International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

# Performance Evaluation of Integrated Hard Real-Time Application and RISC V Processor for Spacecraft on Board Software Application

**Vishwanath Y[1*], Kiran Desai[2], R S Upendra[3], Venkatesh Prasad[1], Sasidhar Babu Suvanam[4], Arun Biradar[1], Supreeth S[5], Rohith S[6]**

**Submitted**: 23/04/2023 **Revised**: 27/06/2023 **Accepted**: 06/07/2023

**Abstract***:* Space industries operating critical missions and safety aspects must operate robust and error-free software systems. A trace amount of error causes failure of the entire spacecraft system; hence, the onboard software of spacecraft systems typically uses Baremetal Cyclic executives to maintain robustness under error-free conditions. Baremetal Cyclic executive software programs are highly predictable in their behavior, and have been evaluated and proven critically for space applications. Technology advent with increased computational power and concurrency necessitates high-end applications of spacecraft systems such as Agriculture, Weather forecasting, communication, and geospatial applications. To meet these challenges, spacecraft systems must be upgraded to handle high-level computational loads, time complexities, and parallelism in activity at the manifold. Currently, space agencies across the globe replace the conventional Baremetal Cyclic executives with advanced RTOS developed on single/multicore processors such as Power PC and LEON4 based on ARINC 653 specifications, which are proprietary, run with the operating system, that is, VxWorks, RTEMS, etc. The execution of the RISC V architecture in the onboard software of a spacecraft system offers advantages, such as openness, modularity, extensibility, and stability. Many RISC-V designs have single/multicore architectures with open-source RTOS support. In the present work, we developed a prototype built on hard real-time satellite application software and evaluated its performance using an RTOS stack on a RISC V series. This research also developed a library to allow portable application development for any flavor of the RISC V architecture.

***Keywords:*** *Space/Avionics, Onboard software system, Baremetal Cyclic executives, RTOS, Multiprocessor, Hard Real Time Systems, RISC V, Closed Loop systems*

## 1. Introduction

Hard Real Time (HRT) systems is a Boolean outcome unlike soft real time systems; works on the assumptions whether a system succeeds in meeting the deadlines or not [1]. Generally, HRT is a critical system, failure to which may result in social/financial loss [2]. Spacecraft on board software is such mission/safety critical software wherein achieving the deadlines are of prime importance [3]. In such systems, selection of microprocessors and corresponding execution environment plays a vital role. Various flavours of microprocessor cores proprietaries such as Power PC, ARM LEON3 (SparcV8) etc. are extensively used in onboard systems of spacecrafts [4]. Most of the applications in such HRTs will be developed employing Baremetal real time cyclic executive software to achieve the required HRT functionalities [5]. Technology advancement demands high end computation power, so that the existing Baremetal real time Cyclic executive software [17, 19] may not serve the purpose and need a replacement with the much-advanced software architecture such as open RISC V. Open RISC V architecture comes along with various advantages such as layered and extensible ISA; common set of shared tools and development resources; flexibility to customize processor; accelerate time to market etc. [6]. With these advantages, manufacturers, and Field-Programmable Gate Array (FPGA) developers all over the world are developing various flavours of RISC-V processors with varied applications [7].

With this interest, present work aimed to explore RISC V processor and to conduct a built-in prototype-based case study applying RTOS in regard with spacecraft on board software application and also to prove the possible implementation of RISC V integrated with RTOS for satellite applications. Present work proposed the usage of

[1]*Professor, School of Computer Science and Engineering, REVA University, Bengaluru-560064, India*
*ORCID ID: https://orcid.org/0000-0003-2210-2297*
[2]*Student, School of Computer Science and Engineering, REVA University, Bengaluru-560064, India*
[3]*Associate Professor, School of Electronics & Communications Engineering, REVA University, Bengaluru-560064, India*
*ORCID ID: https://orcid.org/0000-0003-4749-1719*
[1]*Professor, School of Computer Science and Engineering, REVA University, Bengaluru-560064, India*
[4]*Professor, Department of Computer Science and Engineering, School of Engineering, Presidency University, Bengaluru-560064, India*
*ORCID ID: https://orcid.org/0000-0002-9006-0853*
[1]*Professor, School of Computer Science and Engineering, REVA University, Bengaluru-560064, India*
*ORCID ID: https://orcid.org/0000-0003-2355-7066*
[5]*Assistant Professor, School of Computer Science and Engineering, REVA University, Bengaluru-560064, India*
*ORCID ID: https://orcid.org/0000-0002-7097-6733*
[6]*Associate Professor, Nagarjuna College of Engineering and Technology, Bengaluru-562110, India*
*ORCID ID: https://orcid.org/0000-0001-5709-9492*
** Corresponding Author Email: vishwanath.y@reva.edu.in*

Shakti C Class processors and its compatible FPGA, compiler tool chain to develop a typical Altitude and Orbit Control System (AOCS) for space/avionics applications.

## 2. Literature Review

In [8] stated that in very near future, NASA is bound to face the requirements for reliable space based multicore processing methods and approaches that can deliver under the given extreme constraints of power, mass, and cost features [8]. The usage of multicore processor is extensively discussed in terms of onboard processing power and ability to support fault tolerant environment in the research paper published in [9]. The paper discusses about the experimentation carried out with multicore processor used on board of Mars Rover to evaluate the performance of image processing analysis. The hardware setup includes Tilera TILE64TM 750 MHz with concurrency architecture of 64 processing elements. The developed hardware architecture [18] setup supports a Linux based software architecture which includes the standard GNU C and C++ compilers and other support tools like debugger and object manipulation tools. Hence the development environment supports a full featured Linux platform to development and performance evaluation. The results reported that, multicore processor environment has made an impactful step by qualifying the code for the radiation hardened multicore processor and also there is a ten-fold increase in the average runtime of image processing algorithm has been noticed [9]. A research team developed MIPS R3000 processor supported spacecraft onboard real time software architecture system. The developed architecture system is focused on fault detection and identification (FDI). The system makes use of prony approximation and FFNN (feed forward neural network) to classify the anomalies detected from the wheel and solar panel. The result of the classification is found to have an accuracy of 100% [10].

In the recent past time various countries across the globe has witnessed the tremendous increase in the number of satellites being launched, among all most of them are used for agriculture, commercial and research needs. Furthermost, these satellites carry a good number of sensors which can generate enormous amount of data i.e optical sensing, hyperspectral, infrared imaging data etc. Hence the onboard software system is expected to have large memory to store this huge amount of data generated and make use of the downlink bandwidth to send it back to the earth station for further processing, which can make the system unproductive. The ineffectiveness of the onboard software system can be overcome by having an onboard system capable of processing the data and sending the required data on the downlink. A set of six experiments to demonstrate intelligent data processing capability using AMD A10-8700P and AMD V1605B part of the V1000 family of SOCs using the ROCm HPC stack. The results obtained from the setup reported significant benefits in terms of radiation tolerance and observed that the deep learning training can be performed on the orbit effectively and neural network designed for earth observation application can be applied for onboard data processing and self-learning [11]. Aranda et al. (2020) conducted a study on RISC V soft processor implemented on FPGA which can be used as an onboard computer of space application. Study used fault injection experimentation method to assess the performance of RISC V software processor installed in satellite onboard software system. The experimentation results have shown that, 96% of injected errors didn't create any wrong outcome, the analysis of the result obtained revealed that many of the bits that Xilinx classifies as essential do not influence the behaviour by inducing the errors, hence they cannot be considered as critical errors. Therefore, based on the results study proposed a procedure to compute the empirical percentage of critical bits that helps in assessing the fault tolerance capability of the system [12].

From this critical review it is evidenced that the present onboard processors and operating systems employed in the satellite on board software communication network need to be updated with the multicore based processing system runs with the installed RISC V multicore processor in order to support the storage of enormous data generated by the satellites and also to send the data to downlink station for further processing. Hence it is advised to develop a prototype applying the RISC V multicore processor using real time operating system to know the real time application of multicore processor system in the satellite onboard software system.

## 3. Shakti C Class Processor and Tool Chain

The Shakti processor program was initiated by IIT-M RISE group in 2014. The major aim of the program is to bridge industry and academia by providing innovative and customized solutions without royalties. The Shakti processors designed by IIT-M comes in various classes based on the application at hand to be solved [13]. Any ecosystem built for spacecraft on board software development consists necessary elements as discussed in Figure 1 [14].

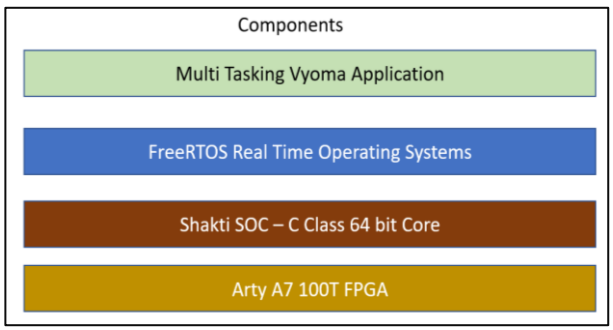



**Fig 1.** Shakti C Class Stack

A. **Processor selection:** Shakti C Class processor was selected; it is a new upgraded version from RISE and represents fault tolerant class (F Class)

B. **RTOS Selection:** FreeRTOS was selected for development among the existing RTOS flavours such as RT Linux, FreeRTOS and Zephyr RTOS.

C. **RISC V GNU GCC Compiler:** An Architecture design of the prototype developed and tested on RISC V Shakti C Class processor and named it as Vyoma Application.

## 4. System Design of Vyoma Application

Vyoma system consists of following architectural components.

**Shakti C Class IP Core:** Shakti C Class IP was compiled using BlueSpec compiler and Vivado Tool Suite; generated by loading bit/mcs file to ArtyA7 100T FPGA. Further, the loaded FPGA was configured as 64-bit single core processor known as Shakti C Class processor [15].

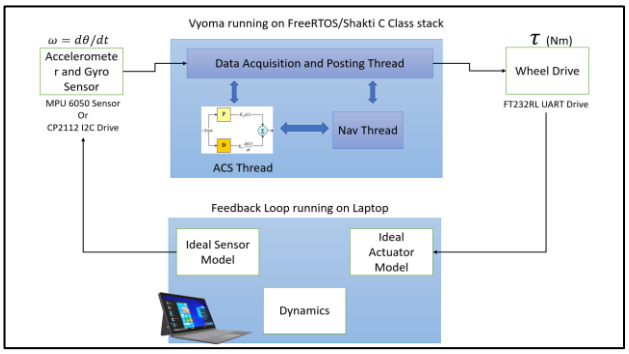



**Fig 2.** Vyoma System Context Diagram

### 4.1 Vyoma Spacecraft on Board Application

Application was developed with a conceptual understanding of a multithreaded application running on a FreeRTOS OS stack. The Vyoma application is a multithreaded application which consists of three main threads [16].

### 4.2 ACS (Attitude Control System) Thread

Thread runs at 16ms frequency with 4 ms interval and is responsible for Attitude Control of the spacecraft, by executing following functions

- Reading the angular rate sensor data acquired through data acquisition thread w (radians/s)
- Convert the rates into quaternions using small angle approximation theorem as in Eq. (1)-(3).

$$\Delta q = \left(\frac{1}{2}\right)\dot{\omega} \quad (1)$$

$$Q = Qbn \times \Delta q \quad (2)$$

$$Q = \text{correct}\,(Q)(\text{ Sign correctness }) \quad (3)$$

Compare with commanded altitude of spacecraft and generate error as in Eq. (4):

$$\tau = -K_{\text{rate}} \times (\omega_{\text{err}}) - K_{\text{prop}} \times (2.0 \times \text{ Qerr }) \quad (4)$$

- Generate torque signal

i. Data Acq Thread: Runs at 4ms frequency with 2ms interval and is responsible to

- Read angular rate using sensor such as MPU6050 real sensor (or) CP2112 I2C driver sensor based on the connection.
- Also, responsible for posting wheel torque to wheel drive (UART FT232RL).

ii. Nav Thread: Help in providing precise navigation information to Vyoma

iii. Software Timer Thread: A novel timer initiated by present study and is responsible for triggering all other threads in a timely manner. Timer runs at 1ms frequency.

- Novel features of developed application:
  - ➢ Nyquist sampling: Two times faster than the PD Controller computational rate.
  - ➢ A timer interrupt: After a brief initialization of all state variables and devices, a timer interrupt initiated at regular intervals, based on the thread frequency specified, triggers execution as in Figure 3.

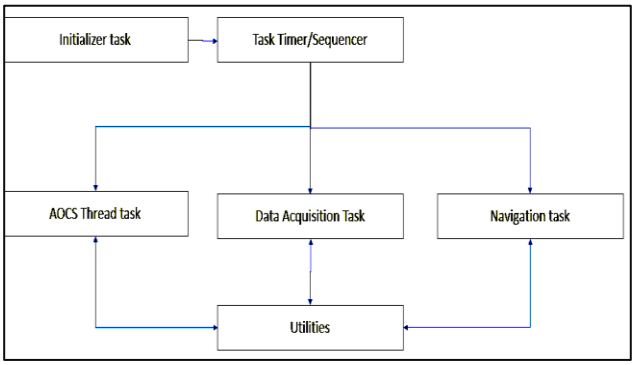



**Fig 3.** Multi-threaded architectural view

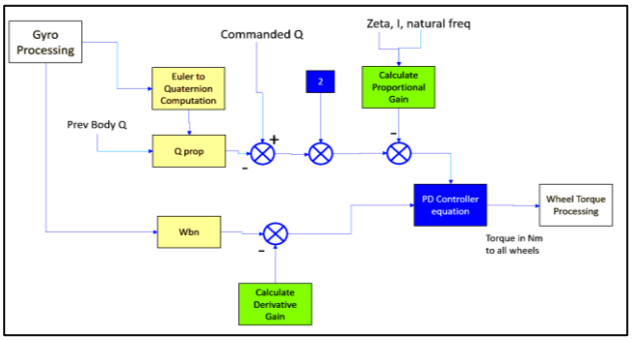



**Fig 4.** Controller details of Vyoma Application

- Developed architecture is portable enough to run on POSIX based RT-Linux OS.

### 4.1. Spacecraft Dynamics:

Simulation to model Spacecraft environment.

**1. Actuator Model (or) Wheel Model:** An ideal wheel was modelled with torque distributed to all the three wheels as in Figure 5. UART driver provides wheel torque to Actuator Model. Based on spacecraft inertia (measured to principal axis Ixx, Iyy, Izz) it is integrated to body momentum, forms the angular rate and wheel momentum. Body rate computed is assessed by sensor model.

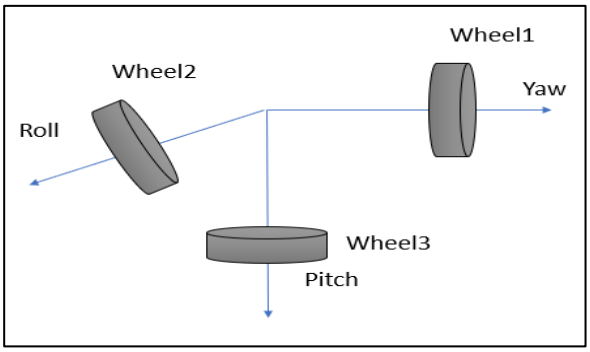



**Fig 5.** Wheel Model with each wheel mounted in principal axis of body

**2. Gyro Sensor Model:** Model takes Body rates as input and screen various sensor characteristics such as: Sensor Bias, Sensor Gaussian Random noise, Misalignment factors and Drift rate. Model was constructed by considering the above discussed parameters, further study computes the Gyro rate and is fed through CP2112 I2C drive and close the loop. The Gyro model provides rate along all the three axes in sensor frame. For ideal conditions, study considered sensor frame and body frame that are aligned with each other.

### 4.3 Brief Architecture of Vyoma Application:

As seen in the Figure 6, the entire software is implemented as a multi-threaded software. The Software consists of three threads, activated by mutex lock release mechanism. This Activation is done by the Software timer, which is started once the scheduler is started. Detailed explanation of what each thread and Software timer performs is provided in subsequent sections.

The scheduler selected from FreeRTOS is Fixed Priority Pre-emptive scheduling. This is also called as "Rate Monotonic" Scheduling. The Prerequisites for this scheduler to be used are;

- Ensure that the thread with smallest quantum to be executed is assigned higher priority
  - ➢ In this case Data Acquisition thread is provided highest priority
- Similar frequency tasks to be given same priority
- Higher period tasks to be given lower priority
  - ➢ In this case Orbit Determination/Nav Thread is given lowest priority

The Vyoma application software consists of 4 major components:

- Software Timer Trigger Module
- Data Acquisition Thread
- Attitude and Orbit Control System Thread
- Navigation, Orbit Determination Thread

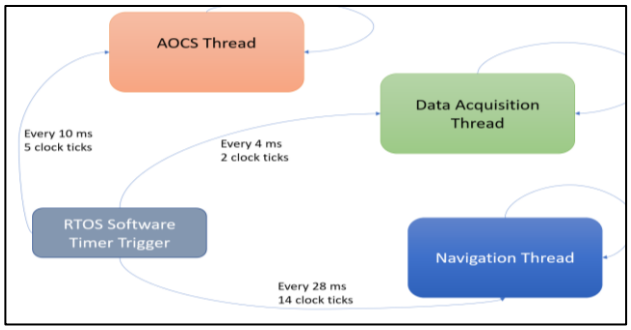



**Fig 6.** Developed onboard power of Vyoma application

#### 4.3.1 Software Timer Trigger Module:

Task Sequencer:

As can be seen in the below flowchart, the task sequencer checks which threads must be unblocked for execution by FreeRTOS, so that the tasks are kept in "ready to execute" state. "Task Notify" functionality is used to release the threads if they are blocked waiting for "Wait till notify" call, which we will see in subsequent sections of thread/task descriptions. This is the routine which decides the frequency in which the tasks are to be executed, which was discussed in previous section.

#### 4.3.2 Data Acquisition Thread

This thread waits for "Task Sequencer" as shown in Figure 7 to release "sem" or "Task Notify". This thread will be in block state at beginning of execution or once its actions are executed for that cycle. This thread is an infinite loop running same tasks in a repeated manner.

Inputs to this thread:

- ulTaskNotifyTake to Thread Output from this thread: Processed Gyro angles in all axes Delta Quaternions with small angle approximations Propagated quaternions representing the body attitude

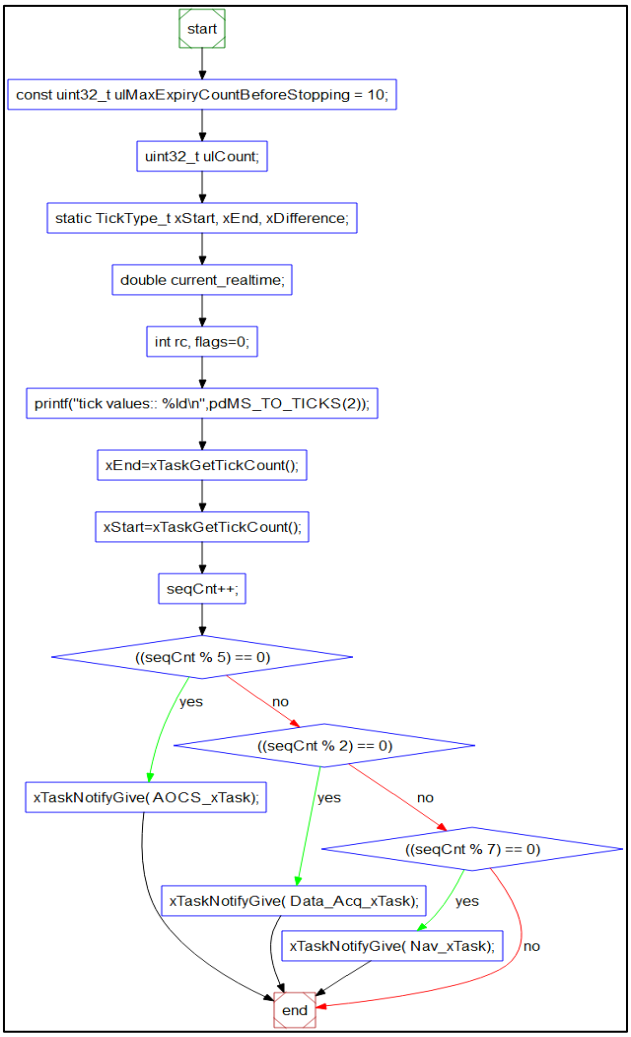



**Fig 7.** vTimerCallBack

### 4.3.3 Attitude and Orbit Control System Thread:

This is the main Controller Thread of the entire System. The job of this thread is to take input, Processed Quaternions, Gyro Rates of all the three axes, execute PD Controller, Compute Body Torque and provide output to Actuators.

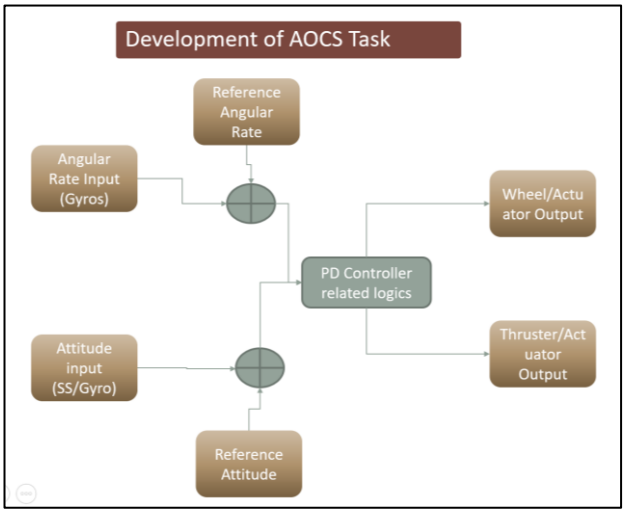



**Fig 8.** rRead_I2C_Gyro_into_QIB

This function reads propagates through QXQ multiplication utility and generates current Body attitude. Inputs to this thread:

Inputs to this thread:

- Processed Gyro Rates
- Current Body attitude (QBN), Reference Attitude (QREF)

Output from this thread: Actuator Torque

### 4.3.4 Navigation, Orbit Determination Thread:

The Spacecraft always needs a reference to which it must point in terms of Orientation and Angular velocity vectors. These vectors are derived using the knowledge of current Orbit/Trajectory and set of Orbital Parameters, which are shown in the below figure and descriptions:

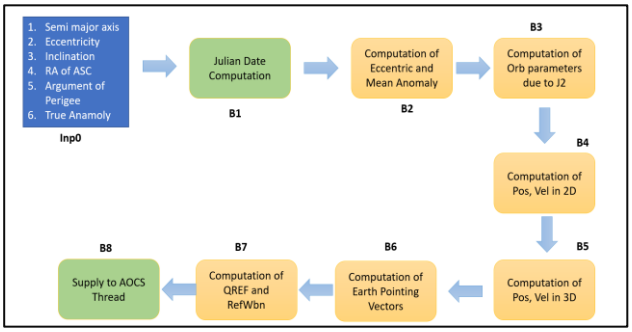



**Fig 9.** Functional flow of Orbit Model computation

Steps followed:

1. Read the Orbital Parameters initialized
2. Julian Date computation
   a. JDN=**(**1461 *** (**year + 4800 + **(**month - 14**)**/12**))**/4 +**(**367 *** (**month - 2 - 12 *** ((**month - 14**)**/12**)))**/12 - **(**3 *** ((**year + 4900 + **(**month - 14**)**/12**)**/100**))**/4 + day - 32075**;**
   b. JD=JDN+UT/**(**cSec_in_SolDay**);**
3. Computation of Eccentric and Mean Anomaly
   a. Mean_anly=(n*(t-to));
   b. Ecc_anly=Prev_Ecc_anly+(Mean_anly+e*sin(Ec c_anly)-Ecc_anly)/(1.0-e*cos(Ecc_anly));
4. Computation of Mean Anomaly and Eccentric Anomaly due to J2:
5. Computation of Position and Velocity in 2D Ellipse
6. Computation of Position and Velocity in 3D axes:
   a. Using Right Ascension of ASC Node and argument of perigee compute
7. Conversion of ECI Frame to ECEF Frame
8. Computation of Earth Pointing Direction Cosine Matrix and Reference Quaternions

```
Qearth.q[3]=1.0+R_3x3DCM[0][0]+R_3x3DCM[1][1]+
    R_3x3DCM[2][2];

Qearth.q[0]=1.0+R_3x3DCM[0][0]-R_3x3DCM[1][1]-
    R_3x3DCM[2][2];

Qearth.q[1]=1.0-R_3x3DCM[0][0]+R_3x3DCM[1][1]-
    R_3x3DCM[2][2];

Qearth.q[2]=1.0-R_3x3DCM[0][0]-
    R_3x3DCM[1][1]+R_3x3DCM[2][2];

find_max();

Ang=euler_angle(Qearth.q[0],Qearth.q[1],Qearth.q[2],Q
    earth.q[3]);

Rate=calculate_Rates(Ang,Prev_Ang);
```

9. Assigning to Reference Attitude Control system Thread.

## 5. Setup and Results of Execution

### 5.1. Hardware setup of onboard Vyoma application system:

Efficacy of the satellite onboard software system prototype Vyoma is principally influenced by the accuracy and robustness of the inbuilt hardware setup. The entire hardware setup of Vyoma application was discussed in the Figure 10 and it consists of,

1) A developed onboard software with Vyoma application system represented in the Figure 10a.

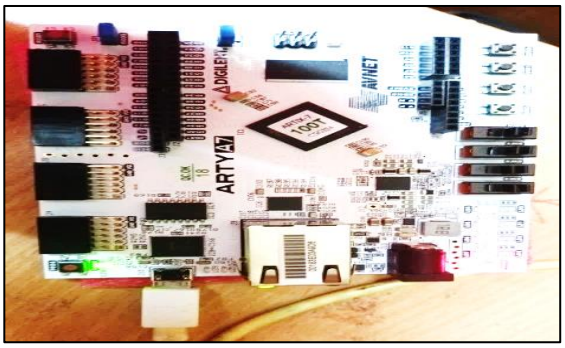
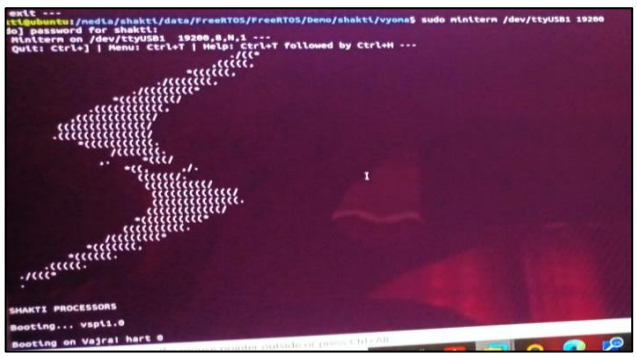


**Fig 10a.** Developed onboard power of Vyoma application

2) Shakti class c processer powered on display outcome was




discussed in the Figure 10b,

**Fig 10b.** Shakti C Class Powered on system

3) Connecting through OpenOCD and GDB represented in the Figure 10c.

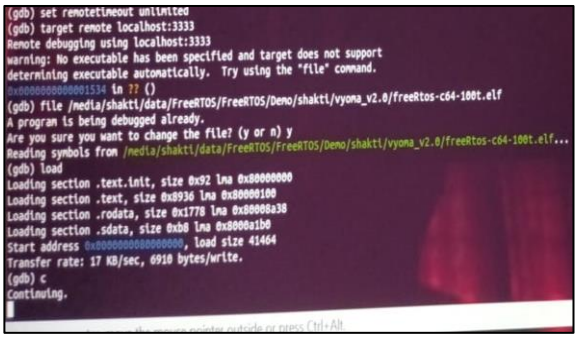



**Fig 10c.** OpenOCD and GDB connected to Shakti C Class IP Core

4) Execution of Threads as discussed in the Figure 10d.

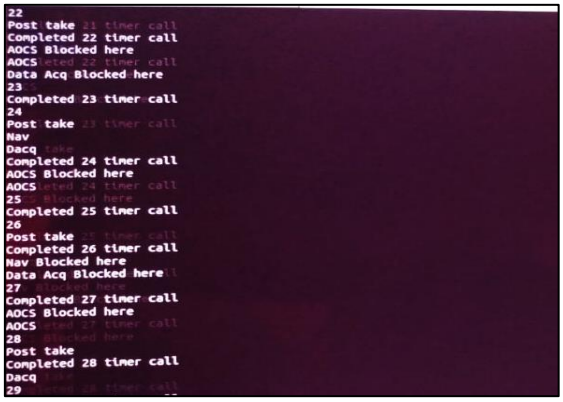



**Fig 10d.** Threads Execution

### 5.2. Time tick response of FreeRTOS

As the threads are of minimum 4ms revisit time, there are two ticks available from FreeRTOS in the developed prototype, which provides necessary trigger for threads to start. FreeRTOS in current implementation provides 1 tick/2ms which was sufficient to achieve the goal of present study (Fig 11). Efforts have been ON to make it much finer of the order of 0.5ms to take care of future requirements.

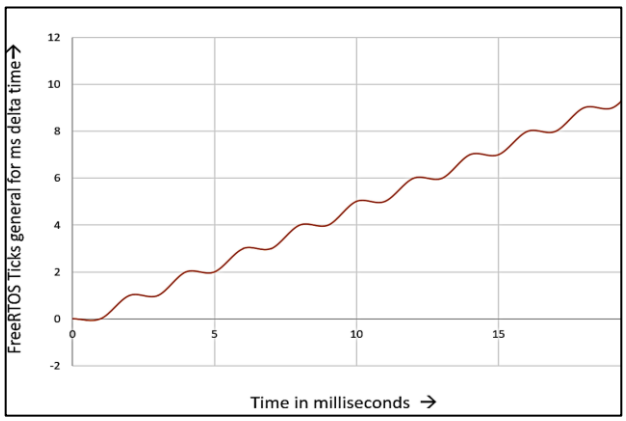



**Fig 11.** Time Tick response of FreeRTOS

### 5.3. Thread Response with Frequency 50 Hz, 20 Hz, 0.066Hz:

The thread response with Frequency 50Hz, 20Hz, 0.066Hz is recorded. The scheduler used in FreeRTOSConfig file for the same is fixed priority preemptive scheduling. As can be seen in Fig 8, since service S1 is of higher priority, pre-emption of S3 takes place as soon as S1 frequency has come into effect. With the above results it is imperative that RISC V and FreeRTOS combination is providing suitable environment for engineers to use the tool chain. Furthermore, closed loop studies indicate its effectiveness.

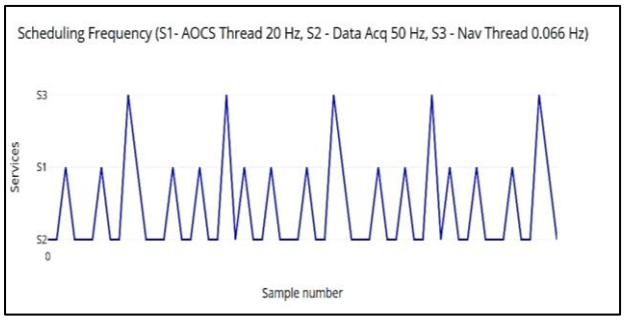



**Fig12.** Execution profile

### 5.4. Safety assessment

The safety aspect of the system is tested by considering a sample scenario data recorded in the excel sheet as shown in Figure 13, and when compared with that of the cheddar output shown in Figure 14 where a one-to-one mapping is considered, it clearly matches and hence we can conclude that it is a safe system.

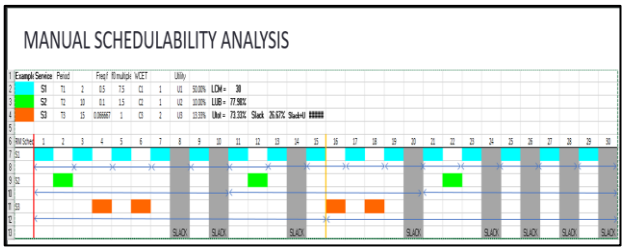



**Fig 13.** Manual Schedulability Analysis

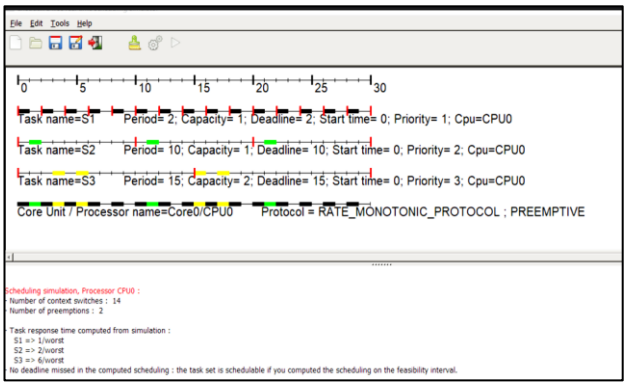



**Fig 14.** Cheddar tool output

## 6. Conclusions

Based on the results obtained from the present study, it can be concluded that the Vyoma architecture defined over the RISC V-FreeRTOS stack for a single core can meet the hard deadlines specified by the application. Hence, it is reported that the RISC V C-class processor and FreeRTOS stack are suitable for hard real-time applications. The current work focused on setting up an architectural and detailed design of spacecraft on board software on a free RTOS RISC V stack on a single core on an experimental basis. The future work pertaining to this work is as follows:

1) Portability prototype development between RT-Linux and FreeRTOS
2) Performance enhancement and evaluation on Multi core Shakti Processors [2]
3) Performance evaluation on Fault Tolerant F-class processors
4) Development of comprehensive Guidance and Navigation Control Software for Satellites for Academic purposes.

### Conflicts of Interest

All authors declare that they have no conflicts of interest.

### Author Contributions

**Vishwanath Y, Kiran Desai** were identified Initial problem identification, algorithm write-up, analysis, drafting of the manuscript, and simulation. **R S Upendra, Venkatesh Prasad** were responsible for the Literature survey and helped in the initial review process. **Sasidhar Babu Suvanam, Arun Birader** were responsible for the Complexity analysis of the research, evaluation of the research work. **Supreeth S, Rohith S** were responsible for the figures, final formatting and applied for the journal. All authors worked together to implement and evaluate the integrated system, and approved the final version of the paper.


### Acknowledgment

The authors deeply acknowledge the support by the Director, the **School of Computer Science and Engineering, REVA University**, and the entire academic team.


## References


[1] L. E. Rubio-Anguiano, J. L. Briz and A. Ramírez-Treviño, "Accounting for Preemption and Migration Costs in the Calculation of Hard Real-Time Cyclic Executives for MPSoCs," in *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 7990-7997, July 2022, doi: 10.1109/LRA.2022.3186489.

[2] D. Palmer and R. S. Holmes, "Extremely Low Resource Optical Identifier: A License Plate for Your Satellite,"

*Journal of Spacecraft and Rockets*, vol. 55, no. 4, pp. 1014–1023, Jul. 2018, doi: 10.2514/1.a34106.

[3] M. B. Quadrelli *et al.*, "Guidance, Navigation, and Control Technology Assessment for Future Planetary Science Missions," *Journal of Guidance Control and Dynamics*, vol. 38, no. 7, pp. 1165–1186, Jul. 2015, doi: 10.2514/1.g000525.

[4] G. Lentaris *et al.*, "High-Performance Embedded Computing in Space: Evaluation of Platforms for Vision-Based Navigation," *Journal of Aerospace Information Systems*, vol. 15, no. 4, pp. 178–192, Apr. 2018, doi: 10.2514/1.i010555.

[5] N. -J. Wessman et al., "De-RISC: the First RISC-V Space-Grade Platform for Safety-Critical Systems," 2021 IEEE Space Computing Conference (SCC), Laurel, MD, USA, 2021, pp. 17-26, doi: 10.1109/SCC49971.2021.00010.

[6] N. -J. Wessman et al., "De-RISC: A Complete RISC-V Based Space-Grade Platform," 2022 *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, Antwerp, Belgium, 2022, pp. 802-807, doi: 10.23919/DATE54114.2022.9774557.

[7] L. A. Aranda *et al.*, "Analysis of the Critical Bits of a RISC-V Processor Implemented in an SRAM-Based FPGA for Space Applications," *Electronics*, vol. 9, no. 1, p. 175, Jan. 2020, doi: 10.3390/electronics9010175.

[8] C. Villalpando, D. Rennels, R. Some and M. Cabanas-Holmen, "Reliable multicore processors for NASA space missions," 2011 *Aerospace Conference*, Big Sky, MT, USA, 2011, pp. 1-12, doi: 10.1109/AERO.2011.5747447.

[9] B. Bornstein, T. Estlin, B. Clement and P. Springer, "Using a multicore processor for rover autonomous science," 2011 *Aerospace Conference*, Big Sky, MT, USA, 2011, pp. 1-9, doi: 10.1109/AERO.2011.5747454.

[10] E. A. Omran, W. A. Murtada and A. Serageldin, "Spacecraft on-board real time software architecture for fault detection and identification," 2017 12th *International Conference on Computer Engineering and Systems (ICCES)*, Cairo, Egypt, 2017, pp. 615-620, doi: 10.1109/ICCES.2017.8275379.

[11] F. Bruhn, N. Tsog, F. Kunkel, O. Flordal, and I. A. Troxel, "Enabling radiation tolerant heterogeneous GPU-based onboard data processing in space," *Ceas Space Journal*, vol. 12, no. 4, pp. 551–564, Jun. 2020, doi: 10.1007/s12567-020-00321-9.

[12] A. E. Wilson, M. Wirthlin and N. G. Baker, "Neutron Radiation Testing of RISC-V TMR Soft Processors on SRAM-Based FPGAs," in IEEE Transactions on Nuclear Science, vol. 70, no. 4, pp. 603-610, April 2023, doi: 10.1109/TNS.2023.3235582.

[13] S. Tiwari, N. Gala, C. Rebeiro, and V. Kamakoti, "PERI," *ACM Transactions on Architecture and Code Optimization*, vol. 18, no. 3, pp. 1–26, Apr. 2021, doi: 10.1145/3446210.

[14] N. Iuga, I. Zagan and V. G. Gaitan, "CPU Execution Time Analysis based on RISC-V ISA Simulators: A Survey," 2022 *International Conference on Development and Application Systems (DAS)*, Suceava, Romania, 2022, pp. 12-18, doi: 10.1109/DAS54948.2022.9786163.

[15] A. Dörflinger et al., "A comparative survey of open-source application-class RISC-V processor implementations," Proceedings of the 18th ACM *International Conference on Computing Frontiers*. ACM, May 11, 2021. doi: 10.1145/3457388.3458657.

[16] G. R. Granholm, P. J. Cefola, and W. L. Harris, "Bridging the Tech Gap: Using STEM Internships to Accelerate Innovation in the U.S. Air Force and Space Force," *AIAA SCITECH 2022 Forum. American Institute of Aeronautics and Astronautics*, Jan. 03, 2022. doi: 10.2514/6.2022-1998.

[17] S. Abdul Halim, M. H. Othman, A. G. Buja, N. N. Abdul Rahid, A. A. Sharip, and S. M. Md Zain, "C19-SmartQ: Applying Real-Time Multi-Organization Queuing Management System Using Predictive Model to Maintain Social Distancing," *International Journal of Interactive Mobile Technologies (iJIM)*, vol. 15, no. 06. International Association of Online Engineering (IAOE), p. 108, Mar. 30, 2021. doi: 10.3991/ijim.v15i06.20597.

[18] A. Obukhov, D. Dedov, A. Siukhin, and A. Arkhipov, "Mobile Simulator Control System for Isolating Breathing Apparatus of Software-Hardware Platform", *Int. J. Interact. Mob. Technol.*, vol. 14, no. 08, pp. pp. 32–42, May 2020.

[19] F. Al Huda, H. Tolle, and R. Andrie Asmara, "Realtime Online Daily Living Activity Recognition Using Head-Mounted Display", *Int. J. Interact. Mob. Technol.*, vol. 11, no. 3, pp. pp. 67–77, Apr. 2017.

[20] Sharma, R., & Dhabliya, D. (2019). A review of automatic irrigation system through IoT. International Journal of Control and Automation, 12(6 Special Issue), 24-29. Retrieved from www.scopus.com

[21] Ana Rodriguez, Kristinsdóttir María, Pekka Koskinen Pieter van der Meer, Thomas Müller. Machine Learning Techniques for Multi-criteria Decision Making in Decision Science. Kuwait Journal of Machine Learning, 2(4). Retrieved from http://kuwaitjournals.com/index.php/kjml/article/view/214