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

Operating system operating systems are used for desktop and laptop computers. Process-based planning is done. Delays are not considered significant. Rtos is for embedded systems only. Time-based programs are made. Delays, even microseconds, are crucial.

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

We also use Stm 32 in industrial projects. We prefer Stm32 because it has an arm processor on it. Because other processors experience interference more easily and are microprocessors used mostly for learning purposes.

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?

Using other I2C bus based on an article. I can use this method if we have enough hardware to provide another I2C bus. I leave the link of the article I read.

https://www.oreilly.com/library/view/programming-embedded-systems/0596009836/ch01.html

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?

There are I2C connection multiplexer modules to solve this problem. And I think it can eliminate this problem.

The TCA9548A module supports up to 8 devices with the same address.

The default I2C address of the card is 0x70 and you can communicate with the I2C device connected to this address by choosing one of the 8 connections of the card with a byte of data you will send to this address. In addition, since the I2C address of this card can be changed between 0x70 and 0x77, 8x8=64 I2C devices can be used by using 8 of these cards at the same time. Thanks to the logic level converters on the board, it can be used easily with 3-5V logic levels.

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?

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?

In 2C communication, there is a master device that controls the communication. Each communication must have one master. In order to ensure communication, at least one slave device must be connected to the communication line. Master and slave devices do not need to be connected to the same supply line. However, the ground lines must be the same in order to ensure communication. In addition, there are two communication lines for data transfer, SDA (Serial Data Line) and SCL (Serial Clock). Among these lines, SDA is the line where data transfer between devices is provided. This line has two-way data transfer. The synchronization of the data transferred to the line is performed by the SCL line. The SCL line contains the clock signal generated by the master device. The communication on the SDA line is regulated according to this signal.