UNIT V

VALIDATION AND DEBUGGING : Host and Target Machines - Validation Types and Methods - Host Testing - Host- Based Testing Setup - Target Testing - Remote Debuggers and Debug Kernels - ROM Emulator - Logical Analyzer Background Debug Mode - In-Circuit Emulator(ICE)

1. Arnold S Berger , "Embedded Systems Design - An Introduction to Processes, Tools and Techniques", Elsevier, New Delhi, 2011.
2. Sriram V Iyer, Pankaj Gupta , "Real-time Systems Programming", Tata McGraw-Hill Publishing Company Limited, New Delhi, 2006.

HOST AND TARGET MACHINES, VALIDATION TYPES AND METHODS, HOST TESTING AND HOST BASED TEST SET UP

Dr.N.Arulanand,
Professor,
Dept of CSE,
PSG College of Technology

HOST AND TARGET MACHINES

HOST MACHINE:

- A computer system on which all programming tools run.
- Embedded software is developed, compiled, tested and debugged.

TARGET MACHINE:

- Once program is written , cross- compiled, assembled and linked, it is moved to the target machine.

PERFORMANCE

PERFORMANCE OF HOST MACHINE:

- It is the development platform.
- It has higher capability processor and more memory.
- These tools are freely available.

PERFORMANCE OF TARGET MACHINE:

- The output binary image created by host is executed on target hardware platform.
- **It contains two entities:**
  - Target hardware – processor.
  - Runtime environment – OS.

DIFFERENCE

HOST SYSTEM

- Writing, editing, compiling and linking is done.
- Software development is done.
- Compiler, linker, assembler, debugger are used.
- Stubs are used.
- Programming centric.

TARGET SYSTEM

- After completion of programming work, it is moved to target system.
- Developed software is shifted to customer from host.
- Cross-compilers are used.
- Real libraries are used.
- Customer centric.

TOOLS USED

- **Compiler:** A software program that converts source code that is written in high level programming language into low level language.
- **A Native-Compiler:** Runs on a computer platform and produces code for the computer platform.
- **A Cross-Compiler:** Runs on one computer platform and produces code for another computer platform.
- **Linker:** Program that takes one or more objects generated by compilers and assembles them into single executable program.
- **Locator:** Tool that performs the conversion from relocatable program to executable binary image.

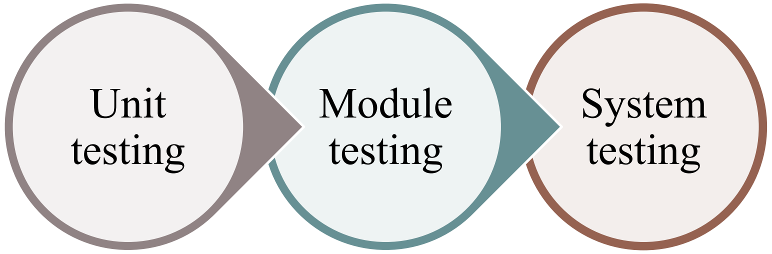
GETTING EMBEDDED SOFTWARE INTO TARGET SYSTEM

- **PROM Programmers:** Creating file in ROM or PROM.
- **ROM Emulators:** Replaces ROM in target system.
- **In-Circuit Emulators:** Uses overlay-memory.
- **Flash:** Software can be directly downloaded into flash without removing the chip.
- **Monitors:** Stores the user software sent from host through communication link to target RAM and runs it.

VALIDATION TYPES AND METHODS

BACKGROUND

LEVELS OF TESTING:



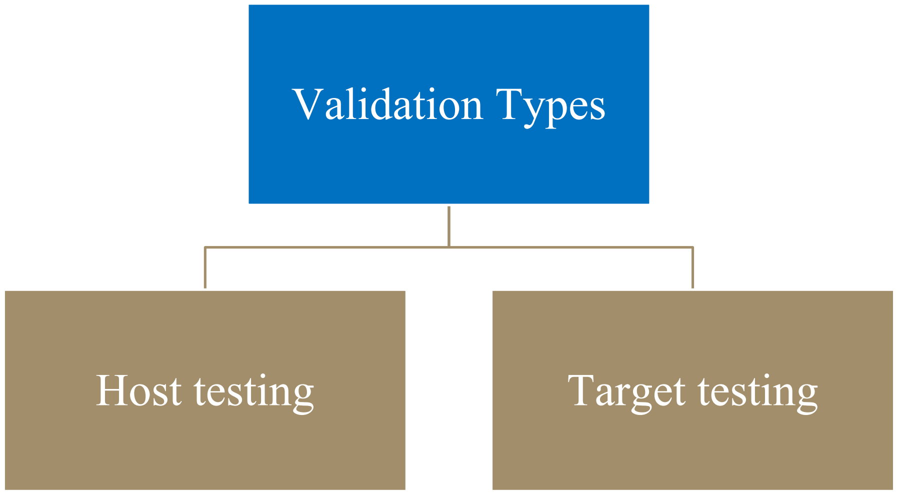
WHY IS SOFTWARE TESTING DIFFICULT?

- **Sensitivity to errors**
  - A near correct value is no better than a wrong value
- **Complexity**
  - Infinite ways to arrive at a code
- **Testing issues**
  - Hard to test all execution paths and decision statements
- **Regression**
  - Should keep an eye on previous bugs resurfacing
- **Simulation of actual environment**
  - Challenging to think of all possible real-time scenario that a software will be put through

HOW IS EMBEDDED TESTING DIFFERENT?

- Embeddedness
- Realtime behaviour
- Difficult to simulate
- Difficult to see outputs
- No downtime

VALIDATION TYPES AND METHODS



HOST TESTING:

- Tested in a host machine
- Software detached from surroundings
- Simulation of neighbouring environment

TARGET TESTING:

- Testing on actual embedded systems
- Actual surroundings or similar replica of conditions

TARGET TESTING IS GOOD, BUT DIFFICULT

INCOMPETE SOFTWARE

- Cannot check quality and validity of individual components
- More expensive

INCOMPETE HARDWARE

- Hardware should be fully up and running before testing
- At least one of hardware or software teams must wait for the other

INCOMPLETE TESTING

- Cannot test all portions of code
- Easier to simulate exceptions triggered by rare failures on host

REGRESSION

- Harder to simulate a bug again in target platform

LACK OF GOOD TRACKING METHODS

- A bug detected once may not appear again
- No storage mechanism to capture all status values in a trace dump

HOST TESTING - LIMITATIONS

NO REALTIME

- Realtime behaviour is difficult to test on host
- Simulators can only help to a limited extent

ACCESS OF PERIPHERALS AND MEMORY

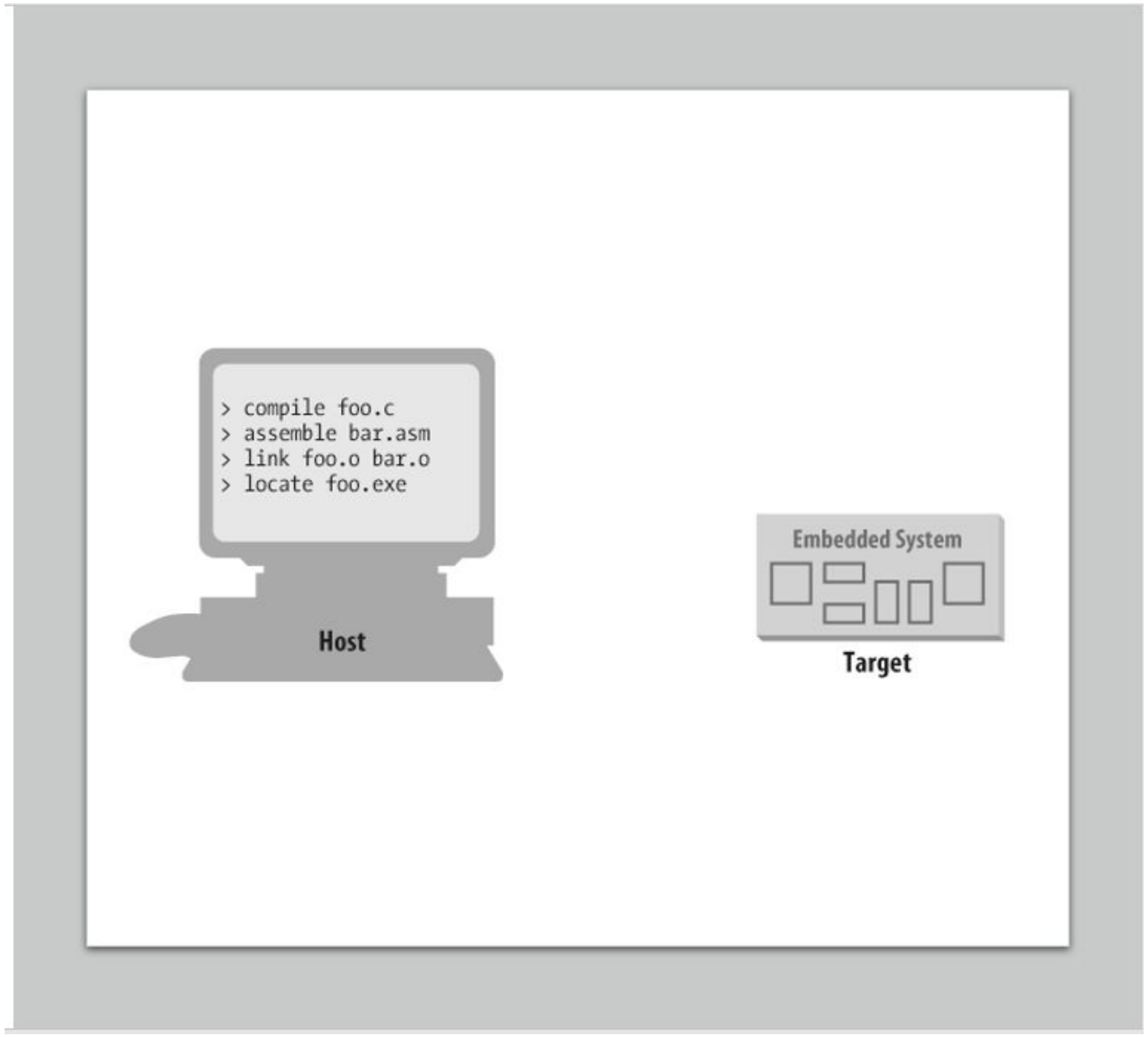
- Shared data problems cannot be detected on host
- Problems related to accessing peripherals and actual field conditions
- Watchdogs can be tested easily in target

HOST TESTING

- Embedded software can be tested either **on-target** or **on-host**.
- In target testing, an application is tested **on the hardware** to be deployed .
- In host testing, an application is tested **on a host computer** that has a different hardware environment to the final application.
- On-host testing is **less resource intensive**, allows testing to be **automated** .
- **Unit testing**, **integration testing** and **system testing** are often used
- When testing embedded software, the best option is to keep your options open so you can perform **both** on-target and on-host testing.

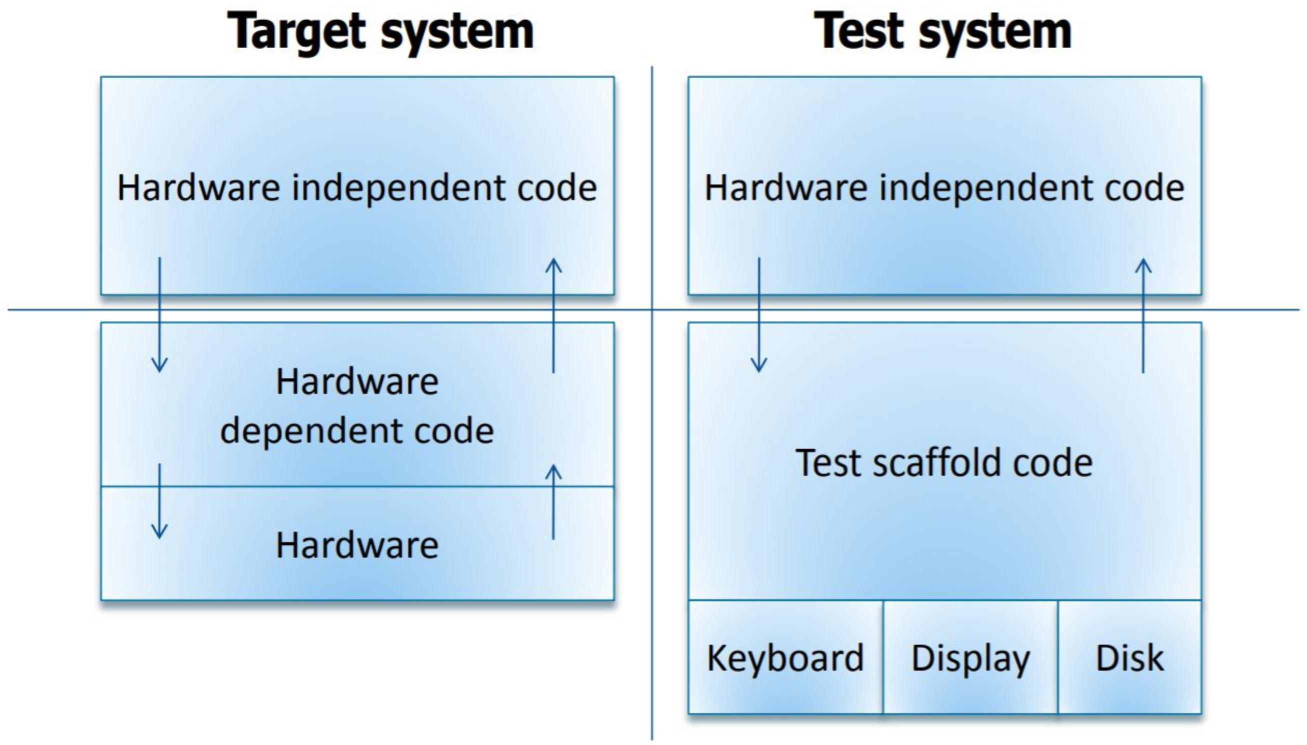
PRIMARY GOALS FOR TESTING ON HOST MACHINE

1. Find bugs in development process
2. Exercise all of the code
3. Repeatable and reusable tests
4. Audit trail of test results



```
> compile foo.c
> assemble bar.asm
> link foo.o bar.o
> locate foo.exe
```

BASIC TECHNIQUE FOR TESTING



- **Scaffold software**, software running on host the target dependent codes and which have same start of code and port and device addresses as at the hardware.
- **Instructions**– given from file or keyboard inputs.
- **Outputs**–at host’s LCD display and saves at file.

CALLING INTERRUPT SERVICE ROUTINES

- Based on the occurrence of interrupts tasks will be executed**.**
- Calling interrupt routines needs to be done from scaffold.
- Difficult? Not really.
- Split code of the ISR in two: HW-dependent and HW-independent.
- Place the HW-independent code in a separate function.
- Test this code extensively.

CALLING TIMER INTERRUPT

- Timer ISR – one of the most used and important parts of the system.
- Alternatives: Emulate HW to "naturally" call this ISR.
- Force calls from the test scaffolds.

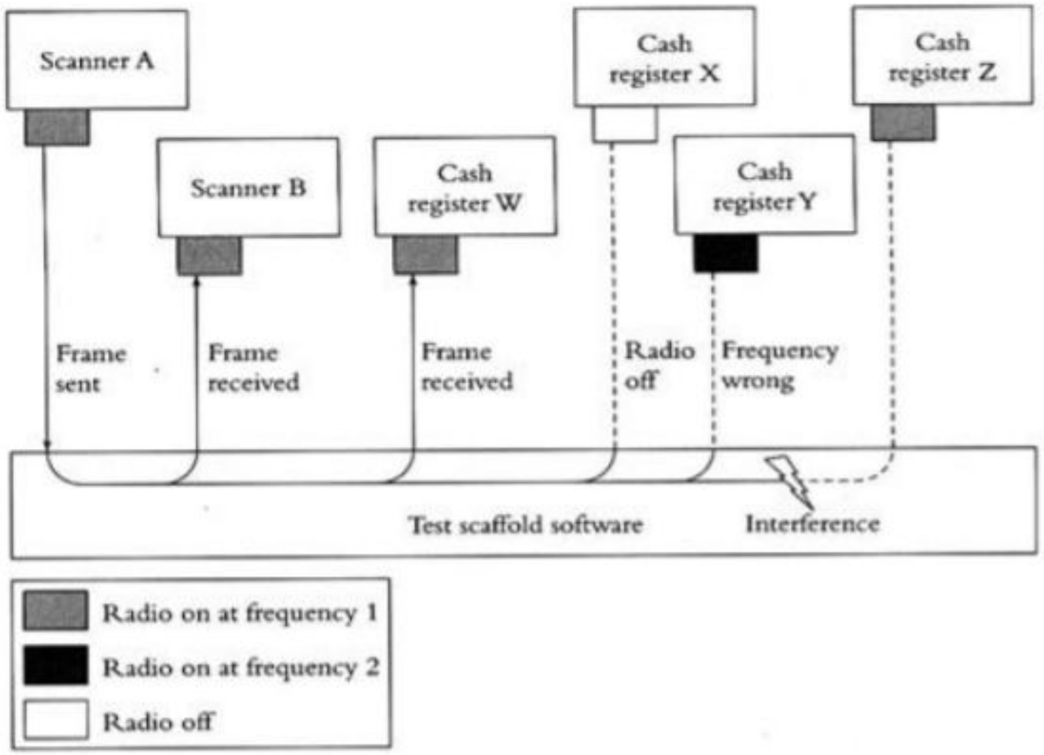
SCRIPT FILES AND OUTPUT FILES

- Script read from - keyboard or from a file.
- **FEATURES OF SCRIPT FILES**

  ✔ Simple commands - two or three letter commands, parser written more quickly.
  ✔ Comments script file indicate - what is being tested
  - results expected
  - version control information etc.

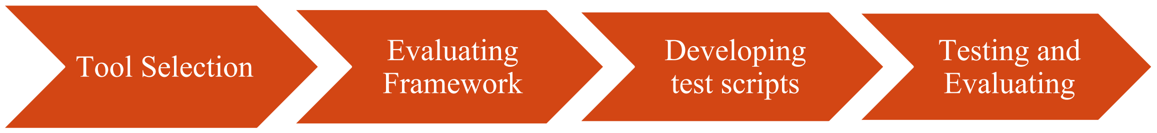
  ✔ Data can be entered in ASCII or in Hexadecimal.

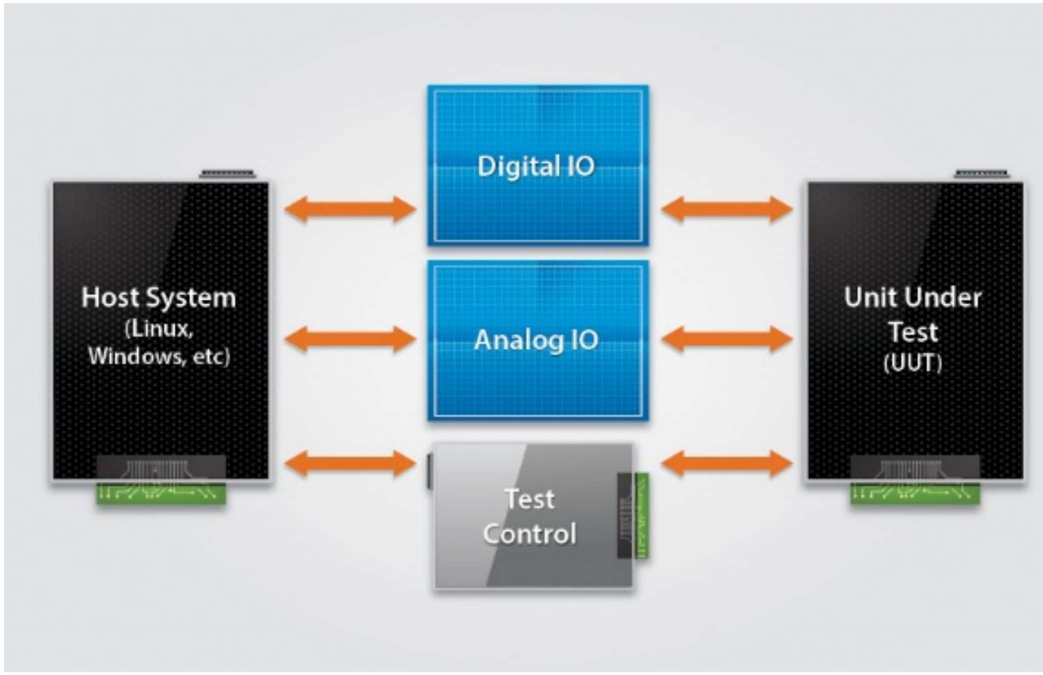
TEST SCAFFOLD FOR THE BAR-CODE SCANNER SOFTWARE



AUTOMATED TESTING

- An automated test is a program that somehow activates parts of the software that is under test and verifies that it behaves as expected.





AUTOMATED TESTS ON THE HOST

Host based unit tests	• Tests a small part (a unit) of the software
Host based integration tests	• Several units can be tested together and their interaction becomes part of the testing
Host based system tests	• Linking all units and the scheduler together on the host

BENEFITS OF AUTOMATED TESTS

- Useful for discovering bugs early
- Brings the embedded software under complete control
- Automated testing is conducted using software tools, so it works without tiring and fatigue unlike humans in manual testing
- Automation process can be recorded. This allows you to reuse and execute the same kind of testing operations

MANUAL VERSUS AUTOMATED TESTING

Manual Testing

- Test cases are executed by a human tester and software.
- Manual testing is time-consuming and takes up human resources.
- Exploratory testing is possible in Manual Testing
- The initial investment in the Manual testing is comparatively lower. ROI is lower compared to Automation testing in the long run.
- Manual testing is not as accurate because of the possibility of the human errors.

Automated Testing

- Automation Testing uses automation tools to execute test cases.
- Automated testing is significantly faster than a manual approach.
- Automation does not allow random testing.
- The initial investment in the automated testing is higher. Though the ROI is better in the long run.
- Automated testing is a reliable method, as it is performed by tools and scripts.

REGRESSION TESTING

Test previously observed bugs in the code

Sanity check:

1 Minimum quality of the software is guaranteed.

Old ghosts:

2 Detect new bugs that were introduced by developers for testing

- Created only after faults have been detected in the system.
- Written on the basis of reports from the field, or from a past error.
- a very remote and complicated path of software has been found to be faulty, software bugs usually exist in groups.
- If a portion of software has been found to be faulty, chances are that more problems can be unearthed by changing the parameters slightly and checking other border conditions.

WHITE BOX TESTING

Performed at a system or module level by a team in order to exercise the most important paths of the source code.

Performed after unit testing

White box testing tests the coverage of the code. It tests the software from the code point of view.

White box testing is unlikely to detect missing code faults. In embedded systems, some parts of the code may be unreachable through this testing.

- White box testing lays its entire focus on the source code being tested.
- In case of slight change in code, because of change in requirements, design or fixing of a major bug, all such tests that depend on the changed code get affected as well.

- The current code needs to be understood and a heuristic needs to be found about what are the most important places in the code that need evaluation.
- In this task, testers need to interact heavily with the developers so that useful insight can be given.

FUNCTIONAL TESTING

01 Looks at the system from a requirements point of view.

02 Functional testing is completely independent of the way the system has been designed and implemented.

03 Functional testing and development teams perform their jobs independent of each other.

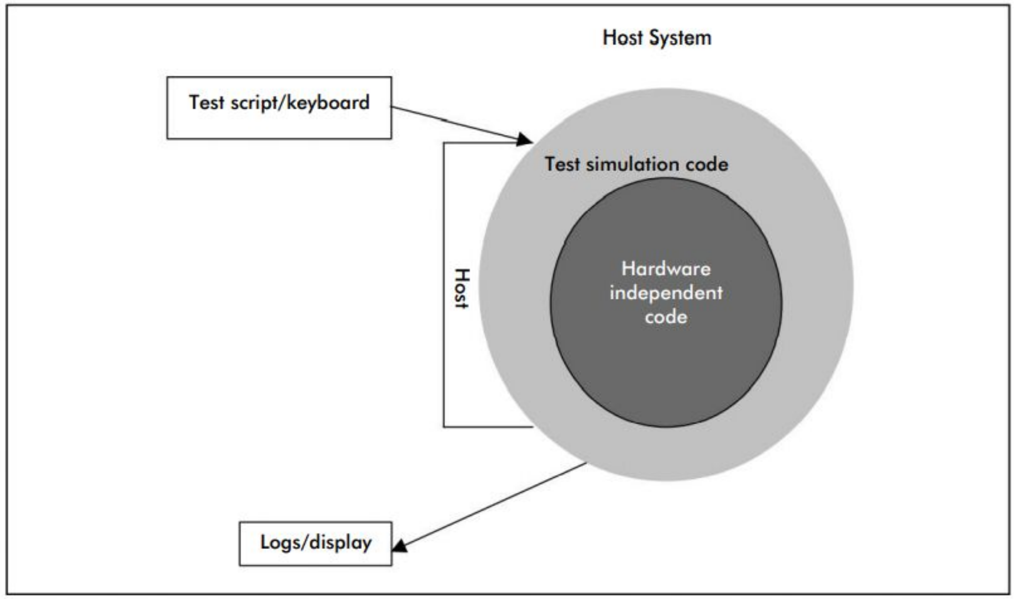
04 Best suited for testing of nonfunctional requirements as well.

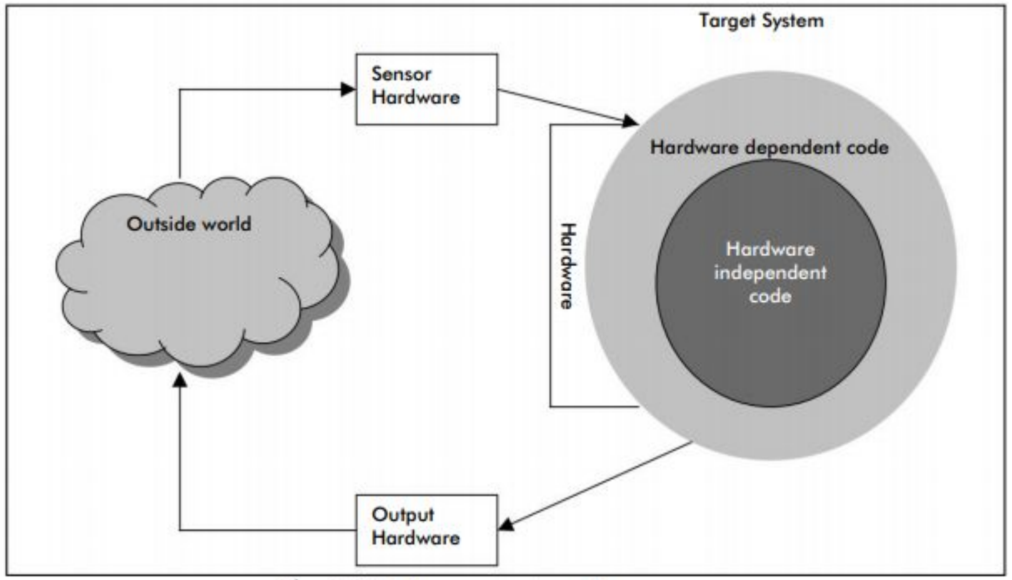
05 Functional testing is heavily dependent on the quality of requirements.

Not a good substitute for code-based testing.

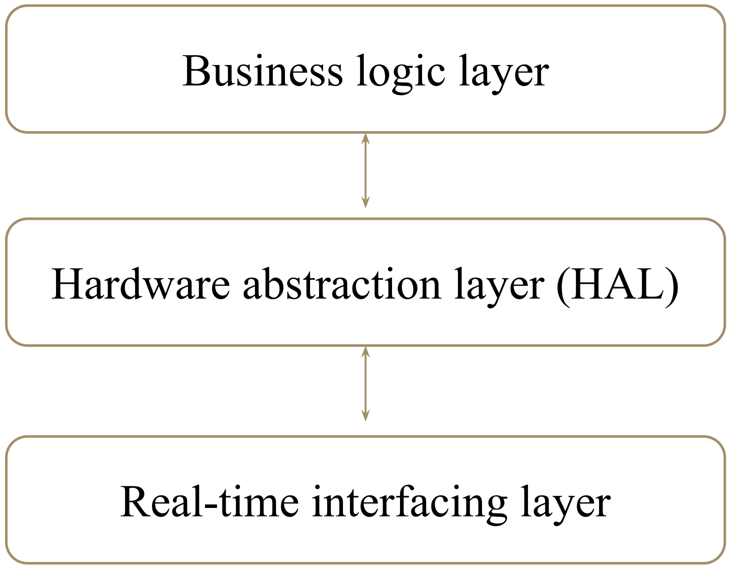
- Once a requirement needs to be coded, it gets blown up by the multitude of subtle conditions that must be considered.
- And it is much more cost-effective to perform code coverage analysis through white box testing.

HOST- BASED TEST SETUP





- A simulation of the complete environment of the system under test (SUT).
- The hardware dependent portions of the code cannot be tested effectively on host, they need to be simulated as well.



This implements the logic of the device: the use cases and features defined in the requirements specification.

This is a selection of functions through which all input to and output from the hardware flows

This is the part of the software that interacts directly with the hardware such as actuators, sensors, input and output devices.

- It is possible to simulate hardware input and output for almost all hardware devices common to small embedded systems.
- The most effort is spent in identifying the hardware abstraction layer, if this is not explicit in the existing software.
- The effort required to simulate the hardware devices after the HAL has been established is usually a matter of a few weeks of coding.

REFERENCES

- Sriram V Iyer, Pankaj Gupta, “Embedded Realtime Systems Programming”.
- Arnold S Berger, “Embedded Systems Design: An Introduction to Processes, Tools, and Techniques”.
- https://ecestudy.files.wordpress.com/2015/11/es-unit-4.pdf
- https://www.rapitasystems.com/embedded-software-testing-tools
- http://epgp.inflibnet.ac.in/epgpdata/uploads/epgp_content/S000007CS/P001072/M023188/ET/1505901416lect-36-f
- https://www.linkedin.com/pulse/host-based-system-tests-embedded-software-mocking-out-korsholm?trk=portfolio_article-card_title
- https://www.linkedin.com/pulse/automated-testing-embedded-software-stephan-korsholm

THANK YOU