**Project Title:** **Wireless Brightness Control with Hand Gestures | ESP32 + OpenCV + TM1637**

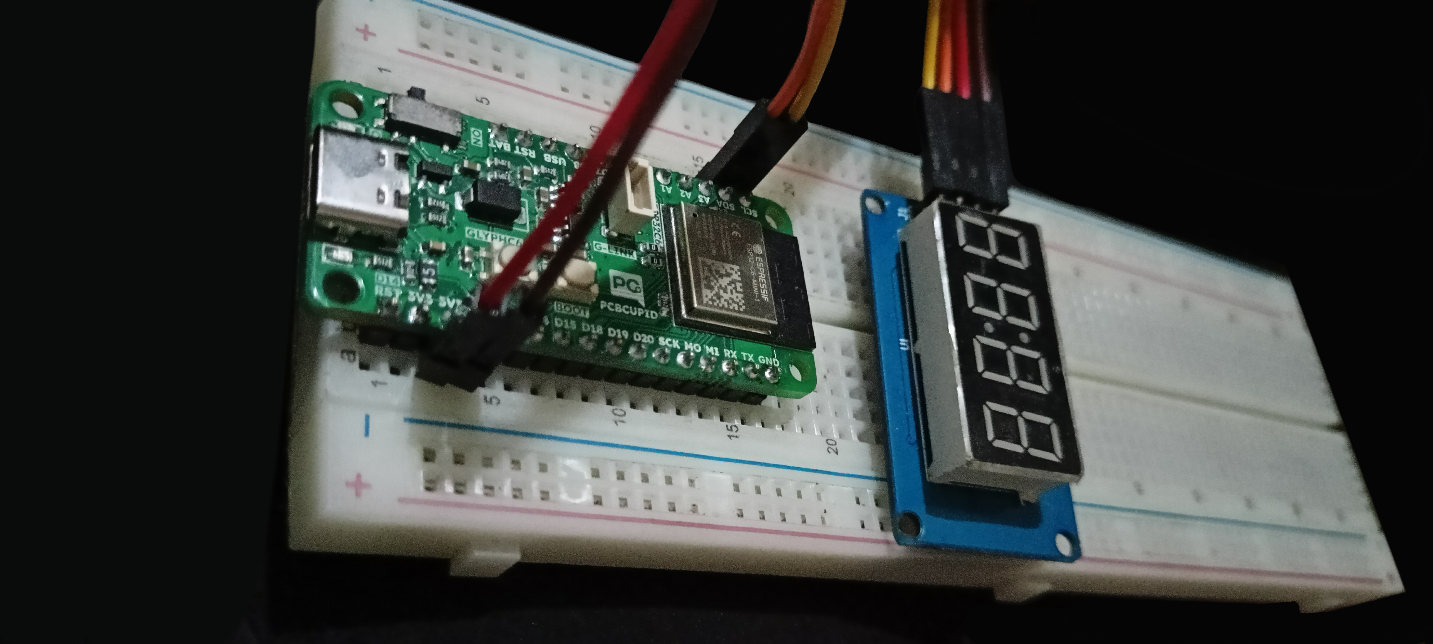
### 🔗 Step 1: Introduction

I’ve always been fascinated by interactive tech — especially the kind you can control without ever touching a button. That curiosity led me to build this project myself: a **wireless brightness controller** that responds to hand gestures in real time.

I used a webcam and OpenCV to track my hand movements — specifically the distance between my thumb and index finger. I wrote the logic to map that distance to a brightness value (ranging from 0 to 255), and then I sent that value wirelessly to an ESP32 using WebSocket.

But instead of just lighting up an LED, I wanted something more visual — so I hooked up a TM1637 7-segment display. It now shows the brightness level and even adjusts its own display brightness based on my hand gesture.

This project was a great way for me to dive into computer vision, real-time wireless communication, and embedded hardware — all in one compact build. It was fun, hands-on, and taught me a lot!



### 🛠️ Step 2: Supplies / BOM

#### Components:

* GLYPH C6 – WiFi / Zigbee Enabled ESP32C6 DEV Board



* TM1637 4-digit 7-segment display (e.g., CL3642BHL)

A close-up of a digital display

AI-generated content may be incorrect.

* Jumper wires
* Breadboard
* USB Cable
* Laptop with Webcam

#### Software:

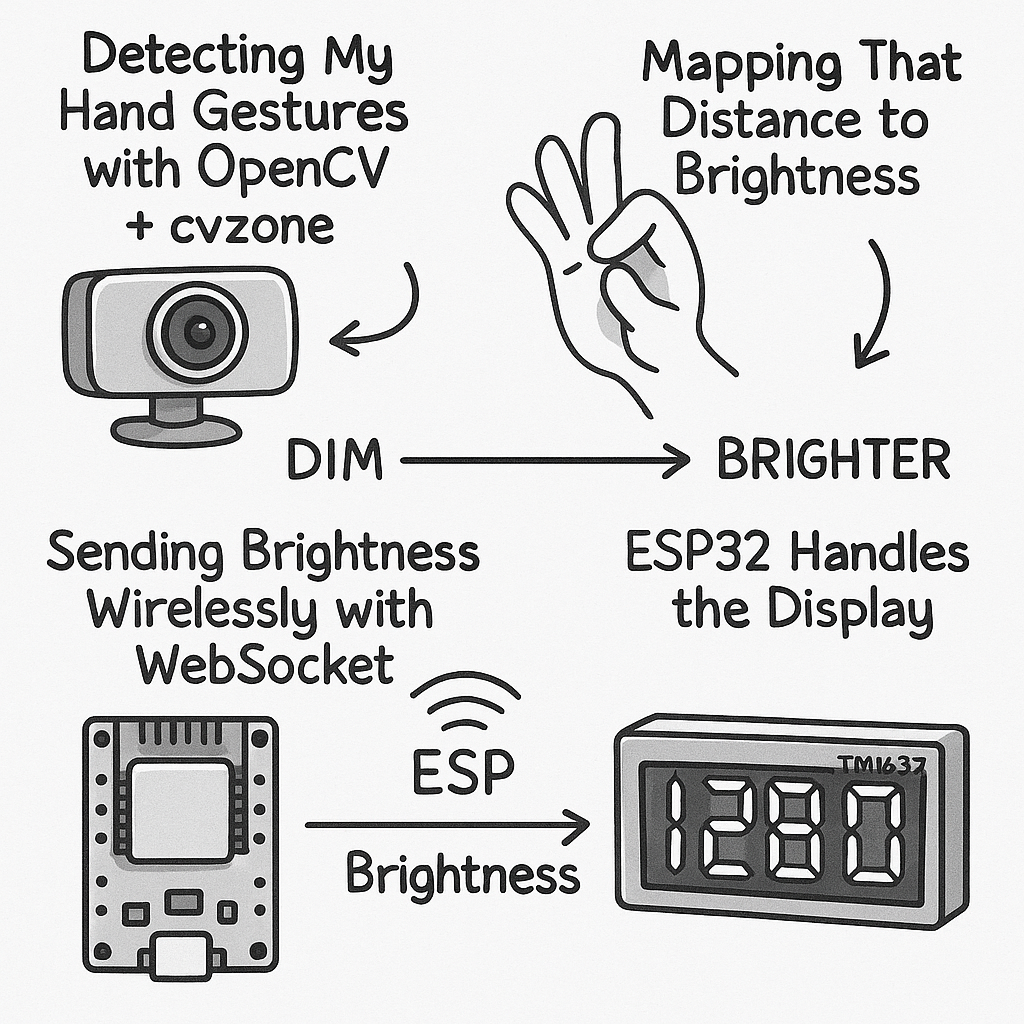
* Arduino IDE (with ESP32 board manager)
* Python 3.10+
* Libraries: OpenCV, cvzone, numpy, websocket-client

#### Optional Tools:

* GitHub (to store code)

### 🧠 Step 3: Concept & Working Principle

This project blends computer vision, wireless communication, and embedded systems — all working together to let me control brightness in real-time, just by moving my fingers.



1. **Detecting My Hand Gestures with OpenCV + cvzone**

At the core of it is OpenCV, paired with cvzone.HandTrackingModule, which simplifies hand tracking using MediaPipe. I used it to detect my thumb tip (ID: 4) and index fingertip (ID: 8) in every frame from my webcam.

By calculating the distance between those two points, I could estimate how far apart my fingers are — a simple and natural gesture that I mapped to brightness (closer = dimmer, farther = brighter).

1. **Mapping That Distance to Brightness**

Once I got the distance, I linearly mapped it to a brightness range of 0 to 255. This made the control feel smooth and intuitive — even small finger movements created responsive changes in brightness.

1. **Sending Brightness Wirelessly with WebSocket**

Originally, I considered using pyserial and USB, but I didn’t want wires or lag. So I set up a **WebSocket server** on the ESP32 (port 81), and wrote a Python client that sends the brightness value in real-time.

This made the entire system wireless, super responsive, and scalable — perfect for expanding later.

1. **ESP32 Handles the Display**

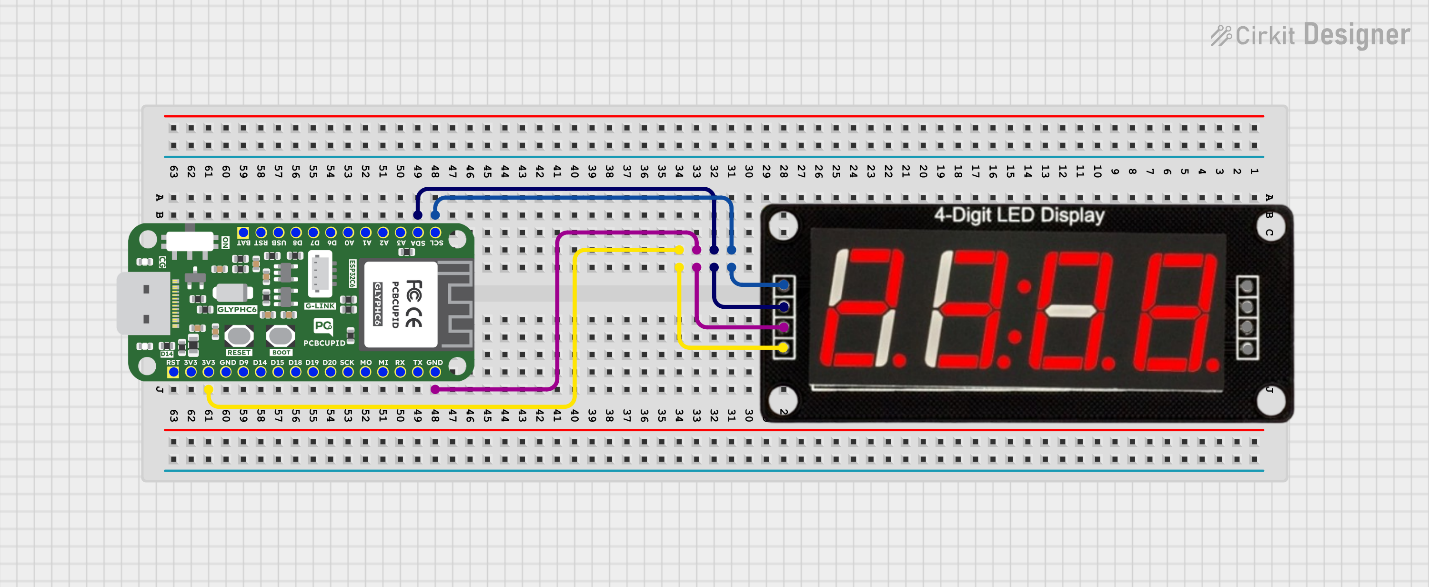
On the ESP32 side, I receive the brightness string inside the onEvent() handler. I convert it into an integer, clamp it between 0–255, and display it on a TM1637 7-segment display using .showNumberDec().

What’s cool is that the TM1637 also supports brightness levels from 0–7, so I mapped the same brightness value to control the display’s intensity too. That way, the display not only shows the number but also **visually dims or brightens** based on my hand gesture — just like an actual light would.

### 🔧 Step 4: Hardware Setup

#### TM1637 Pin Mapping:

* VCC → 3.3V
* GND → GND
* DIO → GPIO 4
* CLK → GPIO 5



ESP32 will host a WebSocket server on port 81. No resistors needed between DIO/CLK and GPIOs.

**Note:** Use 3.3V logic. TM1637 typically works fine with ESP32 voltage levels.

### 📄 Step 5: ESP32 Code (Arduino IDE)

On the ESP32 side, I set it up to connect to Wi-Fi and host a WebSocket server on port 81. Once it's up and running, it just waits for brightness values sent from my Python script.

Whenever it receives data, I convert it into an integer, clamp it between 0 and 255 just to be safe, and then display that value on the TM1637 7-segment display.

I also made sure the **display’s own brightness** adjusts based on that value — so it doesn't just show the number, it actually reacts to it visually. Everything runs wirelessly and in real time, which makes the experience feel super smooth and interactive.

If you want to dive into the actual code, it’s all available in the GitHub repo.

### 💻 Step 6: Python + OpenCV Code

On the Python side, I wrote a script that captures the webcam feed and detects my hand using cvzone and OpenCV. It tracks key landmarks, then calculates the distance between my thumb and index fingertip. That distance is mapped to a brightness value between 0 and 255, and sent directly to the ESP32 via WebSocket.

I also added some real-time visual feedback on the video frame — small white dots on the tips of my thumb and index finger, connected by a dotted white line. Below that, the actual brightness value is displayed so I can see exactly what’s being sent.

**Note :** The full implementation, including helper functions and the WebSocket logic, is available in the GitHub repository.

### 🌐 Step 7: WebSocket vs PySerial

At first, I used pySerial to send brightness data over USB — and while it worked, it wasn’t ideal. There was noticeable delay, and the wired setup felt limiting.

So I switched to WebSocket — and honestly, it made everything better.

**Why WebSocket?**

* It’s completely **wireless**
* The response is **real-time** and snappy
* It’s **cleaner and way more scalable** for future upgrades

In this setup, the ESP32 runs a WebSocket server, and my Python script acts as the client. Once connected, it sends brightness values instantly, no cables needed.

### 📊 Step 8: Testing and Demo

* Upload the ESP32 code
* Run the Python script
* Move your hand in front of the webcam
* The distance between thumb & index controls the brightness
* Watch the TM1637 display update both the number and its brightness

### 🌟 Step 9: Applications and Extensions

* Smart lighting systems
* Gesture-controlled dashboards
* Replace TM1637 with OLEDs or LED strips
* Add MQTT for cloud dashboard
* Multi-device expansion

### 🌐 Step 10: Final Thoughts & Code Repo

Building this project was genuinely rewarding — it brought together everything I love: computer vision, real-time communication, and embedded hardware. It started as a simple idea to control brightness with a gesture, but quickly turned into a full-blown wireless interface that’s clean, fast, and super fun to use.

I see this as a small step toward more natural, gesture-based IoT control systems. And the best part? It's just the beginning — there’s so much potential to build on top of this.

**GitHub Repository:** <https://github.com/prtmxio/handcast-display>

**Contributors:** Pritam Dhiman