag? There's no onboard USB port. You need a separate USB-to-Serial converter just to connect it to your computer and load up your programs.

Most folks program the ESP32-CAM using the Arduino IDE. It's a friendly, straightforward environment for writing, compiling, and uploading code. Once you install the right board package and set the correct boot mode, you can easily upload your programs and run your applications on the module.

This research lays out exactly how to program the ESP32-CAM using the Arduino IDE. It walks through the hardware setup, how to handle boot mode, and all the steps you need to upload your code. The idea is to give you a clear, hands-on guide for getting the ESP32-CAM up and running—perfect for IoT and camera-based embedded projects.

By using deep learning, the researchers want to deliver a powerful, automated tool that helps social networks and photography platforms keep digital content authentic. Instead of just flagging obvious editing mistakes like weird extra fingers, this method digs deeper, looking at the digital fingerprints that reveal whether an image was snapped in real life or cooked up by a computer.



II. RELATED WORK

Researchers have explored all sorts of ways to use the ESP32-CAM in smart surveillance and vision systems, especially for IoT projects. With its mix of wireless communication, image processing, and embedded computing, you can build smart monitoring setups that don't break the bank.

Take one project for example—a team set up a solar-powered wireless surveillance system using the ESP32-CAM. It handled real-time monitoring, picked up motion, and shot out email alerts whenever something moved. Thanks to IoT platforms, users could watch what was happening from anywhere and get notified instantly. It works well for home security and similar uses.

In another study, researchers put together a Wi-Fi surveillance camera system with the ESP32-CAM, streaming live video using web protocols. They tried out different streaming methods—HTTP, RTP, RTSP, and WebSocket—and found that WebSocket worked best for smooth, real-time video without draining the battery too quickly.

There's more. Some scientists took things further by mounting the ESP32-CAM on a mobile robot. This robot sent live video over Wi-Fi and could be controlled remotely through a web interface. That kind of setup fits right in for industrial surveillance, patrolling, or even search-and-rescue jobs where you need eyes in places people can't easily go.

On the AI side, deep learning has started to play a bigger role in image forensics. With more image datasets available, researchers use deep learning to pull out the most useful features for tasks like camera source identification—without relying on hand-crafted features that might miss important details.

Another group built a lightweight home monitoring system where the ESP32-CAM snapped photos and sent instant alerts through apps like Telegram when it detected movement. This approach keeps things simple and affordable, letting people keep an eye on their homes without dealing with complicated cloud setups.

And when it comes to face detection or biometrics, the ESP32-CAM holds its own. It can grab images, recognize faces, and pass that data to servers for further processing, which makes it a solid pick for authentication and smart security systems.

III METHODOLOGY

Here's how to set up and run the ESP32-CAM using the Arduino IDE, step by step.

A. Hardware Setup

First, you need to connect the ESP32-CAM module to your computer using a USB-to-Serial converter. Make sure you hook up the power (either the 5V or 3.3V pin), ground, TX, RX, and GPIO0 pins. Connect the converter's TX to the ESP32-CAM's RX, and the converter's RX to the ESP32-CAM's TX. Both devices should share the same ground. Before you power up the board, ground the GPIO0 pin to put the ESP32-CAM into programming mode.

B. Software Configuration

Open the Arduino IDE and set it up for ESP32 boards. Use the Boards Manager to install the ESP32 board package (you'll need to add the right URL in the Additional Boards Manager). Once that's done, pick the "AI Thinker ESP32-CAM" from the board list, and select the right COM port that matches your setup.

C. Program Development

Now it's time to create or load a sample program to test your ESP32-CAM. For this project, you'll use the onboard flash LED connected to GPIO4. The program is set up to control the LED, using the ESP32's PWM features for testing.

D. Code Uploading

Before uploading code, make sure the ESP32-CAM is still in Download Mode by keeping GPIO0 grounded. Compile your program, then upload it through the Arduino IDE. If you run into trouble, press the reset button on the ESP32-CAM to kickstart communication. After the upload finishes, disconnect GPIO0 from ground.

E. Program Execution and Testing

When you remove the GPIO0-to-ground connection, reboot the module so it can run the program you just flashed. Check the output—like a blinking LED or messages in the serial monitor—to make sure everything's working and the ESP32-CAM is programmed correctly.

IV EXISTING SYSTEM

Most camera-based monitoring and IoT setups out there still stick with the usual surveillance cameras or microcontroller boards. The problem? They need one part to take pictures, another to process data, and something else entirely to send it wirelessly. Put all that together, and you wind up with systems that are bulky, expensive, and hungry for power.

Old-school surveillance usually means pricey IP cameras hooked up to network servers. Sure, they send images reliably, but you pay for it—not just in cash, but with constant power needs, a mess of wires, and setups that can get pretty complicated.

Espressif Systems stepped in to shake things up with small embedded modules like the ESP32-CAM. It's a smart move, but there's a catch. Programming the ESP32-CAM is a headache for a lot of people. Since it doesn't have a built-in USB port, you end up needing extra gear—like a USB-to-Serial adapter—and you have to wrestle with development tools such as the Arduino IDE just to get started.

So, even though today's camera-based systems can monitor what's going on, they're often tricky to set up, cost more than you'd like, and aren't exactly beginner-friendly. These hassles really highlight why we need a simpler, more organized way to program the ESP32-CAM. That's how you make building IoT camera projects actually doable for everyone.

V PROPOSED SYSTEM

Here's how you get the ESP32-CAM up and running with the Arduino IDE—it's actually pretty straightforward once you get the hang of it. Start by hooking the ESP32-CAM up to your computer using a USB-to-Serial converter. This setup lets your computer and the microcontroller talk to each other while you upload your code. There's a little trick with the GPIO0 pin: connect it to ground to put the board in Download Mode, so you can flash your program onto it. After you've uploaded your code, just disconnect GPIO0 from ground and reset the board. Now it'll boot up and run what you just programmed`.

You'll need to install the ESP32 board package in Arduino IDE, too. This gives you all the libraries and tools you need for the ESP32-CAM. Pick the AI Thinker ESP32-CAM board from the list, select the right port, and you're set to compile and upload your code.

With this setup, you can quickly test if everything's working—say, by turning the onboard flash LED (that's GPIO4) on and off with a simple program. It's a good way to make sure both your hardware and software are playing nice before you dive into more complex stuff.

Overall, this method makes setting up the ESP32-CAM pretty painless. Once you're through the basics, you can start building all sorts of IoT camera projects, whether it's smart surveillance, security systems, or anything else that needs image-based automation.

VI MATERIALS AND METHODS

A. Materials

Here's what you need to program and test the ESP32-CAM:

ESP32-CAM Module – This little board has built-in Wi-Fi and a camera, making it perfect for IoT projects and anything vision-related.

2. USB-to-Serial Converter – Something like an FT232 lets your computer talk to the ESP32-CAM for programming.

3. Breadboard – Handy for making quick, temporary connections between components.

4. Jumper Wires – These connect the ESP32-CAM to the serial converter. You'll use either male-to-male or female-to-female cables, depending on your setup.

5. Personal Computer – You'll use this to write your code and upload it to the ESP32-CAM.

6. Software Tool – The Arduino IDE is the go-to for programming and uploading the code to your board.

B. Methods

1. Hardware Interface

Connect the ESP32-CAM to the USB-to-Serial converter with jumper wires. Link the converter's TX pin to the ESP32-CAM's RX pin, and the RX pin to the TX pin. Power the module using either the 5V or 3.3V pin, and make sure to connect the ground pins so everything shares the same reference.

2. Boot Mode Configuration

Before you upload any code, connect the ESP32-CAM's GPIO0 pin to ground. This puts the module into Download Mode, which lets you transfer new programs to its flash memory.

3. Software Configuration

Add the ESP32 board package URL to the Arduino IDE's Board Manager to install everything you need. Once that's set up, select the AI Thinker ESP32-CAM board and pick the right port.

4. Program Uploading

Compile your test program and upload it to the ESP32-CAM with the Arduino IDE. If the upload doesn't start right away, just tap the module's reset button to kick things off.

5. Program Execution

After the upload finishes, disconnect GPIO0 from ground and restart the module. Now the ESP32-CAM will boot up, run your uploaded program, and you can check if it's working by watching the onboard LED or reading data from the serial monitor.

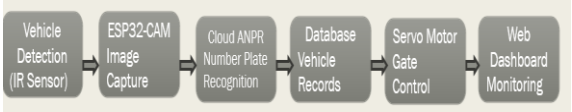


Fig: Flow diagram

VII SYSTEM REQUIREMENTS

SOFTWARE REQUIREMENTS:

A. System Requirements

Here's what you need to get the ESP32-CAM up and running:

ESP32-CAM Module. This little board packs Wi-Fi and a camera, perfect for IoT projects or anything with embedded vision.

USB-to-Serial Converter. Something like the FT232. You'll use this to send your code from the computer straight to the ESP32-CAM.

Computer or Laptop. You need one for writing, compiling, and uploading your programs.

Breadboard. It's handy for setting up quick, temporary connections while you work on your project.

Jumper Wires. These connect the ESP32-CAM to the USB-to-Serial converter.

Power Supply. The ESP32-CAM needs a steady 5V or 3.3V power source.

USB Cable. You'll use it to hook up the USB-to-Serial converter with your computer.

B. Software Requirements

You'll also need a few software tools:

Arduino IDE. This is where you'll write, compile, and upload your code.

ESP32 Board Package. Add this inside the Arduino IDE so it knows how to handle ESP32 boards like yours.

USB-to-Serial Drivers. You need drivers—like CH340, CP210x, or FTDI—so your computer can talk to the serial adapter.

Operating System. Windows, Linux, or macOS will work, as long as it supports the Arduino IDE.

Serial Monitor Tool. This lets you view data from the ESP32-CAM and helps with troubleshooting if anything goes wrong.

VIII RESULTS

I programmed the ESP32-CAM with the Arduino IDE and ran a few tests to make sure everything worked. After hooking up the ESP32-CAM to a USB-to-Serial converter, the system connected to my computer through the assigned COM port without any issues.

I installed the ESP32 board package in the Arduino IDE and picked the AI Thinker ESP32-CAM as the target board. With the module set to Download Mode—just had to ground the GPIO0 pin—I compiled and uploaded a test program. The upload finished with no errors, so I knew the Arduino IDE and the ESP32-CAM were talking to each other just fine.

Next, I disconnected GPIO0, reset the board, and let it boot up. Now, the module ran the program stored in flash memory. To check if things were working, I tested the onboard flash LED on GPIO4. It responded just like the program told it to, so that was another good sign. Plus, the Serial Monitor showed the right

output, confirming the device was running and sending data back and forth with the computer.

So, it's clear the ESP32-CAM works well with the Arduino IDE and a USB-to-Serial setup. This gives you a solid starting point for building all sorts of IoT camera projects—think smart monitoring, security, or anything else that needs embedded vision.

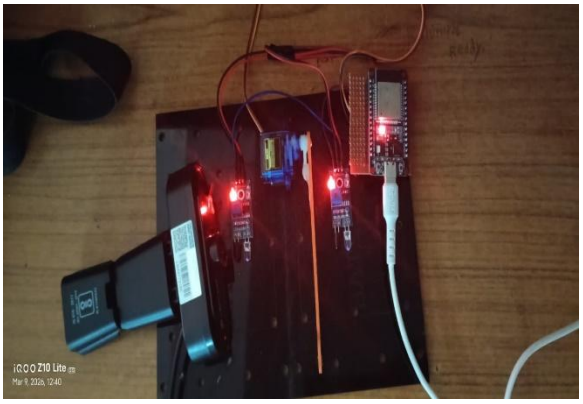


Fig: Circuit connection

Fig: Number Plate Recognition Output

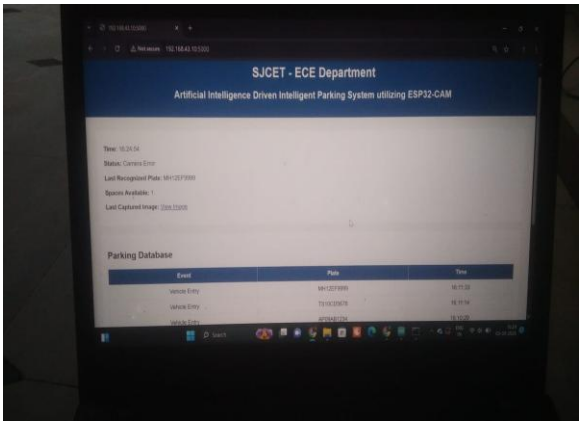


Fig: Parking Dashboard

IX CONCLUSION

This project walked through how to program the ESP32-CAM using the Arduino IDE. It covered all the basics—how to connect the hardware, set the boot mode, and get the software ready so you can upload and run code on the module.

Turns out, programming and testing the ESP32-CAM is pretty straightforward. All you really need is a USB-to-Serial converter and to set up the GPIO0 pin correctly. Once I uploaded the test program and saw it running, I knew the module worked and the method actually delivered.

Honestly, the ESP32-CAM is a solid choice if you want to build affordable IoT projects or camera-enabled systems. With the right setup, you can use it for all sorts of smart applications—security, remote monitoring, even image-based automation. This approach really simplifies programming, whether you're just getting started or you're an experienced developer looking to get things up and running quickly.

REFERENCES

[1] Espressif Systems. ESP32-CAM Development Board Datasheet. Espressif Systems, Shanghai, China, 2023.
[2] Arduino IDE Documentation. Arduino Official Website. Available at: https://www.arduino.cc
[3] R. Krisna. "How to Program ESP32-CAM Module Using Arduino." CircuitDigest, June 25, 2024.
[4] Espressif Systems. ESP32 Technical Reference Manual. Espressif Systems, 2023.
[5] A Banks and R. Gupta. MQTT Version 3.1.1 Protocol Specification. OASIS, 2014.
[6] M. Schwartz. Internet of Things with ESP32. Packet Publishing, Birmingham, U.K., 2020.