ISSN (e): 2250 – 3005 || Volume, 07 || Issue, 07|| July – 2017 ||
International Journal of Computational Engineering Research (IJCER)

Controllers in Embedded Systems

R Yazhini Priyadharshini
4th Year, SRM University

Abstract: *All around us we are filled with electronic gadgets that make up our world. No longer do we hang onto our retro stereos. About almost every part of our lives has become digitalized. The uses of embedded systems are virtually limitless, because every day new products are introduced to the market that utilize embedded computers in novel ways. Microcontrollers, microprocessors are emerging every day cheaper and smaller than the day before. Olden day's big, heavy cameras have been elevated by small, lighter digital cameras. About 85% of the adults use cellphones all over the world for going online according to a recent survey. 74% of adults ages 18-34 own an mp3 player. This paper highlights the most important feature of an embedded system and its various functions by giving out important examples.*

Keywords: *Controllers, device drivers, discrete control, functions, fault handling, fault management, retry loops*

I. INTRODUCTION

An embedded system is a device that has been built to perform a particular function and it cannot be easily changed. It has a processor and a software, input and an output. The word embedded means it is built into the system. It is a permanent part in a larger system. For example, a controller is embedded in a washing machine and tells the motor inside the tub to rotate a number of times based what the user requires. In a television, a decoder is embedded in a satellite television setup box to read a signal from the dish and send it to the TV which understands. Often this type of system must do its work in a specific amount of time. This is called real-time computing. Time is a huge factor which comes in front when designing an embedded system. Take a rocket launch as an example, time plays a vital role. While designing the path of a rocket several factors must be kept in mind. Orbital launch vehicles commonly take off vertically, and then begin to progressively lean over, usually following a gravity turn trajectory. Once above much of the atmosphere, the vehicle then angles the rocket jet, pointing it largely horizontally but somewhat downwards, which permits the vehicle to gain and then maintain altitude while increasing horizontal speed. As the speed grows, the vehicle will become more and more horizontal until at orbital speed, the engine will cut off. As one can notice, the altitude, position changes every second. To feed in this piece of information to the system one requires precision and accuracy. Any fault in this it may lead to the failure of a rocket launch. This paper provides a detailed explanation in simple terms as to the various functions in a controller. It can be used by various technicians to debug a system.

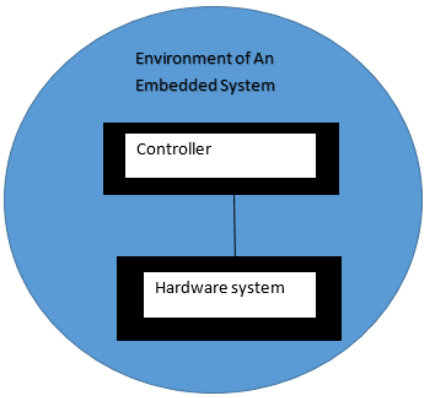

Figure 1- Environment of an embedded system

II. TERMINOLOGIES

DEVICE DRIVERS: The embedded system consists of a controller and a hardware system (rotors, stators, etc.) Device drivers are also included. These device drivers (commonly referred to as a driver) are computer programs that operates or controls a device that is attached to a computer. A driver provides a software interface to hardware devices, enabling operating systems and other computer programs to access hardware functions without needing to know precise details of the hardware being used. A common example is when you install an

app in one's laptop a screen pops out with the words "Welcome to device driver wizard". The major purpose for a device driver is to act as a translator between hardware components and the operating system or the application.

NETWORK INTERFACE CARD (NIC): A network interface card (NIC) is a circuit board or card that is installed in a computer so that it can be connected to a network. A network interface card provides the computer with a dedicated, full-time connection to a network. Personal computers and workstations on a local area network (LAN) typically contain a network interface card specifically designed for the LAN transmission technology.

NETWORK PINS: Network pins are act as connections through which a function/condition/inputs are given to the controlled device so that accordingly the output is also changed according to the various conditions. The input is given as below. If there are 10 pins as below, 2 pins take part in the passage of data. Therefore, the total possible combinations are 10C2 = 1024.

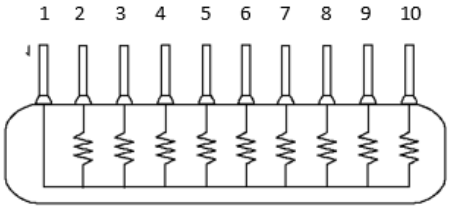

Figure 2- Network Pins

If suppose the first pins 1 and 2 are for a form of data, another combination is used for another set of data. The number of pins changes widely and used according to the data input.

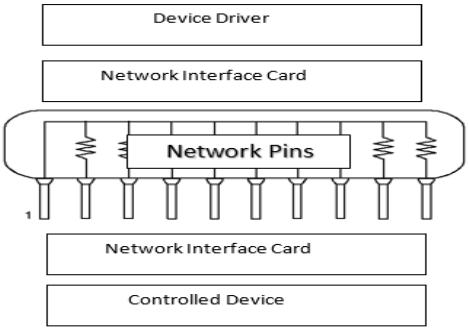

Figure 3- How network pinks are placed in a system

CONTROLLERS: Controllers are devices which can be considered a self-contained system with a processor, memory and peripherals. Most controllers in use today are embedded in other machinery, such as automobiles, telephones, appliances, and peripherals for computer systems. They control the entire system.

III. FUNCTIONS OF CONTROLLERS

A controller has 2 major functions- Lifetime management which leads to the desired output.
For example, take the case of a laptop- initially when we buy a new laptop and start we initialize the laptop that is we set the date, time, and we install certain applications required. Then in case use the system and the applications whenever we require. When initialization, the system itself runs a few background checks on itself. This checking does not stop- it is a never-ending process. The system frequently checks for any update and upgrades. In case of any faults within the system it itself tries to repair the fault else it pings us and we repair it with the help of the system or a software. All of this is just a outlook of what a controller does. Below are the functions of a controller.

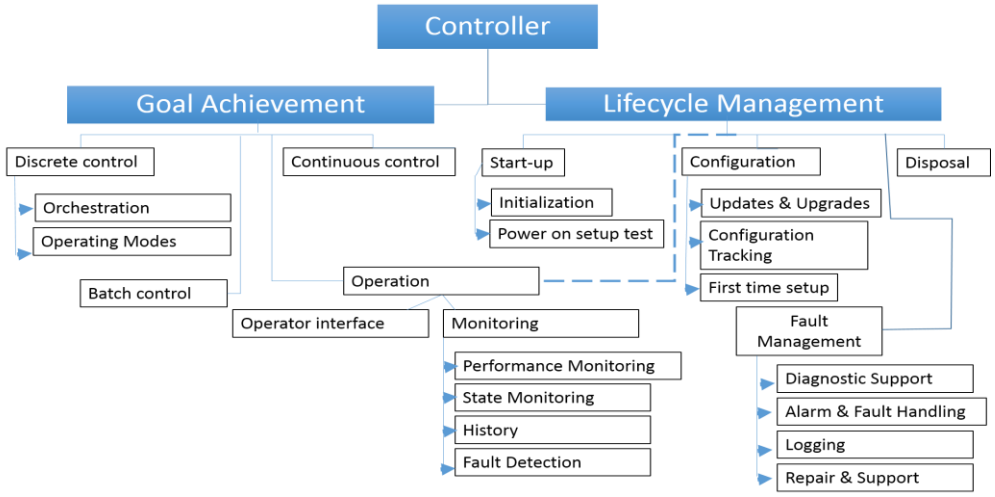

Figure 4- Functions of a controller- flowchart

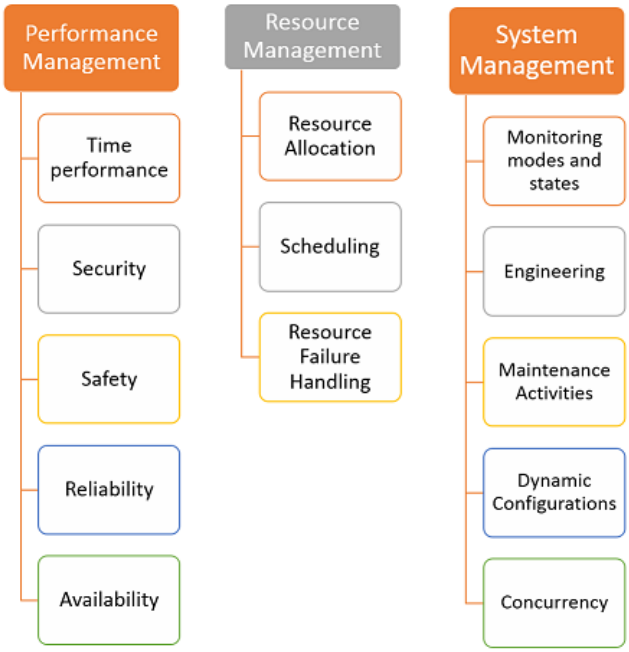

Figure 5- Various subsections in Performance Management, Resource Management and System Management

Both resource management and system management come under state monitoring.
While goal achievement is merely how the goal is achieved and about how much the desired and actual values are same, lifecycle management is looking at a system from its stage of initialization, going over to updates and upgrades, and till the point in which it gets disposed or in other words uninstalled.
The above is explained in detail:

3.1 Configuration:

3.1.1 First- time setup:

Takes places when we start the system for the first time.
It usually is in right relationship with the environment that is, it is sometimes independent of the user's help that is the very systems we use need to change a few settings on its own for its convenience. Why, when we who require the system change its setting to meet up our demands, it which satisfies them all need a few modifications for its content!
When we switch on a device say a laptop, sometimes it starts optimizing itself which is initializing the basic requirements to start or run programs in a laptop.
But this is not always. Most of the time, we are required to setup a system.
For example, in our printer, alignment of ink and paper takes place on its own. For example, when we operate our phones for the first time, we need to set up the date, time, language preferred, etc.

In washing machines, we need to first let the water run through before using it for the first time.

3.1.2 Updates & Upgrades:

Upgrading is the process of replacing a product with a newer version of the same product. Updates are fixing or improving the system. It does not essentially mean changing it completely. This includes fixing security vulnerabilities and other bugs, and improving the usability or performance. Though there are options to update a system, there aren't much options to rollback or in other terms de-upgrade/de-update it.

A rollback is an operation which returns the database to some previous state. Rollbacks are important for database integrity, because they mean that the database can be restored to a clean copy even after erroneous operations are performed. They are crucial for recovering from database server crashes; by rolling back any transaction which was active at the time of the crash, the database is restored to a consistent state. Since the settings are different to that of the previous server, it would be difficult to return to the previous settings. Therefore, it's very difficult to change it without any technical support. Hence rollback is done in very rare cases.

3.1.3 Configuration Tracking:

Configuration tracking is the task of tracking and controlling changes in the software, part of the larger cross-disciplinary field of configuration management. If something goes wrong within the system, configuration tracking can determine what was changed and who changed it. If a configuration is working well, it can determine how to replicate it across many hosts.

The goals of configuration tracking are generally

Configuration identification - Identifying configurations.

Configuration control - Implementing a controlled change process. This is usually achieved by setting up a change control board whose primary function is to approve or reject all change requests that are sent against any system.

Configuration status accounting - Recording and reporting all the necessary information on the status of the development process.

Configuration auditing - Ensuring that configurations contain all their intended parts and are sound with respect to their specifying documents, including requirements, architectural specifications and user manuals.

Build management - Managing the tools used for builds.

Process management - Ensuring adherence to the organization's development process.

Environment management - Managing the software and hardware that host the system.

Teamwork - Facilitate team interactions related to the process.

Defect tracking - Making sure every defect has traceability back to the source.

With the introduction of cloud computing the purposes of configuration tracking tools have become merged in some cases. The tools themselves have become virtual appliances that can be instantiated as virtual machines and saved with state and version. The tools can manage cloud-based virtual resources, including virtual appliances, storage units, and software.

3.2 Start- Up:

3.2.1 Initialization:

The hardware gets ready for processing of the device.

For example, before we start using phones, we charge our phones for a few hours.

In refrigerators, we let it cool for some time before using it.

Actuators and sensors are run through and processed. Actuators are used for opening inlet valves while sensors are used to detect changes in the system (when ink levels are low or if there are no papers in the printer tray or water in not sufficient).

3.2.2 Power On Self-Test:

Also, known as 'built-in self-test', this test is designed to check the machine's output. It is basically comparing the desired value and actual value so that the amount of error can be measured.

The defects in the system can be noted. This is done by the system itself or the environment.

Suppose the paper gets stuck in the printer (often called a paper jam), here both the environment as well as the system participate in the correction. The printer lets us know there is a paper jam and aids us in the removal of the paper.

More often power on self-test is an automatic test that takes place when we switch on a system. This takes place every time we switch the device on.

For example, in our printers, when we switch it on, we would be able to hear a sound that tells us the printer moves the ink to check the conditions of the printer.

In our laptops, we come across a screen which says “System is configuring. Please do not switch-off”.

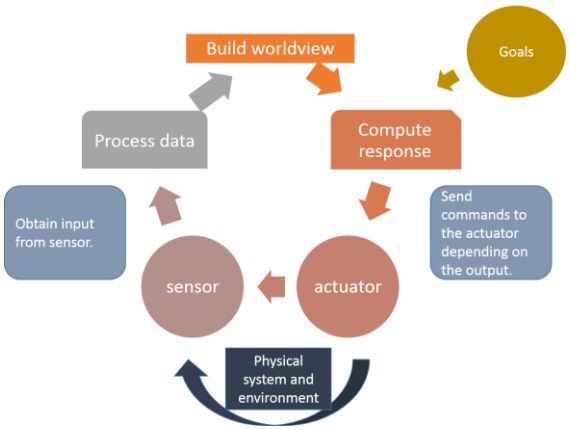

Figure 6- Sensors and Actuators

The above is explained with the help of an example.

Consider a robot which is built to catch a ball. Consider it to be a system. This system requires a sensor, actuator and a proper environment and the entire system is dependent on time. The sensor (motion sensor) is used to monitor the position of the ball which is thrown. The actuator is used to raise or lower the arm of the robot. This actuator is dependent on time. Even if the arm too high or delayed by even one second, the ball would not be caught.

Figure 7- How a goal causes an action

The above is how the goal (required output) causes an action to the output when there is an error. The error is measured by the difference in the actual and observed value. This is sent as feedback to the system so that any faults in the system can be handled.

The control theory plays a crucial role here.

It can be explained using the example of wireless connection.

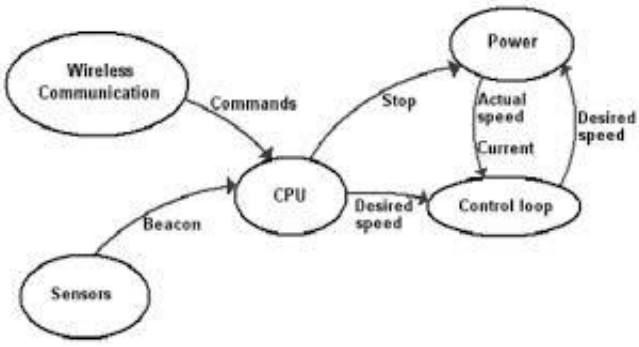

Figure 8- Wireless communication example

Sensors detect the signals (beacon) and send it to the CPU. The desired speed is sent through a control loop which measures the actual speed from the power and the sends the desired speed to the power. Any error occurring (low speed) is notified to the user.

3.3 Goal Achievement
The **goal achievement** is of three types- Discrete, batch and continuous control.

3.3.1 Discrete Control:
They usually happen in steps that is, they have a step wise process.
They are depicted using flowcharts.
Found in many manufacturing, motion and packaging applications. Robotic assembly, such as that found in automotive production, can be characterized as discrete process control. Most discrete manufacturing involves the production of discrete pieces of product, such as metal stamping.

a) Orchestration:
Steps to do a process is decided. A single process is broken step by step.

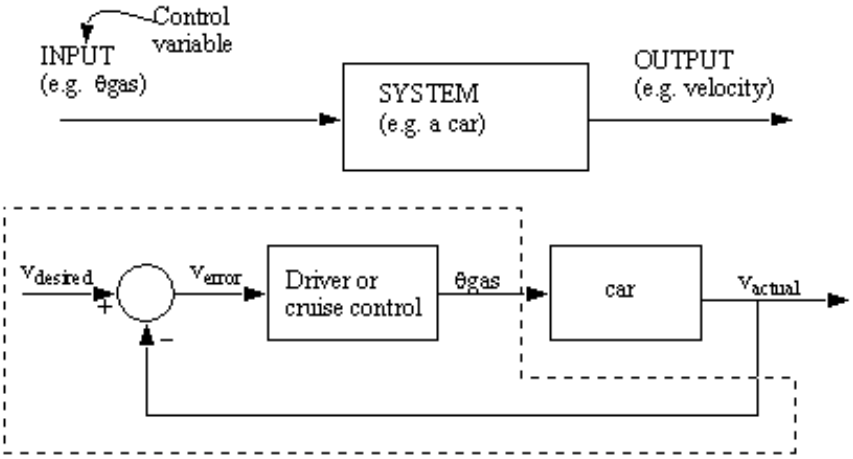

Figure 9- Orchestration steps

Example of A Flowchart: To scan a barcode, a barcode machine does the following steps.

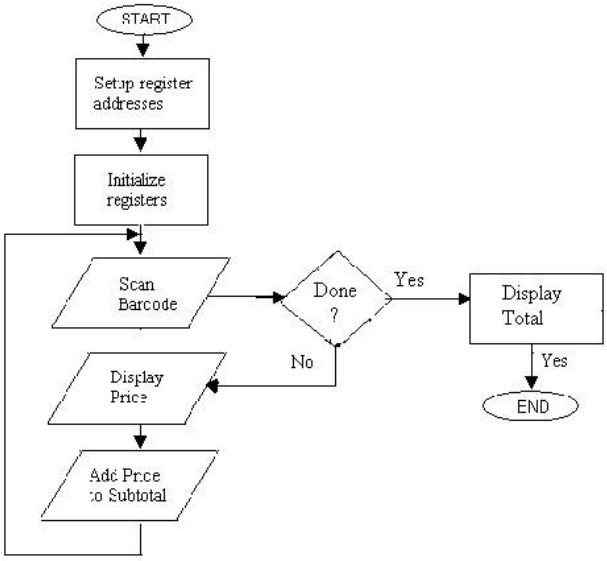

Figure 10- Barcode scanning

Retry Loops:
They are a set of tasks and each has a different procedure- the output varies. Till the correct output is achieved, the system undergoes a series of retry loops where the output is varied. Each procedure may include either a discrete or continuous control.
Example: In flowcharts, we can see two different procedure for ‘yes’ or ‘no’.

b) Operation Modes:

They are several types in which a system behaves. In each mode, the system behaves in a separate way. Each mode has a distinct set of commands.

Examples:

When we use televisions as a computer, some operations we can perform in our computers will be restricted like remotes cannot be used (like how we change channels).

In our mobile phones, we have a list of modes- airplane mode, driving mode, silent mode, vibrate mode, etc. In each a list of features will either be changed or frozen.

3.3.2 Batch

Some applications require that specific quantities of raw materials be combined in specific ways for particular durations to produce an intermediate or end result. One example is the production of adhesives and glues, which normally require the mixing of raw materials in a heated vessel for a period to form a quantity of end product. Other important examples are the production of food, beverages and medicine. Batch processes are generally used to produce a relatively low to intermediate quantity of product per year (a few pounds to millions of pounds).

An example of batch system is as follows:

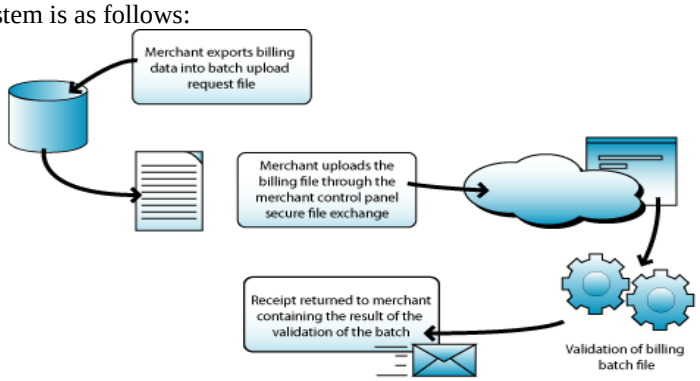

Figure 11- Batch System

3.3.3 Continuous

Often, a physical system is represented through variables that are smooth and uninterrupted in time. The control of the water temperature in a heating jacket, for example, is an example of continuous process control. Some important continuous processes are the production of fuels, chemicals and plastics. Continuous processes in manufacturing are used to produce very large quantities of product per year (millions to billions of pounds).

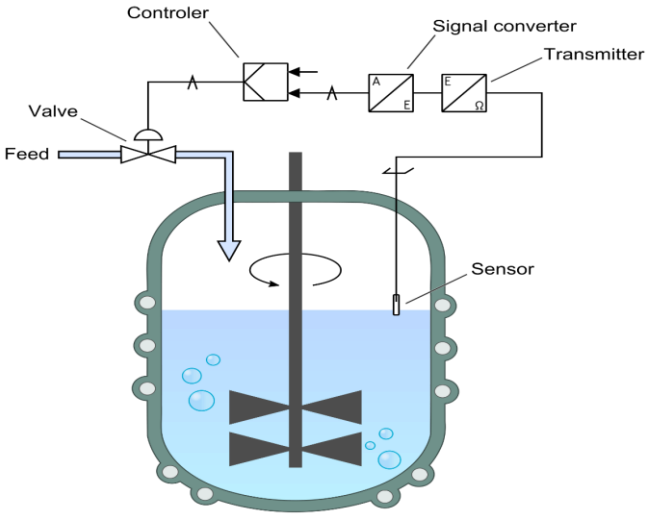

Figure 12- Continuous System

3.4 Operation:

3.4.1 Monitoring:

Monitoring is displaying what is happening within the system. It helps in building a worldwide view. The output depends widely on monitoring. Saving of data helps rectify systems of their previously made errors so that they don't occur again.
It is of 4 types as shown below.

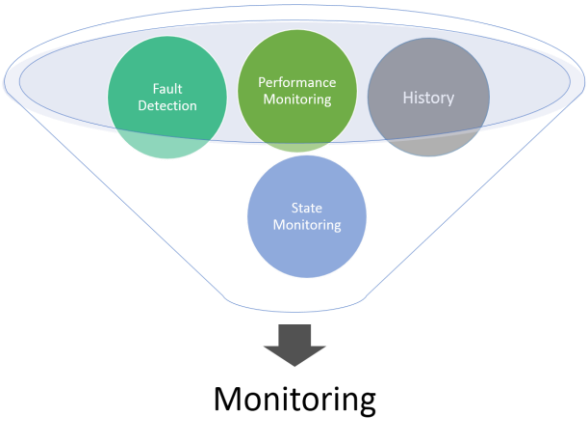

Figure 13-4 types of Monitoring

a) Performance Monitoring and state monitoring:

While performance monitoring widely monitors the performance of a system, State Monitoring indicates whether a managed system is healthy at a given time. Example of performance monitoring is the RAM status in our laptop or the data consumption of a day in our mobiles when we use mobile data.
Example of state monitoring will be the battery levels in our phones or water purity level in a water purifier.

b) History:

It shows the history of the system- the various operations that have been performed. It is later used for fault management in case of any issues in the system or it is merely used for just information.
Examples are the call logs in our mobile phones, the search history on the net.

c) Fault Detection:

Detection of faults within the system is done in four different methods:

- Sensors
- Environment(user)
- Expected vs observatory output
- Software

When the printer says no ink, or shows a notification saying paper jam, those are fault detection taking place by the sensor.
When we notice water coming out of the washing machine or we notice our laptops working very slowly that is the RAM consumption is too high, we become the detectors- environment.
When there is a difference between the expected and the observatory output that also becomes a way of fault detection. Say in an elevator, we select the 4^{th} floor instead if it brings us to the 5^{th} floor, there is some fault within the system or if the washing machine exceeds the time it should run, that too becomes a fault.
Sometimes it is not the system nor the environment but an external source- a software which must detect any faults. Viruses within a laptop are detected with the help of an antivirus software which also has the option of removing it. Though this becomes part of the system, it should be downloaded to be used.

3.4.2 Operator Interface:

Operator Interface is showing what is happening within the system. Also, sending commands to the user to achieve a goal takes place through this interface.
Examples are:
In our mobile phones, the battery level indicator, the information about the various applications running all are a part of the operator interface.
In printers, in case of a paper jam, several steps are shown to the user which helps in removing the jam successfully is also a part of operator interface.

In elevators, in case the door is not closed properly, a notification is done through a beep sound or a voice alert which notifies us of the problem. Now-a-days the doors are closed automatically through a motion sensor.

3.5 Fault Management:

First, we saw how the faults are detected and how they are detected. Next, we will see how they are managed.

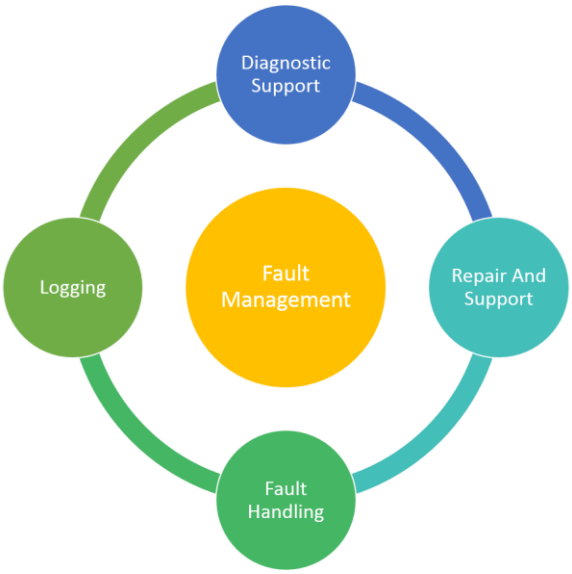

Figure 14- Fault Management

3.5.1 Logging:

Involves recording steps that take place till a time using time as the other coordinate. It involves file handling and the presence of huge memory power. The process of using a computer to collect data through sensors, analyze the data and save and output the results of the collection and analysis. Data logging also implies the control of how the computer collects and analyzes the data. Example is storing the various processes that have taken place in a server for future references. It involves huge file handling. Examples of the types of information a data logging system can collect include temperatures, sound frequencies, vibrations, times, light intensities, electrical currents, pressure and changes in states of matter.

3.5.2 Diagnostic Support:

Diagnostic support is concerned with the development of algorithms and techniques that can determine whether the behavior of a system is correct. If the system is not functioning correctly, the algorithm should be able to determine, as accurately as possible, which part of the system is failing, and which kind of fault it is facing. The computation is based on observations, which provide information on the current behavior.

An example of diagnosis is the process of a garage mechanic with an automobile. The mechanic will first try to detect any abnormal behavior based on the observations on the car and his knowledge of this type of vehicle. If he finds out that the behavior is abnormal, the mechanic will try to refine his diagnosis by using new observations and possibly testing the system, until he discovers the faulty component. It means the mechanic plays a significant role in the vehicle diagnosis. The experience can be provided by a human operator. In this case, the human knowledge must be translated into a computer language. By examples of the system behavior. In this case, the examples must be classified as correct or faulty (and, in the latter case, by the type of fault). Machine learning methods are then used to generalize from the examples.

This is what takes place:

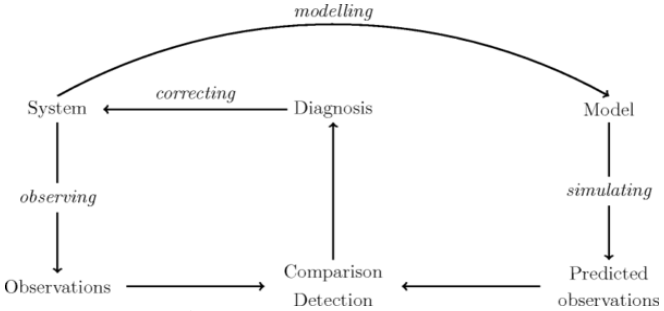

Figure 15- Diagnosis Support

3.5.3 Fault Handling:

Fault handling requires the following steps:

- Fault detection
- Giving notification
- Finding the root cause
- Repairing

The notification is given through an alarm. This brings us again back to the operator interface which is of two types: System and the environment- Either the system itself finds the fault or the environment finds it.

3.5.4 Repair and Support:

Removes the root cause of the fault and involves supporting the system so that the fault does not occur again. Usually both go hand in hand because once a crisis is resolved, it must be stopped from taking place again.

3.5.5 Disposal:

Last but not the least comes disposal. Once a system is not required, a pattern is designed which is used to handle cleanup of data and prevent leaks in runtime environments that use automatic garbage collection. Automatic garbage collection is a form of automatic memory management. The garbage collector, or just collector, attempts to reclaim garbage, or memory occupied by objects that are no longer in use by the program.

Leaks here mean when personal data is not removed but reallocated from one destination to another. The fundamental problem that the pattern aims to solve is that, because objects in a garbage-collected environment have reallocations rather than destructions, there is no guarantee that an object will be destroyed at any deterministic point in time. The dispose pattern works around this by giving an object a method which frees any data the object is holding onto.

IV. CONCLUSION

Embedded systems have virtually entered every sphere of our life, right from the time we work out on tread mills to the cars that we drive today. The possibilities in this field are only limited by our imagination. Many of the embedded systems are managed by human controllers by some sort of man machine interface-for example a cash register, a cell phone, a TV screen or a system interface. Both ways, the system helps us perform our day to day activities in a much easier way than it was before. In this paper, we have seen the most valuable tool every system requires to function properly and to produce the expected output- A controller.

REFERENCES

[1] F. Vahid, T. D Givargis, Embedded Systems Design: A Unified Hardware/Software Introduction

[2] Fernando Pereira, Filipe Moutinho, Luis Gomes, "Model checking frameworks for embedded systems controllers development using IOPT Petri nets", Industrial Electronics (ISIE), 2012 IEEE International Symposium, published on 2012.

[3] D. Sarabia; C. de Prada; S. Cristea, Hybrid Predictive Control of a Simulated Continuous-Batch Process, 2007 IEEE conference oncontrol applications, Year 2007,Pages 1400-1407

[4] Adaptive control strategies for process control: A survey, D. E. Seborg, T. F. Edgar, S. L. Shah, June 1986

International Journal of Computational Engineering Research (IJCER) is **UGC approved** Journal with Sl. No. 4627, Journal no. 47631.

R Yazhini Priyadharshini. " Controllers in Embedded Systems." **International Journal of Computational Engineering Research (IJCER)** 7.7 (2017): 47-56.