TAFITI PROJECT

COMPARATIVE ANALYSIS OF RASPBERRY PI PICO & STM32F103C8T6FOR CUBESAT AND NANOSATELITE APPLICATIONS:

PERFOMANCE, ECONOMY AND ENVIRONMENTAL RESISLIENCE

1.INTRODUCTION

The remarkable reduction in the cost and complexity of CubeSat and Nanosatellite development has effectively lowered the barrier to entry for space exploration, enabling a broader spectrum of participants, such as universities, research centres, and even individual enthusiasts, to design and launch their own satellite missions. This fundamental shift in the space paradigm directly influences the viability of employing Commercial Off-The-Shelf (COTS) components, which were traditionally deemed unsuitable for the rigorous demands of space environments.

At the core of any CubeSat or nanosatellite mission lies the On-Board Computer (OBC), serving as the central intelligence that monitors and controls all satellite operations. The OBC is tasked with critical functions including receiving, validating, and distributing commands to various subsystems, detecting and recovering from anomalies—particularly those induced by space radiation—maintaining precise spacecraft timekeeping, and processing data collected by payloads.

This report undertakes a comprehensive, data-driven comparative analysis of two widely adopted microcontrollers: the Raspberry Pi Pico and the STM32F103C8T6. The evaluation focuses on their technical specifications, economic advantages, and their inherent suitability and limitations when subjected to the harsh conditions of the space environment.

2.PERFOMANCE AND TECHNICAL SPECIFICATIONS

**2.1Raspberry Pi Pico**

The original Raspberry Pi Pico, built around the RP2040 silicon platform, features a dual-core Arm Cortex-M0+ processor operating at a clock speed of 133MHz.This symmetric dual-core architecture provides a solid foundation for various embedded tasks. The newer Raspberry Pi Pico 2 and Pico 2W, which are based on the more advanced RP2350 microcontroller. These variants deliver a significant performance upgrade, incorporating either dual Arm Cortex-M33 cores or dual open-hardware Hazard3 RISC-V cores, both clocked at 150MHz. The Arm Cortex-M33 core, notably, includes a Floating-Point Unit (FPU) and has been observed to be 2x as fast as the M0+ of the RP2040, enhancing its capacity for more complex mathematical computations. This flexibility allows for initial prototyping and development on the more economical original Pico, with the option to transition to a higher-performance variant as mission requirements evolve, thereby reducing the need for a complete redesign and mitigating development risk.

On memory, the original RP2040 is equipped with 264KB of on-chip SRAM and 2MB of on-board QSPI Flash Memory. The RP2350 doubles this to 520KB of SRAM and 4MB of QSPI Flash Memory. The notable increase in flash memory is good in applications using CircuitPython or MicroPython as it provides more space for storing files and application code.

Both the RP2040 and RP2350 have rich peripheral handling capabilities for various applications. Both have 26 General Purpose Input-Output (GPIO) pins (3-Analogue inputs-RP2040, 4 Analogue Inputs- RP2350), 2 UARTs, 2 SPI controllers, 2 I2C Controllers and USB 1.1. The RP2040 has 16PWM channels and 8 PIO (Programmable Input/Output) State machines. Whereas, the RP2350 has 24PWM channels and 12 PIO state machines. The Pico 2W variant specifically integrates single-band 2.4GHz wireless interfaces (802.11n) using the Infineon CYW43439 chip, connected via SPI to the RP2350, offering Wi-Fi and Bluetooth Low Energy (BLE) capabilities.

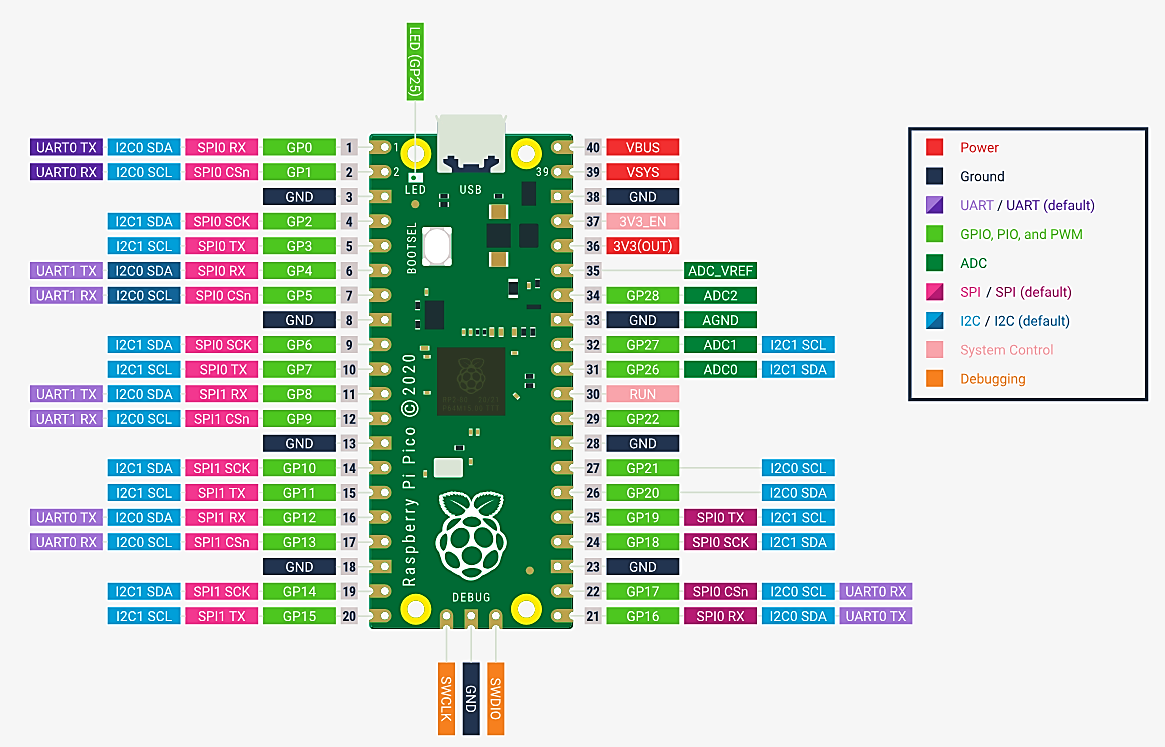


Figure 1: The pinout for a Raspberry Pi Pico Rev3 Board

The power consumption of the RP2040 in the active mode is ~90mA (at 133MHz). In Idle or Sleep Mode the current draw is in the micro-ampere range (specific modes can go down to ~100uA)

**2.2STM32F103C8T6**

The STM32F103C8T6, a prominent member of STMicroelectronics' STM32F103xx medium-density performance line. It incorporates a single-core Arm Cortex-M3 32-bit RISC core. This CPU operates at a maximum frequency of 72MHz, delivering a performance of 1.25 DMIPS/MHz (Dhrystone 2.1) with zero wait state memory access. The core also features single-cycle multiplication and hardware division, which enhances its efficiency for various embedded control applications.

For memory, the STM32F103C8T6 includes 64KB of Flash memory, with some variants in the broader STM32F103xx family offering up to 128KB, and 20KB of SRAM.

The peripheral set and I/O capabilities of the STM32F103C8T6 are more compared to the Raspberry Pi Pico. It has 37 general-purpose I/O pins. These I/Os are highly flexible, being mappable on 16 external interrupt vectors, and notably, almost all are 5V-tolerant, which offers broad compatibility with a wide array of external sensors and actuators.

Communication interfaces are comprehensive, including CANbus, 2 I2C interfaces, IrDA, LINbus, 2 SPI interfaces,3 USARTs and a USB 2.0 full-speed interface. Additional integrated peripherals include a 7-channel DMA controller (supporting timers, ADC, SPIs, I2Cs, and USARTs), motor control PWM, Power-On Reset (POR), Power-Down Reset (PDR), Programmable Voltage Detector (PVD), and a robust set of seven timers. These timers comprise three 16-bit general-purpose timers, one 16-bit motor control PWM timer with dead-time generation and emergency stop, two watchdog timers (independent and window), and a 24-bit SysTick timer. The microcontroller also integrates two 12-bit, 1µs Analog-to-Digital Converters (ADCs) with up to 16 channels, a 0-3.6V conversion range, dual-sample and hold capability, and an integrated temperature sensor.

The power consumption/current draw of the STM32F103C8T6 is ~36mA (at 72MHz) in the active mode, ~24mA in Run Mode, ~5mA in Sleep Mode, ~50uA in Stop Mode and ~2uA in Standby Mode

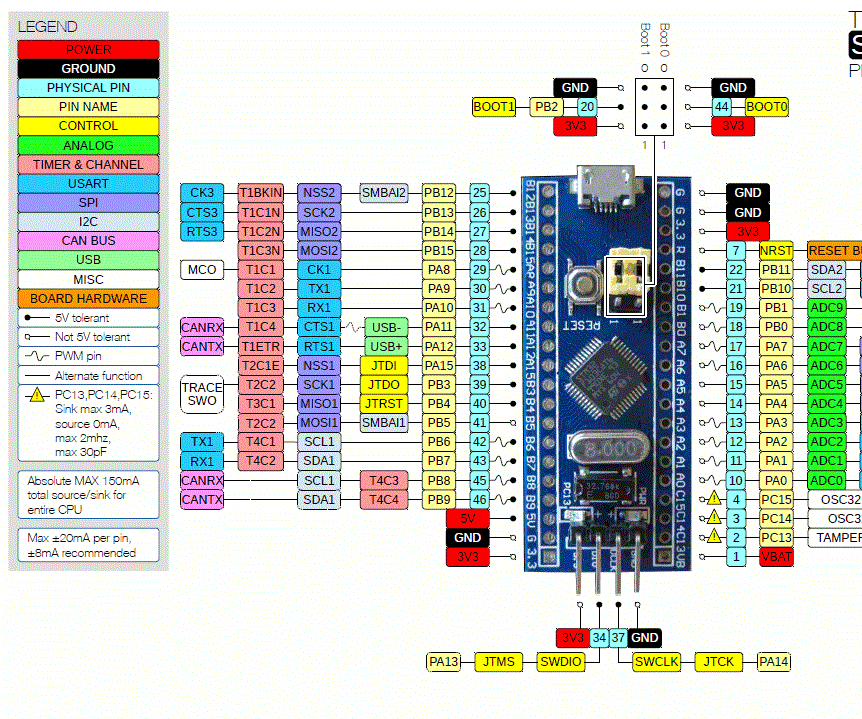


Figure 2: STM32F103C8T6 (Bluepill) pinout

|  |  |  |  |
| --- | --- | --- | --- |
| Feature | Raspberry Pi Pico (RP2040) | Raspberry Pi Pico 2/2W (RP2350) | STM32F103C8T6 |
| CPU Core | Dual-core Arm Cortex-M0+ | Dual Arm Cortex-M33 or RISC-V | Single-core Arm Cortex-M3 |
| Clock Speed (MHz) | 133 | 150 | 72 |
| FPU | No | Yes (Cortex-M33) | No |
| On-chip SRAM (KB) | 264 | 520 | 20 |
| On-board Flash (MB) | 2 | 4 | 0.064 (64KB) |
| Total GPIO Pins | 26 | 26 | 37 |
| Analog Inputs | 3 | 4 | 10 (12-bit ADC) |
| UART Controllers | 2 | 2 | 3 |
| SPI Controllers | 2 | 2 | 2 |
| I2C Controllers | 2 | 2 | 2 |
| PWM Channels | 16 | 24 | 7 (general-purpose, motor control) |
| USB Controller | 1.1 (Host/Device) | 1.1 (Host/Device) | 2.0 Full-Speed |
| CAN Interface | No | No | Yes (2.0B Active) |
| PIO State Machines | 8 (2 blocks) | 12 (3 blocks) | No |
| Wireless (Wi-Fi/BLE) | No | Yes (Pico 2W only) | No |
| Input Voltage Range (V DC) | 1.8–5.5 | 1.8–5.5 | 2.0–3.6 |
| Operating Temp. Range (°C) | -20 to +85 | -20 to +85 | -40 to +85 (up to +105 for family) |
| Dimensions (LxWxH, mm) | 51 x 21 x 3.9 | 51 x 21 x 3.9 | 54 x 22 (board) |
| Weight (g) | 3 | 4 | 9 (board) |

Table 1: Comparative Technical Specifications (Raspberry Pi Pico vs. STM32F103C8T6)

**Power Consumption.**

**Raspberry Pi Pico (RP2040)**

Operating Voltage: 3.3V (typical)

|  |  |  |
| --- | --- | --- |
| Mode | Current | Power (Watts) |
| Active (133MHz) | ~90mA | **0.297W** |
| Sleep Mode | ~100µA | **0.00033W** |

**STM32F103C8T6**

Operating Voltage: 3.3V (typical within 2.0-3.6V range)

|  |  |  |
| --- | --- | --- |
| Mode | Current | Power (Watts) |
| Active (72MHz) | ~36mA | **0.119W** |
| Run Mode | ~24mA | **0.079W** |
| Sleep Mode | ~5mA | **0.017W** |
| Stop Mode | ~50µA | **0.00017W** |
| Standby Mode | ~2µA | **0.0000066W** |

**Power Efficiency**

In the Active Mode: The STM32F103C8T6 consumes 0.119W which is 60% less power than Pi Pico which consumes 0.297W (2.5 times more power consumption).

In the standby mode the STM32F103C8T5 consumes 0.0000066W which is about 50 times more efficient than the Pi Pico in sleep mode the Pi Pico consumes 0.00033W in the sleep mode.

3.ECONOMIC VIABILITY

Both the Raspberry Pi Pico and STM32F103C8T6 fall into the Commercial Off-The-Shelf category with remarkably similar pricing. The Raspberry Pi family offers several options within a narrow price range, with the original Pico costing between Ksh 520 and 1,040, the newer Pico 2 priced at Ksh 650, and the Wi-Fi enabled Pico W available for Ksh 780. The STM32F103C8T6 presents comparable costs, with the standalone microcontroller chip priced at Ksh 665 and the popular Blue Pill development boards ranging from Ksh 1,040 to 1,235.

This analysis reveals that both microcontroller options cost between Ksh 520 and 1,300, making them equally affordable for initial component acquisition. The minimal price difference means that neither platform offers a significant economic advantage over the other in terms of direct purchase costs. A true economic comparison for CubeSat applications must therefore consider broader factors including development time, environmental hardening requirements, and long-term reliability costs.

The cost landscape changes dramatically when comparing these COTS components to space-grade alternatives. Traditional space-qualified microcontrollers command premium prices that are orders of magnitude higher than their commercial counterparts. The VORAGO VA10820, a radiation-hardened Cortex-M0 processor designed specifically for space applications, costs between Ksh 65,000 and 169,000 for the basic chip. Specialized variants can reach extraordinary prices of Ksh 1,575,000 to 2,600,000, while high-performance processors from the Gaisler LEON series are priced on a quote-only basis, reflecting their ultra-premium positioning.

4.SUITABILITY FOR HARSH SPACE ENVIRONMENT

**Space Radiation Challenges**

Space contains high-energy particles that damage electronic components over time. Two main problems affect microcontrollers in space:

**Total Ionizing Dose (TID)** causes gradual damage that builds up over months or years. This leads to higher power consumption, timing changes, and reduced performance. In Low Earth Orbit, commercial parts may fail within a year due to radiation levels of 4-40 krad per year.

**Single Event Effects (SEE)** happen when a single high-energy particle hits the microcontroller. This can cause temporary errors in memory or logic, or permanent damage through destructive short circuits that can destroy the device. These include: They include:

* **Single Event Upsets (SEU):** Non-destructive, temporary bit flips in memory cells or logic states.
* **Single Event Latch-up (SEL):** A destructive event that can create a low-impedance path (short circuit) within the device, potentially leading to permanent damage or destruction if not mitigated.

**4.1Raspberry Pi Pico (RP2040)**

Has a radiation tolerance up to 50 krad, which is much better than expected for a commercial part. The main processor and input/output functions remained stable during testing. However, the onboard flash memory and SD card components failed under radiation, creating system interruptions. Replacing the flash memory with radiation-tolerant alternatives like MRAM could significantly improve reliability.

**STM32F103C8T6**

Has a radiation tolerance of up to 50 krad, with a slight clock speed reduction as radiation dose increased, eventually failing at 107 krad primarily due to flash memory failure. In space, the processor may run at about 85% of its programmed speed due to radiation effects. STMicroelectronics produces specialized radiation-hardened versions of STM32 microcontrollers specifically for space applications, providing 50 krad immunity without requiring additional qualification testing.

**Mitigation Requirements**

Commercial microcontrollers need additional protection systems to work reliably in space. These include physical shielding, backup systems, specialized radiation-tolerant memory, watchdog circuits, and sophisticated error-correction software. While the microcontroller itself costs only a few dollars, these protection systems add significant weight, complexity, and cost to the overall design.

**Economic Reality**

The initial low cost of commercial microcontrollers becomes misleading when all necessary protection systems are included. For short missions with lower radiation exposure, commercial parts with minimal protection may be cost-effective. However, for longer missions requiring high reliability, the total system cost including all mitigation measures can make purpose-built radiation-hardened components more economical despite their much higher initial price.

Microcontroller Radiation Performance and Cost Comparison:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Microcontroller Model | CPU Core | Typical TID Tolerance (krad(Si)) | SEL Immunity (LETth in MeV-cm²/mg) | Key Radiation Hardening Features | Approximate Unit Cost (Ksh) |
| Raspberry Pi Pico (RP2040) | Dual-core Arm Cortex-M0+ | ~50 krad(Si) (observed) | Not specified for COTS, potential vulnerability | COTS, requires external mitigation for flash/SD card | Ksh 520 - 1,040 |
| STM32F103C8T6 | Single-core Arm Cortex-M3 | ~50 krad(Si) (observed for ARM Cortex-M) | Not specified for COTS, potential vulnerability | COTS, requires external mitigation for flash | Ksh 421 - 1,234 |
| VORAGO VA10820 | Arm Cortex-M0 | >300 | Latch-up immune | Rad-hard, DICE latches, TMR, EDAC, specific packaging | Ksh 65,000 - 2,600,000 |
| Gaisler GR716B | LEON3FT SPARC V8 | 100 | >118 | Radiation-hardened, fault-tolerant, EDAC, hermetic package | Price on request |
| Gaisler GR740 | Quad-core LEON4FT SPARV8 | 300 | >125 | Radiation-hardened, fault-tolerant, ceramic/plastic package | Price on request |

**5. STM32F103C8T6 Communication Challenges for CubeSat Applications**

**1. Processing Power Limitations**

**Insufficient Data Rate Handling**

The STM32F103C8T6 microcontroller, based on a single-core 72 MHz architecture, lacks the computational capacity to handle high-data-rate communication protocols efficiently. Its limited processing bandwidth creates difficulty when simultaneously managing communication tasks and onboard data processing. These constraints are especially critical in real-time scenarios, where the microcontroller cannot perform complex modulation or demodulation while executing other time-sensitive operations.

**Mathematical Processing Limitations:** The absence of a hardware floating-point unit (FPU), forcing the system to rely on slow software-based floating-point computations. This inadequacy severely impacts digital signal processing (DSP) capabilities, making it impractical to implement advanced modulation schemes. Complex algorithms, such as forward error correction and encryption, place heavy demands on the CPU, consuming excessive cycles and introducing latency that further hampers communication responsiveness.

**2. Memory Constraints**

**Severe RAM Limitations:** With only 20 KB of SRAM, the STM32F103C8T6 cannot allocate sufficient memory for essential communication buffers or full protocol stacks. Modern communication protocols typically require significantly more memory for queuing and handling packet data, making them incompatible with the microcontroller’s limited resources.

**Flash Memory Restrictions**

The 64 KB flash memory imposes additional limitations by restricting the size and complexity of the firmware that can be stored. Communication libraries with comprehensive protocol support often exceed this capacity, forcing developers to make difficult trade-offs between features. Furthermore, the limited flash space complicates over-the-air firmware updates, which are critical for long-duration missions requiring post-launch software enhancements.

**3. Modern Communication Interface Gaps**

**Missing Critical Interfaces**

The STM32F103C8T6 does not feature native support for modern wireless standards such as Wi-Fi or Bluetooth, necessitating external modules that increase system complexity and integration overhead. Additionally, the microcontroller lacks high-speed peripheral interfaces, such as full USB 2.0 and Ethernet, which are often required for advanced communication modules. Compared to alternatives like the Raspberry Pi Pico, which offers more capable USB implementations, the STM32 falls short in interface support.

**Advanced Protocol Support Issues**

Implementing modern communication standards and technologies such as software-defined radio (SDR) is infeasible due to the microcontroller’s limited processing power. It cannot accommodate adaptive protocols that require real-time switching or dynamic resource allocation. Additionally, its limited capacity for secure computation restricts the implementation of modern encryption and authentication protocols, critical for mission data security.

**4. Advanced Modulation and Signal Processing**

**Digital Signal Processing (DSP) Inadequacy**

The STM32F103C8T6 lacks the resources to execute complex modulation schemes such as Quadrature Amplitude Modulation (QAM) or Orthogonal Frequency Division Multiplexing (OFDM) and cannot implement sophisticated error correction algorithms. Real-time signal quality analysis is also infeasible, which undermines the ability to dynamically adjust system parameters based on link conditions. Adaptive communication systems, which are increasingly essential in space applications, are beyond the microcontroller’s capabilities.

**Software-Defined Radio Limitations:** The microcontroller is unsuitable for software-defined radio applications due to its inadequate computational performance and memory. It cannot support frequency agility, generate complex waveforms, or operate on multiple frequency bands simultaneously. These limitations prevent the implementation of advanced, flexible communication systems tailored to dynamic mission requirements.

**High-Data-Rate Downlinks:** The STM32F103C8T6 cannot efficiently handle the downlink of high-resolution images or large volumes of scientific data. Bulk data transfers, which are common in CubeSat missions, are significantly hampered by limited throughput. Additionally, the microcontroller lacks the resources to perform real-time data compression, a vital technique for maximizing downlink efficiency.

Summary on critical limitations:

The STM32F103C8T6 is fundamentally constrained in several key areas:

1. It lacks the computational capability to support modern, high-data-rate communication.
2. It has insufficient memory to implement full-featured communication protocol stacks.
3. It cannot perform real-time digital signal processing or error correction.
4. It is incapable of supporting software-defined radio functionality.
5. It does not natively interface with modern high-speed communication hardware

To overcome these challenges, a hybrid architecture is recommended. The STM32F103C8T6 should be relegated to support functions such as radio power management, basic UART or SPI communication, and watchdog monitoring. Meanwhile, a more capable microcontroller like the Raspberry Pi Pico can serve as the primary communication controller.

This approach leverages the strengths of both systems:

* The STM32 handles low-level interfacing, power control, and auxiliary communication tasks.
* The Pi Pico is responsible for computationally intensive communication processing, including protocol handling, modulation/demodulation, and error correction.
* The combined system balances reliability with performance, providing a robust solution for CubeSat communications in low-Earth orbit and beyond.

1. RECOMMENADTION

**6.1 Use Cases and Subsystem Suitability**

**Individual Use**

When used individually, the Raspberry Pi Pico (RP2040 or RP2350) is suitable for**:**

1. Payload Control: For non-critical scientific instruments, cameras, or sensor arrays where data processing is required. Its dual-core nature and PIO can handle complex timing and parallel operations.
2. On-board Data Processing: Due to its higher clock speed and larger RAM than STM32F103, it's better suited for light image processing, data compression, or machine learning inference (edge computing) for experimental payloads.
3. Communication Interface: Can serve as a robust interface for S-band or X-band transceivers, handling modulation/demodulation if the data rates are within its capability.

It is less suitable for critical flight software ADCS (Attitude Determination and Control System) primary control due to its radiation susceptibility and reliance on external flash.

When used individually the, STM32F103C8T6 is suitable for:

1. Power Management Unit (PMU): Due to its low power modes and integrated ADCs, it can effectively monitor battery voltage, current, and control power switching to various subsystems.
2. Attitude Determination and Control System (ADCS) - Secondary/Backup: Can handle sensor interfaces (magnetometers, gyroscopes, sun sensors) and basic attitude algorithms. Not recommended as the sole ADCS controller unless significant rad-hardening is implemented.
3. Command and Data Handling (CDH) - Distributed Nodes: As an intelligent peripheral controller or a watchdog for other subsystems.
4. Temperature Monitoring and Thermal Control: Its ADCs and GPIOs are suitable for reading temperature sensors and controlling heaters

The STM32F103C8T6 is less suitable for high-throughput data processing, primary flight software for critical missions without heavy redundancy.

**6.2 Used Together (Heterogeneous Architecture):**

Using both microcontrollers together in a heterogeneous architecture can leverage their strengths and mitigate individual weaknesses, offering a more robust and flexible system compared to using either alone.

**Primary/Secondary Controller Setup:**

1. STM32F103C8T6 as a "Safe Mode" Controller / Watchdog: Its lower power consumption and slightly better (though still COTS-level) observed radiation tolerance could make it a good candidate for a critical, low-power "safe mode" controller. It can monitor the health of the RP2040, trigger resets, power cycles, or switch to a minimal operational mode if the RP2040 encounters issues due to radiation.
2. Raspberry Pi Pico (RP2040) as the "Main" Payload Processor/High-Performance CDH: The RP2040 can handle the computationally intensive tasks of the main payload (e.g., image processing, advanced algorithms), high-rate data acquisition, and primary communication protocols. It offloads these tasks from the STM32, which can then focus on critical "housekeeping" functions.

**Distributed Subsystem Control:**

1. STM32F103C8T6 for PMU and ADCS Front-end: One STM32 could manage the power subsystem, monitoring power lines and controlling switches. Another could interface directly with ADCS sensors and actuators (magnetorquers) for basic control, reporting data to the RP2040.
2. Raspberry Pi Pico for Central CDH and Payload Interface: The RP2040 acts as the central command and data handling unit, orchestrating communication between the STM32-controlled subsystems and the ground station, processing scientific data from the payload, and executing higher-level mission commands. Its USB and PIO capabilities make it versatile for interfacing with various specialized sensors or transceivers.

**Redundancy and Fault Tolerance:**

1. By using both, we can implement *cross-strapping* or *redundancy schemes*. If one fails, the other can attempt to take over critical functions (e.g., safe mode, communication).
2. The STM32 could act as a hardware watchdog for the RP2040, and vice-versa, or a more complex health monitoring system could be implemented.
3. The RP2040's higher processing power allows for more sophisticated software-based error detection and correction (EDAC) for its own internal memory and potentially for data received from other subsystems.

**6.3 Why this combination is better and more economical than using only one or a rad-hard MCU:**

1. Cost-Effectiveness: The combined cost of multiple Pico and STM32 boards is still orders of magnitude lower than a single rad-hard microcontroller. This allows for increased redundancy and distributed intelligence without breaking the budget.
2. Performance vs. Reliability Trade-off: The RP2040 provides the necessary processing power for modern CubeSat payloads and data handling, while the STM32 can handle more robust, lower-power critical functions. This allows for a balance between cutting-edge capabilities and fundamental system survival.
3. Flexibility and Modularity: Designing with distinct microcontrollers for different subsystems promotes modularity. Issues in one subsystem are less likely to bring down the entire system, and development can proceed in parallel.
4. Risk Mitigation: While neither is rad-hard, using them together with intelligent power management, watchdogs, and software fault tolerance can significantly improve the overall system resilience against radiation-induced failures compared to a single COTS device. The STM32 can provide a basic "limp mode" capability if the more complex RP2040 fails