1. **What are the main features of Real-Time Operating Systems and how do they differ from regular OS’s?**

**It is perhaps the simplest way to describe normal operating systems, where there are many files in the background on computers and they all work separately but at the same time. The processing of this working principle on hardware and running there are also called real-time operating systems. This is the clearest difference between real-time operating systems and normal operating systems. The most used area of real-time operating systems is the application of critical and attention to control systems. It is expected that the hardware will take action very quickly while the systems are being used. However, getting fast results from hardware does not put that system in the category of real-time operating system. For this, it is expected to receive feedback at the same speed for similar priority tasks at the same time.**

1. **When is it preferable to use STM32 over Arduino, ESP32, PIC or other comparable embedded system?**

Arduino and ESP32 module are certainly rich in terms of both being readily available in the market and having quick and comprehensive access to source codes and built examples, as they share the common IDE. PIC, on the other hand, continues to be used by those who are still starting out. Today, it is frequently used in real machine systems, both PLC and military systems, keyboards or devices used in the control of games. Considering the prevalence of use in professional works, it is quite appropriate to prefer it.

1. Let's assume that a few of the peripherals you use have the same hardcoded I2C address. What solution would you use to solve this problem?

The following comes to my mind to solve this problem: First, I would define different GPIO pins as I2C software and call them as pins, not as addresses.

Secondly, while designing the pcb as hardware, after deciding how many I2C lines are suitable for the system from the multiplexer chips and how many I2C lines are connected to it, I would deactivate it from the microprocessor.

1. Let's assume that there is a hardware module that is attached to the motherboard with a mezzanine connector. Due to the small form factor, the number of pins on the mezzanine connector is much less than the number of pins you need. Assuming that the communication speed is not important, how would you solve this problem?

For example, I have 64-bit data, but if the sending part is only in an 8-bit area, I would try to send it by connecting serial, while there is normally a parallel communication for the data to pass and transmit to the other side. So I would send 64 by 8 ways.

1. Let's say you want about 30 hardware modules to communicate with each other. There is one STM32 microcontroller on each module, and the modules can be removed and installed instantly. Which communication standard would you use for these modules to communicate effectively with each other? Why?

As a result of my research, they used hotplug communication and this system is provided by CANBUS.

1. 3 devices with identical embedded software and hardware are required to communicate over I2C. What kind of solution would you develop so that devices can be dynamically addressed and recognize each other?