1

Hello everyone. We are group of 10.

2Today I will present on the topic of real-time software engineering.

3

The main content includes 4 parts. These are embedded (em be đệt) system design, Architectural (a chi tếch trồ) Patterns for Real-Time Software, Timing Analysis and Real-Time Operating Systems.

4

we will go through some overview content
Firsly, about the **Embedded software**

# 5 Responsiveness

To continue, I will talk About the **Responsiveness**

Responsiveness in real-time is the critical difference between embedded systems and other software systems, such as information systems, web-based systems or personal software systems.

For non-real-time systems, correctness can be defined by specifying how system inputs map to corresponding outputs that should be produced by the system.

> This means that non-real-time software will focus on accuracy. For example, if you choose a create command in a patient (pây sừn) information system, then the correct system response is to create a new patient record in a database and to confirm that this has been done. Within reasonable limits, it does not matter how long this takes

6

In a real-time system, the correctness depends both on the response to an input and the time taken to generate that response. If the system takes too long to respond, then the required response may be ineffective.

> For example, if embedded software controlling a car's braking system is too slow, then an accident may occur (ờ cơ) because it is impossible to stop the car in time

# 7Definition

Therefore, time is fundamental (phăn đơ men tồ) in the definition of a real-time software system.

8

* However there is a note: Timely response is an important factor in all embedded systems, but not all embedded systems require a very fast response.

# 9Characteristics of embedded systems

As well as the need for real-time response, there are other important differences between embedded systems and other types of software system.

Here are some special characteristics of embedded systems:

12

# Next, I will talk about fisrt content, it is : Embedded system design

14

## About Reactive systems

✧ Real-time systems are often considered to be reactive systems. Given a stimulus(sờ ti mơ lợt), the system must produce a reaction or response within a specified(sờ peo sơ phai) time.
=> . A stimulus is an event occurring in the software system's environment that causes the system to react in some way; a response is a signal or message that the software sends to its environment.

Slide 15

## Stimuli and response for a burglar alarm system :

You can determine the behavior of a real-time system by listing the received stimuli of the system, the responses involved, and the point in time at which the response must be made is produced. For example, the figure below shows possible stimuli and the system's response to them theft alarm system.

You can see …. and similarly with the remaining stimulus

## Slide 16 A general model of an embedded real-time system :

The figure below depicts how excitation from the sensor variables in the environmental system and the response are sent to the set the media,

## Slide 17 : Architectural considerations

There are several architectural(a chi tếch trồ) considerations when designing embedded systems

## Slide 18 Sensor and actuator processes

: It is a general design guideline for real-time systems to have separate control procedures(pờ rồ xi điu) for each type of sensor and actuator.

## 21. Design process activities

There is no standard embedded system design process. Rather, different processes are used that depend on the type of system, available hardware, and the organization that is developing the system. The following activities may be included in a realtime software design process:

# 30 : Mutual exclusion

To avoid this problem, you should implement information exchange using a shared buffer and use mutual exclusion mechanisms(me chơ nít dừm) to control access to that buffer.

# 31 Producer/consumer processes sharing a circular buffer

31 : Figure 21.4 illustrates the organization of a shared buffer.

This is usually implemented as a circular queue, using a list data structure. Mismatches in speed between the producer and consumer processes can be accommodated(ờ căm mơ đây tựt) without having to delay process execution.

A circular buffer is a data structure that when data is added, it automatically returns to its original position if it has reached the end.

This makes good use of buffer space and creates an efficient mechanism(me chơ nít dừm) for sharing data between processes. We have learned about this model in operating systems class.

# 33 : State machine model of a petrol (gas) pump

this Figure is another example of a state model that shows the operation of a fuel delivery software system embedded in a petrol (gas) pump. The rounded rectangles represent system states, and the arrows represent stimuli that force a transition from one state to another

# 36 : The second main content is about Architectural patterns for real-time software

# 37 Architectural patterns for embedded systems

37 : Architectural patterns are abstract, stylized descriptions of good design practice.

The difference between real-time and interactive software means that there are distinct architectural patterns for real-time embedded systems. Real-time systems' patterns are process-oriented rather than object- or component-oriented. In this section, I discuss three real-time architectural patterns that are commonly used:

# 39 : The observe and React pattern

We will start with Observe and React .

This is a table presenting the main content about Observe and react :

The input values of a set of sensors of the same types are collected and analyzed. These values are displayed in some way. If the sensor values indicate that some exceptional condition has arisen, then actions are initiated to draw the operator’s attention to that value and, in certain cases, to take actions in response to the exceptional value.

it receives stimulation as values from sensors attached to the system.

The reactions it performs are: Outputs to display, alarm triggers, signals to reacting systems.

Included processes are: Observer, Analysis, Display, Alarm, Reactor.

Be used in : Monitoring systems, alarm systems.

40 : This is **Observe and React process structure :**

**Stimuli from the sensor are processed and responses are given**

# 43 The environmental Control pattern

43: The next architectural pattern is: The Environmental Control pattern.

The control system can use the Environmental Control pattern, which is a control pattern generally includes sensor and actuator processes. This pattern is described in Image

**44. : this is Environmental Control process structure**

# 45Control system architecture for an anti-skid braking system

45 :  For example, a car's anti-slip braking system monitors the car's wheels and braking system (system environment). It looks for signs of wheel slippage when brake pressure is applied. In this case, the system will adjust brake pressure to stop wheel locking and reduce the possibility of skidding.

You can see how this pattern is used in the figure below, which illustrates an example of a controller for a car brake system. The starting point for the design is to associate an instance of the template with each actuator type in the system. In this case, there are four actuators, each controlling the brakes on one wheel. Individual sensor processes are combined into a single wheel monitoring process to monitor sensors across all wheel. This monitors the state of each wheel to check whether the wheel is spinning or locked. A separate process monitors the pressure on the driver's brake pedal.  The Wheel monitor process monitors whether or not each wheel is turning. If a wheel is skidding (not turning), it informs the Analysis process. This then signals the processes associated with the wheels that are skidding to initiate anti-skid braking

# 46 : **The Process Pipeline pattern**

The last architectural pattern is Process Pipeline

Many real-time systems are concerned(cần sơ) with collecting analog data from the system's environment. They then digitize(đi di thai) that data for analysis and processing by the system. The system may also convert digital data to analog data, which it then sends to its environment.

The data processing involved in many of these systems has to be carried out very quickly. Otherwise, incoming data may be lost and outgoing signals may be broken up because essential information is missing. The Process Pipeline pattern makes this rapid processing possible by breaking down the required data processing into a sequence of separate transformations

Here is a brief description of pipeline partern

47 : this figure shows the process architecture for this pattern

# 48 : Neutron flux data acquisition

An example of a system that may use a process pipeline is a high-speed data acquisition(ách qui si xừn) system.

This figure is a simplified model of a data acquisition system that might be part of the control software in a nuclear reactor. This system collects data from sensors monitoring the neutron flux (the density of neutrons) in the reactor. The sensor data is placed in a buffer from which it is extracted and processed. The average(e vơ rueet ) flux level is displayed on an operator's display and stored for future processing.

# 49: The third main content section is about: timing analysis

# 50 : **Timing analysis**

It includes some information such as:

- The correctness of a real-time system depends not just on the correctness of its outputs but also on the time at which these outputs were produced.
- In a timing analysis, you calculate how often each process in the system must be executed to ensure that all inputs are processed and all system responses produced in a timely way. = > Timing analysis for real-time systems is particularly difficult when the system has to deal with a mixture of periodic and aperiodic stimuli and responses. Because aperiodic stimuli are unpredictable, you have to make assumptions about the probability of these stimuli occurring and therefore requiring service at any particular time
- The results of the timing analysis are used to decide how frequently each process should execute and how these processes should be scheduled by the real-time operating system.

# 51 : **Factor in timing analysis**

When you are analyzing the timing requirements of embedded real-time systems and designing systems to meet these requirements, you have to consider three key factors:

# 53Powerfailure timing analysis

53: To continue the example of a power supply failure, let's calculate the worstcase execution time for a process that switches equipment power from mains power to a battery backup. this Figure presents a timeline showing the events in the system:

# 56Timing requirements for the burglar alarm system

56 : The starting point for timing analysis in a real-time system is the timing requirements, which should set out the deadlines for each required response in the system.

This figure shows possible timing requirements for the office building burglar alarm system discussed before

# 57 : Stimuli to be processed

To simplify this example, let us ignore stimuli generated by system testing procedures and external signals to reset the system in the event of a false alarm. This means there are only two types of stimulus processed by the system:

# 58: **Frequency and execution time**

To ensure that you meet the deadlines defined by the timing requirements, you then have to decide how frequently the related processes have to run and how many sensors should be examined during each execution of the process. There are obvious trade-offs here between frequency and execution time:

# 59 : **Alarm process timing**

When you have completed the timing analysis, you may then annotate the process model with information about frequency of execution and their expected execution time. Here, periodic processes are annotated(e nơ tây tựt) with their frequency, processes that are started in response to a stimulus are annotated with R, and the testing process is a background process, annotated with B.

The final step in the design process is to design a scheduling system that will ensure that a process will always be scheduled to meet its deadlines

In allocating process priorities, you have to consider the deadlines of each process so that processes with short deadlines receive processor time to meet these deadlines.

# 60 : The last main content section is Real-time operating systems

The execution platform for most application systems is an operating system that manages shared resources and provides features such as a file system and runtime process management. However, the extensive functionality in a conventional operating system takes up a great deal of space and slows down the operation of programs. Furthermore, the process management features in the system may not be designed to allow fine-grain control over the scheduling(sờ két đu linh) of processes. For these reasons, standard operating systems, such as Linux and Windows, are not normally used as the execution platform for real-time systems.

# 62 **Real-time operating systems**

What is the realtime operating systems

# 63 **Operating system components**

The components of an RTOS depend on the size and complexity of the real-time system being developed. For all except the simplest(sim- pồ - lít) systems, they usually include:

# 64 **Non-stop system components**

64 Real-time operating systems also include **Non-stop system components, there are :**

# 65 Components of a real-time operating system

This is a model of real-time operating system components, inclue some component I talked before

# 71RTOS actions required to start a process

The actions taken by the operating system to manage the periodic process are shown in the figure below

# 73 Scheduling strategies

At any one time several processes, all with different priorities(pờ roai ơ ri ti), could be executed. The process scheduler implements system-scheduling policies that determine the order of process execution. There are two commonly used scheduling strategies:

# KeyPoint

Finally, there are key-points about the content of the real-time software - engineering section

# Question

Next, I have some questions for you.\

Can anyone give me the answer?

A, Please answer the question

…

This is the end of my group's presentation. Thank you teacher and everyone for listening. Please give my team feedback.