28/04/21
Wednesday

CS 404
EMBEDDED SYSTEM.

Christy Varghese
CSE-A Roll No. 34

(1) Embedded System — General Purpose Computer

Operating System

It may or may not contain an operating system. — It contains general purpose operating system.

Key factors

Application specific requirements are key factors — Performance is the key factor.

Power consumption

Less — More.

(3) Embedded firmware design Approaches.

It depends on the — Speed of operation required
— Complexity of the func. to be performed.

(1) Super loop based approach

- This approach is applied for the applications that are not time critical and the response time is not so important.
- Similar to the conventional procedural programming where the code is executed task by task.
- It does not require an OS, since there is no need for scheduling which task is to be executed from the point where it got interrupted.

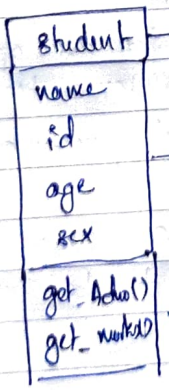
(iii) Embedded OS based Approach

- Contains OS which can be either a general purpose OS (GPOS) or real Time OS (RTOS).
- GPOS based design is very similar to the conventional PC based Application development where the device contain an OS and you will be creating & running user applications on top of it.
- RTOS respond in a timely and predictable manner to events.

Code of temperature checking systems

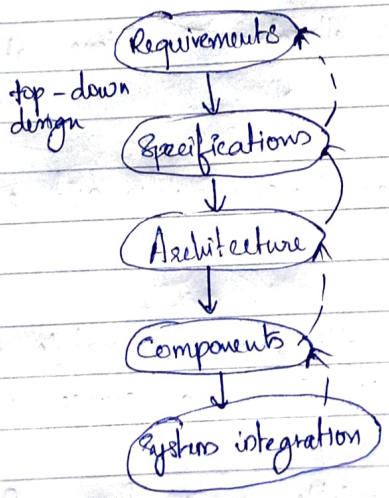
* Get a temperature measurement from the patient.
* If the temperature is above 100°F then the LED indicator blinks fast.
* If when the temperature is below 100°F then the LED indicator blinks slow.

(4) Unified Modeling Languages (UML) was designed to be useful at many levels of abstraction in the design process. UML is useful because it encourages design by successive refinement and progressively adding details to the design.



- The class has the name used as an instance of class student.
- The student class defines the personal details of the students. The student will have its own details so that different students of the same class have their own values for the attributes.
- Other classes can examine and modify class attributes if we have to do something more complex than use the attributes directly, we define a behaviour to perform that function.
- A class defines both the interface for a particular type of student and the student's implementation.
- When we use an object, we don't directly manipulate its attributes.

(5) Embedded System Design Process.



→ In top-down view, we start with the system requirements.
→ In the next step, specification, we create a more detailed description of what we want. The specification states only how the system behaves, not how it is built.
→ The details of the system's internals begin to take shape when we develop the architecture, which gives the system structure in terms of large components. Once we know the components we need we design those

components, including both s/w & h/w. Based on those we can build a complete system.

POS)

For sensor

ASSEMBLY LANG to Machine Code

* Each source module is written in assembly and is stored in .src or .asm file.

* Each file can be assembled separately

* On assembling of each .src/.asm file a corresponding object file is created with extension .obj.

* It can be placed at any code memory location & it is responsibility of the linker/loader to assign absolute address for the module.

* Absolute address allocation is done at absolute object file creation stage.

* Each module can share variables and subroutine among them.

* Exporting a variable from a module is done by declaring that variable as PUBLIC in source module.

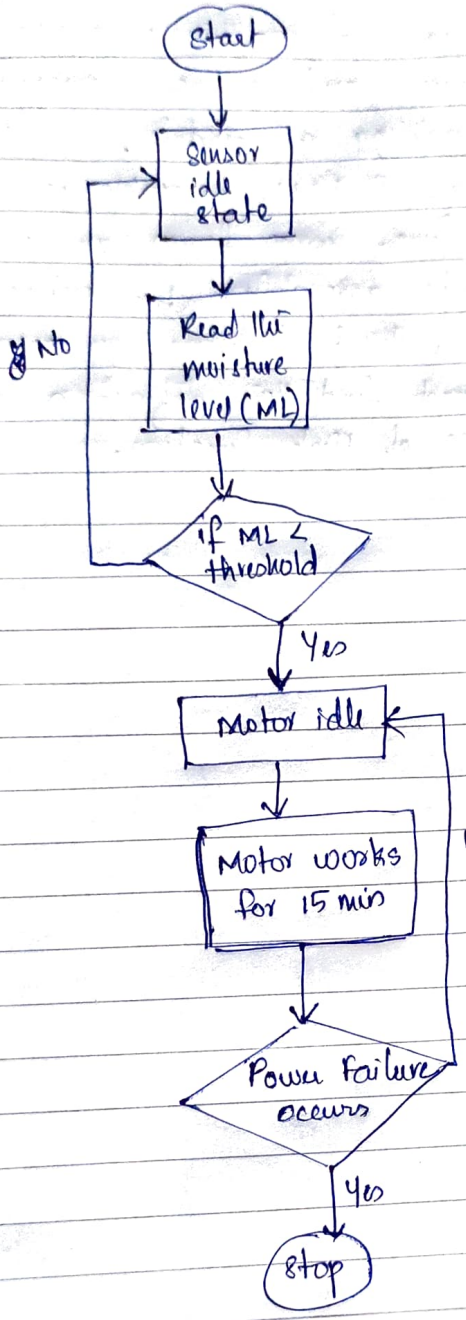
* Importing a variable or a func.. from a module is done by declaring that variable or func. as EXTRN in the module. where it is going to be accessed.

* ~~Corresponding~~ to a variable/function declared as PUBLIC in a module

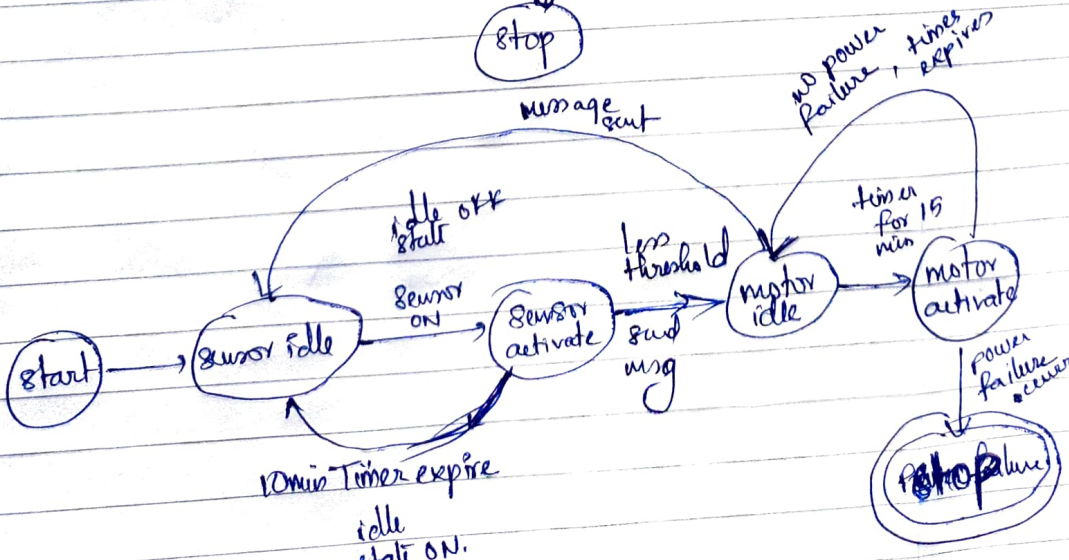
(*) PUBLIC keyword inform the assembler that the variable / func.. need to be exposted.

(*) while assembling a module, on seeing variable / func. with keyword EXTRN, assembler understand that these variable or func. come from an external module and it proceeds assembling the entire module ~~and it proceeds as~~ without throwing any error, through the assembler cannot find the definition of variable & implementation of that function.

(2)


Start
Sensor idle state
Read the moisture level (ML)
if ML < threshold
No
Yes
Motor idle
Motor works for 15 min
NO.
Power failure occurs
Yes
Stop

(6)


start
Sensor idle
Sensor ON
Sensor activate
Less threshold
Send msg
motor idle
timer for 15 min
motor activate
power failure occurs
Stop failure
message sent
idle OFF state
No power failure, timer expires
10min Timer expire
idle state ON.