From Trash to Treasure: Refurbishing Routers into Mesh WiFi Extenders with ESP32

Aeon Flex, Elriel Assoc. 2133 [NEON MAXIMA]

Aeon Flex, Elriel Assoc. 2133 [NEON MAXIMA]

Photo by Vishnu Mohanan on Unsplash

Ever stumbled across an old router collecting dust in a drawer or tucked away in a closet? Maybe it's a relic from your college days or an outdated model replaced by a shiny new one. Instead of tossing it into the e-waste bin, what if you could breathe new life into it? Better yet, what if you could transform it into a powerful mesh WiFi extender using an affordable microcontroller like the ESP32? This guide is all about turning trash into treasure, showing you how to repurpose old routers into a modern, functional mesh network that boosts your WiFi coverage. It's a fun, hands-on project that's equal parts techy, sustainable, and rewarding. Let's dive in!

Why Refurbish Old Routers?

Before we get our hands dirty, let's talk about why this project is worth your time. Old routers might seem obsolete, but they often have solid hardware that can still perform well with a bit of tweaking. Most routers from the last decade have decent processors, memory, and wireless capabilities that can be repurposed for modern needs. By refurbishing them, you're not only saving money but also reducing electronic waste, which is a growing problem. According to the United Nations, the world generated 62 million metric tons of e-waste in 2022, and only about 22% was properly recycled. Repurposing your router keeps it out of landfills and gives you a chance to flex your DIY skills.

Plus, WiFi coverage is a common pain point. Dead zones in your home or office can make streaming, gaming, or even Zoom calls frustrating. A mesh WiFi system, where multiple devices work together to create a seamless network, is a great solution, but commercial mesh systems can cost hundreds of dollars. By combining an old router with an ESP32, a low-cost microcontroller with built-in WiFi, you can create a custom mesh extender for a fraction of the price. It's a win for your wallet, your tech skills, and the planet.

What You'll Need

To get started, let's gather the tools and components. This project is beginner-friendly, but it does require some basic electronics knowledge and a willingness to tinker. Here's what you'll need:

- An old router: Look for one with a functional WiFi chip and at least one LAN port. Brands like TP-Link, Netgear, or Linksys from the last 10–15 years are good candidates.

- ESP32 module: The ESP32 is a versatile microcontroller with dual-core processing and built-in WiFi/Bluetooth. Popular options include the ESP32-WROOM-32 or NodeMCU ESP32, available for $5-$10 online.

- USB-to-serial adapter: This is for flashing firmware onto the ESP32. A common choice is the FTDI FT232R or CP2102, costing about $3-$5.

- Jumper wires and a breadboard: These are for connecting the ESP32 to the router.

- Soldering kit (optional): If you need to make permanent connections or modify the router's internals.

- Computer with Arduino IDE: You'll use this to program the ESP32.

- Basic tools: A screwdriver, wire cutters, and maybe a multimeter for troubleshooting.

- A power supply: Ensure it matches your router's voltage (usually 5V or 12V) and has enough current capacity.

- Micro USB cable: For powering and programming the ESP32.

You might already have some of these items lying around. If not, the total cost should be under $20, assuming you have the router.

Step 1: Assessing Your Router

First, let’s figure out what you’re working with. Dust off that old router and check its specs. Look at the label on the bottom or back for details like the model number, WiFi standard (e.g., 802.11n or 802.11ac), and power requirements. A quick Google search for the model number can reveal its chipset, flash memory, and RAM. For this project, you want a router with a working WiFi module and at least one functional LAN port. If the router powers on and can connect to a network (even if it’s slow), it’s likely a good candidate.

If the router’s firmware is outdated or locked by the manufacturer, don’t worry. We’ll bypass most of its original software by using the ESP32 to control its networking functions. However, if the router is completely dead (no power, no lights), you might need to troubleshoot its power supply or consider using a different one. A multimeter can help confirm if the router is getting power.

Step 2: Understanding the ESP32

The ESP32 is the star of this project. It’s a tiny but powerful microcontroller developed by Espressif Systems, known for its affordability and robust WiFi capabilities. The ESP32 can act as a WiFi client, access point, or even a mesh node, which makes it perfect for extending your network. For this project, we’ll use the ESP32 to bridge your router’s WiFi signal and create a mesh network, allowing multiple devices to share the same network seamlessly.

The ESP32 runs on a dual-core processor, supports 2.4 GHz and 5 GHz WiFi bands, and has plenty of GPIO pins for connecting to external hardware like your router. We’ll program it using the Arduino IDE, which simplifies the process with a user-friendly interface and tons of community libraries.

Step 3: Flashing the ESP32 with Firmware

To make the ESP32 work as a mesh WiFi extender, we need to load it with the right firmware. Here’s how to set it up:

1. Install the Arduino IDE: Download and install the Arduino IDE from the official Arduino website. It’s free and works on Windows, macOS, and Linux.

2. Add ESP32 support: Open the Arduino IDE, go to File > Preferences, and add the following URL to the Additional Boards Manager URLs: `https://dl.espressif.com/dl/package_esp32_index.json`. Then, go to Tools > Board > Boards Manager, search for “ESP32,” and install the ESP32 board package by Espressif.

3. Connect the ESP32: Plug your ESP32 into your computer using the micro USB cable. Use the USB-to-serial adapter if your ESP32 doesn’t have a built-in USB interface.

4. Load the mesh firmware: We’ll use the ESP-MESH framework, which allows ESP32 devices to form a self-healing mesh network. You can find sample code in the Arduino IDE under File > Examples > ESP32 > Mesh. For this project, we’ll use a basic mesh extender sketch.

Here’s a simplified version of the code to get you started:

```
#include <WiFi.h>

#include <esp_wifi.h>

#include <esp_mesh.h>


#define MESH_PREFIX “MeshNetwork”

#define MESH_PASSWORD “MeshPassword123”

#define MESH_PORT 5555


Void setup() {

Serial.begin(115200);

WiFi.mode(WIFI_STA);

Esp_mesh_init();

Esp_mesh_set_topology(MESH_TOPO_TREE);

Esp_mesh_set_ssid_prefix(MESH_PREFIX, strlen(MESH_PREFIX));

Esp_mesh_set_password(MESH_PASSWORD,
strlen(MESH_PASSWORD));
Esp_mesh_start();
Serial.println("Mesh network started");
}

Void loop() {
// Handle mesh communication
Esp_mesh_comm_loop();
}
```

This code initializes the ESP32 as a mesh node. You'll need to customize `MESH_PREFIX` and `MESH_PASSWORD` to match your network. Upload the code to your ESP32 via the Arduino IDE by selecting the correct board and port under Tools. If you run into issues, double-check your connections and ensure the ESP32 drivers are installed.

Step 4: Connecting the ESP32 to the Router

Now, let's integrate the ESP32 with your old router. The goal is to use the router's WiFi hardware to extend the mesh network created by the ESP32. Most routers have a serial interface or LAN port that we can use to communicate with the ESP32.

1. Open the router: Use a screwdriver to carefully open the router's casing. Be gentle to avoid damaging the circuit board.

2. Locate the serial pins: Many routers have a serial interface (usually labeled TX, RX, GND, and VCC) for debugging or flashing firmware. Check online forums or the OpenWrt wiki for your router's model to find the pinout.

3. Connect the ESP32: Use jumper wires to connect the ESP32's GPIO pins to the router's serial pins. For example, connect ESP32's TX to the router's RX, ESP32's RX to the router's TX, and GND to GND. If your router requires 3.3V logic, ensure the ESP32's VCC is connected to a 3.3V source.

4. Power considerations: The ESP32 typically runs on 5V or 3.3V, while routers might use 12V. Use a voltage regulator if needed to avoid frying your ESP32.

If your router doesn't have accessible serial pins, you can use a LAN port to communicate via Ethernet. This requires additional configuration on the

ESP32 to act as an Ethernet-to-WiFi bridge, which we'll cover in the next section.

Step 5: Configuring the Router as a WiFi Extender

To make the router work as a WiFi extender, we need to configure it to connect to the ESP32's mesh network. If your router supports OpenWrt or DD-WRT (open-source firmware), flashing it with one of these can simplify the process. Check the OpenWrt or DD-WRT websites for compatibility with your router model.

1. Flash OpenWrt (optional): If compatible, download the OpenWrt firmware for your router and follow the flashing instructions. This usually involves uploading the firmware via the router's web interface or using a TFTP client.
2. Configure the router: Access the router's web interface (usually at 192.168.1.1) and set it to "Client" or "Repeater" mode. Enter the SSID and password of the ESP32's mesh network (`MESH_PREFIX` and `MESH_PASSWORD` from the code).
3. Test the connection: Power on both the router and ESP32. The router should connect to the ESP32's mesh network and extend its coverage. Use a WiFi analyzer app to check signal strength and confirm the mesh is working.

If your router doesn't support custom firmware, you can still use it as a dumb access point by connecting it to the ESP32 via a LAN port and configuring it manually. This might require setting a static IP and disabling DHCP on the router.

Step 6: Expanding the Mesh Network

The beauty of a mesh network is its scalability. You can add more ESP32 modules or refurbished routers to extend coverage further. Each ESP32 node will communicate with the others, creating a seamless network. To add a new node:

1. Flash another ESP32 with the same mesh firmware.
2. Connect it to another router or use it standalone as a WiFi extender.
3. Power it on and let it join the mesh network automatically.

The ESP-MESH framework handles node discovery and communication, so you don't need to manually configure each device. Just ensure all nodes use the same `MESH_PREFIX` and `MESH_PASSWORD`.

Troubleshooting Tips

If things don’t work as expected, here are some common issues and fixes:

- ESP32 not connecting: Double-check your serial connections and ensure the correct board is selected in the Arduino IDE.
- Router not joining the mesh: Verify the router’s WiFi settings match the ESP32’s mesh network. Check for firmware compatibility issues.
- Weak signal: Reposition the router or ESP32 to avoid interference from walls or electronics. Consider upgrading the router’s antenna if possible.
- Power issues: Use a multimeter to confirm the router and ESP32 are getting the correct voltage.

Why This Project Matters

This project isn’t just about saving a few bucks or getting better WiFi. It’s about creativity, sustainability, and learning. By refurbishing an old router, you’re giving new purpose to hardware that might otherwise end up in a landfill. You’re also diving into the world of microcontrollers, networking, and open-source software, which are valuable skills in today’s tech-driven world. Plus, there’s something incredibly satisfying about turning a forgotten gadget into something useful.

The ESP32’s versatility opens up endless possibilities. Beyond WiFi extenders, you could use it to create IoT devices, home automation systems, or even a retro gaming console. The skills you gain from this project, like programming, soldering, and network configuration, can be applied to countless other DIY endeavors.

Final Thoughts

Transforming an old router into a mesh WiFi extender with an ESP32 is a fantastic way to blend sustainability with technology. It’s a project that’s accessible to beginners but offers enough depth to keep seasoned tinkerers engaged. By following these steps, you’ll not only boost your WiFi coverage but also gain a deeper understanding of how networks and microcontrollers work. So, dig out that old router, grab an ESP32, and start building. Your home network (and the planet) will thank you!

If you want to take this further, consider experimenting with additional features like QoS (Quality of Service) settings to prioritize bandwidth or integrating sensors with the ESP32 for smart home functionality. The possibilities are endless, and the journey from trash to treasure is just the beginning.

Mesh Networks

Esp32

DIY