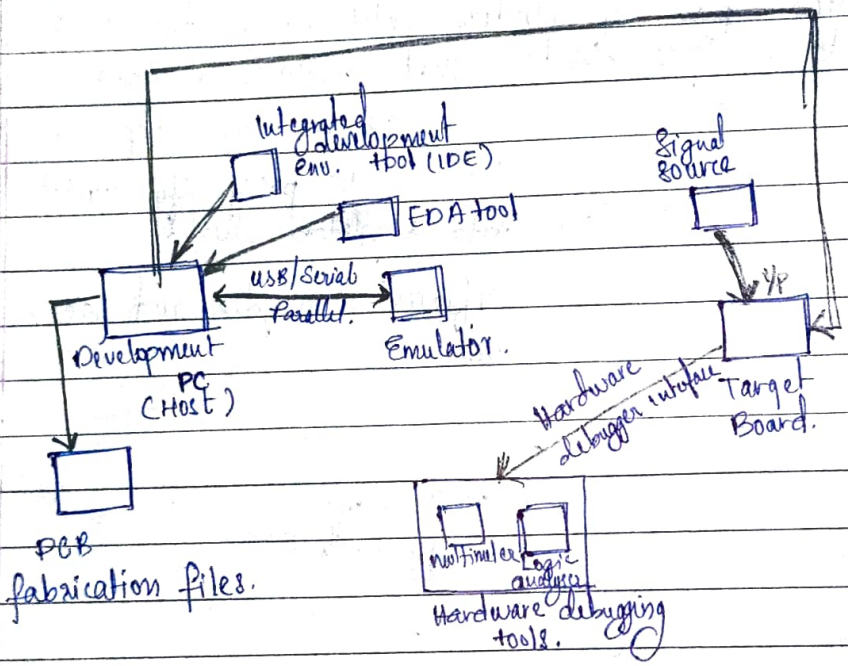
09/06/2021

CS 404
Embedded system.

Christy Varghese
CSE - A
Roll no. 34.

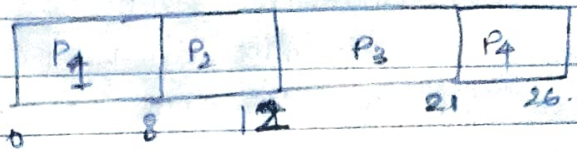
## (1) Embedded system development environment.




- Host Computer: Acts as the heart of development environment.
- IDE Tools: Tools for firmware design & development.
- Electronic Design Automation Tools
  - Embedded Hardware Design: * IDE & EDA are selected based on the target hardware.
    * They are supplied are installable files.
- Emulator hardware: Debugging target board.
- Signal Source (func. generator)
- Target H/w.
- Target H/w Debugging tools.
- CRO, Multimeter, Logic Analyser.

(6)

(a) FCFS.




$P_1 \rightarrow 0-0 = 0$

$P_2 \Rightarrow 8-1 = 7$

$P_3 \Rightarrow 4-2 = 2$

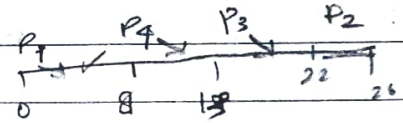
$P_4 \Rightarrow 9-3 = 6$

average waiting time $= \frac{(0+7+2+6)}{4} = \frac{15}{4} = 3.75$ ms

average turnaround time $= \frac{(8+12+21+26)}{4} = 16.75$ ms.

(b) LCFO

| | AT | BT | Priori | Comp time | TAT | WT |
|---|---|---|---|---|---|---|
| $P_1$ | 0 | 8 | 4 | 8 | 8-0=8 | 0 |
| $P_2$ | 3 | 5 | 2 | 13 | 13-3=10 | 5 |
| $P_3$ | 2 | 9 | 3 | 22 | 22-2=20 | 11 |
| $P_4$ | 1 | 4 | 1 | 26 | 26-1=25 | 21 |




avg TAT $= \frac{8+10+20+25}{4} = 15.75$ ms

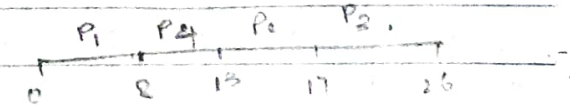
avg WT $= \frac{0+5+11+21}{4} = \frac{37}{4} = 9.25$ ms

(c) Non Preemptive Priority based.

| | Priority | AT | BT | CT | TAT | WT |
|---|---|---|---|---|---|---|
| $P_1$ | 4 | 0 | 8 | 8 | 8 | 0 |
| $P_2$ | 2 | 1 | 4 | 17 | 16 | 12 |
| $P_3$ | 3 | 2 | 9 | 26 | 24 | 15 |
| $P_4$ | 1 | 3 | 5 | 13 | 10 | 5 |

avg TAT $= (8+10+16+24)/4 = 14.5$ ms.

avg WT $= (0+5+12+15)/4 = 8$ ms



Page No :
Date :

(3) (*) Each device event has the codes for an ISR, which executes only on scheduling it by the RTOS and provided only on scheduling it by the RTOS and provided an interrupt is pending for its service.

(*) Assume that using an RTOS, the touch screen ISR, ISR_TouchScreen has been created using a function OS_ISR_Create().

(*) The ISR can share the memory heap with other ISRs.

(*) A func., but connect connects the touch screen event with the event identifier in an interrupt handler, ISR_handler.

(*) when a touch screen event occur on tap at the screen to select icon or menu the OS sends signal on behalf of the ISR handler to the waiting ISR_Touch Screen.

(*) ISR_Touch Screen runs on an interrupt call message.

(*) ISR_touch Screen executes as per its priority, IST_touch screen priority among the other pending ISRs before it starts executing.

(*) Before return from the ISR_TouchScreen, it sends a message to kernel using a function OS_eventPost() or OS_ISR_Exit() just before the end of the codes in the ISR_TouchScreen.

(2) A simulator is designed to create an environment that contains all of the s/w variables and configurations that will exist in an applications actual production environment.
An emulator does attempt to mimic all the hardware features of a production environment, as well as s/w features.
The emulator h/w contains necessary emulation logic and it is hooked to the debugging application running on the development PC on one end and connects to the target board through some interface on the other end. In summary, the simulator 'simulates' the target board CPU and the emulator 'emulates' the target board CPU

(5) There are diff. methods for embedding firmware in hardware:

(i) Out of Circuit Programming: The processor or memory chip into which the firmware needs to be embedded is taken out of the target board and its programmed with the programming device.
The programmer contain zif socket locking pin to hold the device to be programmed.

(ii) In System Programming: The code or data is programmed one byte at a time by supplying the address and data together with the appropiate write instruction. The selected memory location is first erased before the new data is written. After successfully programming the device, set RST to low or turn off the chip power supply & turn it to ON to commence the normal option.

(iii) In Application Programming: It's a technique used by firmware running on the target device for modifying a selected portion of the code memory. It modifies the program code memory under the control of embedded application including updating calibration data, look up tables etc. which are stored in embedded applications.

(iv) Use of factory Programmed chip: Here embeds the firmware into the target processor/controller memory at the time of chip fabrication itself. Such chips are called factory programmed chip.
They are bit expensive. It is not recommended as the firmware goes frequent modifications.

(4) Recent trends in Embedded computing

* Visualization in Real Time.: Developers currently lack tools for monitoring and visualizing the embedded industrial systems in real time. The industry is working on real-time visualization tools that will give s/w engineers the ability to review the embedded s/w execution. These tools will enable developers to keep a check on key metrics.

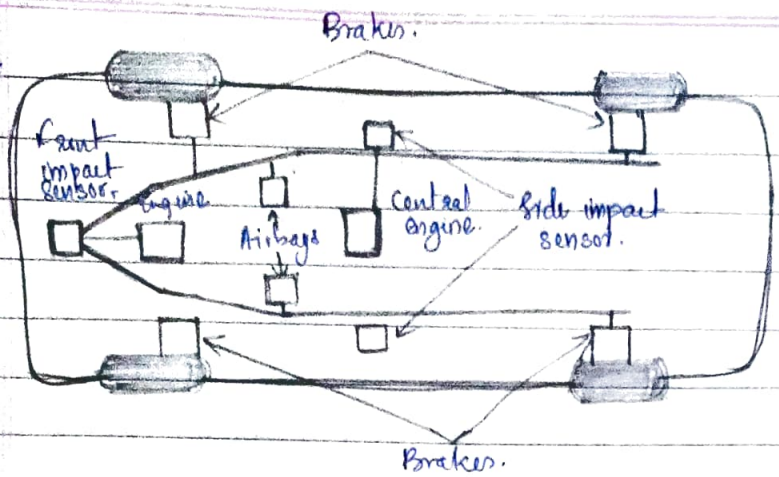
* Deep Learning : It represent a rich, yet unexplored embedded systems market that has a range of applications from image processing to audio analysis.

* Healthcare ; Electronic medical device and other technological innovations with the convergence of biotech, nanotech, sensor technologies are making epic transformations in healthcare delivery and creating new health care paradigms.

* Improved security : with the rise of IOT, the primary focus of developers and manufacturers is on security.

* cloud computing & Mesh n/w: Getting embedded industrial systems connected to the internet and cloud can take weeks and months in the traditional development cycle.

(ग).




Automated braking system of a automated motor car.

A distributed embedded system can be organized in many different ways, but its basic units are the PE and the n/w. A PE may be an instruction set processor such as DSP, CPU or microcontroller, as well as a nonprogrammable unit such as the ASICs used to implement PE. It is also possible that the system can use more than one n/w.

~~The distributed are~~

In the distributed system architecture of the autonomous car, the functional subcomponents are implemented into several local computing units. The computational complexity of the system can be reduced through the DSA. The overall autonomous driving algorithm can be large and complex depending on the mission objective. The planning function determines the behaviour and motion of the autonomous car based on the information from preception and localization. The control function follow the derived command from the planning function by steering, accelerating and braking the autonomous car.