

Secure the Quality of OS in Spacecraft

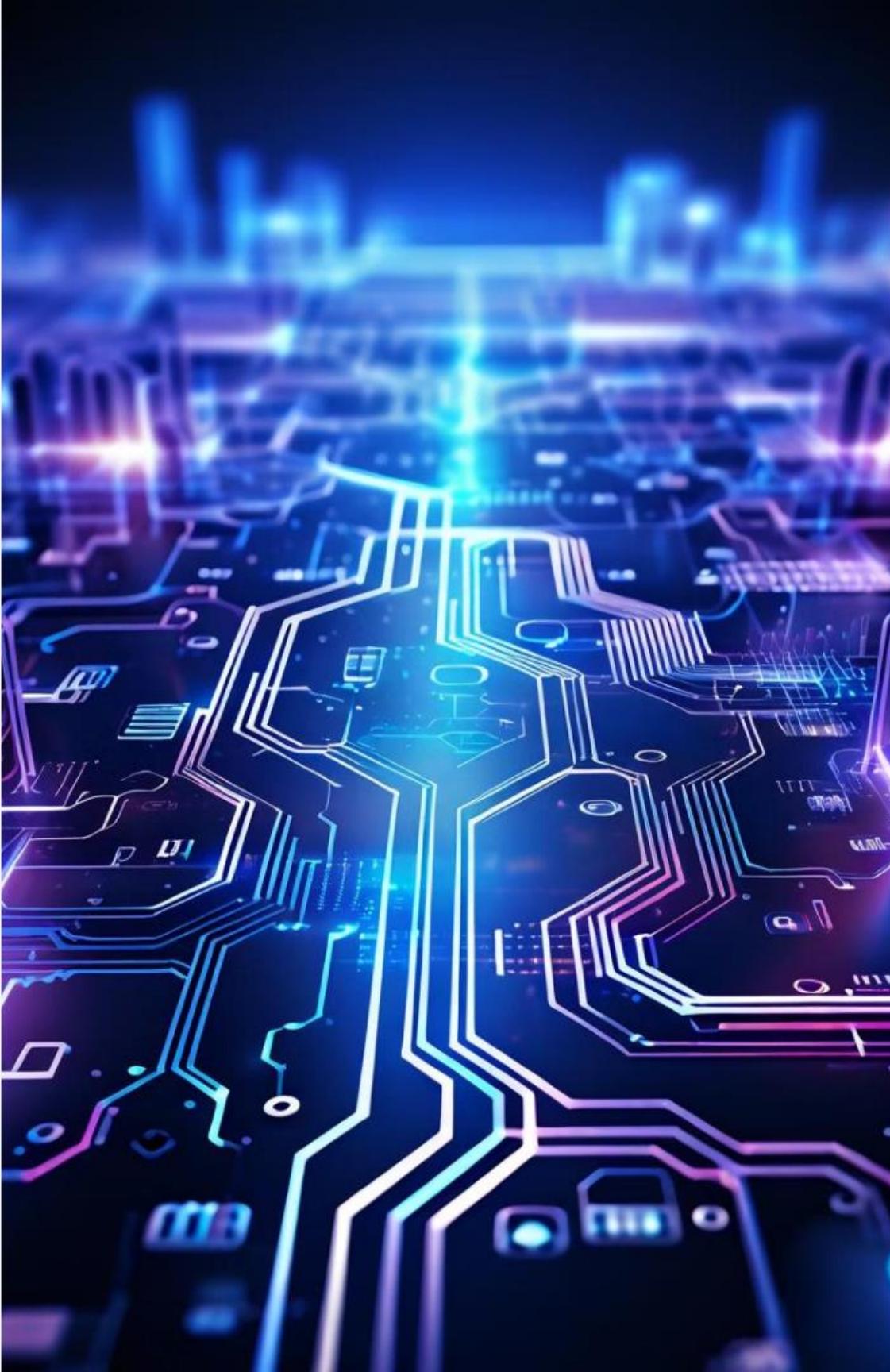
○ by Team Wicked Crew



Team Information

- **Group Leader:** Ashutosh Khadse
- **Research & Reporting:** Anastasia Baranova, Thant Kyaw Min
- **Development & Testing:** Harishik Dev Singh Jamwal, Ashutosh Khadse
- **Presentation Maker:** Harishik Dev Singh Jamwal
- **Presentation:** Shohujakhon Muhsin Ugli Zikirov, Ashutosh Khadse





Instruction Professor



**PROFESSOR KIM YOUNG
AI & BIGDATA**



CONTENTS

- Project Goal
- Key Achievements
- Limitations
- Project Summary/Recommendation
- Conclusion
- Demonstration

Understanding FreeRTOS Task Management

1 Task Creation

Efficient task creation and initialization

2 Task Scheduling

Reliable and responsive scheduling of APIs with each other

3 Task Synchronization

Seamless synchronization between concurrent tasks

4 Completing 55 APIs

Optimal distribution of API between Team

Key Achievements

Completed 54/55 API

One API need Arduino any micro controller supporting FreeRTOS to implement.

Created a Mock Environment

To run few APIs, we created a mock (virtual) environment.

Solving Errors

Solved the errors by using different ways.

Project Limitations

Error Indication and Solution

Error: vTaskResumeFromISR undefined

Solution: Setting Correct Configuration

Stack Depth Error

Static memory being barrier, need to use
dynamic memory allocation using mock
environment

Hardware Limitations

Need a micro controller to test few APIs
without the mock environment.

Scalability

Supporting different operating systems
(like windows and MacOS).



Project Purpose

Verify Correct Functionality

1

Detect and Fix Bugs

2

Measure Performance

3

Prepare Documentation

4

Assignment Of APIs to Test

Ashutosh APIs

1. xTaskAbortDelay
2. xTaskCallApplicationTaskHook
3. vTaskDelay
4. taskDISABLE_INTERRUPTS
5. taskENABLE_INTERRUPTS
6. xTaskGetCurrentTaskHandle
7. xTaskGetIdleTaskHandle
8. uxTaskGetNumberOfTasks
9. vTaskGetRunTimeStats
10. xTaskGetSchedulerState
11. uxTaskGetStackHighWaterMark
12. uxTaskGetSystemState
13. uxTaskPriorityGet
14. vTaskPrioritySet
15. taskENTER_CRITICAL
16. taskEXIT_CRITICAL
17. vTaskSetApplicationTaskTag()
18. xTaskGetApplicationTaskTag()
19. xTaskCreate()
20. vTaskStartScheduler()

Harishik APIs

1. xTaskCreate()
2. vTaskStartScheduler()
3. vTaskDelayUntil()
4. xTaskGetTickCount()
5. vTaskGetInfo()
6. vTaskSuspendAll()
7. xTaskResumeAll()
8. vTaskSuspend()
9. vTaskResume()
10. xTaskNotifyGive()
11. xTaskNotifyStateClear()
12. vTaskDelete()
13. taskYIELD()
14. ulTaskNotifyTake()
15. vApplicationStackOverflowHook()

Minh APIs

1. xTaskGetTickCountFromISR()
2. XTaskList()
3. XTaskNotify()
4. xTaskNotifyAndQuery()
5. xTaskNotifyAndQueryFromISR()
6. xTaskNotifyFromISR()
7. xTaskResumeFromISR()
8. xTaskSetTimeOutState()
9. vTaskStepTick()

Anastasia APIs

1. vTaskAllocateMPURegions()
2. xTaskCheckForTimeOut()
3. vTaskSetTimeOutState()
4. xTaskCreateStatic()
5. taskENTER_CRITICAL_FROM_ISR()

Muhsin APIs

1. taskEXIT_CRITICAL_FROM_ISR()
2. xTaskGetHandle()
3. vTaskGetRunTimeStats()
4. eTaskGetState()
5. pcTaskGetName()

Description of Final Design

Built a Project

- Folder for each Team member.

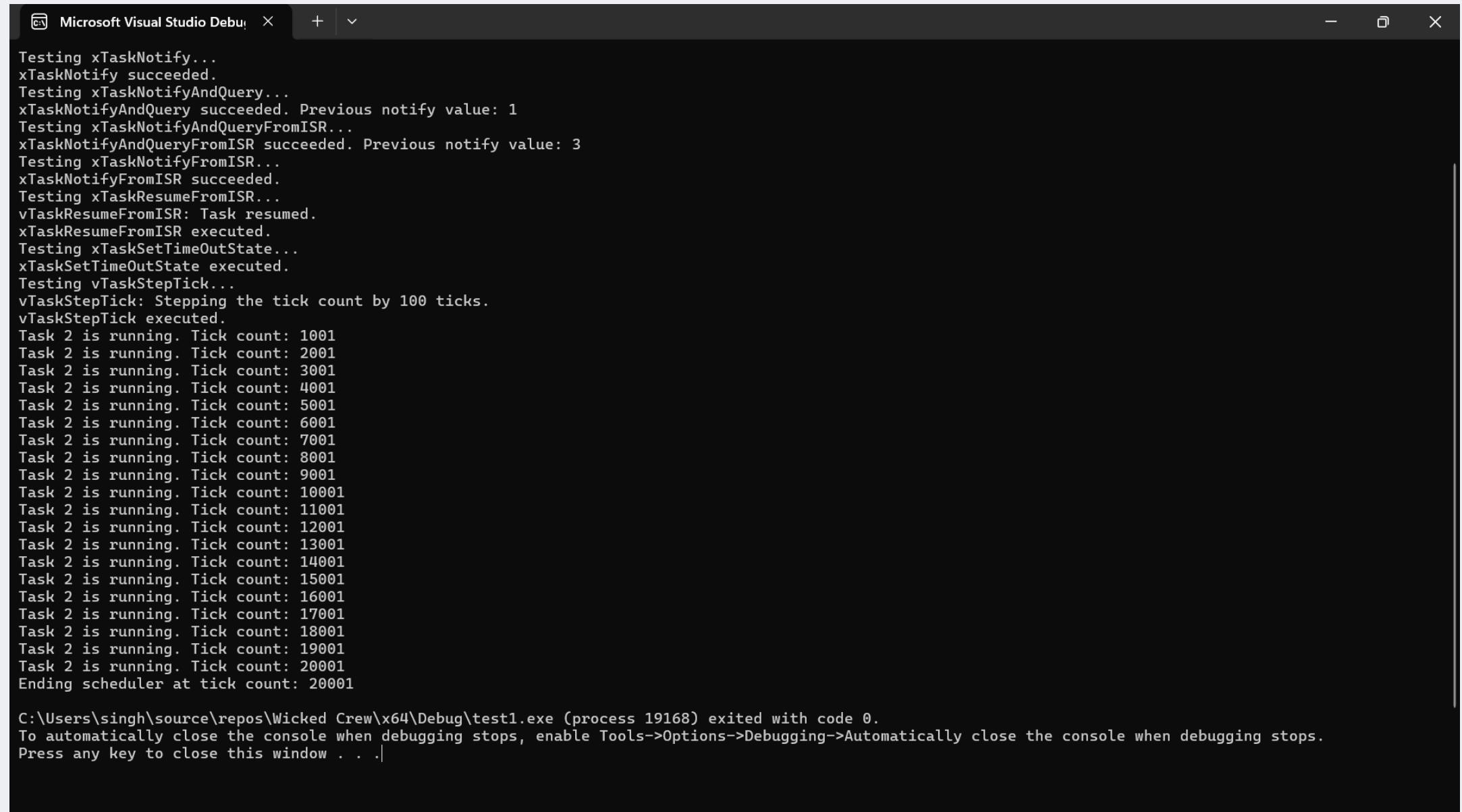
The screenshot shows a code editor interface with the following details:

- Solution Explorer:** Displays a project named "Harishik" containing six sub-projects: Anastasia, Ashutosh, Harishik, Minh, Muhsin, and Wicked Crew. Each sub-project has a "test1" folder containing various files like test1.c, manifest.yml, and other C/C++ source files.
- Code Editor:** The main window displays a file named "test1.c". The code includes standard library headers, FreeRTOS.h, task.h, timers.h, and semphr.h. It defines task function prototypes for vTask1 and vTask2, and provides implementations for vTaskGetTickCountFromISR, vTaskNotify, vTaskNotifyAndQuery, vTaskNotifyAndQueryFromISR, vTaskNotifyFromISR, vTaskResumeFromISR, vTaskGetTimeOutState, vTaskStepTick, and memory management functions pyPortMalloc and vPortFree. It also includes application-specific hook functions vApplicationMallocFailedHook and vApplicationStackOverflowHook.
- Output Window:** At the bottom, the "Output" window shows build errors. It lists several paths where it cannot find the "FreeRTOS.h" header file, such as "C:\Users\ashut\OneDrive\Desktop\Wicked Crew\Wicked Crew\FreeRTOS\Source\timers.c(3,10)" and "C:\Users\ashut\OneDrive\Desktop\Wicked Crew\Harishik\test1\test1.c(3,10)".

Implementation of Final Design

Output Window

- xTaskNotify
- xTaskNotifyAndQuery
- xTaskNotifyFromISR()
- xTaskResumeFromISR()
- xTaskSetTimeOutState()
- xTaskGetTickCountFromISR()
- xTaskList()
- vTaskStepTick()



The screenshot shows the Microsoft Visual Studio Debug Output window. The window title is "Microsoft Visual Studio Debug". The output content is as follows:

```
Testing xTaskNotify...
xTaskNotify succeeded.
Testing xTaskNotifyAndQuery...
xTaskNotifyAndQuery succeeded. Previous notify value: 1
Testing xTaskNotifyAndQueryFromISR...
xTaskNotifyAndQueryFromISR succeeded. Previous notify value: 3
Testing xTaskNotifyFromISR...
xTaskNotifyFromISR succeeded.
Testing xTaskResumeFromISR...
vTaskResumeFromISR: Task resumed.
xTaskResumeFromISR executed.
Testing xTaskSetTimeOutState...
xTaskSetTimeOutState executed.
Testing vTaskStepTick...
vTaskStepTick: Stepping the tick count by 100 ticks.
vTaskStepTick executed.
Task 2 is running. Tick count: 1001
Task 2 is running. Tick count: 2001
Task 2 is running. Tick count: 3001
Task 2 is running. Tick count: 4001
Task 2 is running. Tick count: 5001
Task 2 is running. Tick count: 6001
Task 2 is running. Tick count: 7001
Task 2 is running. Tick count: 8001
Task 2 is running. Tick count: 9001
Task 2 is running. Tick count: 10001
Task 2 is running. Tick count: 11001
Task 2 is running. Tick count: 12001
Task 2 is running. Tick count: 13001
Task 2 is running. Tick count: 14001
Task 2 is running. Tick count: 15001
Task 2 is running. Tick count: 16001
Task 2 is running. Tick count: 17001
Task 2 is running. Tick count: 18001
Task 2 is running. Tick count: 19001
Task 2 is running. Tick count: 20001
Ending scheduler at tick count: 20001

C:\Users\singh\source\repos\Wicked Crew\x64\Debug\test1.exe (process 19168) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .|
```

Implementing Enhancements & Demonstrating Improvements

1



Simulation & Validation

Verify performance and compliance

2



Rigorous Testing

Ensure reliability and robustness

3



Comprehensive Documentation

Facilitate understanding and adoption

Final Conclusions & Suggestions

Recommendation	Benefit
Implement Dynamic Scheduling Algorithms	Improve task prioritization and responsiveness
Enhance Memory Management Strategies	Optimize resource utilization for embedded systems
Develop Comprehensive Error Handling	Ensure reliable and fault-tolerant operation
Facilitate Cross-Team Collaboration	Align with Space-X requirements and objectives

What If we had two more weeks

Detailed Documentation

Creating detailed documentation
of APIs

Advanced Memory Management using Hardware

Try to implement dynamic
memory allocation using
Arduino

More qualified Test

Combining APIs to make
code more efficient

Conclusion

By addressing the task management challenges in FreeRTOS, we have optimized its performance and reliability, ensuring that it meets the stringent requirements of Space-X's manned spacecraft.

