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

While the response times of real-time operating systems do not change depending on the system load, the response time of general purpose operating systems increases as the system load increases.

In real-time operating systems, as a rule, it is important to perform operations in a limited time, while in general-purpose operating systems, the number of operations that are performed in a certain time is important.

In real-time operating systems, a task running on the processor must always be the task with the highest priority, otherwise a system failure will occur. In general-purpose operating systems, low-priority tasks can run while high-priority tasks are waiting for the cpu because general-purpose operating systems always try to keep the number of processes done.

While in real-time operating systems, the transition times between tasks and from task to interrupt are fixed, this is not the case in general-purpose operating systems.

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

It can be used in applications that use peripherals that need more than one clock source. It is used in high speed applications. In applications that require multiple i2c, spi, uart, stm32 provides them while others cannot.

STM32 processors offer many case options, while others generally cannot.

In STM32 processors, it is possible to access peripherals on peripherals and to configure these peripherals in detail. Memory management on STM32 processors is more flexible and offers more options.

3. **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?**

To solve this problem, i2c multiplexer IC can be used. In addition, an output pin that controls the power connection of each peripheral can be communicated when only one of these peripherals is powered.

4. **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?

I use i2c or spi based pin Multiplexer to solve this problem.

5**. 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?**

I use the CAN-BUS communication protocol in this system. Because : CAN-BUS communication protocol is message-based, not address-based. it also consists of two lines and the line only needs one 120ohm resistor. Since there is no master and slave in this system, there will be no problem if any module is removed. In addition, since it has the feature of debugging faulty messages with CAN-BUS, the wrong messages received due to the noise that occurs when inserting and removing the modules are removed.

6. 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?

First of all, when all devices are energized, they automatically switch to the master mode and query the addresses on the line, and even take one more than the highest value address from the addresses they find as an address and switch to the slave mode. Periodically, they exit the slave mode and perform address recognition and send a message identifying themselves to all devices on the line. While in slave mode, the devices on the line listen to the incoming messages and record the message ID that introduces itself. If there is no device on the line, the first connected device will receive the address 0x01, while a device will send a message to a device with a certain id, it will send a message to the address corresponding to that id in its own memory.