TDD and Embedded Systems

Day 1:

- **Introduction to TDD and Embedded Systems:** This topic will provide an overview of Test-Driven Development (TDD) and its benefits for embedded systems development. It will also introduce the basic concepts of embedded systems and how TDD can be applied to them.
- **Setting Up the Development Environment:** This topic will cover how to set up a development environment for embedded C TDD, including choosing a compiler, editor, and testing framework.
- **Writing Your First Test:** This topic will guide you through writing your first test for an embedded C application, using a unit testing framework like Unity or CMock.
- **Red-Green-Refactor Cycle:** This topic will introduce the Red-Green-Refactor cycle, which is the core of TDD. You will learn how to write failing tests, make the tests pass by writing minimal code, and then refactor the code to improve its quality.

Day 2:

- **Test-Driven Design:** This topic will discuss how to use TDD to design your embedded C applications. You will learn how to break down your application into smaller, testable components and how to use tests to guide your design decisions.
- **Advanced Testing Techniques:** This topic will cover more advanced testing techniques, such as mocking, dependency injection, and property-based testing.
- **Integration Testing:** This topic will discuss how to test the integration of different components of your embedded C application.
- **Continuous Integration and Continuous Delivery (CI/CD):** This topic will introduce the concepts of CI/CD and how they can be used to automate the testing and deployment of your embedded C applications.

Day 3:

- **Testing Legacy Code:** This topic will discuss how to test legacy code that was not written using TDD.
- **Performance Testing:** This topic will cover how to test the performance of your embedded C application.
- **Test Coverage:** This topic will discuss how to measure test coverage and how to achieve high test coverage in your embedded C application.
- **Advanced Topics:** This topic will cover some more advanced topics in embedded C TDD, such as testing real-time systems and testing for security vulnerabilities.

Table of Contents:

- **Day 4**

  TDD Fundamentals and FreeRTOS Integration

  Morning:

  - **Introduction to TDD:**
    - Benefits of TDD in embedded systems development
    - Red-Green-Refactor cycle
    - Choosing a testing framework (e.g., Unity, CMock)
  - **Setting Up the Development Environment:**
    - Tools and libraries for TDD in embedded C
    - Integrating FreeRTOS and testing framework
  - **Writing Your First Test:**
    - Creating a simple FreeRTOS task and writing a unit test for its functionality
    - Red-Green-Refactor cycle in practice

  Afternoon:

  - **Test-Driven Design with FreeRTOS:**
    - Designing FreeRTOS tasks and components with testability in mind
    - Mocking and stubs for isolating individual components
    - Testing task creation, deletion, and communication
  - **Integration Testing with FreeRTOS:**
    - Testing interactions between multiple FreeRTOS tasks and components
    - Using test doubles to simplify complex interactions
    - Continuous integration practices for embedded systems

  Hands-on Lab:

  - Throughout the day, participants will have hands-on lab sessions to practice the concepts learned in lectures.
  - Lab exercises will focus on:
    - Setting up the development environment
    - Writing unit tests for simple FreeRTOS tasks
    - Testing task communication and synchronization
    - Implementing advanced testing techniques

  Key Takeaways:

  - Understand the fundamentals of TDD and its benefits in embedded C development.
  - Be able to set up a development environment for TDD with FreeRTOS integration.
  - Write unit and integration tests for FreeRTOS tasks and components.
  - Apply TDD principles to design testable embedded C applications.

  Additional Notes:

  - This is a 1-day intensive training, so basic understanding of embedded C and FreeRTOS concepts is recommended.
  - The specific testing framework and tools used might vary depending on instructor preference and participant needs.
  - Feel free to ask questions and request further clarification on any topics during the training.