The module is significantly cheaper than the W5500, but there may be connection issues. Spent a lot of time figuring out why the internet scheme didn't work, so I'll save the experience here. On the internet, there were unanswered questions about why this scheme also didn't work for some, had to figure it out myself.

On the internet, mostly schemes for connecting to the module specifically ESP32 Devkit C, which is an extended version of ESP32 Devkit V1. It has an additional 7 pins, which are not desirable for regular users, as they clutter the ESP32.

- One of these is GPIO0, if it's set to logic low (or even logic high from an external 1.8V logic, or if this logic is floating), then during ESP32 startup, it will interpret it as entering programming mode, and it won't start the program. On the module, GPIO0 is connected to the FLASH (BOOT) button and pulled up with a resistor of 4.7k...10k to +3.3V. Sometimes on the module, the button goes through a 470 ohm resistor, which may be too much for the module to respond to it when connected to something else. Same goes for the Reset button.

- 6 pins through which ESP32 boots from its memory chip. The issue here was that the Chinese labeled one pin as "GND" instead of CMD.

- Plus, a common small mistake is using a 0.1µF capacitor or its absence in parallel with the Reset button, whereas it should be 1µF. Because of this, it was necessary to press the FLASH (BOOT) button to upload the sketch.

The use of Devkit C is chosen due to the possibility of clocking (synchronization) at 50MHz from LAN8720 towards ESP32, thanks to the hardware capabilities of GPIO0. But this can be bypassed with settings in the sketch, maybe not in favor of Ethernet stability in combination with Devkit V1 module.

I assembled it "on the fly" based on a common wiring diagram found online (then cross-referencing it with the schematic).

One 4.7k resistor is placed between GND and NC (a pin not connected anywhere, reserved), with a wire to the Enable pin of the active quartz. It turns off the quartz during program startup, then the program via GPIO17 sends logic high through this NC pin, and it starts the quartz by pulling it out of the Z-state. Otherwise, when power is applied, the quartz output via the nINT/RETCLK pin will interfere with GPIO0, and ESP32 will fail to start. The second resistor is already present on the ESP32 module (it pulls GPIO0 BOOT to +3.3V), so I didn't add it.

The sketch is standard \WiFi\examples\ETH_LAN8720_internal_clock from the ESP32 Arduino set. It appears after selecting the ESP32 Dev Module board, which allows selecting the reflashed flash from 32Mbit (4MB) to 128Mbit (16MB) later in the "Tools" tab. The microchips in ChipDip are GD25Q127csigr, GD25Q127cyigr, W25Q128jveiq, etc.

The sketch obtains a dynamic IP from your router via LAN, connects to Google, and retrieves date and time from it. The entire process is output to the serial port monitor.

And there's another type of connection where the clocking is done from ESP32 to LAN8720. I'm not sure about its impact on booting and processor stability. There are several variations:

1. Clocking from GPIO0.

In the sketch, you need to output 50MHz on GPIO0 instead of receiving it. No quartz control will be needed in this case.

Disconnect the wire from GPIO17, freeing up this pin. This means you can do without resistors by shorting the Enable pin of the quartz to the nearby GND on the LAN8720. The NC pin is not used.

2. Clocking from GPIO17.

This option is suitable if you only have the ESP32 Devkit V1, as wiring to GPIO0 on the module for implementing the first option might not be aesthetically pleasing. In the sketch, specify clocking from GPIO17.

Hardware-wise, similar to above, the quartz should be turned off, and the nINT/RETCLK pin should be connected to GPIO17, not GPIO0. The NC pin is not used. Judging by the comment in the sketch, this option has been tested and works fine.

// Additionally, I've noticed that the LAN8720 module can rarely hang. So, if it's in a remote location, it would be good to power cycle it using a free ESP32 pin. LAN8720's current consumption during operation is up to 90mA, which is much less than W5500. The ESP32's onboard regulator can easily handle it. Perhaps the current also slightly depends on the length of the wire to the router/switch.